Indexing Sparse Graphs for Similarity Search

Divyashree Bhoye
M. E. Student, Dept. of Computer Engg. Vishwabharati Academys College of Engineering, Ahmednagar, Maharashtra, India.
divyashree.bhoye@gmail.com

Abstract: Graphs are widely used model for representing complex structure data and similarity search for graphs has become a fundamental research problem. Data which is schema less, such as chemical compounds, can be efficiently modelled using graph structure. Many applications include chemical compounds, road networks, bio-informatics, social networks, pattern recognition etc. Managing large amount of graph data in these domains is a very challenging problem. A fundamental query is to efficiently index and perform similarity search on a large collection of graphs. Exact matching is often too restrictive on complex structures hence similarity search on graph related application is an important operation. The project includes an efficient indexing mechanism on graphs structure for similarity search. The project achieves this by decomposing graphs into k-Adjacent Tree patterns. Using these k-Adjacent Tree patterns and the lower bound estimation of their edit distance filtration is done to obtain the candidate set of graphs for further similarity search. The project focuses optimizing the value of 'k' to guarantee the filtering ability of the index for graph set. In this way the project helps to implement a better and more optimized similarity search.

Keywords: Graph edit distance, Similarity Search, k-adjacent tree.

I. INTRODUCTION

There has been rapid growth in use of graphs as data models recently, such as protein interaction in bio-informatics, chemical compounds, road networks, social networks, pattern recognition. Similarity search over a large data set of graphs is a crucial and fundamental issue in graph based application. For example, in the road network, the cities can be considered as vertices of the graph and the roads connecting these cities can be considered as edges across the corresponding vertices.

In similarity search, we look for graphs in a database which are similar to a query graph. Two graphs are said to be similar to each other by judging the size of their maximum common subgraph. Suppose that graph in Figure 1 is used as a query graph; then {g3} is the result of the subgraph search. Such queries are very useful for an exploration purpose in many applications like drug design and pattern recognition, to extract and identify a small set of molecules and graph models for further analysis.

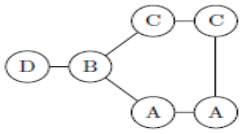

Fig 1. Query graph

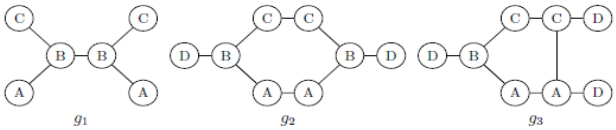

Fig 2. Sample Graph dataset

Sequential searching on a large data set of graphs introduces a huge computational cost. As sequential search has low efficiency, another method is employed called as a filter-and verification method, to speed up the search efficiency of graph similarity matching over a graph set and indexing can be done on the graph data set to filter and reduce the candidates[9].

 © 2016, IJREAM All Rights Reserved.

Indexing method focuses on breaking the graphs into pieces and evaluates the similarity between them by pairing corresponding pieces between the database and the query. Motivated by the Q-Gram [2] indexing technique of strings, the graphs are decomposed into k-adjacent trees (AT) patterns, then use their common k-AT patterns for edit distance estimation and then indexed to solve the similarity search of sparse graphs.

The rest of this paper is organized as follows: Section 2 introduces the research work related to this paper, Section 3 presents the k-adjacent tree index and filtering principle based on the concept of k-adjacent tree. Section 4 includes the proposed work, section 5 gives the mathematical model. Section 6 shows performance evaluation on the proposed work, and Section 7 concludes the paper

II. RELATED WORK

Sequential scan is very costly method, as one has to access the whole database in one by one fashion. Also subgraph isomorphism is an NP-complete problem[1]. Thus, filter-and-verification method is employed to speed up the search efficiency of graph similarity search over a graph data set. Here, firstly one generates a set of candidates that satisfy necessary conditions of the edit distance constraints, and then verify them with edit distance computation.

Filtering is the main phase to improve the search efficiency; a lot of indexing techniques are proposed recently to speed up the filtering phase. Most of this research work can be categorized into two groups; *frequent-subgraph* based indexing and *graph-decomposition* based indexing.

Yan et al. introduce a novel indexing technique *G-Index* based on frequent subgraph patterns[3,4]. The basic indexing features uses frequent subgraph structures. Shang et al. propose a novel indexing technique *QuickSI* to efficiently compute verification phase for testing subgraph isomorphism[5]. Cheng et al. propose a nested inverted-index called FG-index to avoid candidate verification by exploiting frequent subgraphs and edges as indexing features. Frequent-sub graph-based indexing methods have two main shortcomings: The effectiveness and efficiency of such kind of methods depend on the quality of selected features and it is difficult to construct and maintain the index because the frequent subgraph mining algorithm usually takes a very long time to compute[3,4].

Williams et al. have developed three kinds of graph decomposition schemas, Clique Decomposition, Modular Decomposition, and Node Label Decomposition (NLD) to decompose a graph dataset, and to describe the results of a graph decomposition, Directed Acyclic Graphs (DAGs) are constructed and a Graph Decomposition Index (GDI) is proposed to support graph similarity search[8]. Tian and Patel have proposed an indexing method by incorporating graph structural information in a hybrid index structure called NH-index. Graph decomposition indexing methods suffer from two main drawbacks[8]: They have to enumerate all connected sub graphs, and therefore, complexity of graph decomposition is exponential to the graph size that is being decomposed and the frequency information existing in the graph decomposition results is not utilized for improving the efficiency of graph similarity search[8].

This paper includes a novel graph decomposition method called k-Adjacent Tree (k-AT). It is inspired by the idea of "Q-Gram" from string matching[8]. A graph is decomposed into a set of k-Adjacent Trees and the decomposed results are indexed by a k-AT index. A lower bound of the edit distance between graphs is derived and used for filtering graphs. This guarantees the absence of false negatives. This method incorporates both graph decomposition methods and frequent subgraph methods.

III. K-ADJACENT TREE INDEX

First, we will discuss how to use k-Adjacent tree pattern decomposition for lower bound estimation[8]. This indexing method uses the idea of Q-Grams index which is used in

 © 2016, IJREAM All Rights Reserved.

string matching[9]. To avoid the structural complexity of graph patterns, we use adjacent tree patterns for index construction. Adjacent tree is able to preserve its structural information well. It is much faster to do similarity search on adjacent tree than on graph grams. The following figure shows representation of the k-adjacent tree for a given graph.

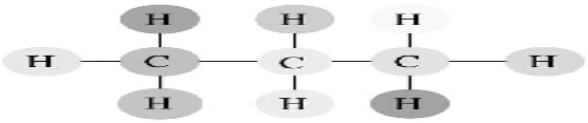

Fig. 3 Given Graph

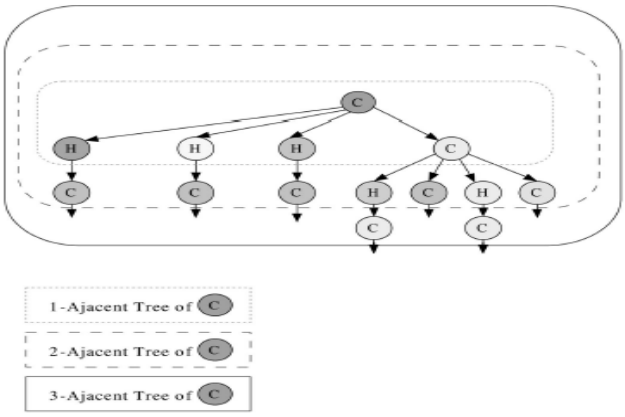

Fig. 4 Representation as k-adjacent tree

We generate all the k-ATs of each graph in the graph data set and store them in a table. For a query graph Q, we also generate its k-ATs, and for each graph G in the data set we calculate the number of common k-ATs of Q and G. Then we use inequality given below to check whether graph G belongs to the candidate set of the query or not.

$|Candk|=|\{G|G \in D,| k\text{-}ATS(Q) \cap k\text{-}ATS(G)| \geq |V(G)| - €.\delta(G)k\}|$

The following figure represents a basic block diagram for the system.

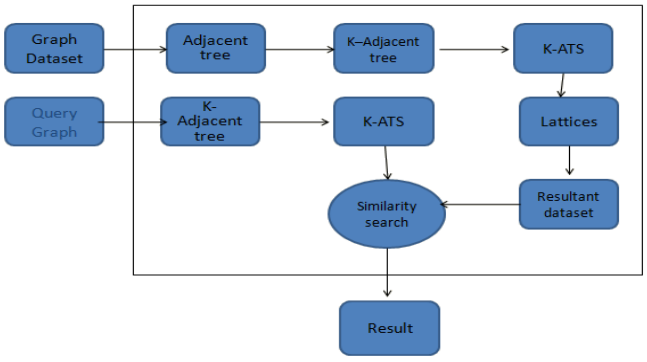

Fig. 5 Block diagram of system

IV. PROPOSED WORK

A. *Graph Similarity*

The key idea in graph similarity is that a node in one graph is similar to a node in another graph if their neighbourhoods are similar. The methods in the related work section that solve the graph similarity problem yield results that are not very intuitive. As our system combines both Frequent-sub graph-based and graph decomposition based indexing methods so manages to capture both the local and global topology of the graphs[8]; therefore, it is able to spot small differences between the graphs and give results that agree with intuition.

B. *Edit Distance*

The Graph Edit Distance between two graphs G1 and G2 is the minimum number of GEOs needed to transform G1 to a graph isomorphic to G2. The definition of edit distance of two graphs gives us a measurement to quantify the difference of two graphs [9].

The GEO can be one of the following six operations:

1. Delete an edge from the graph.
2. Insert an edge between two disconnected vertices.
3. Delete an isolated vertex from the graph.
4. Insert an isolated vertex into the graph.
5. Change the label of a vertex.
6. Change the label of an edge.

The inequality given below is used to check whether graph G belongs to the candidate set of the query or not.

$|Cand_k|=|\{G|G \in D,| k\text{-}ATS(Q) \cap k\text{-}ATS(G)| \geq |V(G)| - €.\delta(G)^k\}|$

One of the limitations of system is that it is sensitive to the selection of the parameter k, which specifies the number of components that the decomposition should produce [8].

The problem that is how to choose the proper value of ‘k’ to guarantee the filtering ability of the index for a specific graph set. Question arises is smaller value of k better? The answer is negative because a matching on small adjacent tree sets may

 © 2016, IJREAM All Rights Reserved.

lead to mismatches on the whole graph. For example, two graphs can have all their 1-ATs matched with the other, but all in the wrong places[9]. This can be analogized with the Q-Gram method on string matching[9]; both too large and too small length Q-Grams can weaken the filtering ability of Q-Gram index. This is because too small grams cannot reflect the global structure of the string while too large grams may not be preserved under minor edit operations, this causes the trade-off in gram size selection. Due to the structural complexity of graph, it is difficult to give a theoretical evaluation on how to choose the optimum k.

However, when k=1, it leads to the solution for nonsparse graphs. To index nonsparse graphs, we can choose 1-AT index, without worrying about that the large k will weaken the filtering ability of the index. Although this is a feasible solution for non sparse graphs, it is better to use this index only on sparse graphs because when k is too small, the index sometimes can show unsatisfactory performance on candidate filtering [2].

We propose an algorithm to optimise the value of 'k' that is the depth of lattice. It can be achieved using following algorithm.

Algorithm Depth_Lattice(q, B)

Step 1: For the given lattice, find a "short" and quite "orthogonal" basis called the reduced basis.

Basis reduction is naturally associated with the problem of finding the shortest lattice vector

Step 2: Enumerate all lattice points falling inside a sphere, which is centered at the query point, for the nearest lattice point.

Here, q is the query point and B is the basis matrix. The sphere to be enumerated must have a radius such that it contains at least one lattice point.

V. MATHEMATICAL MODEL

A. *Problem description*

Let S be the System which indexes the graph data such that

S={G,I,KID,F,R}

Where

G={gc | g € G ^ c € N}

I={ic | ic=f(gc) ^ g € G ^ c € N}

F={fk | fk=f(I,KID) ^ k€KID}

R={ri | ri€G ^ i€N}

S_1={$v_{11}e_{11}$,v12e12,…..v1n,e1n}

S_2={$v_{21}e_{21}$, v22e22,…..v2n,e2n }

S_3={v_{31}, e_{31}, v32e32,…..v3n,e3n }

$_{A1}$={$v'_{11}e'_{11}$, $v'_{12}e'_{12}$…….. $v'_{1n}e'_{1n}$}

$_{A2}$={$v'_{21}e'_{21}$, $v'_{22}e'_{22}$…….. $v'_{2n}e'_{2n}$}

$_{A3}$={$v'_{31}e'_{31}$, $v'_{32}e'_{32}$…….. $v'_{3n}e'_{3n}$}

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of K-AT index by comparing with other indexes. Since our proposed method is a combination of graph-decomposition-based indexing method and frequent-subgraph-based indexing method, we choose two indexing techniques to compare with our proposed method[8]. FG-index is a kind of frequent-subgraph-based indexing technique while DAG index is a kind of graph-decomposition-based indexing technique[8]. We used AIDS dataset as it is a widely used real chemical compounds data set.

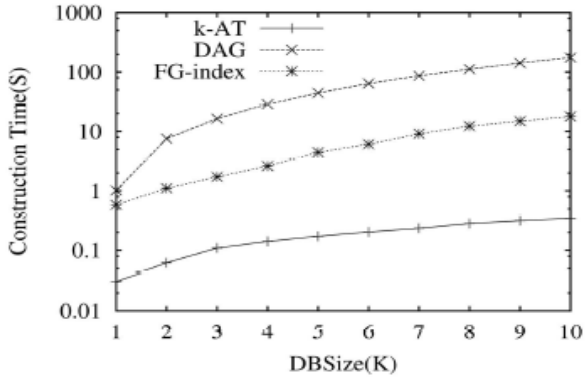

Fig. 6 Index construction time

 © 2016, IJREAM All Rights Reserved.

Our indexing method K-AT incorporates graph decomposition-based indexing techniques with frequent-subgraph-based indexing techniques; therefore, it has less space costs than DAG. Fig. shows that K-AT is superior to the other two indexing methods in index construction time. In the procedure of constructing indexes, the main CPU cost is subgraph isomorphism test.

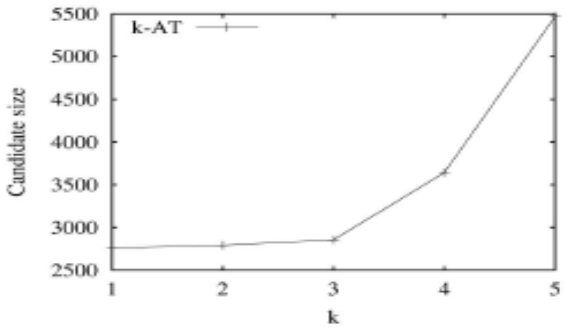

Fig. 7 Scalability versus K for candidate set

Fig. 11 shows the scalability performances of k-AT when k varies. We can see that k-AT has a good filtering ability when k is smaller than 4. This shows that the query processing response time increases linearly as K increases[6,7].

VII. CONCLUSION

We evaluate the global similarity between the graphs by decomposing them into smaller pieces (k-ATs) and pairing up these pieces. k-AT records more structured information than a normal graph decomposition based indexing method and also maintaining the simple structure of tree[8]. This gives us a method for indexing and candidate filtering for similarity search in a graph data set. Also experimental results evince that when applied to large graph data set filtering on k-AT index can be both fast and accurate.

ACKNOWLEDGMENT

The authors thank a lot to all who supported them in their work. And they would like to express their sincere gratitude and appreciation to all the staff members for the patience, guidance, help and for being their greatest source of information.

REFERENCES

[1] T.H. Cormen, "*Np Completeness Introduction to Algorithms*," W. Yu, ed., second ed., vol. 7, pp. 620-630. China Machine Press, 2007.

[2] *Efficiently Indexing Large Sparse Graphs for Similarity Search* Guoren Wang, Bin Wang, Xiaochu Yang, Member, IEEE Computer Society, and Ge Yu, Member, IEEE IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 3, MARCH 2012.

[3] X. Yan, P.S. Yu, and J. Han, "*Graph Indexing: A Frequent Structure-Based Approach*," Proc. ACM SIGMOD, pp. 335-345,2004.

[4] J.H. Xifeng Yan and P.S. Yu, "*Graph Indexing Based on Discriminative Frequent Structure Analysis*," ACM Trans. Database Systems, vol. 30, no. 4, pp. 960-993, 2005.

[5] H. Shang, Y. Zhang, X. Lin, and J.X. Yu, "*Taming Verification Hardness: An Efficient Algorithm for Testing Subgraph Isomorphism*," Proc. 34th Int'l Conf. Very Large Data Bases, pp. 364-375, 2008.

[6] J. Cheng, Y. Ke, W. Ng, and A. Lu, "*Fg-Index: Towards Verification-Free Query Processing on Graph Databases*," Proc. ACM SIGMOD, pp. 857-872, 2007.

[7] J. Cheng, Y. Ke, and W. Ng, "*Efficient Query Processing on Graph Databases*," ACM Trans. Database Systems, vol. 34, no. 1, pp. 1-44,2009.

[8] O. Johansson, "*Graph Decomposition Using Node Labels*," doctoral dissertation, Royal Inst. Of Technology, 2001.

[9] Y. Tian and J.M. Patel, "*Tale: A Tool for Approximate Large Graph Matching*," Proc. 24th Int'l Conf. Data Eng., pp. 963-972, 2008.

 © 2016, IJREAM All Rights Reserved.