ISSN: 0975-833X

Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 8, Issue, 10, pp.39771-39774, October, 2016

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

RESEARCH ARTICLE

RETRIEVING SIMILARITY SET USING SUB GRAPH APPROACH

***,[1]Ravi Chinapaga, [1]Harshitha, G., [2]Bal Raju, M. and [3]Subhash Chandra, N.**

[1]Computer Science and Engineering, TKR Collegef Engineering and Technology, Hyderabad, India
[2]Computer Science and Engineering, Krishna Murthy Institute Technology & Engineering, Hyderabad, India
[3]Computer Science and Engineering, Holy Mary Institute Technology & Science, Hyderabad, India

ARTICLE INFO

Article History:
Received 20th July, 2016
Received in revised form
22nd August, 2016
Accepted 28th September, 2016
Published online 30th October, 2016

Key words:

Subgraph, Retrieving similarity set,
Dynamic weight, DS algorithm,
Structured based pruning.

ABSTRACT

In graphs like social networks, Semantic Web and biological networks, every vertex has high information, which can be design by a set of tokens or elements. In this paper, we are studying about. Retrieving similarity set using sub graph approach, which gives subgraph that is structurally isomorphic to the query. Here we apply the apriori algorithm to find sub transaction set for actual transaction set in with each item get separated depending upon the category they are, which accomplish the frequent item set with its dynamic weight. In this we design a lightweight signature for both query vertices and data vertices. Structure-based pruning, which accomplishment the individual features of both (dynamic) weighted set similarity. We design an efficient algorithm to perform sub graph matching based on the dominating set of query graph.

Citation: Ravi Chinapaga, Harshitha, G., Bal Raju, M. and Subhash Chandra, N. 2016. "Retrieving similarity set using sub graph approach", *International Journal of Current Research*, 8, (10), 39771-39774.

INTRODUCTION

In applications like social networks Semantic Web, and biological networks, graph databases have been generally used as important tools to design and query complex graph data. Much aforementioned work has broadly studied different types of queries over graphs, in this subgraph retrieving is a basic graph query type. Given a query graph Q and a large graph G, a archetypal subgraph retrieved query gives the subgraph in G that matches with Q in circumstances of both structure of graph and vertex label. In any way some graph applications, each vertex has high set of tokens or elements denoting features of vertex, and the explicit matching of vertex label is sometimes not appropriate. By the above observation, in this paper we put focus on various of the subgraph retrieving query, called Retrieving similarity set using sub graph approach , in this every vertex contains set of elements with dynamic weights in behalf of single label. The weights of an element are mentioned by admin in unique query's depending upon the item requirement. In detail given a query graph Q with n vertices ui (i = 1,...,n), the retrieving similarity set query fetch all the subgraphs X with n vertices vj (j = 1,...,n) in a large graph G, such that (1) the weighted set similarity between S(ui) and S(vj) is larger than a user speci ed similarity threshold, where S(ui) and S(vj) are sets associated with ui and vj, respectively; (2) X is structurally isomorphic to Q with ui mapping to vj. Before exhibiting our method, we discuss an example to determine the usefulness of retrieving similarity set queries.

****Corresponding author: Ravi Chinapaga,***
Computer Science and Engineering, TKR Collegef Engineering and Technology, Hyderabad, India.

Example

The DBLP computer science bibliography provides a reference graph *G* (Fig. 1(b)) in which vertices represent papers and edges represent reference relations between papers. Each paper contains a set of keywords, in which each keyword is assigned a weight to measure the importance of a keyword with regard to a paper. In reality, a user wants to search for similar papers from DBLP based on both reference relationships and paper content similarity.

For example, a user wants to find papers on subgraph matching that are cited by both social network papers andpapers on protein interaction network search in DBLP. Furthermore, she/he requires papers on protein interaction network search being cited by social network papers. Such query can be modeled as an retrieving similarity set query, which obtains subgraph matches of the query graph *Q* (Fig. 1(a)) in *G*. Each paper (vertex) in *Q* and its matching paper in *G* should have

similar set of keywords, and each reference relation (edge) exactly follows the user requirements. Thestimulus examples show that retrieving similarity set queries are very useful in many real-world applications. Best of our knowledge, no preceding work studied the subgraph matching problem in structural isomorphism and set similarity with dynamic element weights (called *dynamic weighted set similarity*). Normally weighted set similarity that focuses on fixed element weight not exactly on the dynamic weighted set. Due to this previous techniques on exact or approximate applied to answering retrieving similarity set queries. It is difficult to utilize both dynamic weighted set similarity and structural constraints to efficiently answer retrieving similarity set queries. There are two algorithms that answer retrieving similarity set queries by modifying existing algorithms. Our approach adopts a "filter-and-refine" framework, which exploits unique ifeatures of both graph topology and dynamic weighted set similarity. In the filtering phase, by apriorialgorithmwe find frequent item set of *element set*s of vertices in data graph *G*. Then, data vertices are encoded into signatures, and organized into *signature bucket*s. In the refinement phase, we propose a *dominating set* 2(DS)-based subgraph matching algorithm to find retrieving similarity set

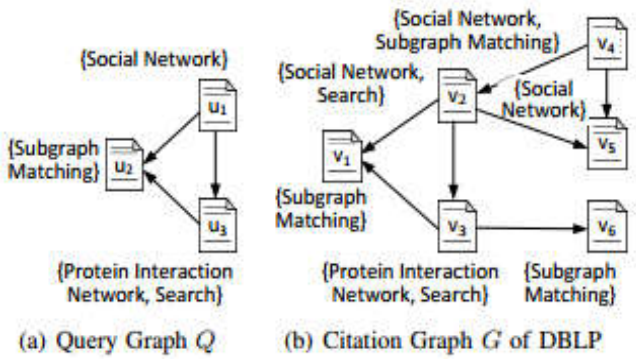


Fig. 1. An Example of Finding Groups of Cited Papers in DBLP that Match with the Query reference Graph

Related work

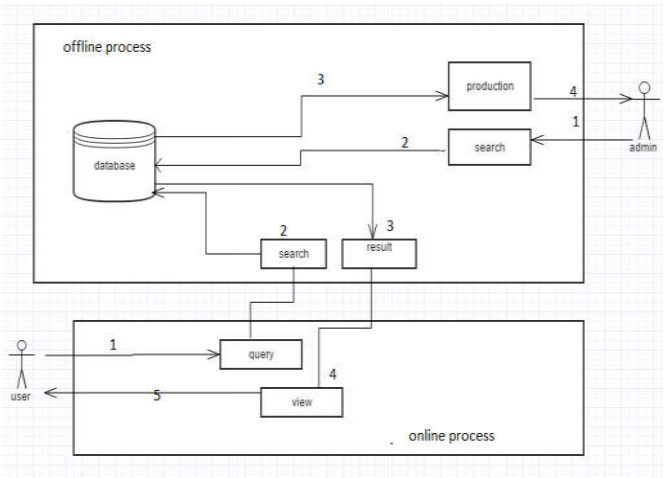
Exact subgraph matching query requires that all the vertices and edges are matched exactly. The Ullmann's subgraphisomorphism method (Ullmann, 1976) and VF2 (Cordella *et al*., 2004) algorithm do not utilize any index structure, thus they are usually costly for large graphs. Tree pi indexes graph databases using frequent sub trees as indexing structures. gaddi is a structure distance based subgraph matching algorithm in a large graph. (Shang *et al*., 2008) proposedQuickSI algorithm for subgraph isomorphism optimized by choosing an search order based on some features of graphs. SING (Di Natale *et al*., 2010) is a novel indexing system for subgraph isomorphism in a large scale graph. GraphQL (He and Singh, 2008) is a query language for graph databases which supports graphs as the basic unit of information. Sun *et al*. (2012) utilized graph exploration and parallel computing to process subgraph matching query on a billion node graph. Recently, an efficient and robust subgraph isomorphism algorithm Turbo*ISO* (Han *et al*., 2013) was proposed. RINQ (Gulsoy and Kahveci, 2011) and GRAAL (Memisevi˘ c and Pr´ zulj, 2012) are graph alignment algorithms for biological networks, which can be used to solve isomorphism problems. However, a query graph is much smaller than the data graph in subgraph isomorphism problems, while the two graphs usually have similar size in graph alignment problems. To solve subgraph isomorphism problems, graph alignment algorithms introduce additional cost as they should first find candidate subgraphs of similar size from the large data graph. In addition, existing exact subgraph matching and graph alignment algorithms do not consider weighted set similarity on vertices, which will cause high postprocessing cost of set similarity computation. Recently, several novel subgraph similarity search problems have been investigated. studied the problem of *graph simulation* by enforcing duality and locality conditions on subgraph matches. NeMa (Khan *et al*., 2013) focuses on the subgraph matching queries that satisfy the following two conditions (1) many-to-one subgraphmatching with a cost function, and (2) label similarity of matching vertices. *S* 4 system finds the subgraphs with identical same structure and semantically similar entities of query subgraph. SMS 2 query differs from the above problems in that it considers both one-to-one structural isomorphism and dynamic set similarity of matching vertices. Zou *et al*. proposed a top-k subgraph matching problem that considers the similarity between objects associated with two matching vertices. This work assumes that all vertex similarities are given, and does not exploit set similarity pruning techniques to optimize subgraph matching performance. As for the weighted set similarity query, Hadjieleftheriou *et al*. proposed various index structures and algorithms. Recently, a Heaviest First strategy has been proposed for efficiently answering all-match weighted string similarity query. However, dynamic element weights (i.e. query dependent weights) in retrieving similarity set queries make most of existing index structures and query processing techniques for weighted set similarity inefficient, or even infeasible. The reason is that these methods rely on element canonicalization according to fixed weights, while elements with dynamic weights cannot be canonicalized in advance.

Architecture diagram

User: The user raise query the result is search from data base and displayed to user to view the result all this process will comes under online process.

Admin: The admin gives query in search box the production takes information from data base and processes it to give the result and the result will be displayed to the admin

Production: In production all operation regarding this project will be done i.e filtering and refinement phase work



4) System design

Structure-based pruning

A matching subgraph should not only have its vertices (element sets) similar to that in query graph *Q*, but also preserve the same structure as *Q*. Thus, in this section, we design lightweight signatures for both query vertices and data vertices to further filter the candidates after set similarity pruning by considering the structural information.

Structural Signatures

We define two distinct types of structural signature, namely *query signature Sig*(*u*) and *data signature Sig*(*v*) for each query vertex *u* and data vertex *v*, respectively. To encode structural information, *Sig*(*u*)*/Sig*(*v*) should contain the element information of both *u/v* and its surrounding vertices. Since the query graph is usually small, we generate accurate query signatures by encoding each neighbor vertex separately. On the contrary, the data graph is much larger than the query graph, so the aggregation of neighbour vertices can save a lot of space. The pruning cost can be also reduced due to limited number of data signatures. Specifically, we first sort elements in element sets *S*(*u*) and *S*(*v*) according to a predefined order (e.g., alphabetic order). Based on the sorted sets, we encode the element set *S*(*u*) by a bit vector, denoted by *BV*(*u*), for the former part of *Sig*(*u*). In particular, each position *BV*(*u*)(*i*) in the vector corresponds to one element *ai*, where $1 \leq i \leq |U|$and$|U|$ is the total number of elements in the universe *U*. If an element *aj*belongs to set *S*(*u*), then in bit vector *BV*(*u*), we have *BV*(*u*)(*j*) = 1; otherwise (if *aj*∈/ *S*(*u*)), *BV*(*u*)(*j*) = 0 holds. Similarly, *S*(*v*) is also encoded using the above technique. For the latter part of *Sig*(*u*) and *Sig*(*v*) (i.e., encoding surrounding vertices), we propose two different encoding techniques for *Sig*(*u*) and *Sig*(*v*), respectively. The difference is that, we encode every neighbor vertex separately in *Sig*(*u*), but aggregate all neighbor vertices in *Sig*(*v*).

Dominating-set-based subgraph matching

In this section, we propose an efficient dominating-set(DS)-based subgraph matching algorithm(denoted by DS-Match) facilitated by a dominating set selection method.

DS-Match Algorithm

DS-Match algorithm first finds matches of a dominating query graph QD formed by the vertices in dominating set DS(Q), then verifies whether each match of QD can be extended as a match of Q. Second, we can speed up subgraph matching by only finding matches of dominating query vertices. The candidates of remaining (non-dominating) query vertices can be filled up by the structural constraints between dominating vertices and non dominating vertices. In this way, the size of intermediateresults during subgraph matching can be greatly reduced. In the following, we formally define the dominating set.

V (QD) _ V (QD) !E(QD),and there is an edge (ui; uj) in E(QD) iff at least one ofthe following conditions holds:

1) ui is adjacent to uj in query graph Q
2) jN1(ui) \ N1(uj)j >0
3) jN1(ui) \ N2(uj)j >0
4) jN2(ui) \ N1(uj)j > 0

Distance Preservation Principle: Given a subgraph match XD of QD in data graph G, QD and XD have n vertices u1; :::; un and v1; :::::; vn, respectively, where vi 2 C(ui). Considering an edge (ui; uj) in QD, then all the following distance preservation principles hold:

1) if the edge weight is 1, then vi is adjacent to vj ;
2) if the edge weight is 2, then jN1(vi) \ N1(vj)j > 0;
3) if the edge weight is 3, then jN1(vi) \ N2(vj)j > 0 or jN2(vi) \ N1(vj)j > 0

DQG- Match

Input: a dominating query graph *QD*, an intermediate state *s*; the initial state *s*0 has *M*(*s* 0) = *φ*

Output: the mapping *M*(*QD*) between *QD* and *G*'s subgraph

1 **if** *M*(*s*) *covers all the vertices of QD* **then**
2 *M*(*QD*)=*M*(*s*);
3 Output *M*(*QD*);
4 **else**
5 Compute the set *PA*(*s*) of all the pairs (*u*, *v*), where *u* ∈ *V*(*QD*) and *v* ∈ *V*(*G*);
6 **for** *each pair (u, v) in PA*(*s*) **do**
7 **if** *(u, v) satisfies threshold value* **then**
8 Add (*u*, *v*) to *M*(*s*) and compute current state *s 0*;
9 Call DQG-Match(*s 0*);

the current state *s*is initially set to *φ*. We build a candidate pair set *PA*(*s*) containing all the possible candidate pairs (*u,v*) and add it to the current state *s*(Line 5). When a candidate pair (*u,v*) is added to the current mapping *M*(*s*), we verify if the partial match *M*(*s*)satisfies threshold value (Lines 6- 7). If yes, we continue to explore the state until a subgraphmatch of *QD* is found (Lines 8-9). Otherwise, the corresponding search branch is terminated. In the following, we propose DS-match algorithm (Algorithm 2). Algorithm 1 is first called to find the mappingfunction M(QD) (Line 2). The mapping function M(s) ofcurrent state s is initialized to M(QD) (Line 3). Then, DS-match extends each match of the dominating query graph QD to a subgraph match of the query graph Q. If M(s) covers all the vertices of Q, then we output themapping function M(Q) (i.e. a subgraph match of Q)(Lines 4-6). Otherwise, for each non-dominating vertex u0 ∈ V (Q) - V (QD), we considering one-hop and two hopneighboring dominating vertices of u0 (i.e. N1(u0) andN2(u0)) (Lines 7-9). Note that, for each dominating vertexui∈ N1(u0) and uj∈ N2(u0), candidate vertex sets C(ui)and C(uj) have been found by Algorithm 1. Based ondistance preservation principle, each candidate vertex v0of u0 must be one-hop neighbor and two-hop neighbor ofthe vertices in C(ui) and C(uj), respectively (Line 10).Then, we check whether the candidate pair (u0; v0) satisfies conditions of subgraph match with set similarity (Definition1) (Line 11). If yes, (u0; v0) is added to the current state (Line 12). We continue to explore the state until all nondominatingvertices are considered (Line 13).

Algorithm 2: DS-Match

Input: a query graph Q, a dominating query graph QD, and an intermediate state s; the initial

state s0 has M(s0) = φ

Output: the mapping M(Q) between query graph Q and G's subgraph

1 if M(s0) = φ then
2 Call Algorithm 1 to find M(QD);
3 Initialize M(s) with M(QD);
4 if M(s) covers all the vertices of Q then
5 M(Q) = M(s);
6 Output M(Q);
7 else
8 for each u0 $\in$ V (Q) - V (QD) do
9 for each dominating vertex ui$\in$N1(u0) and dominating vertex uj$\in$N2(u0) do
10 for v0 U$v\in C$ ($u\ i$)$N1(v)$ and U$v\in C$ ($u\ j$)$N2(v)$
11 if pair (u0, v0) satisfies conditions of Retrieving similarity set then
12 Add (u0, v0) to M(s) and compute current state s0;
13 Call DS-Match(s0);

Conclusion

In this paper, we study the problem of subgraph matching with set similarity, which exists in a wide range of applications. To tackle this problem, we propose efficient pruning techniques by considering both vertex set similarity and graph topology. structural signature buckets are designed Finally, we propose an dominating-setbasedsubgraph match algorithm to find subgraph matches. experiments have been conducted to demonstrate the efficiency and effectiveness of our approaches compared to state-of-the-art subgraph matching approach.

REFERENCES

Cordella, L. P., P. Foggia, C. Sansone, and M. Vento, 2004. "A (sub) graph isomorphism algorithm for matching large graphs," *PAMI*, vol. 26, no. 10.

Di Natale, R., A. Ferro, R. Giugno, M. Mongiov`ı, A. Pulvirenti, and D. Shasha, 2010. "Sing: Subgraph search in nonhomogeneous graphs," *BMC bioinformatics*, vol. 11, no. 1, p. 96, 2010.

Gulsoy G. and T. Kahveci, 2011. "Rinq: Reference-based indexing¨ for network queries," *Bioinformatics*, vol. 27, no. 13, pp. 149–158.

Han, W.-S., J. Lee, and J.-H. Lee, 2013. "Turboiso: Towards ultrafast and robust subgraph isomorphism search in large graph databases," in *SIGMOD*.

He H. and A. K. Singh, 2008. "Graphs-at-a-time: query language and access methods for graph databases," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, pp. 405–418.

Khan, A., Y. Wu, C. C. Aggarwal, and X. Yan, 2013. "Nema: Fast graph search with label similarity," *Proceedings of the VLDBEndowment*, vol. 6, no. 3, pp. 181–192.

Memisevi˘ c V. and N. Pr´ zulj, 2012. "C-graal: Common-neighbors-˘ based global graph alignment of biological networks," *Integrative Biology*, vol. 4, no. 7, pp. 734–743.

Shang H., Y. Zhang, X. Lin, and J. X. Yu, 2008. "Taming verification hardness: An efficient algorithm for testing subgraphisomorphism," *Proceedings of the VLDB Endowment*, vol. 1, no. 1.

Sun, Z., H. Wang, H. Wang, B. Shao, and J. Li, 2012. "Efficient subgraph matching on billion node graphs," *PVLDB*, vol. 5, no. 9.

Ullmann, J. R. 1976. "An algorithm for subgraphisomorphism," *Journal of the ACM*, vol. 23, no. 1.

****