A Survey of Vertices Associated Subgraph Mining Methods

Pathan Vasim A. and Bhavsar Swati A.,

1,2Department of Computer Engineering, MCCORE, Nashik, Maharashtra, India

Abstract—In this survey we overview the definitions and methods for mining subgraphs, which are associated with a given set of vertices. We review different methods for finding frequent subgraphs. Then we discuss difference between frequent subgraphs and subgraphs associated with particular set of vertices, methods for finding subgraphs associates with given set of vertices and how this method can help to vertex associated subgraphs in a Protein-protein interactions (PPI).

Keywords—Subgraph Mining, PPI, Subgroup Discovery.

I. INTRODUCTION

Graph data is used as important resource in various research fields such as social networking, social sciences, bioinformatics. Graphs are structures formed by a set of vertices (also called nodes) and a set of edges that are connections between pairs of vertices. A single graph may contain large amount of information so it is a measure challenge to discover information of interest in a given graph dataset. There are many data mining techniques are used to extract new or useful knowledge from a graph dataset. This problem can be defined in another way as frequent subgraph discovery in a graph dataset. Frequent subgraphs are those that occur most often in a given graph dataset than a provided support threshold value. Frequent subgraph discovery problem is relevant to problem of frequent itemsets for Boolean association rules. The Apriori algorithm is an influential and fundamental algorithm for mining frequent itemsets for Boolean association rules. Apriori algorithm based on following key concepts

・Frequent Itemsets:
The sets of item which has minimum support i.e. this set of item occur more or equal number of times than given support threshold (denoted as Ls for i-th itemset).

・Apriori Property:
Any subset of frequent itemset must be frequent i.e. if \{AB\} is a frequent itemset, both \{A\} and \{B\} should be a frequent itemset.

・Join Operation:
To find \(L_k\) a set of candidate k itemsets are generated by joining \(L_{k-1}\) with itself.

A. Algorithm

\(C_k\): Candidate itemset of size k
\(L_k\): frequent itemset of size k.
\(L_k\) = \{frequent itemset\}

K = 1
while (\(L_k\) = NULL) do
\(C_{k+1}\) = candidates generated from \(L_k\);
for each transaction \(t\) in database do
Increment the count of all candidates in \(C_{k+1}\) that are contained in \(t\)
\(L_{k+1}\) = candidate in \(C_{k+1}\) with min_support.
End
End

IV. GSPAN: GRAPH-BASED SUBSTRUCTURE PATTERN MINING

Xifeng Yan and Jiawei Han proposed an algorithm for frequent subgraph mining called as gSpan[3]. gSpan builds a new lexicographic order among graphs, and maps each graph to a unique minimum DFS code as its canonical label. Based on this lexicographic order and adopts the depth-first search strategy to mine frequent connected subgraphs efficiently. gSpan reduces the search space by using two techniques DFS lexicographic order and minimum DFS code which form a novel canonical labelling system to support DFS search.

A. DFS Lexicographic Order

・Each graph map to DFS code (a sequence), building a new lexicographic ordering among these codes and constructing a search tree based on this lexicographic order.

・While performing a DFS in a graph a DFS tree is constructed. One graph can have several different DFS trees. For example, graphs in Fig. I(b)-(d) are isomorphic to that in Fig. I(a). The thickened edges in Fig. I(b)-(d) represent three different DFS trees for the graph in Fig. I(a). The depth-first discovery of the vertices forms a linear order.
B. DFS Code

In DFS code each edge represented using 5 tuple, \((i, j, l_i, l_j, l_{ij})\) where \(l_i\) and \(l_j\) are labels of \(v_i\) and \(v_j\) respectively and \(l_{ij}\) is the label of edge between them.

C. DFS Code Tree

In a DFS Code Tree, each node represents a DFS code, the relation between parent and child node complies with the parent-child relation. The relation among siblings is consistent with the DFS lexicographic order. That is, the pre-order search of DFS Code Tree follows the DFS lexicographic order:

In gSPAN algorithm all possible edges with two vertices are constructed and for each edge all the frequent subgraphs containing that edge are discovered. This procedure repeats until all possible subgraphs are covered and when the subgraph becomes enough large only graphs which containing

![DFS Code Tree](image)

Figure 1: Depth-First Search Tree

Subgraphs are considered. This procedure recursively called to grow and find all their frequent subgraphs.

V. FSG ALGORITHM

Michihiro Kuramochi and George Karypis proposed frequent itemset discovery algorithm called as FSG [4]. FSG is computationally efficient and finds all frequent connected subgraphs in a large graph dataset. The significant features of FSG are as below:

i. it uses a sparse graph representation that minimizes both storage and computation.

ii. it increases the size of frequent subgraphs by adding one edge at a time, allowing it to generate the candidates efficiently.

iii. it includes various optimizations for candidate generation and frequency counting which enables it to scale to large graph databases and

iv. it uses classy algorithms for canonical labelling to uniquely identify the various generated subgraphs without having to recourse to computationally expensive graph and subgraph isomorphism computations.

FSG algorithm based on Apriori principle which increase the size of frequent itemsets by adding a single item at a time, this algorithm increases the size of frequent subgraphs by adding an edge one by one. FSG initially enumerates all the frequent single and double edge graphs. Then, it extends each subgraph by single edge. Further, it counts the frequency for each of these candidates, and prunes subgraphs that do not satisfy the support constraint. Discovered frequent subgraphs satisfy the downward closure property of the support condition, which allows us to effectively prune the lattice of frequent subgraphs.

VI. VERTICES ASSOCIATED SUBGRAPHS

A. Vertices Associated Subgraphs

Pieter Meysman and et al proposed an approach to discover subgraphs most associated with a given set of vertices [6]. For example, in a biological graph subgraphs of interest might be those that are associated with a particular disease. Here PPI networks are considered as biological graphs and each vertex in PPI network represent protein information. An example of subgraphs associated with vertices is found in figure 2. Here circular vertices represent protein of one type and other shaped vertices represent proteins of another types. Frequent subgraphs associated with orange colored are shown in figure 2. Proposed subgraph mining algorithm uses a canonical labelling to reduce search space and remove isomorphic graphs.

![Example graph data set with subgraph associated with the orange vertices](image)

Figure 2: Example graph data set with subgraph associated with the orange vertices.

B. Canonical Labeling

In canonical labelling/representation, each vertex in the subgraph is given a unique numeric identifier. Each edge in the representation is then made of two vertex identifiers and two sets of vertex labels if the vertices are labelled. This procedure is based on the canonical representations used by several other graph mining algorithms, which are used to avoid redundant subgraph matching. Figure 3 shows an example of a graph with its canonical representation.

![Example subgraph](image)

Figure 3: Example subgraph (left) and its canonical representation (right).

C. Significant Subgraph Mining Algorithm

Subgraphs are significant if they have a higher frequency than expected from random networks. The Significant Subgraph Mining algorithm proposed by Pieter Meysman includes following measure steps

a) Candidate Generation

In this step all possible subgraphs are constructed. Initially subgraph consist of a single edge and one or two vertices. For a single vertex there is edge from the vertex to itself called as self-loop. For two vertices subgraphs constructed in twice by considering each vertex as source and destination respectively. Subgraph will extend if pass the predefined support cut-off frequency.

b) Candidate Pruning
Generated candidate subgraphs can be pruned based on their significant enrichment probability value calculated using hypergeometric probability distribution. Candidate subgraphs which satisfy threshold probability value are selected for further extension and these subgraphs extended until their size larger than the provides size.

Acknowledgment

I would like to thanks Prof. S. A. Bhawsar for her kind help, guidance and support. I also express my special thanks to Department of Computer Engineering Matoshri College of Engineering, Nashik, for their timely support and valuable guidance.

References


