I. J. Computer Network and Information Security, **2017, 4, 48-56**
Published Online April 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2017.04.06

A Survey on Graph Queries Processing: Techniques and Methods

Hamed Dinari
Research Assistant, Web and Search Engines Lab, Department of Computer Engineering Iran University of Science and Technology (IUST) Narmak, Tehran, Iran
E-mail: dinari@comp.iust.ac.ir, dinari.hamed@yahoo.com

Abstract—Graphs are widely used to model complicated structures and link them with each other. Some of such structures are XML documents, social networks, and computer networks. Information and model extraction from graph databases is a graph mining process. Efficient query search in graph databases, known as query processing, is one of the heated debates in the field of graph mining. One of the query processing techniques is sequential search over the whole dataset and isomorphism test on all sub-graphs in the database, which is not an optimal technique as to response time and storage. This problem brought in the issues of indexing graph databases to improve query processing performance. As the method implies, part of the database where the answer is expected to be found there is pruned and the number of needed isomorphism tests decreases. It might not be easy to compare the methods and techniques of graph query techniques as different techniques have different objectives. For instance, similarity search techniques reduce query time, while they cannot compete with exact matching techniques as to accuracy and vice versa. Input data volume might be also effective on query time as with immense datasets, similarity search techniques are more preferred than exact matching techniques. The present study is a survey of graph query processing techniques with emphasis on similarity search and exact matching.

Index Terms—Graph Mining, data mining, graph database Indexing, graph query processing, pattern matching.

I. Introduction

Databases are widely used for structured and complicated data management including stringdata, stream data, video, images, trees, and graphs [1]. Graph data is more complicated and general structure and it is widely used to picture combination of proteins and compare their structure [2], relationship networks, medicine design [3], social networks [4-5], road networks [6], video indexing [7], web information [8] computer vision, pattern detection, and chemical/biological informatics. Searching graph to extract required information is one of the main fields of graph mining [9-10]. In most of the cases, value and applicability of a graph data application depends on performance of its graph query. This is one of the key issues in graph mining field, which by definition suppose a graph query q and a graph database D = $\{ g_1, g_2, g_3, \dots, g_n \}$ the answer of query q is the number of isomorphism graphs in database D by the query. The preliminary graph query techniques extracted all isomorph super-graphs by a graph query in the database. Clearly, this technique is not so efficient and like isomorphism test, sequential scan of each database graphs needs great processing and storage resources. To accelerate graph query process, therefore, we need to index the database. As a simple type of graph query, several techniques have been proposed for XML databases indexing [7, 11-13]. There are, however, key issues to deal with in graph query processing such as 1- how to store database graphs to achieve more efficient processing? 2- how to define similarity of graphs? 3- How to create an efficient indexing structure to accelerate pattern match and graph search and improve the performance? [14] The rest of the paper is designed as follows. The next section discusses the basic concepts of graph mining and the readers with graph mining background can skip it. Section three discuss available graph query processing techniques from different aspects such as patterns exact matching, patterns inaccurate matching (similarity), mining and non-mining query processing techniques and methods, data structures to store indices, and input graphs to improve performance. Finally, conclusion and future works are represented by section four.

Basic Concepts: Some technical terms and definitions about graph mining are introduced in what follows:

***Graph*:** a graph is displayed as $G(V, E)$, where V stands for a set of heads and $\mathrm{E} \subseteq \mathrm{V} \times \mathrm{V}$ is a set of edges that connect the heads.

***Subgraph*:** suppose two graphs $\mathrm{G_1}(\mathrm{V_1}, \mathrm{E_1}, \mathrm{L_{v1}}, \mathrm{L_{E1}}, \varphi_{\mathrm{v1}}, \varphi_{\mathrm{E1}})$ and $\mathrm{G2}(\mathrm{V_2}, \mathrm{E_2}, \mathrm{L_{v2}}, \mathrm{L_{E2}}, \varphi_{\mathrm{v2}}, \varphi_{\mathrm{E2}})$, where $\mathrm{G_1}$ and $\mathrm{G_2}$ are the subgraphs that meet the following conditions:

$$\mathrm{V_1} \subseteq \mathrm{V_2}, \ \forall \mathrm{v} \in \mathrm{V_1}, \varphi_{v1}(\mathrm{v}) = \varphi_{v2}(\mathrm{v})$$

$$\mathrm{E_1} \subseteq \mathrm{E_2}, \forall(\mathrm{u}, \mathrm{v}) \in \mathrm{E_1}, \varphi_{E1}(\mathrm{u}, \mathrm{v}) = \varphi_{E2}(\mathrm{u}, \mathrm{v})$$

In addition, G2 is a supergraph of G1.

Isomorph Graphs: two graphs are isomorph when there is one-by-one correspondence between the heads and the edges. For instance, graphs G and H are isomorph ($H \cong G$), when the following conditions are met:

$With\ xy \in E \leftrightarrow \Phi(x)\,\Phi(y) \in E'\ for\ all\ x,y\ in\ V$
$V \to V'\ \Phi$:

Single Graph Database: the data are represented as a supergraph (e.g. Facebook, Twitter, Google +, or telecommunication connections).

Transactional Graph Database: The data are represented as a set of several independent graphs (e.g. protein, amino acids, and chemical/biological informatics databases). Fig. 1. illustrates a graph database comprised of 3 graphs.

***Support*:** the number of possible repetition modes of pattern S in graph database G. For instance, suppose $freq(s)$ stands for number of possible repetitions of pattern S in the database, and |D| stands for the database size, then $Sup(s)$ is defined as equation (1).

$$Sup\ (s) = freq(s)\ /\ |D| \quad (1)$$

In this type of support, $freq(s)$ is equal with number of graphs where the pattern s takes place. In addition, pattern S is counted once even when it appears in a graph for several times.

Frequent pattern*:* The pattern of which the support is more or equal with a threshold defined by the operator.

Closed pattern: The pattern for which any possibly bigger pattern has the same number of supports. In other words, if a patters, bigger than the first pattern, exists with the same number of support, then the second pattern is closed.

Query Processing: suppose a database $\{g_1, g_2, g_3, \dots, g_n\}$ and query q. graph query q processing; Graph query process (q) refers to all q_is belonging to D including the query q.

Figure 1. Illustrates a graph database and Figure 1.(d) illustrates a graph query where the answer of graphs query is $\{b, c\}$.

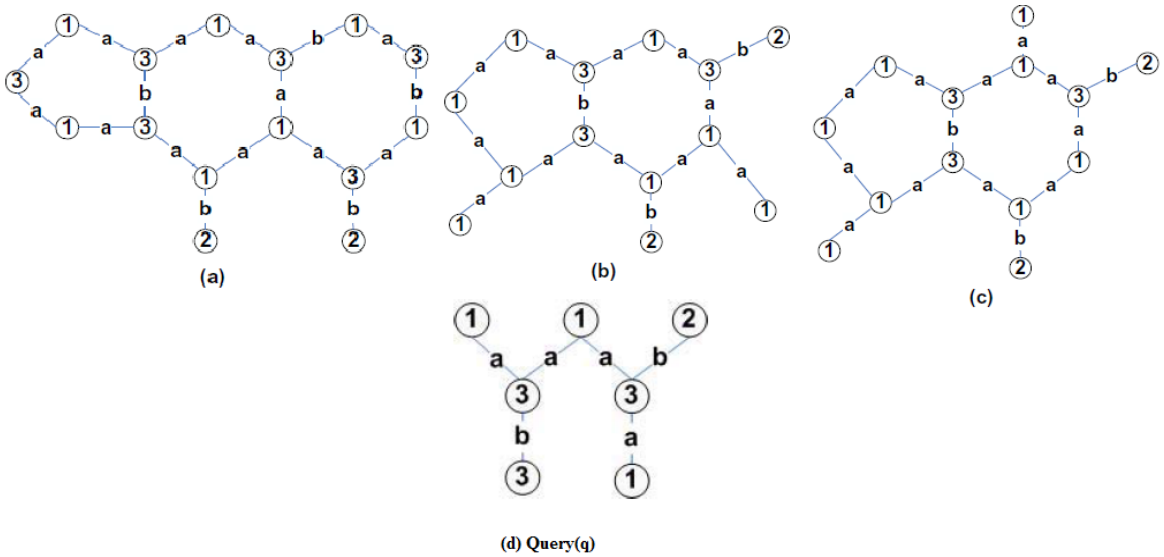


Fig.1. Graph Database Including 3 Graphs and One Query [**1**]

II. Related Works

In [15], several programming languages including GraphLog [16],G [17], and GraphDB [18], which have developed during 25 years ago to process graph queries, have been considered from diverse aspects such as syntax, and applications used. Furthermore, the types of graphs on which these languages perform such as directed/undirected/labeled/unlabeled graphs as well as graph representation to process the queries have been discussed. Some of these languages first transform input graphs and queries into collections of strings, distinct paths, and algebraic expressions, then process them. GOOD is another graph query language based on an object-oriented model [19]. In this paper, methods that are used for exact pattern matching, finding similar queries, and ranking answers of queries are discussed.

The main idea of [20] is to improve the processing of graph queries using parallel techniques and frameworks such that MapReduce (Hadoop). In addition to weaknesses and restrictions of these methods, techniques to resolve them is pointed out. In [21] in order to process the queries the frameworks that use multi-join for processing is applied which employ relational databases and don't store graphs. as well as top-k graph pattern problem is argued.

In [22] pattern matching problem is argued such as:(1)

pattern graphs that specify search conditions and (bounded) connectivity, and (2) bounded simulation.

In [23] the features which is extracted from graphs, is used to process similar queries (similarity search), moreover some of the algorithms is considered plus weakness and the strengths of them.

In [24] some of the methods that is used for processing of queries is discussed such as: Exact vs. inexact matching, Optimal vs. approximate solutions ‹Structural vs. semantic matching, as well as in terms of graph database types that queries search on them is done such as Single-graph vs. graph- transaction setting. In [25] tree patterns matching techniques, efficient join-based algorithms, and optimization techniques for graph pattern matching is discussed comprehensively. In [26] applications and algorithms that is used for finding graph and tree patterns on the graph databases is mentioned, furthermore some of the algorithms in grate details, as well as the way that is index constructed and stored for processing of the queries is discussed.

III. Classification of Graph Query Processing

A. Non-minding based indexing techniques

Rather than indexing few selected indices, the techniques under this classification index all structures of the database. Some of disadvantages of this method are 1- poor pruning power and 2- expensive comparisons during filtering. Regarding the advantages, updating capability with no expense to update the selected features as index and rebuilding are notable. Ullmann [27] proposed a subgraph matching algorithm based on state space search using back tracking technique. Given that this method is featured with testing isomorphism of subgraphs (an NP-Complete problem), it is not effective on supergraphs or where number of the database graphs increases [28].

A.1 GraphGrep technique

The technique, called path-based, was proposed by Giugno and Shasha (2002). Index structure uses counted paths as the index features to filter the irrelevant graphs. All the available paths to a maximum length are counted and number of the occurrences number of each path is stored. Thus, each row and column in the index table represents a path and a graph respectively. In addition, each entry of the table represents number of path occurrence in the graph. To find the candidate graphs sets, which include the paths in the query structure, and to check whether the number of the paths exceeds the threshold of query, indexed paths are used by the query process. Afterward, each candidate graph is surveyed at verification stage using isomorphism of the subgraph to obtain the results. An advantage of this path is that path indexing process is fast for the paths with limited length; however, with increase of the graph database size, the size of indexing paths increases exponentially while the expense of verification increases with increase of candidate set size. The technique is recommended for small databases with small number of graphs [29].

A.2 GD-Index

The technique was proposed by Williams in 2007. As the technique implies, all the subgraphs connected to and induced from a graph are determined at first. Then, a graph of the size of n encompassing $2\text{^}n$ subgraphs, each labeled with unique label is generated. However, due to isomorphism among the graphs determined by one complete graph, the subgraphs with the same labels might in turn be decomposed to more subgraphs. When all the labels are identical, a complete graph with the size of n is decomposed into $n+1$ subgraphs. A directed acyclic graph (DAG) is used to model the decomposed graphs and the links between them. DAG always has a node that represents the whole graph G and a node that is known as null graph. Children of the node P are the graphs encompassing Q with a directed link between P and Q in the DAG. In addition, grandchildren of the node P are all the nodes of DAG accessible from P. This technique is not recommended for large graphs. Figure 2. pictures an example of reducing the graph size through GD-Index method [11].

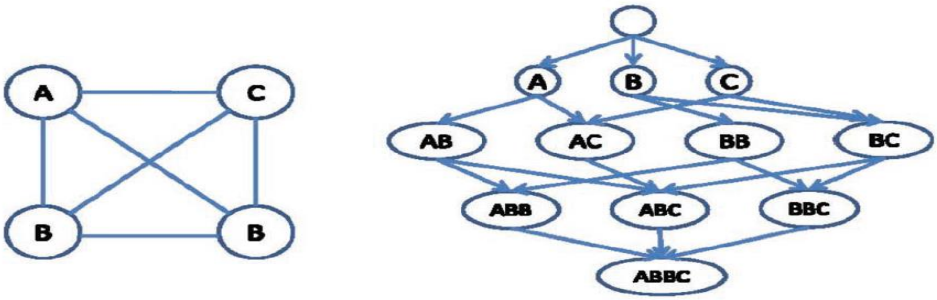


Fig.2. Parsing a Graph by GD-Index Technique [**10**]

A.3 GString

Jian (2997) introduced Gstring as a method to examine the structural concepts and put more emphasis on modeling biochemical objects with basic structures such as line, star, and cycle structures that bear concepts and are used as features of index.The line structure is comprised of connected heads, the cycle structure is

comprised of heads that form a closed loop, and finally, star structure is comprised of a head at center that is connected to several other heads. For a given graph g, GString first extracts all cycle structures followed by star structures and line structures (Figure 3) GString transforms the graphs and queries on the graphs as a sequence of strings and transforms the subgraph search into a sub-string matching problem. In addition, the strings are displayed by a prefix tree structure. A prefix tree data structure is used to display all strings and match the strings in an efficient way GString is comprised of three elements including 1- type; 2-size; and 3- edited sets.

Number of heads is given for the line and cycle and in the case of star, capacity of the central node is known. A disadvantage of GString is transformation of searched queries to matching problem, which is nowhere near efficiency especially when the size of graph database or query increases. Additionally, GString is mainly used for decomposition of chemical compounds into basic structures that represent biochemical concepts. The method is not recommended for other applications [30].

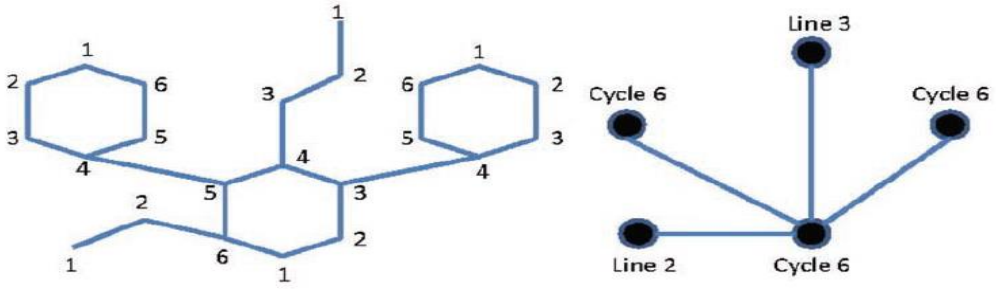
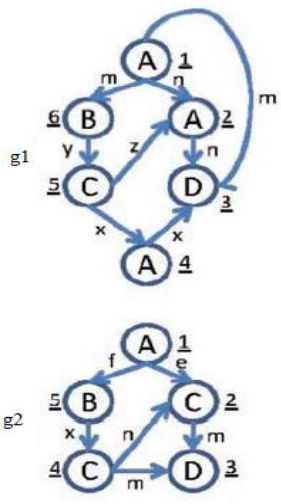


Fig.3. Graphs under GString Technique [10]

A.4 GraphREL

Sakr introduced GraphREL in 2009 for graph query processing. The graph database, under this method, is encoded as a head-edge relationship table (Figure 4). Afterward, graph query is encoded as a SQL string to work on the stored table. One of the issues of GraphREL is the expense of joining large number of the tables for further processing. Main GraphREL optimization technique is based on observations that influence mid results and performance of SQL scripts. Therefore, it stores frequency of the nodes and the edges in simple tables of graph database [10].



graphID	vertexID	vLabel
1	1	A
1	2	A
1	3	D
1	4	A
1	5	C
1	6	B
2	1	A
2	2	C
2	3	D
2	4	C
2	5	B

Vertex Table

graphID	sVertex	dVertex	eLabel
1	1	2	n
1	1	3	m
1	2	3	n
1	4	3	x
1	5	4	x
1	6	5	y
1	5	2	z
1	1	6	m
2	1	2	e
2	2	3	m
2	4	3	m
2	4	2	n
2	5	4	x
2	1	5	f

Edge Table

Fig.4. Encoding the Graphs by GraphREL [**10**]

In a graph query q, statistical information is stored to determine pruning points on the structure (nodes and edges with lower frequency are filtered). In the case of large graph queries, GraphREL uses a decomposition mechanism to convert large and complicated SQLs into mid-query strings (using temporary tables before computing the final results). Using the statistical information stored, the decomposition mechanism attenuate mid-results obtained at each stage.

B. Mining-based Graph indexing techniques

Mining-based graph indexing runs a graph mining algorithm on graph database, and indexing patterns are implemented on these patterns after mining. Some of the

most common techniques and methods of this type are discussed in what follows.

B.1 Gindex (Graph-based Index)

Yan (2004) introduced the first pattern mining technique based on graph database indexing. His method was based frequent graphs as the base indexing unit. The key point of the proposed method was that the graph-based index considerably improves query performance comparing with path-based indexing. However, a limitation of using subgraphs as indexing unit is that the number of graphing structures is usually more than the paths in a graph database. To solve this, GIndex method only surveys the frequent subgraphs. Therefore, to avoid uncontrolled increase of frequent graphs, support threshold increases when number of the subgraphs increases. Suppose a graph query denoted by q, if it is a frequent subgraph, then the set of the query's answers is reviewed with no need to check the candidate as q is indexed. When graph query of q is infrequent, the subgraph exists only in few graphs in the graph database, which means, number of graph isomorph tests is decreases [31] [10].

B.2 Tree+Delta (Tree-based Mining)

A method based on frequent sub-trees as indexing unit for **TreePI** graph structures was introduced by Zhang (2007). The idea behind this technique is based on two key points: i) the data are arranged in tree structure, routs patterns are more complicated and the trees can store more structural information of the frequent subgraph pattern; ii) frequent sub-trees mining process is relatively simpler than that of the subgraphs. Thus, by mining frequent trees on the graph database and then selecting a set of frequent tress, TreePI is an index pattern. To process query for the graph query q and obtain a candidate set, the frequent sub-trees of q are determined and compared with a set of index features. For verification, isomorphism of the stored information is tested. Considerable improvement of indexing and searching query can be achieved as a canonical form of each tree is computed in multinomial time frame. Additionally, comparing with graphs, operations such as isomorphism or normalizing are performed easier on trees. Such operations on graphs usually are of NP-Complete problems [10] .Many key structures in biology and chemistry are trees (e.g. RNA).

B.3 FG-index

The process starts by mining the closed frequent subgraphs, then, an inverse index is created on the frequent subgraphs. The invers index includes:

1. An array called Graph Array (GA), which is used to store the closed frequent subgraphs. So that, GA[i] is the i^{th} entry of GA and showed by g. Suppose G is a set of frequent subgraphs, then the relevant set is displayed by $CLOS(g)$ when g encompasses a set of subgraphs and g is the frequent subgraphs of the set. What we have here is a nested list so that GA[i] refers to $GLOS(g)$.
2. An array called Edge Array (EA), which stores a set of different edges of G.
3. Every different edge in EA refers to an ID-entries list and each ID-entries in turn refers to a list of arrays known as ID-entries. Each ID-array includes a set of IDs, and the IDs in turn are set of graphs. For instance, if $f_1, f_4, f_5, f_8, f_9, f_{13}$ are a set of closed frequent subgraphs and a, b, and c are sets of the different edges of the frequent subgraphs. Edge Array – i.e. EA[0] – can be seen in the first row (Figure.1) so that edge a in GA[2] ({2}) mentioned in Size-2 ID-entry, occurs once (the left side number of the entry). Moreover, the same edge in Size-4 ID-entry occurs for three times in a subgraph in GA[6]({3}). The query operation can be accomplished using this table and the nested lists. An advantage of this method is its pace when the input query is a closed frequent subgraph; otherwise, the query is performed using inverse index and subgraph search techniques. Figure. 5. illustrates inverted index for processing graph queries [32].

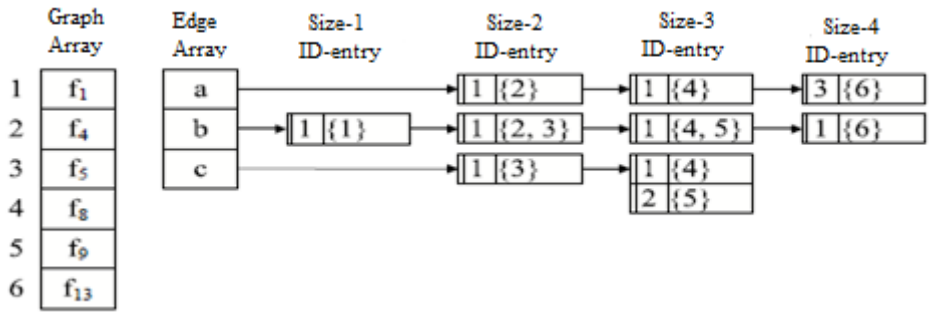


Fig.5. Inverted index [**32**]

This method, however, is recommended only for applied programs with small sub-structures. In addition, CDIndex [33] is used for the graph with limited size so that all the subgraphs in a database are determined first and then stored in a mixing table using canonical labeling code technique. The obtained table is used to answer the

queries. Moreover, C-tree [12] employs R-tree data structure to index and answer graph queries.

C. Similarity search methods and techniques

Similar subgraph query is one of the areas of interest [34]. These methods are featured with a query and a database of graphs and the aim is to find subgraphs similar to the query. Thus, these methods can employ node mismatch and node gap (node gap is the node that cannot be mapped to other nodes in the database) along with structural differences of the graphs. Graph approximated matching techniques are used for crowded databases or when part of the data is missed. These techniques outperform exact matching with such data. Similarity queries can be divided into two general groups [9-10]:

1. K-NNs query
2. Range query

K-NNs queries return data with more similarity as answer of query, while range query returns all the graphs that match a default range of the queried data. There is variety of similarity measures including:

1. Edit distance
2. The longest distance of common subgraph, which is used frequently

Edited distance between two graphs represents minimum expense for transforming a graph. The transformations have to do with adding or removing heads and edges. MCS distance between two graphs G1 and G2 is defined as equation (2):

$$d_{MCS}=1-\frac{|V_{MCS}|}{\max(|V1|,|V2|)} \quad (2)$$

Where, V_{MCS}, V_1, and V_2 are number of heads of the graphs MCS, G1, and G2 respectively.

C.1 Daylight Finger Print method

As the method implies, all the routs to a specific length (length 7) are extracted as descriptor and a molecule is indexed by its descriptor as a bit string [35].

C.2 Grafil (Graph Similarity Filtering) method

Yan (2005) proposed a feature-based structural filtering algorithm, called Grafil, to find similar queries in a graph database. Among the features of this method is that the queries are modeled as a set of features of the model and that it filters several graphs though comparing similarity. Two matrix data structures – graph features matrix and edge features matrix - are used to find similar queries. Graph features matrix is used to compute difference between features of a graph query and graphs available in a dataset; so that the columns represent a graph in the database graph and the rows represent the indexed features. On the other hand, edges features matrix uses multi filter decomposition strategies so that each filter utilizes a specific set of features. The filters are made based on one dimensional lustering algorithms and hierarchy; while the features are grouped into set of features based on similar selectivity. Throughout matrix query processing, Features of the graph are used to compute number of different features between each members of graph database g_i and query q and when the differences > d_{max} the graph is removed from comparison list and the candidate answer set is comprised by the remaining matrices [10].

C.3 G-Hash method

The method starts by extracting all features of the nodes and edges of the graphs and then the index is built using a hashing table. To process the queries, the method needs extracting features of the nodes and edges and returns k closest index records of data query as the output. The method achieves a great reduction in the time needed to build the index and to find query answer thanks to hashing table [10].

C.4 Substructure index-based approximate graph alignment (SAGA) matrix

SAGA is an approximate matching graph query method that computes similarity between graphs with equal distance from each other so that similar graphs are closer to each other. Distance model includes 3 elements:

1. StructDist: refers to structural difference to match pair-nodes in two graphs;
2. Node-Mismatches: the expense of matching two nodes based difference;
3. Node gaps: measures expense of gap nodes based on graph query.

SAGA index is built on fragment indices of the graphs in a database. Each piece is a set of *k* nodes in the graphs of a database and *k* is defined by the operator. The index does not count all the sets with *k* nodes. The parameter $dist-max$, which is defined by the user is used to avoid indexing each pair of nodes in a piece of graph when the distance measure > $dist-max$. SAGA pieces do not always answer the linked subgraphs. To have an efficient assessment of subgraph distance, another index called "distance index" is stored between graph query and database graph. The new index is used to go through the pre-computed distance between each pair of nodes in a graph. The matching process is featured with three steps:

I. The search space is divided into smaller fragments; then, "fragment index" is probed to find database pieces similar to query pieces.
II. Larger candidates are generated when the probing is successful. An edge is removed when the probing the index is successful and only and if only two pieces of queries are zero or more nodes are shared.
III. To generate the answer of the set, each candidate must be tested. Gap node percentage for each candidate is checked and the candidate matching

is neglected when the gap node percentage is bigger than the user defined threshold P_g ; otherwise, Distance Index is probed to compute subgraph-matching distance. When the distance of two matches is equal and one is sub-match of the other, only the bigger matching is examined [10].

C.5 A Tool for approximate Large graph matching (TALE)

The method is known as NH-Index from the rooted Neighborhood Index. Index volume in some methods that use rout, tree, and subgraph to build index is increased considerably with increase of database size. However, TALE uses neighborhood of each node as the index structure as it is used for very large databases and it prunes great deal of the data graph (Most Pruning Power). Index structure size in this method increases with the size of database. Moreover, NH-index is a disk-based index that enables it to handle graph database without needing extra memory space. Each neighbor node is defined as an induced subgraph from the same node and the neighbors (adjacent nodes). Three features of the index used to describe the neighbors are number of neighbors, the way the neighbors are linked to each other, and the neighbors' label. Moreover, TALE method is recommended for implementing index structure based on structure of the data such as bit array and B+-tree thanks to its high performance [14]. Furthermore, the PIS method [36] selects several similar graphs, based on the query, and builds the preliminary set. APEX method [37], on the other hand, employs index structures matching to process graph queries. Figure 6. illustrates general classification of the graph query processing methods and the techniques. And Table 1, Table 2, and Table 3 list graph query processing methods and techniques with their strengths and weaknesses.

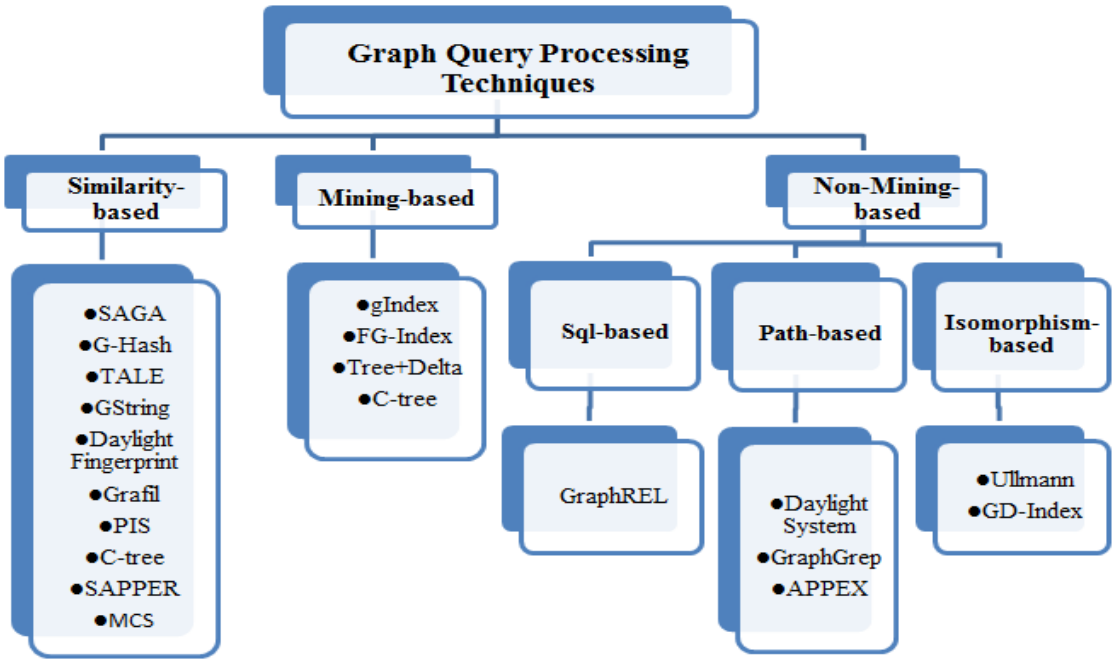


Fig.6. General Classification of the Graph Query Processing Methods and Techniques

Table 1. Non-Mining-based Graph Query Processing Approaches

Weakness	Strength	Index Features	Approaches
-Failure to detect loops, chains, and closed loops -Increase of index volume with increase of database size	Good performance with small databases	Save all paths to a specified length	GraphGrep
Poor performance with big databases	- High accuracy with small databases - No need to rebuilding the index after transforming the graphs	Save all the connected subgraphs	GDIndex
Do not use in areas with the exception of chemistry	Good performance in chemical databases	Save linear, star, cycle structures That have special meaning	GString
join tables to each other is expensive with large databases	SQL syntax compatible and no need for writing codes	Node and Edge Features-based (Vertex-Edge Schema)	GraphREL

Table 2. Mining-based Graph Query Processing Approaches

Weakness	Strength	Index Features	Approaches
-Updating and rebuilding index in the case of graph change	-Performance improvement if query indexed and not need to candidate verification	Save frequent subgraphs	GIndex
-Updating and rebuilding index in the case of graph change - Not a good pruning the state space	- Performance improvement if query is indexed and not need to candidate verification -Subtrees mining simpler than subgraphs	Save frequent subtrees	C-Tree
- Index construction is costly because frequent closed subgraphs must be mined -Updating and rebuilding index in the case of graph change	Good performance if query is indexed	Save closed frequent subgraphs	FG-index

Table 3. Strengths and Weaknesses of Graph Processing Methods

Weakness	Strength	Technique	Method
- Poor performance with big databases - Isomorphism test	- High accuracy with small databases - No need to rebuilding the index after transforming the graphs	Isomorphism-based	Non-Mining-based
- Failure to detect loops, chains, and closed loops - Increase of index volume with increase of database size	- Good performance with small databases	Path-based	
- join tables to each other is expensive with large databases	- High accuracy with small databases - SQL syntax compatible and no need for writing codes	Sql-based	
- Needed time to build the index - Updating and rebuilding index in the case of graph change - Needed memory to store subgraphs	- Performance improvement - No need to check the candidate (during indexing query)	Tree+Graph-based	Mining-based
- Good performance with large databases	- poor accuracy (approximate answers)	Node and Edge Features-based	Similarity-based

IV. Conclusion And Future Works

After a brief review of graphs and their applications, some of the common terms and definitions pertinent to graph mining were introduced. Section three brought in some of the techniques and methods of graph mining and section four gave a general classification of graph query processing such as non-mining and mining methods and techniques and many other methods known for processing graph queries. One may conclude that methods with better pruning performance promise higher performance regarding needed memory space and faster isomorphism tests. Future works may focus on using graph nodes (min, max, total) and nodes label for pruning state space. Moreover, the algorithm used for frequent patterns mining on dynamic graph databases can be employed for online query processing and to eliminate delay to build inverse index.

References

[1] S. Zhang, M. Hu, and J. Yang, "TreePi: A Novel Graph Indexing Method," in *IEEE 23rd International Conference on Data Engineering*, Istanbul, 2007, pp. 966-975.

[2] J. Huan, W. Wang, D. Bandyopadhyay, J. Snoeyink, J. Prins,and A. Tropsha, "Mining Protein Family Specific Residue Packing Patterns from Protein Structure Graphs," in *in Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, 2004, pp. 308-315.

[3] C.Borgelt, and Michael, R. Berthold, "Mining Molecular Fragments: Finding Relevant Substructures of Molecules," in International Conference on Data Mining (ICDM), 2002, pp. 51-58.

[4] Pizzuti, Clara. , "GA-Net: A genetic algorithm for community detection in social networks.," Parallel Problem Solving from Nature–PPSN, 2008; pp. 1081-1090.

[5] Misra, Sudip, Romil Barthwal, and Mohammad S. Obaidat, "Community detection in an integrated internet of things and social network architecture," in Global Communications Conference (GLOBECOM), IEE 2012; pp. 1647-1652.

[6] H.Dinari, H.Naderi, "A Survey of Frequent Subtrees and Subgraphs Mining Methods," International Journal of Computer Science and Business Informatics, Jun. 2014; vol. 14:1, pp. 39-57.

[7] B. T. Messmer, and H. Bunke, "A Decision Tree

Approach to Graph and Subgraph Isomorphism Detection," Pattern Recognition, Dec. 1999; vol. 32, no. 12, pp. 1979-1998.
[8] S. Raghavan and H. Garcia-Molina, "Representing Web Graphs," in IEEE Inernationall Conference on Data Engineering , 2003, pp. 405-416.
[9] Wang, Haixun. Ed. Charu C. Aggarwal. , Managing and mining graph data, V. 40, Ed. New York, USA: Springer, 2010.
[10] S.Sakr, E.Pardede, Graph Data Management: Techniques and Applications. United States of America: Information Science Reference (an imprint of IGI Global), 2012.
[11] Johansson, "Graph Decomposition Using Node Labels," Doctoral Dissertation, Royal Institute of Technology, 2001.
[12] He, Huahai, and A.K. Singh, "Closure-tree: An index structure for graph queries," in 22nd International Conference on Data Engineering (ICDE), Atlanta, Georgia, 2006; pp. 38-50.
[13] Chen, Zhiyuan, et al, "Index structures for matching XML twigs using relational query processors," in Data & Knowledge Engineering, 2007; pp. 283-302.
[14] Tian, Yuanyuan, and J. M. Patel, "Tale: A tool for approximate large graph matching," in IEEE 24th International Conference on Data Engineering (ICDE), 2008; pp. 963-972.
[15] Wood, Peter T, "Query languages for graph databases," ACM SIGMOD Record ,2012; vol. 41, no. 1, pp. 50-60.
[16] M. P. Consens and A. O. Mendelzon, " Expressing structural hypertext queries in GraphLog," In ACM Hypertext,1989; pp. 269–292.
[17] F. Cruz, A. O. Mendelzon, and P. T. Wood, "A graphical query language supporting recursion," In SIGMOD, May 1987; pp. 323–330.
[18] R. H. G ̈uting, " GraphDB: Modeling and querying graphs in databases," In VLDB, 1994; pp. 297–308.
[19] M. Gyssens, J. Paradaens, and D. V. Gucht, " A graph-oriented object database model," In PODS,1990; pp. 417–424.
[20] Doulkeridis, Christos, and Kjetil Nørvåg, "A survey of large-scale analytical query processing in MapReduce," The VLDB Journal, 2014; vol. 23, no. 3, pp. 355-380.
[21] Cheng, Jiefeng, and Jeffrey Xu Yu. , "A Survey of Relational Approaches for Graph Pattern Matching over Large Graphs," Techniques and Applications Graph Data Management, 2011; pp. 112-141.
[22] Fan, Wenfei, et al. , "Graph pattern matching: from intractable to polynomial time," " Proceedings of the VLDB Endowment, 2010; vol. 3, no. 1-2, pp. 264-275.
[23] Yan, Xifeng, et al. , "Feature-based similarity search in graph structures," ACM Transactions on Database systems (TODS), 2006; vol. 31, no. 4, pp. 1418-1453.
[24] Gallagher, Brian, "Matching structure and semantics: A survey on graph-based pattern matching," AAAI FS 6, 2006; pp. 45-53.
[25] Bhargavi, B., and K. P. Supreethi, "Graph pattern mining: A survey of issues and approaches," International Journal of Information Technology, 2012; pp. 401-407.
[26] Shasha, Dennis, Jason TL Wang, and Rosalba Giugno, "Algorithmics and applications of tree and graph searching," Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.ACM, 2002.
[27] J. R. Ullmann, "An algorithm for subgraph isomorphism," ACM, 1976; pp. 31-42.
[28] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness.," W. H. Freeman & Co., 1979.
[29] Giugno, Rosalba, and D.Shasha, "Graphgrep: A fast and universal method for querying graphs," in IEEE, 16th International Conference on Pattern Recognition, 2002; pp. 112-115.
[30] Jiang, Haoliang, et al, "Gstring: A novel approach for efficient search in graph databases," in 23rd International Conference on Data Engineering(ICDE) IEEE, 2007; pp. 566-575.
[31] Yan, Xifeng, S.Philip Yu, and J.Han. , "Graph indexing: a frequent structure-based approach," in ACM SIGMOD international conference on Management of data, 2004; pp. 335-346.
[32] Cheng, James, et al., "Fg-index: towards verification-free query processing on graph databases," in ACM - international conference on Management of data(SIGMOD), 2007; pp. 857-872.
[33] Williams, W.David , J.Huan, and W.Wang, "Graph database indexing using structured graph decomposition," in 23rd International Conference on Data Engineering(ICDE) IEEE, 2007; pp. 976-985.
[34] X. Yan, P. S. Yu, and J. Han, "Substructure similarity search in graph databases," in international conference on Management of data. ACM, 2005; pp. 766-777.
[35] James, C. A., D. Weininger, and J. Delany, "Daylight Theory Manual. Daylight Chemical Information Systems," 2003.
[36] Yan, Xifeng, et al, "Searching substructures with superimposed distance," in 22nd International Conference on Data Engineering (ICDE). IEEE, 2006; pp. 88-88.
[37] Chung, Chin-Wan, Jun-Ki Min, and K. Shim, "APEX: An adaptive path index for XML data," in SIGMOD international conference on Management of data, 2002; pp. 121-132.

Authors' Profiles

Hamed Dinari received his B.SC. Degree in Computer Engineering (Software) from University of Ilam, Ilam, IRAN, M.SC. in Computer Engineering (Software) from Iran University of Science and Technology (IUST), Tehran, IRAN, in 2012, 2014 respectively. His research interests are about Database Systems, Data Mining, Graph Mining, and indexing.

How to cite this paper: Hamed Dinari,"A Survey on Graph Queries Processing: Techniques and Methods", International Journal of Computer Network and Information Security(IJCNIS), Vol.9, No.4, pp. 48-56, 2017.DOI: 10.5815/ijcnis.2017.04.06