Efficient Algorithm for Mining Frequent Subgraphs (Static and Dynamic) based on gSpan

K.Lakshmi
Asst.Prof, Departmentof MCA,
Sir M.Visvesvaraya Institute of Technology,
Bangalore.

T.Meyyappan, PhD.
Professor, Department of Computer Science & Engineering,
Alagappa University, Karaikudi.

ABSTRACT
Frequent sub graph mining is another active research topic in data mining. A graph is a general model to represent data and has been used in many domains like chemo informatics and bioinformatics. Mining patterns from graph databases is challenging since graph related operations, such as sub graph testing, generally have higher time complexity than the corresponding operations on item sets, sequences, and trees.

We investigated new approaches for frequent graph-based pattern mining in graph datasets and found that a novel algorithm called span (graph-based Substructure pattern mining), has been used as the standard for comparing performance of new algorithms. It is based on the pattern growth approach of frequent sub graph mining and hence discovers frequent substructures without candidate generation. span is based on a lexicographic order, and maps each graph to a unique minimum DFS code as its canonical label. Based on this lexicographic order, gSpan adopts the depth-first search strategy to mine frequent connected sub graphs efficiently. Based on the literature survey done, algorithms based on pattern growth approach and DFS strategy are found to be better in performance than algorithms based on Apriori approach and BFS strategy. In this paper we propose a new algorithm based on gSpan, for a special class of graphs characterised by the existence of unique node lablels.

General Terms
Graph mining, subgraph, Lexicographic ordering, labeled graphs.

Keywords
Parallel programming, frequent subgraph mining, DFS code, Isomorphism.

1. INTRODUCTION
Graph mining has become a very active area of research. Frequent subgraph mining refers to the problem of subgraph isomorphism detection. In sub graph isomorphism detection, a mapping f from the nodes of one given graph g1 to the nodes of another given graph g2 is a bijection that preserves all edges and labels. The main problem with this sub graph isomorphism is its high computational complexity. Also it is a known fact that it is NP-complete. Many heuristics have been developed to speed up sub graph isomorphism by using special canonical labeling of the graphs; none of them, however, can avoid an exponential worst-case computation time.

Contribution. In this paper, we propose a new algorithm based on gSpan. It can mine frequent sub graph from a special kind of graphs, characterized by nodes with unique node labels. It also targets to reduce the time complexity, using parallel programming. If the entire graph dataset can fit in main memory, the proposed method can be applied directly; To the best of our knowledge, the two techniques, DFS lexicographic order and minimum DFS code, introduced in gSpan are the best , which form, a novel canonical labeling system, to support DFS search . But still the problem of finding minimum DFS code used in gSpan is also NP- complete. The proposed algorithm addresses this issue by using a modified DFS representation. It retains all the advantages of gSpan, while taking advantage of the multi core processing technology by using the concept of parallel programming to improve the performance of the algorithm. Number of duplicate graphs generated may be comparatively little more than gSpan algorithm as mining of sub graphs from frequent single edge graphs are done in parallel.

The remainder of the paper is organised as follows. In Sect. 2, we introduce our basic concepts and terminology. Graphs with unique node labels and their possible representations are discussed in Sect.3. Proposed graph mining algorithm based on gSpan and its time complexity is discussed in Sect. 4. In Sect. 5, we discuss the possible applications of the graphs with unique labels. Theoretical conclusions from the proposed work are discussed in Sect. 6.

2. BASIC CONCEPTS AND NOTATIONS
In this section, the basic concepts and terminology used is introduced. In this paper we focus on a special class of undirected labelled simple graphs, graphs with unique no labels. For any graph G and any pair of vertices x,y., the condition L(x) <> L(y) holds if x <>y. Throughout the rest of this paper we consider graphs from this class only.

2.1 Definition1
A Labeled graph can be represented by a 3-tuple G=(V,E,L) where V is a set of vertices, E is a set of edges, L is a set of labels for the vertices. For simplicity we assume the labels for all the edges to be empty. According to this definition a Labelled graph can be represented by a 3-tuple G=(V,E,L) where V is a set of vertices, E is a set of edges, L is a set of labels.

2.2 Definition2
Given a graph G = (V, E), a graph Gs = (Vs , Es) is a subgraph of G if Vs ⊆ V and Es ⊆ E, and is denoted by Gs ⊆ G. Two graphs G1 = (V1, E1) and G2 = (V2, E2) are isomorphic, if they are topologically identical to each other, that is, there is a vertex mapping from V1 to V2 such that each edge in E1 is mapped to a single edge in E_2 and vice versa. In the case of labeled graphs, this mapping must also preserve the labels on the vertices and edges. When a set of graphs {Gi } are isomorphic to each other, they all are said to belong to the same equivalence class.

Given two graphs G1 = (V1, E1) and G2 = (V2, E2), the problem of sub graph isomorphism is to find an isomorphism between G2 and a sub graph of G1. In other words, the sub graph isomorphism problem is to determine whether or not G2 is embedded in G1.

3. GRAPHS WITH UNIQUE NODE LABELS AND THEIR REPRESENTATIONS

Graphs with unique node labels are represented as a labeled graph with the labels mapped into a sequence of integers.

3.1 Example

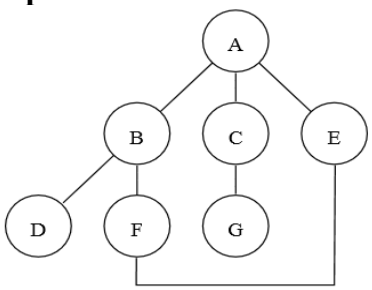

Fig. 1a. Example graph g

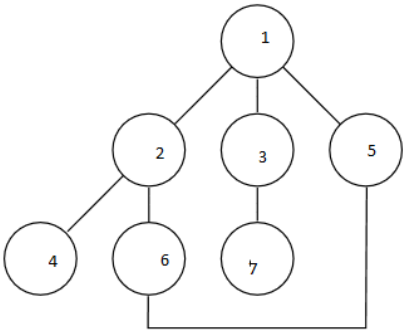

Fig. 1b Label representation of g

3.2 DFS Tree

When performing a depth-first search [3] in a graph, we construct a DFS tree. One graph can have several different DFS trees, if the DFS traversal is done by selecting vertices arbitrarily. We assume to follow lexicographic ordering of the vertices in DFS traversal, hence there will be unique DFS tree generated for each graph.

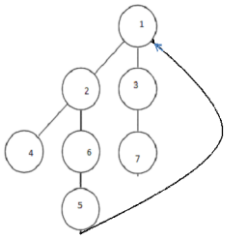

Fig. 1 DFS tree for graph g

3.3 DFS Code

Each graph is mapped into an edge sequence (ei), where i = 0,1,2,..|E|-1, called the DFS code of the graph. It is generated from the DFS tree for the given graph

Basically a DFS code is generated in the following way. Start with a new vertex in the lexicographic order. Add a new vertex and a forward edge that connects one vertex in the old code with this new vertex. Then add all backward edges that connect this new vertex to all other vertices in old code. Repeat this until all edges are added to the code.

3.4 DFS Code Tree

A DFS code tree represents the relationship between different Graphs in a given graph data set. In a DFS code tree each node represents a DFS code.

In a DFS code tree, the n+1 th level represents graphs with n-edges.

3.5 Right Most Extension

Rightmost Extension [2] is a kind of extension methods used widely in graph mining. It is based on the Depth First Search (DFS). The rightmost path is the straight path from root of DFS Tree to rightmost node. The extension to a graph on rightmost path is named rightmost extension. Rightmost extension includes forward extension and backward extension. Backward extension extends a graph by adding an edge between the rightmost node and another node on the rightmost path. Forward extension extends a graph by appending a new node to one of nodes on the rightmost path.

4. gSpan ALGORITHM

gSpan is a novel algorithm which discovers frequent substructures without candidate generation. It maps each graph into a minimum DFS code based on lexicographic ordering. The search strategy used is depth first search. The algorithm has very good parallel and scale up properties.

4.1 The gSpan algorithm

Algorithm gSpanMining(D,MinSup,S)
1:sort labels of the vertices and edges in D by frequency;
2:remove infrequent vertices and edges;
3:relabel the remaining vertices and edges (descending);
4:S0=code of all frequent graphs with single edge;
5:sort S0 in DFS lexicographic order;
  S=S0;
6: for each code s in S0 does
7: gSpan(s, D, MinSup, S);
8: D: =D-s;
9: if |D|<Minus;
10: break;

Algorithm gSpan(s, D, MinSup, S)
1: if s! =min(s), then
2: return
3: insert s into S
4: set C to {}
5: scan D once; find every edge e such that s can be right-most
  Extended to frequent s*e;
  Insert s*e into C;
6: sort C in DFS lexicographic order;
7: for each s*e in C do
8: Call gSpan(s*e, D, MinSup, S);
9: return

5. PROPOSED ALGORITHM

We propose a new algorithm based on gSpan [2], for a special class of graphs that are characterized by nodes with unique labels[1]. Also we incorporate the concept of parallel programming, to take advantage of the existing multicore processor technology to reduce the time complexity.

Algorithm ProposedMining(D,MinSup,S)

1:sort labels of the vertices and edges in D by frequency;
2: remove infrequent vertices and edges;
3: relabel the remaining vertices and edges (descending);
4:S0=code of all frequent graphs with single edge;
5: sort S0 in DFS lexicographic order;
S=S0;
6: for each code s1, s2, s3… in S0 do
7: proposed (s1, D, MinSup, S);
Proposed (s2, D, MinSup, S);
Proposed (s3, D, MinSup, S);
proposed (s..,D,MinSup,S); ……
// Use parallel programming here
8: if |D|<MinSup;
9: break;

Algorithm Proposed(s, D, MinSup, S)

1: insert s into S
2: set C to {}
3. Scan D once, finds every edge e such that s can be right-most
Extended to frequent s*e;
4: insert s*e into C;
5: sort C in DFS lexicographic order;
6: for each s*e in C do
7: Call proposed(s*e, D, MinSup, S);
8: return.

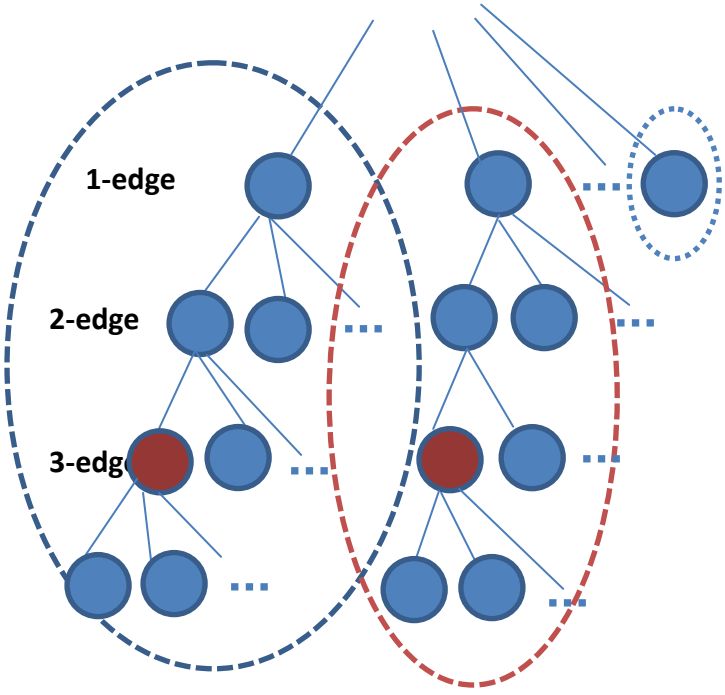

Fig. 2 Parallel programming in gSpan

Fig 1 shows how the concept of parallel programming can be applied in gSpan algorithm. It shows a hierarchical search space, ie. a DFS code tree. Portions covered with dotted ellipses can be executed in parallel. Figure 2. Shows the pruning step applied in gSpan to avoid generation of duplicate frequent sub graphs. It is not required in the proposed algorithm as it is for the special class of graphs.

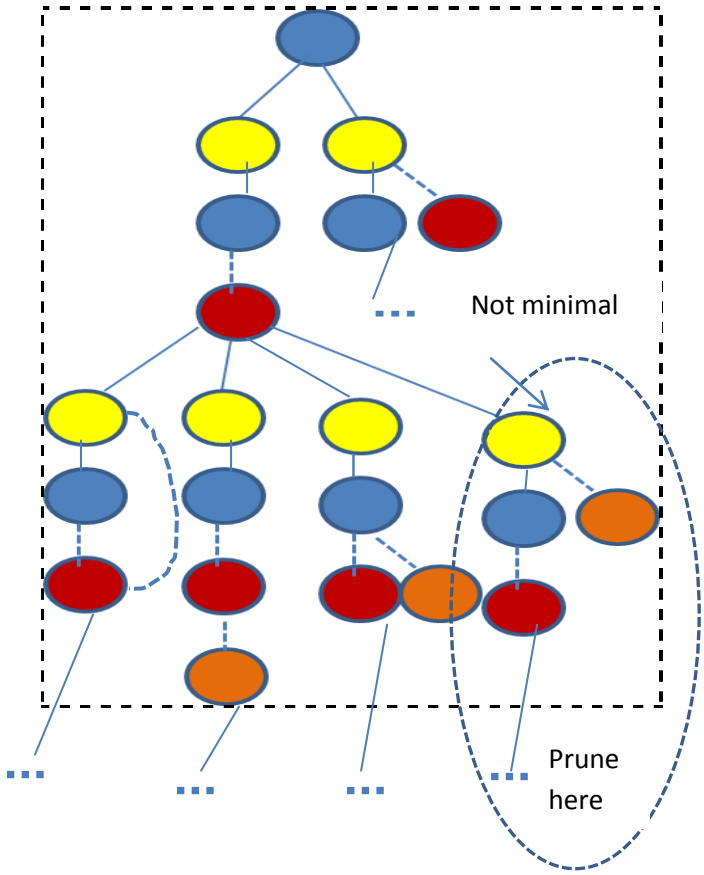

Fig. 3 Pruning in gSpan to avoid generation of duplicate graphs

In gSpan, the problem of finding the minimum DFS code for a graph itself is NP-complete. The proposed algorithm based on gSpan using the concept of parallel programming is for a special category of graphs. The time complexity of DFS code generation for such graphs is shown to be quadratic.

5. APPLICATIONS OF GRAPHS WITH UNIQUE LABELS

Graphs with unique node labels have got several applications. Biological network analysis, Web document analysis, social network analysis, network monitoring are a few to mention.

5.1Network Monitoring

Computer networks can be modeled as graphs with unique node labels, as each node in a network a client, a server or a router can be uniquely identified with its IP address, or the MAC address. Techniques are required to improve network monitoring and proactive detection of network anomalies, by frequent sub graph mining of time series of graphs, representing the states of a network over a period of time can be helpful for this.

5.3 Web Document Analysis

Another important application of graphs with unique node labels is analysis of Web documents. Each term that is present in a document is represented as a node in the graph. Terms that are present multiple times are represented only once.

5.4 Social Network Analysis

Another important application of graphs with unique node labels is social network analysis. Social interactions among individuals can be naturally and commonly represented as information networks[4]. Due to diverse entities types and relationships, along with temporal information of interactions, temporal heterogeneous information networks (HIN) can be used to capture this kind of relational structures. In essence, HIN is a directed graph, in which nodes are individuals of different entity types and edges stand for multiple kinds of relationship. Hence, the social activities in a certain time period can be represented as a temporal snapshot. By collecting networks in a series of periods, we can construct a transaction database of networks, in which each HIN stands for a graph recording social interactions in a certain period.

6. CONCLUSION

In this paper, we propose a new algorithm based on gSpan using parallel programming for frequent sub graph mining. The proposed algorithm is for a special class of graphs. If constraints are imposed on any class of graphs, we usually lose some representational power. But despite the restrictions imposed, the special class of graphs has some interesting applications which are discussed in this paper. We would take the application discussed in this paper for experimental verification of the efficiency of the proposed algorithm. In our proposed work, we modify the gSpan algorithm (i) to incorporate parallel programming, (ii) eliminate pre-pruning as it is not required (iii) simplify DFS code generation (iv) apply it for both static and dynamic graphs (v) apply it for both directed and undirected graphs.

6. REFERENCES

[1] P. Dickinson, H. Bunked, A. Dadejiichand M. Kraetzl. On graphs with unique node labels. In Proceedings of the IAPR Workshop on GBR, pages 13, 2003.

[2] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. Proc. 2nd IEEE Int'l Conf.

[3] Yuhua Li Quan Lin Gang Zhong Dongsheng Duan Yanan Jin Wei Bi, A Directed Labeled Graph Frequent Pattern Mining Algorithm based on Minimum Code. In the proceedings of Third International Conference on Multimedia and Ubiquitous Engineering 2009.

[4] Chang Hun You, Lawrence B. Holder and Diane J. Cook :Graph-based Data Mining in Dynamic Networks: Empirical Comparison of Compression-based and Frequency-based Subgraph Mining in International Conference on Data Mining Workshops IEEE, 2004.

[5] Hsun-Ping Hsieh, Cheng-Te Li, Mining Temporal Subgraph Patterns in Heterogeneous Information Networks: In the proceedings of IEEE International Conference on Social Computing / IEEEInternational Conference on Privacy, Security, Risk and Trust.

[6] Nijssen, S. and Kok, J., A quickstart in frequent structure mining can make a difference. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery andData Mining, ACM, 2004, pp. 647–652.

[7] Bianca Wackersreuther, Peter Wackersreuther, Annahita Oswald : Frequent Subgraph Discovery in Dynamic Networks, ACM 978-1-4503-0214-2 2010.

[8] L. T. Thomas, S. R. Valluri, and K. Karlapalem. Margin:Maximal frequent subgraph mining. Proc.6th IEEE Int'l Conf. Data mining (ICDM '06), pp. 1097-1101, 2006.

[9] M. Kuramochi and G. Karypis. Grew-a scalable frequent subgraph discovery algorithm. In ICDM, pages 439–442,2004.

[10] J. Huan, W.Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism. UNC computer science technique report TR03-021, 2003.

[11] J. Huan, W. Wang, J. Prins, and J. Yang. Spin: Mining maximal frequent subgraphs from graph databases. UNC Technical Report TR04-018, 2004.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein.Introduction to Algorithms. MIT Press, 2001, Second Edition.

[13] J. Huan, W.Wang, and J. Prins. Efficient mining of frequent subgraph in the presence ofisomorphism. UNC computer science technique report TR03-021, 2003.

[14] [4] A. Inokuchi, T.Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In PKDD'00.

[15] M.Kuramochi and G. Karypis . Frequent Subgraph Discovery. In ICDM'01.

[16] M. Deshpande, M. Kuramochi, and G. Karypis. Frequent sub-structure based approaches forclassifying chemical compounds. In Proc. of 2003 IEEE International Conference on Data Mining(ICDM), 2003.

[17] Yong Liu, Jianzhong Li, Hong Gao, JPMiner: Mining Frequent Jump Patterns From Graph Databases. In the proceedings of Sixth International Conference on Fuzzy Systems and Knowledge Discovery 2009.

[18] Nijssen, S. and Kok, J., Faster association rules for multiple relations. In IJCAI'01: SeventeenthInternational Joint Conference on Artificial Intelligence, 2001, vol. 2, pp.891–896.