CS423: Data Mining

Introduction to Graph Mining

Jakramate Bootkrajang

Department of Computer Science
Chiang Mai University

Outlines

- Motivation
- Graph Theory Refresher
- Apriori-based Frequent Subgraph Mining Algorithm

Motivation

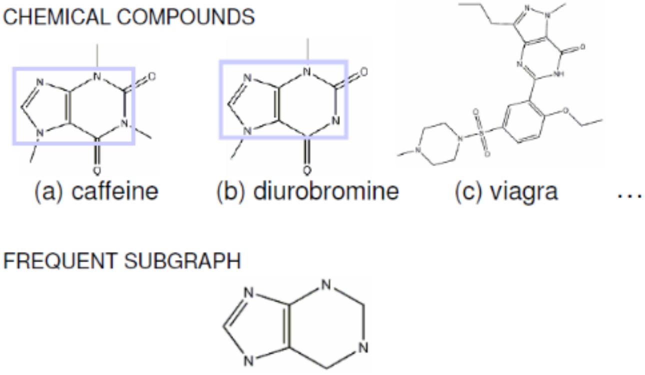
- Graph is a useful modelling tool for representing entities and their relationships.
- Example
  - Internet
    - Vertices: computers, smartphones, routers
    - Edges: communication links
  - Social Network:
    - Vertices: users
    - Edges: friendship
  - Chemical molecule:
    - Vertices: atoms
    - Edges: chemical bonds

Graph Mining: what kind of knowledge ?

- What are the characteristics of these graphs?
- Are there any interesting patterns in these graphs?
- How to differentiate abnormal social network from a normal one?
- How do these graph evolve over time?
- And so on ...

Patterns mining from graph

- In this class, we will learn about frequent subgraph mining



K. Borgwardt and X. Yan (KDD'08)

Areas where frequent graph patterns are useful

- Program control flow analysis
  - Detection of malware/virus
- Network intrusion detection
- Anomaly detection
- Graph compression

Graph theory refresher

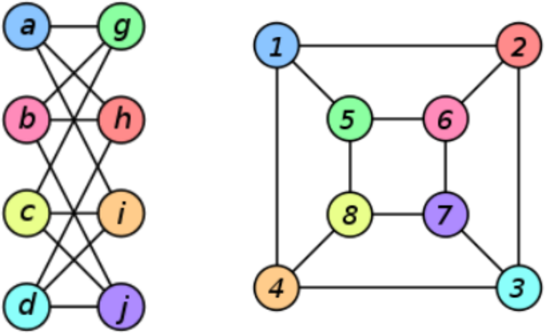
- A graph $G(V, E)$ is a structure which comprised of two set
  - V is a set of vertices
  - E is a set of edges
- A labelled graph $G(V, E, L_V, L_E)$ is a graph where vertices and edges have names.
  - L_V is a set of vertex labels
  - L_E is a set of edge labels
- Labels need not be unique
  - For example, labels may represent chemical elements

Graph theory refresher [2]

- A graph is said to be **connected** if there is path between every pair of vertices
- A graph $G_s(V_s, E_s)$ is a **subgraph** of another graph $G(V, E)$ iff
  - $V_s \subseteq V$ and $E_s \subseteq E$
- Two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are **isomorphic** if they are topologically identical
  - one can be transformed into the other simply by renaming vertices

Graph Isomophism

- The transformation is $f(a) = 1, f(b) = 6, f(c) = 8, f(d) = 3, ...,$



Subgraph Isomophism

- Given two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$: find an isomorphism between G_2 and a subgraph of G_1
- NP-complete problem

Frequent Subgraph Mining: Problem definition

- Given
  - D: a set of undirected, labelled graphs
  - σ: support threshold, $0 < \sigma \leq 1$
- Goal:
  - Find all connected, undirected graphs that are sub-graphs in at least $\sigma|D|$ of input graphs

Frequent subgraph mining methods

- Apriori-based approach (in this class)
- Pattern-growth approach

Apriori algorithm recap

C_k = candidate itemset of size k, L_k = frequent itemset of size k

1. Find frequent set L_{k-1}
2. Joining step
   - C_k is generated from joining member in L_{k-1}
3. Pruning step
   - k-itemset which one of its $(k-1)$-item(sub)set is not frequent cannot be frequent, and should be removed
4. Repeat until C_k is empty.

Apriori algorithm recap

Set of transactions : { {1,2,3,4}, {2,3,4}, {2,3}, {1,2,4}, {1,2,3,4}, {2,4} }

min_support: 3

L_1

Item	Support
1	3
2	6
3	4
4	5

C_2

Item	Support
{1,2}	3
{1,3}	2
{1,4}	3
{2,3}	4
{2,4}	5
{3,4}	3

L_2

Item	Support
{1,2}	3
~~{1,3}~~	~~2~~
{1,4}	3
{2,3}	4
{2,4}	5
{3,4}	3

L_3

Item	Support
{1,2,4}	3
{2,3,4}	3

{1,2,3} and {1,3,4} were pruned as {1,3} is not frequent.

FSG: Frequent Subgraph Mining Algorithm

- Proposed by Kumarochi & Karypis in 2001 and revised in 2004

  Notation: k-subgraph is a subgraph with k edges.

  Init: Scan the transactions to find L_1 and L_2, the set of all frequent 1-subgraphs and 2-subgraphs, together with their counts;

  For(k=3; $L_{k-1} \neq \emptyset$, k++)

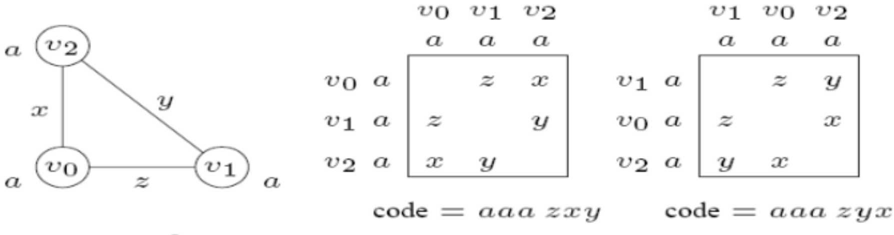
  1. **Candidates generation**: C_k from the set of frequent $k-1$-subgraphs
  2. **Candidates pruning**: Requires that each of $k-1$-subgraphs of the candidate is also frequent
  3. **Frequency counting**: Scan the database to count the occurrences of $c \in C_k$
  4. $L_k = \{c \in C_k | c \text{ has counts no less than } \sigma\ \}$
  5. Return $L_1 \cup L_2 \cup L_3 \cup \ldots L_k$

FSG: Candidates generation

- Follows apriori algorithm by joining two frequent k-subgraph that has common (k-1)-subgraph
  - Need to check if (k-1)-subgraphs are isomorphic (time consuming)
- To avoid that FSG uses canonical labelling to encode graph structure into string.
  - Comparing two subgraphs is just string comparison

Canonical label of graph

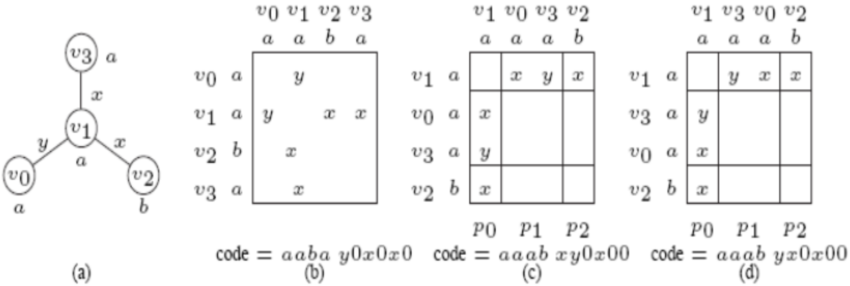
- Lexicographically largest (or smallest) string obtained by concatenating upper triangular entries of adjacency matrix in column-wise manner (after symmetric permutation).
- Uniquely identifies a graph and its isomorphs
  - Two isomorphic graphs will get same canonical label



(a) G^3 (b) (c)

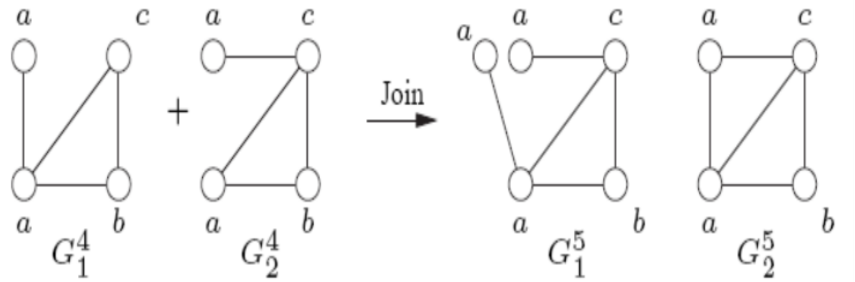
Canonical label of graph

- Canonical labelling is also difficult problem. There are $|V|!$ permutations to try.
- FSG uses inherent properties of vertices that don't change across isomorphic mappings to reduce the size of canonical label set.
  - It groups vertices by **degree and label** and only permute within the groups.

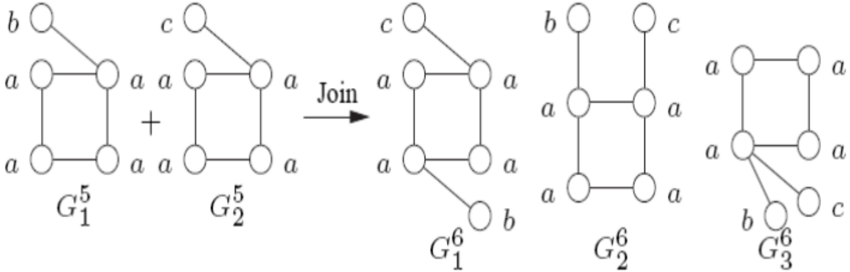


FSG: subgraph joining

- Two k-subgraphs which have (k-1)-subgraph are combined to form (k+1)-subgraph

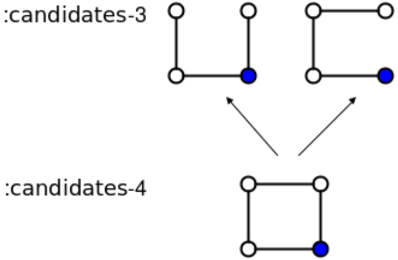


FSG: subgraph joining [2]



FSG: Candidate pruning

- Every (k-1)-subgraph must be frequent (downward closure property)
- For all the (k-1)-subgraphs of a given k-candidate, check if downward closure property holds
- FSG also uses canonical labels to remove duplicate candidates



FSG: Frequency counting example

- Subgraph isomorphism check for each candidate against each graph transaction in database
  - naive and so computationally expensive
- FSG uses transaction identifier (TID) lists
  - For each frequent subgraph, keep a list of TID that support it
- To compute frequency for G^{k+1}
  - Find intersection of TID lists of its subgraphs
  - If size of intersection $<$ minsup: prune G^{k+1}
  - Else: Subgraph isomorphism check only for graphs in the intersection

FSG: Frequency counting

Transactions

$g^{k-1}_1, g^{k-1}_2 \subset T1$
$g^{k-1}_1 \subset T2$
$g^{k-1}_1, g^{k-1}_2 \subset T3$
$g^{k-1}_2 \subset T6$
$g^{k-1}_1 \subset T8$
$g^{k-1}_1, g^{k-1}_2 \subset T9$

Frequent subgraphs

$TID(g^{k-1}_1) = \{ 1, 2, 3, 8, 9 \}$

$TID(g^{k-1}_2) = \{ 1, 3, 6, 9 \}$

Candidate

$c^k = join(g^{k-1}_1, g^{k-1}_2)$

$TID(c^k) \subset TID(g^{k-1}_1) \cap TID(g^{k-1}_2)$

$$\Downarrow$$

$TID(c^k) \subset \{ 1, 3, 9 \}$

- Perform subgraph-iso to T1, T3 and T9 with c^k and determine $TID(c^k)$
- Note, TID lists require a lot of memory.

References

- Introduction to Graph Mining by Sangameshwar Patil. Systems Research Lab. TRDDC, TCS, Pune. http://www.iiserpune.ac.in/~pgoel/Sangam-Intro2GraphMining.ppt
- Graph Mining by Ehud Gudes. https://www.cs.bgu.ac.il/~orlovn/presentations/graph_mining_seminar_2009.ppt