Subgraph Matching Algorithm for Graph Database

Maninder Rajput
PG student,
Dept. of Computer engg.,
K.K.W.I.E.E.R.,
Savitribai Phule Pune University,
Nashik(MH), India

Snehal Kamalapur, PhD
Associate Professor,
Dept. of Computer engg.,
K.K.W.I.E.E.R.,
Savitribai Phule Pune University,
Nashik(MH), India

ABSTRACT
A graph is a symbolic representation of data and its relationships. It is used in many domains like bioinformatics, semantic web and chemical structures applications. Subgraph matching is a technique to retrieve set of subgraphs from dataset which are similar to query/input graph. Subgraph matching is a NP-hard. Graph $S(V_S, E_S)$ is subgraph of graph $G(V_G, E_G)$ if $V_S \subseteq V_G$ and $E_S \subseteq E_G$. Work here aims to fetch all subgraphs S(VS, ES) from graph $G(V_G, E_G)$ which are similar to query graph $Q(V_Q, E_Q)$ using subgraph matching algorithm. Work carried out in two phases, offline phase and online phase. Offline phase grnertaes index over data graph G. Online phase retrieves set of subgraphs from data graph G which are similar to query graph Q. A cost function is introduced for checking similarity of query node with data graph node which efficiently reduces intermediate results by converting vertices into vector points and extracts similar subgraphs by calculating nearest distance of these vector points.

General Terms
Data mining, Graph mining.

Keywords
Graph database, Offline phase, Online phase, Subgraph matching..

1. INTRODUCTION
The graph is an attractive tool to represent and model a data since it allows simple and flexible representation of complex objects. Day by day increasing data in graphs requires new techniques to extract results for graph queries in shorter time. Real world graphs are very large in size that is having millions number of nodes and edges. Web graphs, Bioinformatics, protein interaction, social networks are some examples. Subgraph matching is a technique to retrieve subgraphs which are similar to query/input graph. Subgraph matching is also called subgraph isomorphism.

1.1 Subgraph
A graph[1] H = (v,e) here v is a set of vertices and e is a set of edges, then H is said to be a subgraph of graph G = (V,E) if v ⊆ V and e ⊆ E, and each edge in graph H should have same ending vertices in graph G also.

1.2 Isomorphism
Two graphs[1] G and H are isomorphic (G ≈ H) if and only if there exist a bijective function f, for vertex set of G and H such that,

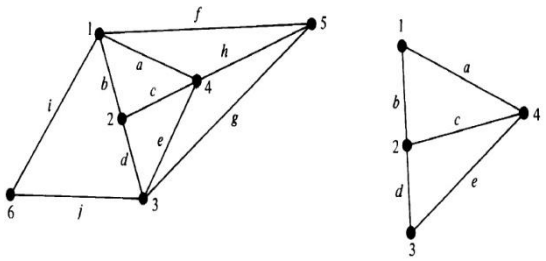

Fig.1 A graph and its subgraph.

$$f : V(G) \rightarrow V(H).$$

This is a mapping function which will map vertices of query/input graph to the vertices of dataset. If u and v are two vertices of G and H respectively, then G and H are isomorphic iff (u, v) ϵ E(H). Any two vertices u and v of G are adjacent in G if and only if they are adjacent in H.

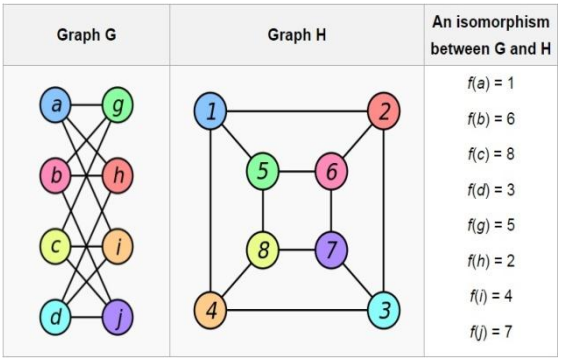

Fig.2 Graph Isomorphism

In fig.2 vertex (a) of graph G has matching vertex (1) in graph H, represented by f(a) = 1 and vertex (b) has matching vertex (6) in graph H, represented by f(b) = 6, similarly rest of the matching's are shown in figure 2.

For a query/input graph Q and a data graph G, subgraph matching algorithm will extract all those subgraphs from G, which are isomorphic to Q. Subgraph matching approaches are generally classified into two categories: Exact and approximate subgraph matching approaches. Subgraph matching approaches aim to find out an exact mapping or matching between the vertices and the edges of the query graphs and data graphs. Approximate subgraph matching approaches aim to compute a distance between vertices of graphs by converting vertices into points in vector space using embedding techniques. Approximate subgraph matching approaches converts pattern match queries into distance based

queries. Approximate matching is useful for rank based applications where the distance between the objects to be compared is needed. Several subgraph matching approaches have been proposed in the literature. The aim of this paper is to provide a survey of recent and current subgraph matching approaches on large graphs.

Work describe in detail the existing approaches and can categorize them into two classes i.e., exact and approximate subgraph matching approaches. The advantages, disadvantages and the differences between these approaches are also highlighted here.

2. LITERATURE SURVEY

Many subgraph matching algorithms have been introduced in recent years. These subgraph matching algorithms can be classified into two classes i.e. exact subgraph matching and approximate subgraph matching.

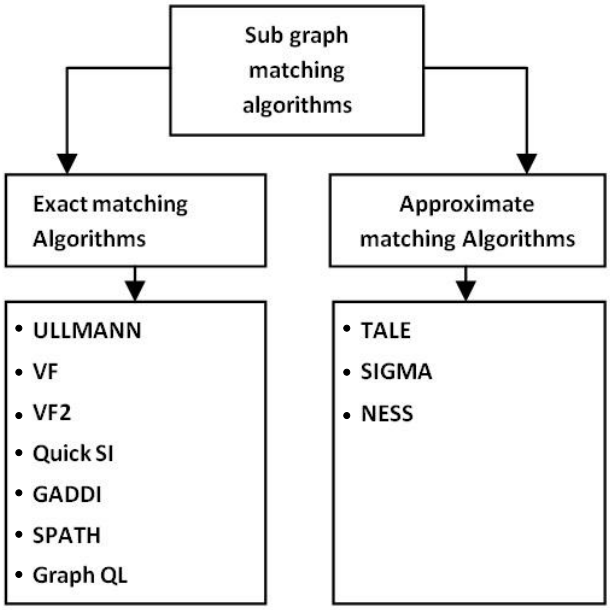

Fig. 3 Subgraph matching algorithms classification

Ullmann algorithm[2] is a backtracking algorithm. It detects subgraph isomorphism using brute force tree search and enumeration procedure. Enumeration algorithm was designed to generate an adjacency matrix of query graph H and large graph G and then using this adjacency matrix, isomorphism was detected. Its refinement phase was used to prune undesired matches from adjacency matrix of possible future matched node pairs. The memory requirement for Ullmann algorithm is O (N3) for N number of nodes, which is very high. This memory requirement has been reduced to O (N) in VF2 algorithm.

L. P. Cordella et al. [3] proposed a matching algorithm for both graph isomorphism and subgraph isomorphism. It was a deterministic matching algorithm. It started with first vertex, selects sequential vertices, search for a subgraph match, and backtracks if not. It has used state space representation for the matching process. A set of feasibility rules were introduced to insure consistency of partial solutions and for pruning a search tree. Here are some limitations for the above proposed algorithm, search with this method is not based on an index, so it is costly as compared to the new methods. It was designed for graphs with thousands of nodes only. Haichuan Shang Ying Zhang Xuemin Lin et al. [4] Introduced an efficient algorithm for subgraph isomorphism. It computed the triple frequencies of labels of vertices of data graph in advance that is before the search procedure started. B+ tree was used for storing frequencies of all vertex labels. Weight of each query vertex was computed and assigned using pre-computed edge label frequencies and a minimum spanning tree was formed using modified Prim's algorithm.

Yuanyuan Tian et al. [5] Proposed a tool [TALE] for approximate matching in large graphs. It is based on the assumption that approximate matching could generate more and nearby results to the input query as compared to exact matching. It distinguishes nodes on the basis of their relative importance (importance here was decided on the basis of degree centrality of node in the database) in the structure of the graph. Firstly, it matches only important nodes of input graph and then these nodes lead the remaining matching procedure by considering adjacent nodes of previously matched nodes in a second step. It has better effectiveness as compared to Gramelin . Its index size scales linearly to dataset size and index construction time grows steadily.

Shijie Zhang et al. [6] Put forward a technique which relied on Neighboring Discriminating Structure distance. It first selects a query vertex appeared first in input graph and then performs DFS (Depth First Search) to find next query vertex for comparison. To refine a candidate set for query vertices, vertices of data graph were pruned on the basis of NDS. Binary search was used to locate distances between any given pair of vertices. But it works efficiently only for biological networks. Lei Zou et al. [7] proposed a pattern match query in the large graph database based on distance and joins. Firstly vertices were transformed into points in vector space using LLR embedding because it is cheap to calculate distance between two vertices then to find nearest vertex of any existing vertex. A cost model was also proposed to guide a join order selection that is this model generated a cheap input query from the query entered by a user. It has used costly join operations.

Peixiang Zhao et al. [8] introduced a new graph indexing technique. Graph matching was performed in a manner, searching for a query path rather than searching for a query vertex. It was a first method to search in this fashion. Its main aim was to reduce N, where N is the number of vertices of query graph. An algorithm GraphQL [9] was introduced earlier to SPath algorithm and SPath has better performance as compared to GraphQL. Both perform neighborhood's signature based pruning before starting actual subgraph matching procedure. There is a difference in indexing technique of these two algorithms i.e. GraphQL indexes nodes of data graph while SPath indexes nodes of datagraph using their neighbour information. SPath has better performance but its average cost for recursive calls is more than GraphQL.

Liang Hong et al. [10] proposed a subgraph matching algorithm in large graph database. It efficiently extracts subgraphs from the large data graph which are isomorphic to query graph. It works in two steps; firstly it builds a lattice based index over data graph, and a data signatures and signature buckets for neighbourhood information about vertices. In the second step, for a query graph Q entered by the user a cost efficient dominating set was generated for it.

3. MOTIVATION

Due to emergence of too much graph data in areas like social networks, information network, technological networks and so on, it is necessary to speed up the search procedure. Graph database is being widely used as an important tool to model and ask questions and to generate answers for it from graph

data. Subgraph matching is a technique to find out the set of subgraphs from a large graph database which are isomorphic to query graph. Some methods use index to fasten the search procedure whiles some not.

It is very important to reduce the search space, as a search space directly affects performance of any subgraph matching method. It's been observed during a literature survey that pruning plays important role in reducing search space, as it reduces the number of comparisons. Pruning is a technique to eliminate or evict unwanted vertices of large graph from being getting compared to that of query graph. Along with pruning, if one can reduce the number of vertices from a query graph, then it will greatly improve performance of existing subgraph matching methods.

3.1 Existing system

Liang Hong et al. [10] proposed a subgraph matching algorithm in large graph database. It efficiently extracts subgraphs from large data graph which are isomorphic to query graph.

It works in two steps; firstly it builds a lattice based index over data graph, and a data signatures and signature buckets for neighborhood information about vertices. In the second step, for a query graph Q entered by the user a cost efficient dominating set was generated for it. Two types of pruning strategies have been introduced, set similarity and structure based pruning. These pruning techniques greatly reduce the size of intermediate results.

Two algorithms have been proposed to find final solution, input for the first algorithm is the dominating query graph and mapping between dominating query graph and subgraph of the data graph will be found out here, and for second algorithm input is query graph, dominating query graph and this will find mapping between query graph and subgraph of the data graph.

3.2 Goal

The goal of this paper is to design an efficient algorithm for subgraph matching which will improve performance of existing system.

3.3 Problem Definition

To design and implement a subgraph matching algorithm along a graph similarity cost function and compare performance with existing system.

4. PROPOSED ARCHITECTURE

4.1 Proposed Methodology

Iterative subgraph matching algorithm is used to find out similar subgraphs from large data graph. Whole process of proposed method will be carried out in two phases' offline and online phase.

4.2 Offline Processing

In offline phase an inverted index will be build over data graph and another index that is neighbourhood index will be constructed to store neighbourhood information of each vertex of data graph. Neighbourhood information of each vertex is stored in form of vector points. Vertices first converted to multidimensional vector using standard formula

$$A(v,l) = \sum_{i=1}^{h} \alpha^i \sum_{d(u,v)=i} I(l \in L(u))$$

Where for a node v ∈ Data graph and u ∈ Query graph A(v,l) represents strength of element weight at vertex v in data graph, α is constant whose value lies between (0,1),

I(l ∈ L(u)) is indicator function such that:

$$I\left(l \in L(u)\right) = \begin{cases} 1 & if\ l\ is\ in\ element\ set\ of\ u \\ 0 & otherwise \end{cases}$$

d(u,v) is distance between u and v.

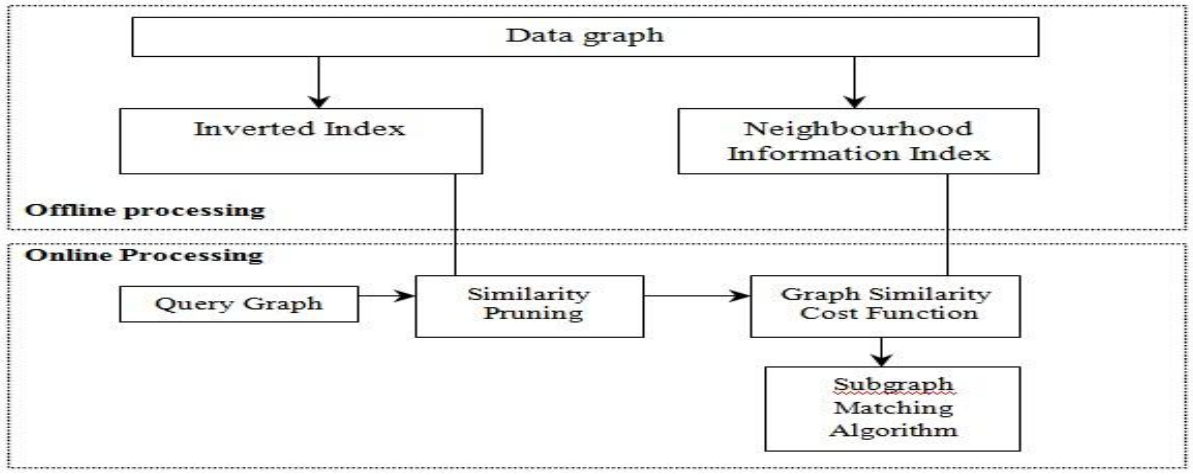

Fig.4 Block Diagram of Proposed system

4.3 Online Phase

In online phase, user enters a query graph and using inverted index similarity pruning will be performed over data graph, here jaccard similarity is used. Further pruning will be done by graph similarity cost function using neighbourhood information index.

For the following data and query graphs:

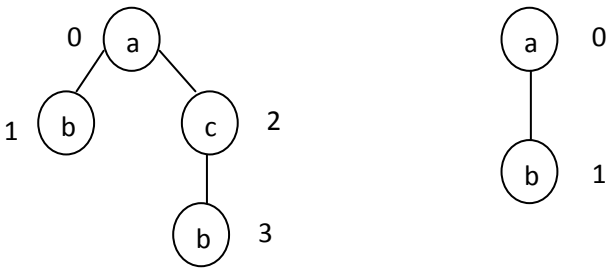

Fig. 5 Data graph **Query Graph**

Distance between nodes is calculated for both query and data graph. One node distance is notated as 0.5 and two node distance is notated as 0.25.

For query graph Distance (Dq) from node a to b is 0.5, and Distance (Dq) from node b to a is 0.5. denoted as follow:

$Dq(a,b) = 0.5$

$Dq(b,a) = 0.5$

For data graph Distance (Dm1) that is for mapping m1, is denoted as follow:

$Dm1(a,b) = 0.5$

$Dm1(b,a) = 0.5$

$Dm2(a,b) = 0.25$

$Dm2(b,a) = 0.25$

Cost of matching (CF) is calculated between query graph and two mappings (m1, m2) of data graph as follow:

$$CF_{m1} = [D_q(a\text{-}b) - D_{m1}(a\text{-}b)] + [D_q(b\text{-}a) - D_{m1}(b\text{-}a)]$$
$$= [\ 0.5 - 0.5\ ] + [\ 0.5 - 0.5\ ]$$
$$= [\ 0\ ] + [\ 0\ ]$$
$$= 0$$

$$CF_{m2} = [D_q(a\text{-}b) - D_{m2}(a\text{-}b)] + [D_q(b\text{-}a) - D_{m2}(b\text{-}a)]$$
$$= [\ 0.5 - 0.25\ ] + [\ 0.5 - 0.25\ ]$$
$$= [\ 0.25\ ] + [\ 0.25\ ]$$
$$= 0.5$$

So, cost of matching is least for mapping one and hence mapping m1 of final solution for query graph.

4.4 Subgraph Matching Algorithm

Finally a set of subgraphs similar to query graph will be generated using subgraph matching algorithm. Subgraph Matching Algorithm is:

1. Select a node from query graph and match it with some node of data graph which satisfies cost function.
2. Discard element set of unmatched nodes.
3. Recalculate neighborhood vectors for nodes that have match with query node. Repeat step 1 until it converges.

5. DATASETS

We used two real datasets Freebase and Dbpedia. The datasets are described below:

1. Freebase is a collection of large knowledge base of structured data. Graphs are in form of entity (vertices) relationship (edges). Weight of features represents its significance that is normalized to range [0, 1]. Freebase dataset is used for efficiency, durability and effectiveness examination.
2. DBpedia collaboration graph dataset is collection of names of authors their publications, co authors and citations.

Niels Henrik David Bohr (Danish: ; 7 October 1885 – 18 November 1962) was a Danish physicist who made foundational contributions to understanding atomic structure and quantum theory, for which he received the Nobel Prize in Physics in 1922. Bohr was also a philosopher and a promoter of scientific research.Bohr developed the Bohr model of the atom, in which he proposed that energy levels of electrons are discrete and that the electrons revolve in stable orbits around the atomic nucleus but can jump from one energy level (or orbit) to another. Although the Bohr model has been supplanted by other models, its underlying principles remain valid. He conceived the principle of

Fig. 6 Dataset Snapshot

6. RESULT AND DISCUSSION

In this section experimental results are presented to demonstrate the efficiency and effectiveness of proposed subgraph matching algorithm. Performance measures for subgraph matching are: Index construction time

- Space cost
- Pruning time
- Number of candidates generated
- Query response time

Complexity in terms of time and space is O(n), where n is number of vertices in data graph.

Table 1: Results

Performance mearsures	**Time (sec.)**
Inverted Index Construction Time	1213
Neighborhood Index Construction Time	2772
Elements Weight Calculation Time	10
Pruning time	0.49
No. of candidates generated after similarity pruning	6377(12754)

7. SUMMARY AND CONCLUSION

The subgraph matching algorithm extracts all similar subgraphs to query graph from the data graph. Various approaches have been discussed in literature survey which aims to minimize query processing time and intermediate results space. Work here aims to reduce intermediate results space by pruning unwanted vertices of the data graph. A graph similarity cost function is introduced which prune unwanted vertices using neighborhood information vertex very efficiently. This will improve performance of existing system.

8. ACKNOWLEDGMENTS

With immense pleasure, I am presenting this Project Report on "Subgraph Matching Algorithm for Graph Database" as a part of the curriculum of M.E. Computer Engineering at K.K.W.I.E.E.R., Nashik It gives me proud privilege to complete this Project Dissertation Stage-I Work under the valuable guidance of Prof. Dr. S. M. Kamalapur. I am also extremely grateful to Prof. Dr. S. S. Sane (HOD of Computer Department) and Prof. Dr. K. N. Nadurkar Principal for providing all facilities and help for smooth progress of Project Work. I would also like to thank my friends and my family members who have directly or indirectly guided and helped me for completion of this work.

9. REFERENCES

[1] Nar Singh Deo, "Graph theory with applications to engineering and computer science".

[2] J. R. Ullmann, "An algorithm for subgraph isomorphism," J. ACM, vol. 23, no. 1, pp. 31–42, 1976.

[3] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 10, pp. 1367–1372, Oct. 2004.

[4] H. Shang, Y. Zhang, X. Lin, and J. X. Yu, "Taming verification hardness: An efficient algorithm for testing subgraph isomorphism," Proc. VLDB Endowment, vol. 1, no. 1, pp. 364–375, 2008.

[5] Y. Tian and J. M. Patel, "Tale: A tool for approximate large graph matching," in Proc. 24th Int. Conf. Data Eng., 2008, pp. 963–972.

[6] S. Zhang, S. Li, and J. Yang, "Gaddi: Distance index based subgraph matching in biological networks," in Proc. 12th Int. Conf. Extending Database Technol.: Adv. Database Technol., 2009, pp. 192–203.

[7] L. Zou, L. Chen, and M. T. Ozsu, "Distance-join: Pattern match query in a large graph database," Proc. VLDB Endowment, vol. 2, no. 1, pp. 886–897, 2009.

[8] P. Zhao and J. Han, "On graph query optimization in large networks," Proc. VLDB Endowment, vol. 3, nos. 1/2, pp. 340–351, 2010.

[9] H. He and A. K. Singh, "Graphs-at-a-time: Query language and access methods for graph databases," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2008, pp. 405–418.

[10] Liang Hong, Lei Zou, Xiang Lian and Philip S. Yu, "Subgraph Matching with Set Similarity in a Large Graph Database," in Proc. IEEE VOL. 27, NO. 9, pp, 2507–2521,2015.