International Journal of Computer Science and Informatics

Volume 4 | Issue 2 Article 9

October 2014

VFG – INDEX: A NOVEL GRAPH INDEXING METHOD

A. PANKAJ MOSES MONICKARAJ
Dept. of Computer Science, Bharathiar University, Coimbatore, India, apankaj.moses@gmail.com

VFG – INDEX: A NOVEL GRAPH INDEXING METHOD

A. PANKAJ MOSES MONICKARAJ[1], K. VIVEKANANDAN[2], D. RAMYA CHITHRA[3]

[1]Doctoral Scholar, [3]Assistant Professor, Dept. of Computer Science, Bharathiar University, Coimbatore, India
[2]Professor, BSMED, Bharathiar University, Coimbatore, India

Abstract- The latest advancements in science and technology have observed large quantity of complicated structures and schema less data such as proteins, circuits, images, Web, and XML documents which can be modeled into various types of which one of the most dominant now a days are graphs. This has caused Graph Mining, one of the budding areas of research happening throughout. In this paper, we investigate the various algorithms for Graph Indexing which makes use of Frequent Sub graph as a key term for indexing, of which one the most dominant is FG-index algorithm. This algorithm is tested with AIDS dataset and an improved algorithm is proposed for effective indexing.

Keywords- *Graph, Graph Mining, Graph Indexing, Frequent Sub Graph*

1. INTRODUCTION

Data Explosion is one of the current major threats of which saving those data are again a problem. Effective storage and retrieval of data are one of the major key problems faced in this web world. Converting these data of any kind to any other format and storing may improve the space and the time complexity while retrieving the data. Recently, graphs are the one of the most effective method for modeling data of any kind say Chemical bonding [1], Proteins [2], three dimensional mechanical parts [3], Computer vision and images [4] .

To retrieve information from the database, queries are given and data are extracted. Similarly in a graph database, the sub graphs relevant to the given query are extracted. Consider a graph dataset G, for a given query q, the set of all sub graphs present in G with relevant to q are extracted as sub graphs. During sub graph search for a given query, if one graph is a sub graph of another leads to sub graph isomorphism [5]. To minimize the sub graph search problem, sub graph features are mined by means of the indexing algorithms to build an initial index [6]-[10]. Query processing is done by means of two steps filtering and candidate verification. In the initial step, false results are eliminated by using the index and candidate answer set is generated.

This set is verified whether the query is a sub graph of the database. As the candidate sub graph is much smaller than the complete database, it is easy to improve the efficiency. Let us consider 11-graph dataset, the sub graph features are mined to obtain four key features and an index is built. Take for the feature p1, the respective related sub graphs would be g_1, g_4, g_5, g_7. Given a query q, which contains the features of P_2 and P_4 (Figure:1).

Therefore $\{P_2 \cap P_4\} = \{g_3, g_6, g_7\} \cap \{g_1, g_2, g_8, g_{10}\}$ are the candidate graphs obtained as others are already filtered out. These results include the super graph of q which has both P_2 and P4. Hence, only these graphs should be evaluated to check sub graph isomorphism issue. Graph database: g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9, g_{10}, g_{11} Sub graph features: p_1, p_2, p_3, p_4

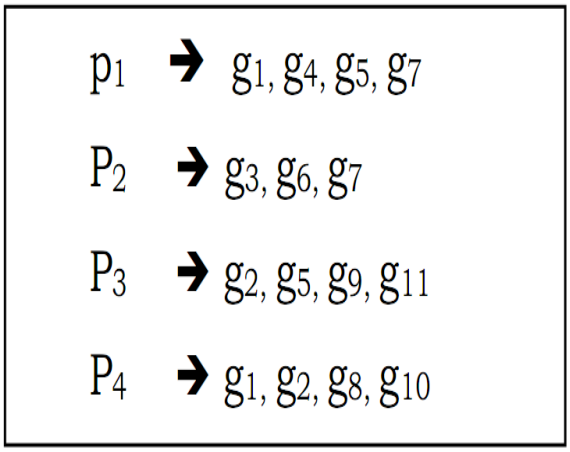


Figure: 1 sub graph query processing

The features can be extracted by any of the mine at once algorithms. These algorithms scan throughout the graph dataset and construct initial index.

In this paper, we propose a novel indexing method similar to FG- index. Initially FG-index is implemented and tested with AIDS dataset then the same dataset is tested with the proposed algorithm.

2. PRELIMINARIES

A graph G is called a sub Graph of another graph G' such that as G ⊆ G' or G' is called the superset of G.

Consider two graphs are isomorphic g_1 to g_2 is injective, f: V_1 -> V_2 ∃ ∀ $(u_1,v_1) \in E_1$, $(f(u_1),f(v_1)) \in E_1$, $l_1(u_1) = l_2(f(u_2))$, $l_1(v_1) = l_2(f(v_2))$, and $l_1(u_1, v_1) =$

$l_2(f(u_1), f(v_1))$ say $g_1 = (V_1, E_1, L_1, l_1)$, $g_2 = (V_2, E_2, L_2, l_2)$, V_1, E_1, L_1, l_1 be the set of vertex, edges, labels and label functions of the graph g_1 that maps each vertex or edge to label in L_1.

Consider a graph database G, and a graph g, freq (g) is the frequency of g in G. Hence, {g': g' €G, g ⊆ g'}. A graph g is said to be a frequent sub graph (FG) [11, 12] if freq (g) ≥σ. /G/ such that (0≤σ≤1) is a user specified minimum frequency threshold.

Let us consider F be the set of all FGs that are mined from graph database G. A graph g is called a Maximal Frequency sub graph (MFG) [13] if g € F and there exist not g' €F such that g' C g and freq (g) =freq (g').

Let G= {g1, g_2, g_3----- g_n} be a graph database. For a query q, the Graph Query answer set G_q = {g_i/q⊆ g_i, g_i € G}.

3. RELATED WORKS

There are plenty of indexing methods for graph databases. There are diverse types of data and each type has to be more selective in processing the particular kind of data. Algorithms like Day light [14], AnMol [15] are for structured data, where as Data Guides [16], T-index [17], Index Fabric[18], APEX [19] are for semi structured and XML data. These are the few varieties or types of algorithms concentrating on particular types of data.

There are hands full of indexing techniques for various graph models. GraphDB [20] and SUBDUE [21] concentrates on query processing from a large graph database while it automatically provides the relevant sub graphs. Of the various methods of indexing one of the dominant is GraphGrep[22] which works on the basis of paths of the graph. Consider if the paths in a graph are huge, the performance of the index will definitely be degraded. To overcome this, a graph based approach named gIndex [23] is reported. A(k) – index [24] uses the k-bisimilarity to make use of similar patterns in a semi strucured graph database. Every path in a tree is treated as string and it is stored in Patricia trie by Index Fabric[25].

Washio and Matoda [26] introduced graph based data mining to mine the graph databases. The frequent sub graphs have to be analyzed for which Inokuchi et al. [28] and Vanetik et.al [27] have used Apriori-based algorithm. For the generation of sub graphs other than algorithms Yan and Han [29] and Borgelt and Berthold [30] have applied pattern growth approach. In this paper, we test the existing FG index [31] with the AIDS dataset and to find the improvement in memory and time complexity with a newly proposed algorithm VFG index (valorous Frequent Graph Index).

4. ANALYSIS ON FG – INDEX GRAPH INDEXING:

Let G be the graph database, F be the set of all frequent FG from G. For a query q, such that q€G, the answer set Gq can be generated without candidate verification step but the reliability is low. So as to make the obtained candidate set a reliable without sub graph isomorphism, verification process is needed. Incase if the number of FG is high, it is hard to index complete set of FG graph.

In GIndex [23], a method named discriminative FG's (F_d) is proposed and used which would be smaller than F. (i.e) F_d C F. Suppose for a query, the candidate set $C_q = (\cap_{f\in F_d \wedge f\subseteq q} D_f)$. To obtain the value of G_q verification test is done on the candidate set which is quite costly. The response time for a query can be calculate by

$$T_r = T_s + /C_q/ \times T_{i/o} + /C_q/ \times T_{sg} \quad (1)$$

T_r - Response time for processing a query
C_q - candidate answer set
T_s - Time taken to compute C_q using index
$T_{i/o}$ - Time taken for fetching candidate graph from memory
T_{sg} - Time taken for testing Sub graph Isomorphism
T_{sg} takes the major part T_r as it is quite expensive.

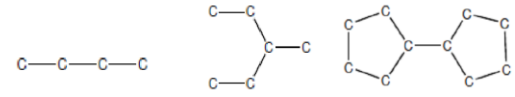
To overcome this cost, a new method without candidate verification was proposed by cheng et al. [31].This reduces the response time but if F is large, then the built index would comparatively large. To solve this, new notation of δ tolerance or δ- TCFGs is proposed. Let T be the set of δ- TCFGs. G can be divided into /T/ disjoint partitions based on T. So obviously, each graph g will contain set of FGs in any of the partition. The size is reduced than G. Hence nested index is built to sum up and provide the right answer set for the particular set of queries. In this paper, FG-index algorithm is tested and validated with the results obtained in the paper [31]. Then the memory complexity and the time complexity are improved to obtain a better algorithm than FG-index with AIDS dataset.

5. VFG – INDEX

In a graph database, to process a query two steps have to be performed.

1. Index Construction

As a preprocessing step, the algorithm scans throughout the graph database and extracts the selective features in the graph database say G. These extracted set of features are initially sorted and sequentially arranged with a weighted preferences for indexing. Say for graph feature f€F, Gf be the set of graphs containing the feature f, then Gf = {gi/f⊆ gi, gi€G). This is an example taken from AIDS antiviral screening database.



(i) (ii) (iii)

Figure 2: A Sample Database

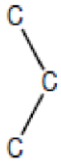


Figure 3: A Sample Query

Assume a query say fig 3, it checks out the bonding type c, c-c, c-c-c in the index. We get a subset of all three as sub graphs incase if taken with the bonding.

2. Query Processing

Once the query is obtained, it searches all the features relevant to the given query in the index to generate the candidate set which contains all the extracted features.

$$Cq = \cap fDf\ (f \subseteq q \text{ and } f \in F) \qquad (2)$$

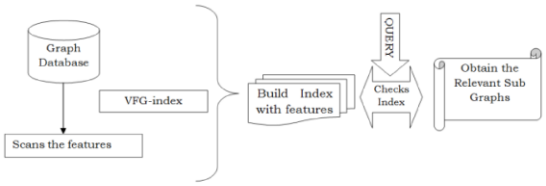


Figure: 4 Overview of index construction and Query processing

Figure 4 gives the complete overview of how the index is constructed by VFG index algorithm initially. Then once the query is obtained, it is checked with the index and then irredundant sub graph are retrieved with reference to the previously constructed index.

6. EXPERIMENTAL RESULTS:

6.1 AIDS ANTIVIRAL SCREEN DATASET:

To test the scalability, database index construction etc, we test with the dataset from National Cancer Institute AIDS antiviral, 3 classes (B) July 29, 2004 By Fei Yuan. The potency data is at http://dtp.nci.nih.gov/docs/aids/aids_data. html. NCI A and NCI B have slightly different sets of molecules and different descriptor sets. Otherwise, the potencies are the same.

6.2 PERFORMANCE OF AIDS DATASET:

6.2.1 INDEX CONSTRUCTION:

After importing the dataset, the algorithm scans throughout to get an initial index. Here the time taken and the memory used by both the algorithms during index construction are compared by using the AIDS dataset are given below:

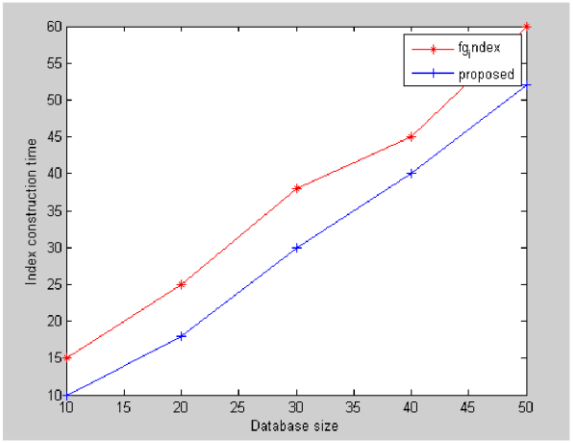


Figure: 5 Index construction time

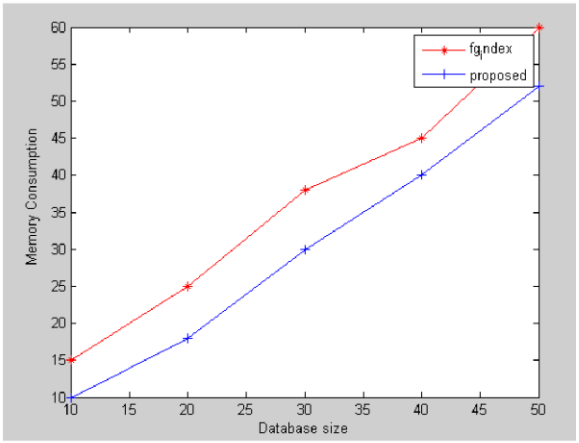


Figure:6 Memory Consumption

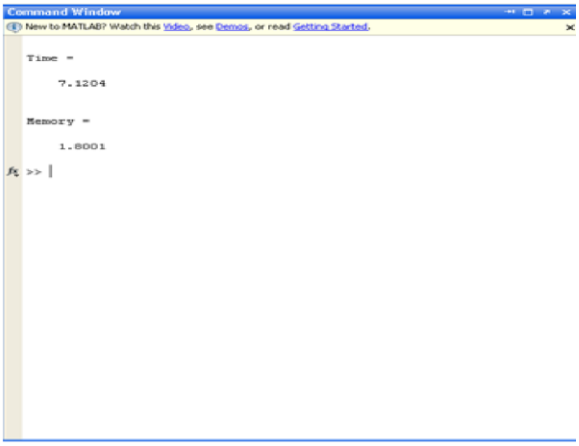


Figure: 7 Screen of memory and time consumption for VFG Index

Table: 1 Comparing FG index and VFG Index during index construction

	FG-index (Sigma 0.1)	Proposed
Memory Consumption	2.5	1.8
Time(sec)	9.75	7.12

In table: 1 the memory and the time taken by FG-index is 2.5 and 9.75 for building the index. Instead VFG has just taken 1.8 and 7.12 seconds to build the

very same index. This shows the improvement in the index construction of the graph dataset.

6.2.2 QUERY PROCESSING:
For a particular query, the amount of time in which it is taken to search from the index and provide the sub graphs for both the AIDS are listed below :

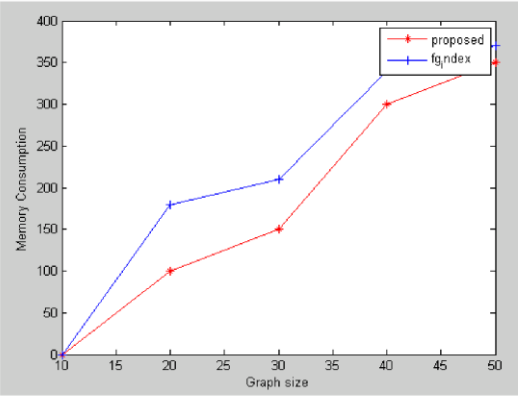


Figure: 8 Memory Consumption

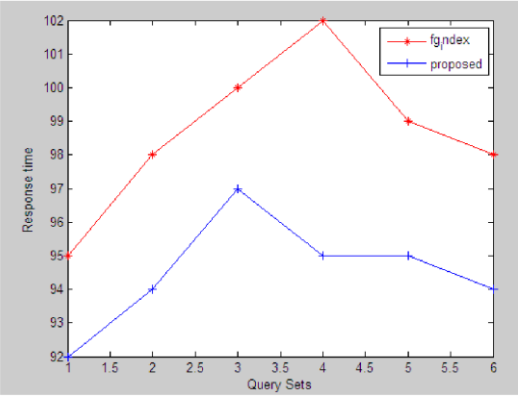


Figure: 9 Response time

Figure: 10 Screen of memory and time consumption for VFG Index

Table: 2 Comparing FG index and VFG index query processing

	FG INDEX	PROPOSED (VFG INDEX)
Memory	2	1.42
Response time	0.031	0.021

In table 2, to process a query the response time of FG-index is 0.031and memory used is 2. [31]. But for the VFG-index the response time is 0.021 and the memory usage is 1.42 which is comparatively lower than FG- index. Thus, the time and the space complexity are improved.

6.3 REQUIRED ENVIRONMENTAL
Front end designed with MATLAB (r2012a) with i3 processor and 80 GB memory. RAM size is 4 GB with Intel mother board.

7. CONCLUSIONS

Graph indexing plays a vital role in the efficiency of Graph databases. Data of various kinds can be made as graph, to retrieve things from database efficient indexing causes improvement by memory and time usage. In this paper, we have explored the various types of indexing techniques. Of which we have specifically focused on FG – Index which has been tested with AIDS dataset [31].

The obtained result by James et al.[31] are improved to get a new algorithm named VFG- Index(valorous Frequent Graph Algorithm). From table: 1 and table 2, VFG –Index algorithm performs better both in space and time complexity than the FG – index algorithm. In future work, the efficiency can still be improved by using pruning techniques in preprocessing steps. Also, we are planning to test the VFG- index algorithm for a bigger dataset.

8. ACKNOWLEDGEMENT

We would like to thank E.D. Boopalan, Doctoral Scholar, Department of Statistics, Bharathiar University, Coimbatore-46, Tamil Nadu, India for his thought provoking tips while modeling and my friend Ramkumar, Doctoral Scholar, Department of Microbial Bio-Technology, Bharathiar University, Coimbatore-46, Tamil Nadu, India for his constant encouragement. We would also like to thank our friend M. Pradeep who helped in programming throughout. Also, would like to convey my regards to V. Kannan for synchronizing during this work.

REFERENCES:

[1] N. Trinajastic , Chemical Graph Theory, 2nd ed. CRC Press, 1992, vol.1,2.

[2] B. Schalkopf, K. Tsuda, and J.P. Vert, Eds., Kernel Methods in Computational Biology. MIT Press, 2004.

[3] M. El-Mehalawi and R. A. Miller, “A database system of mechanical components based on genometric and topological similarity.” J.CAD, vol. 35, no. l, pp.83-94,2003.

[4] D. Conte, P. Foggia, C. Sansone, and M. Vento, “Graph matching applications in pattern recognition and image processing,” in ICIP, 2003.

[5] S. A. Cook, "The complexity of theorem-proving procedures," in STOC, 1971,pp. 151–158.

[6] B. Sun, P. Mitra, and C. L. Giles, "Irredundant informative subgraph mining for graph search on the web," in CIKM, 2009.

[7] J. Cheng, Y. Ke, W. Ng, and A. Lu, "Fg-index: Towards verification-free query processing on graph databases," in SIGMOD, 2007.

[8] S. Zhang, M. Hu, and J. Yang, "Treepi: A novel graph indexing method," ICDE, pp. 966–975,2007.

[9] P. Zhao, J. X. Yu, and P. S. Yu, "Graph indexing: tree + delta <= graph," in VLDB, 2007, pp. 938–949.

[10] X. Yan, P. S. Yu, and J. Han, "Graph indexing: a frequent structure-based approach," in SIGMOD, 2004.

[11] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In Proc. of PKDD, pages 13–23, 2000.

[12] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In Proc. of ICDM, pages 313–320, 2001.

[13] J. Huan, W. Wang, J. Prins, and J. Yang. Spin:mining maximal frequent subgraphs from graph databases. In Proc. of KDD, pages 581–586, 2004.

[14] Daylight theory manual - daylight version 4.9. Daylight Chemica Information Systems, Inc. www.daylight.com, 2006.

[15] S. Srinivasa and S. Kumar. A platform based on the multi-dimensionaldata model for analysis of bio-molecular structures. In Proc. of VLDB, pages 975–986, 2003.

[16] R. Goldman and J. Widom. DataGuides: Enabling query formulation andptimization in semistructured databases. In Proc. of VLDB, pages 436– 445, 1997.

[17] T. Milo and D. Suciu. Index structures for path expressions. In Proc. Of ICDT, pages 277–295, 1999.

[18] B. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon. A fast index for semistructured data. In Proc. of VLDB, pages 341–350, 2001.

[19] C.-W. Chung, J.-K. Min, and K. Shim. Apex: an adaptive path index for xml data. In Proc. Of SIGMOD, pages 121–132, 2002.

[20] R. H. G¨uting. GraphDB: Modeling and querying graphs in databases. In Proc. of VLDB, pages 297–308, 1994.

[21] L. Holder, D. Cook, and S. Djoko. Substructure discovery in the subdue system. In Proc. of KDD, pages 169–180, 1994.

[22] D. Shasha, J. T. L. Wang, and R. Giugno.Algorithmics and applications of tree and graph searching. In Proc. of PODS, pages 39–52, 2002.

[23] X. Yan, P. S. Yu, and J. Han. Graph indexing basedon discriminative frequent structure analysis. ACM Trans. Database Syst., 30(4):960–993, 2005.

[24] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for e±cient indexing of paths in graph structured data. In Proc. 2000 Int. Conf. Data Engineering (ICDE'00), San Jose, CA, Feb. 2002.

[25] B. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon. A fast index for semistructured data. In Proc. 2001 Int. Conf. Very Large Data Bases (VLDB'01), pages 341{350, 2001.

[26] T. Washio and H. Motoda. State of the art of graph-based data mining. SIGKDD Explorations, 5:59{68, 2003.

[27] N. Vanetik, E. Gudes, and S. E. Shimony. Computingfrequent graph patterns from semistructured data. In Proc. 2002 Int. Conf. on Data Mining (ICDM'02), pages 458{465, Maebashi, Japan, Dec. 2002.

[28] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In Proc. 2000 European Symp. Principle of Data Mining andKnowledge Discovery (PKDD'00), pages 13{23, Lyon, France, Sept. 1998.

[29] X. Yan and J. Han. CloseGraph: Mining closed frequent graph patterns. I Proc. 2003 Int. Conf. Knowledge Discovery and Data Mining (KDD'03), pages 286{295, Washington, D.C., Aug. 2003.

[30] C. Borgelt and M. R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In Proc. 2002 Int. Conf. on Data Mining (ICDM'02), pages 211{218, Maebashi, Japan, Dec. 2002.

[31] J. Cheng, Y. Ke, W. Ng, and A. Lu. "Fg-index: towards verification-free query processing on graph databases," In Proc. Of the ACM SIGMOD international conference on Management of data, pp. 857-872,2007.

◈◈◈