A Modified gSpan for Computing Rare Substructures in Graph Databases

M. Azaouzi[1], L. B. Romdhane[2]
MARS (Modeling of Automated Reasoning Systems) Research Group
[1]Faculty of Sciences, University of Monastir
Tunisia
[2]High School of Sciences and Technology of Hammam Sousse
University of Sousse, Tunisia
mehdi.azaouzi@gmail.com, Lotfi.Ben.Romdhane@Usherbrooke.Ca

ABSTRACT: *Graphs can represent any kind of data, as e.g. biology or chemistry, biological networks or chemical compounds. A graph database is a frequently used means to efficiently implementation of these data. With the increasing usage of graph databases, it has become more and more demanding to efficiently process graph queries. Querying graph databases is costly since it involves a test of structure matching of graphs, which is an NP-complete problem. Thus, to improve the performance of querying, must be reduced the number of subgraph isomorphism tests. Therefore efficient methods have been proposed to avoid most of these tests but still allow to identify all graphs containing the query pattern. In this paper, we propose a novel indexing feature, called RAre subGraphs (RGs). The rare subgraphs are candidates occurs in only a small number of graphs in the database. Since discovering patterns is an important problem in data mining, to discover these substructures must have an efficient algorithm. Classics mining systems provide a restricted mechanism on patterns with frequency higher to a minimum support. A major challenge, there are no efficient algorithms for the extraction of the patterns with an exact frequency. For this, we have proposed an adaptation of gSpan to compute the set of subgraphs for a given frequency. By adopting this model, we can extract the rare substructures.*

Keywords: Graph Databases, Graph Querying, Graph Indexing, Graph Algorithm

Received: 28 February 2013, Revised 1 April 2013, Accepted 5 April 2013

1. Introduction

Conceptually, graphs can represent any kind of data. Then, graphs play an important role in representing and understanding objects and their relationships in various domains. For example, in computer vision domain, graphs are used to represent complicated relationships in images, such as organization of entities, which can be used in identifying objects and scenes. In the scientific domain, the structure of chemical compounds can be modeled as graphs by associating a vertex with each atom and an edge with each chemical bond [1]. In semantic web, schema of heterogeneous web-based data sources and e-commerce sites can be modeled as graphs. In Biological networks of interactions between components in cells (e.g. proteins, genes) can be represented with graphs. Because of the wide use of graphs and to achieve the goal of understanding a collection of graphs, it is very important to provide users with effective ways to organize, access and analyze these data graphs. A graph database is adopted to accelerate and assist the understanding process and the accessing process. In many cases, the efficiency of a

database is presented in the responsiveness with the query processing. Let $D = \{G_1, G_2, .., G_n\}$ be a graph database that contains n graphs and a graph query Q. A subgraph-querying algorithm retrieves all graphs $G_i \in D$ containing Q as a subgraph. For example, searching a molecular structure in a database of molecular compounds is useful to detect molecules that preserve chemical properties associated with a well known molecular structure. Deciding whether one graph is a subgraph of another is referred to as the subgraph isomorphism problem; since subgraph isomorphism testing is known as an NP-complete problem [2]. So it is inefficient to perform a sequential search and check whether Q is a subgraph of G_i. Clearly, the solution is to index the graph database to reduced the number of graph comparisons. Previous work presented some effective indexes on graph databases. The query processing using these indexes can be divided in two phases. In the first step (the preprocessing phase), we analyze the database of graphs and an index is built. The second step is the filtering-and-verification framework. The filtering step uses the index to eliminate part of the false results and produces a candidate set C (Q) that may potentially contain the query. The set of candidates is then verified (verification step) by a subgraph isomorphism algorithm and all the resulting matches are reported, in ordre to finally obtain the answer set D (Q). Recently, indexing techniques for graphs databases have been developed with the purpose of reducing the number of subgraph isomorphism tests involved in the query process and minimizing the response time to the query. The appeal of the procedure subgraph isomorphism is done mainly during the verification phase. Thus, these approaches having the goal of maximizing the filtering rate, provide a minimization of the number of subgraph isomorphism tests. Thus, a good feature selected as a basic index structure is the feature that reduces the size of candidate set C (Q) to a minimum and avoids the false scandidate.

There are already a number of methods published that can be classified according to how the features are obtained. A first category of approaches can be distinguished according to the manner of indexing. A feature-based graph indexing, first analyze the database and extract features that to be used as index. Among the methods based on an feature, *GraphGrep* [3] and *GraphFind* [4] this are path-based indexing methods. This methods indexes all paths up toa threshold length from each graph. As second type of feature, Yan and al. [5] proposed in *gIndex*, only frequent and discriminative subgraphs as indexed features. More recently, Cheng and al. propose *Fg-index* [6] that uses compressed frequent patterns as index features. To compress the frequent patterns set, Cheng and al. introduce the concept of δ-*tolerance closed frequent subgraph*. A tolerance closed frequent subgraph can be considered as a representative of a group of frequent graphs. Recently, a novel indexing system SING [7] (Subgraph search In Non-homogeneous Graphs) is presented. The method uses the notion of feature, which can be a small subgraph, subtree or path. Each graph in the database is annotated with the set of all its features. The key point is to make use of feature locality information. The main drawback of these models is that the use of frequent patterns generate an exhaustive candidate set C(Q). Moreover, since paths or fragments carry little information about a graph, the lost of information at the filtering step seems to be unavoidable.

Considering the second group, all graphs of the base will form a cumulative structure clustered hierarchically. Therefore, a second category of approaches can be distinguished according to how to represent this structure. A Recently, hierarchical indexing systems have been proposed. *Closure-Tree* [8] and Graph Decomposition Index (*GDIndex*) [9] are two examples of hierarchical graph indexing. In these methodes a top-down search manner is appliqued, which allows an direct access to each graph of graph database. Thus, they avoid the faults answers and they decrease a size candidate set C(Q).

In this paper, we explore this direction and we introduce a novel indexing feature based on subgraphs with low frequency. Thus, our way of extraction these substructures set. A key feature is that it indexes a graph with the most discriminative rare structure. Intuitively, a rare structure is a subgraph with the minimal frequency that discriminatives the indexed graph from the rest of the graph database. In fact, we present the main changes affected to gSpan [10] to retrieve each structure.

2. Preliminaries

We denote a undirected labeled graph G can be represented by a 4-tuple $G = (V, E, \Sigma, l)$ where V is a vertex set, E is an edge set and is the alphabet of labels. Vertices and edges have labels determine by the function l. A graph database is a set of graphs $D = \{G_1, G_2, ..., G_N\}$ (see figure. 1). We focus on undirected graphs in which vertices have a single label as their attribute and edges have unspecified but identical labels. However, the concepts and techniques described can be extended to other kinds of graphs.

We assume the usual definition of graph isomorphism and the frequency of graph.

Definition 2.1: (Subgraph Isomorphism) Given two graphs, $G_1 = (V_1, E_1, \Sigma_1, l_1)$ and $G_2 = (V_2, E_2, \Sigma_2, l_2)$, a subgraph isomorphism

from G_1 to G_2 is an injective function f:

$V_1 \rightarrow V_2$ such that $\forall(u, v) \in E_1, (f(u), f(v)) \in E_2, l_1(u) = l_2(f(u)), l_1(v) = l_2(f(v))$, and $l_1(u, v) = l_2(f(u), f(v))$.

A graph $G_1 = (V_1, E_1, \Sigma_1, l_1)$ is said to be a *subgraph* of another graph $G_2 = (V_2, E_2, \Sigma_2, l_2)$ iff $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.

Definition 2.2: (Support measure) In a graph database *D*, *The support* of a subgraph *g* in *D* is the number of graphs $G \in D$ such that g is a subgraph. The graphs containing g in *D* comprise the *supporting set* of g, *D*(*g*). The frequency of *g*, *frequency* (g), is the ratio between g's supporting set and $|D|, |D(g)| / |D|$.

In this paper, we used the frequency of *g* is the supporting set. A substructures *P* is called frequent if the measure of the supporting set is greater or equal to a minimum support threshold θ.

3. Our Model For Computing Rare Substructures

3.1 Rare Substructures

We first introduce the concept of the frequency of a subgraph and explain how the frequency value can be applied in graph indexing. The subgraph g is a frequent subgraph if and only if its support is greater than a minimum support threshold, minSup. As one can see, frequent graph is a relative concept. Whether a graph is frequent depends on the setting of minSup. In gIndex, Yan et al. [5] propose an index on a set of discriminative frequent subgraph. However, the frequent subgraph in general included in many graph of database. Then, C_Q obtained, after a search with these indexes, can be very large. Thus, it likely contains many false candidates answers. To avoid this disadvantage, Cheng and al. in *Fgindex* [6], compress the frequent fragments set by determine a representative subgraph for each group of frequent subgraph. In this paper, we avoid completely the indexing with the frequent subgraph. The basic idea of our model is to associate, if possible, *a unique index* to each graph in the base. In our model, we try to assign the uniqueness constraint to our index. Based on definition 2.2, extract a unique index is equivalent to determine the fragments with a support equal to 1 (1- *frequency*). Assume that all the graphs in the database are indexed by substructures with 1-*frequency*. Whenever the request comes, if *Q* has a 1-*frequency*, then the set of candidate answers |CQ| is of small size and can be recovered directly since *Q* is indexed, in this way we reduce the time of the verification phase and thus the total response time. Otherwise, we look for the index containing *Q* as a subgraph. These indexes are 1-*frequency* and the set of candidate answers $|C_Q|$ is of small size and the exact set of query answers is returned without performing candidate verification.

One of the problems that can arise when using these fragments of frequency 1 is that we may have *incomplete* indexing, because not all graphs of the database containing fragments of frequency 1. This requires other fragments to index all the graphs. Therefore, we focus primarily on determining a threshold frequency. The frequency of index will be less than or equal to this threshold. This hreshold represents the minimum frequency that can be indexed to our base, we call f_{min}. The determination of f_{min} is done by the recursive call to an extraction procedure. The instructions in this procedure starts with $f_{min} = 1$. Then we extract all fragments in the database with frequency equal to 1. Thereafter, we test if the database is fully indexed, ie, each graph in the database contains a graph of frequency 1. If so, all indexes are find and f is obtained. Otherwise, we incrementing f_{min} by one, thus increasing the chance to choose other fragments that can index the rest of database. This process is repeated until the base is completely indexed. A flowchart of our approach is shown in Figure 2. We will adapt these tools to define the rare fragments used as index.

Definition 3.1: Let $D = \{G_1, G_2, ..., G_n\}$ be a graph database that contains a set of *N* graphs and f_{min} the minimum frequency that must be indexed *D*. A fragment *pr* is said rare relative to *D* if frequency (*pr*) <= f_{min}.

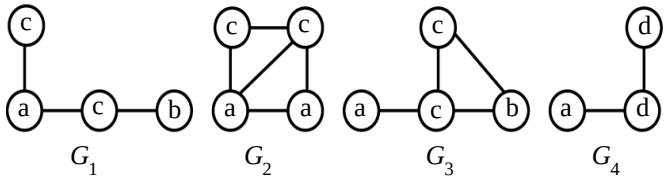

Figure 1. A graph database

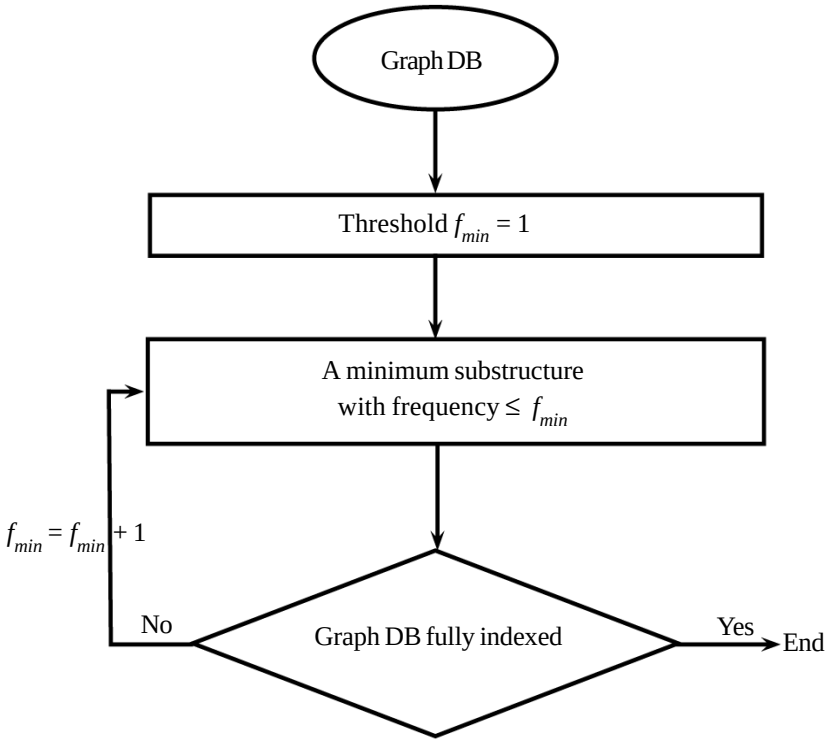

Figure 2. A flowchart of our algorithm

Figures 3 and 4, are retrieved from the substructures of the base shown in Figure 1. At this stage, we meet a challenge made in the extraction of substructures with *k-frequency*. Likewise,

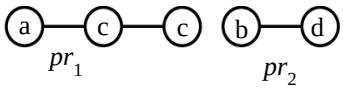

Figure 3. A Fragments with 1-*frequency*

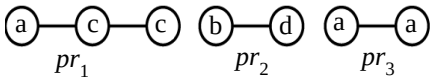

Figure 4. Rare Fragments

3.2 gSpan Modified

The problem of computing frequent-pattern sets is a common task in graph mining and deeply studied in the research literature. One of the most known algorithm is gSpan [10]. However, computing graph substructures whose frequency is exactly equal to a given value is not a trivial problem. Unfortunately, there are no efficient algorithms for the extracting of the patterns with an exact frequency.

The main difficulty in this step is the retrieval of all subgraphs with an exact frequency. The main propose of this section is to a propose a modified *gSpan* algorithm for computation of graph substructures with a specified frequency. Given a set of data graphs, $D = \{G_0, G_1, ..., G_n\}$ and a value of frequency *f*, the problem is to find all subgraphs *G* such that *frequency* (*g*) = *f*.

Several approaches have recently been developed such as gSpan, MOFA, FFSM and gaston [11] to extract the frequent subgraph. The algorithm *gSpan* [10] is considered among the best methodologies that exploit the depth-first search (DFS) for searching frequent graphs. Two techniques are used, the DFS lexicographic order [10] and the minimum DFS codes [10]. *gSpan* combines the growing and checking of frequent subgraphs into one procedure, thus accelerating the mining process. gSpan has the following desirable properties. It reduces the generation of duplicate graphs. It does not need to look for the previously

discovered frequent graphs for detecting the duplicates and it does not extend the duplicated graph; while ensuring discovery of the complete set of frequent graphs. For these reasons, the formalism of algorithm *gSpan* was chosen in our work for modeling the problem of extracting substructures with an exact frequency. We propose "*gSpan-Modified* ", an algorithm for pattern mining with an exact frequency. Here, we illustrate the major changes that we have allocated in the formulation of our algorithm. All the advantages of *gSpan* are called in our algorithm, such as DFS lexicographic order and the minimum DFS code. The changes realized at the level of the stop condition and at the level of selection criteria of substructures. Instead of recursively called of *gSpan* to expand a graph model until the support of a graph newly formed is less than the threshold *minSup* or until its code is not a minimum code, *gSpan-Modified* is called to extend a graph model until the frequency of the graph newly formed *g* is equal to the input frequency *f* or until its code is not a minimum code. So we stop the extension of a subgraph *g* when frequency (*g*) = *f* or DFS code of *g* is no longer minimal. On the other hand, because the frequency is lowered with the edge-growing strategy, the extension in our algorithm includes only the models with a frequency greater than *f*. In fact, due to the apriori property, the extension of a subgraph whose frequency is strictly less than *f* will produce supergraphs whose frequency also is strictly less than *f* [**?**]. Hence, instead of extending the set of frequent subgraphs of size *k* to determine whether a graph of size (*k* + 1) is frequent, the extension in *gSpan-Modified* is performed on all the subgraphs of size *k* whose frequency is greater than *f*.

Algorithm 1 gSpan-Modified (*s, D, f, S*)

Input: A DFS code **s**; a graph data set **D**; and the frequency **f**

Output: A graph set **S** with frequency *f*

```
1  begin
2      if s ≠ min(s) then
3          return
4      end
5      if support(s) = f then
6          insert s into S
7          return
8      end
9      C ← φ
10     Scan D once, find all the edges e such that s can be
       right-most extended to s.e; insert s.e into C and count
       its frequency
11     Sort C in DFS lexicographic order
12     foreach s.e in C do
13         if support (s) ≥ f then
14             gSpan-Modified (s.e, D, f, S)
15             return S
16         end
17     end
18 end
```

Details *gSpan-Modified* are represented in Algorithm 1. As gSpan, *gSpan-Modified* uses a sparse adjacency list representation to store graphs. Assume we have a label set {*A, B, C, ...*} for vertices, and {*a, b, c, ...*} for edges, the first round will discover all the subgraphs with frequency *f* containing an edge $A \overset{a}{-} A$. The second round will discover all the subgraphs with frequency *f* containing $A \overset{a}{-} B$, but not any $A \overset{a}{-} A$. gSpan- Modified test if s is a minimum DFS code pattern to eliminate duplicate subgraphs and their descendants (line 2-3). In fact, extensions are only allowed on minimal DFS codes to avoid conputation of duplicate graphs. We compute the frequency value of *s*, if it is equal to *f*, then gSpan-Modified stops the extension (line 7) and returns s as a substructure with the frequency *f* (line 6) and gSpan-Modified investigates other candidates for expansion (line 12). In this case, *s* is a subgraph with frequency *f*.

gSpan-Modified is recursively called to extend the graphs with a frequency higher than *f* and it stops either when the support of a new graph is less than *f* (line 13). In this case, *gSpan-Modified* stops searching without return *s* and without extending since

all super graph of a graph with frequency lower than f is also with frequency lower than *f*. If the frequency of s greater than *f*, we must extend *s*, and we make a recursive call to each super graph of *s*. This extension stops when we discover a model with a frequency equal to *f* (line 5-8), this model will be a minimum rare subgraph with a frequency *f* and it contains no a subgraph with this frequency value. In this way, all the substructures will be returned with a frequency *f* and they are minimal.

With the algorithme gSpan-Modified at hand, now we are able to compute graph subutructures for a database *D* with a specified frequency *f*. In addition, the computed subgraphs are of minimum size; ie, rare minimum substructures. Now, we are ready to ouline our proposal for computing the index using minimum rare substructures for a given graph database.

3.3 Illustration

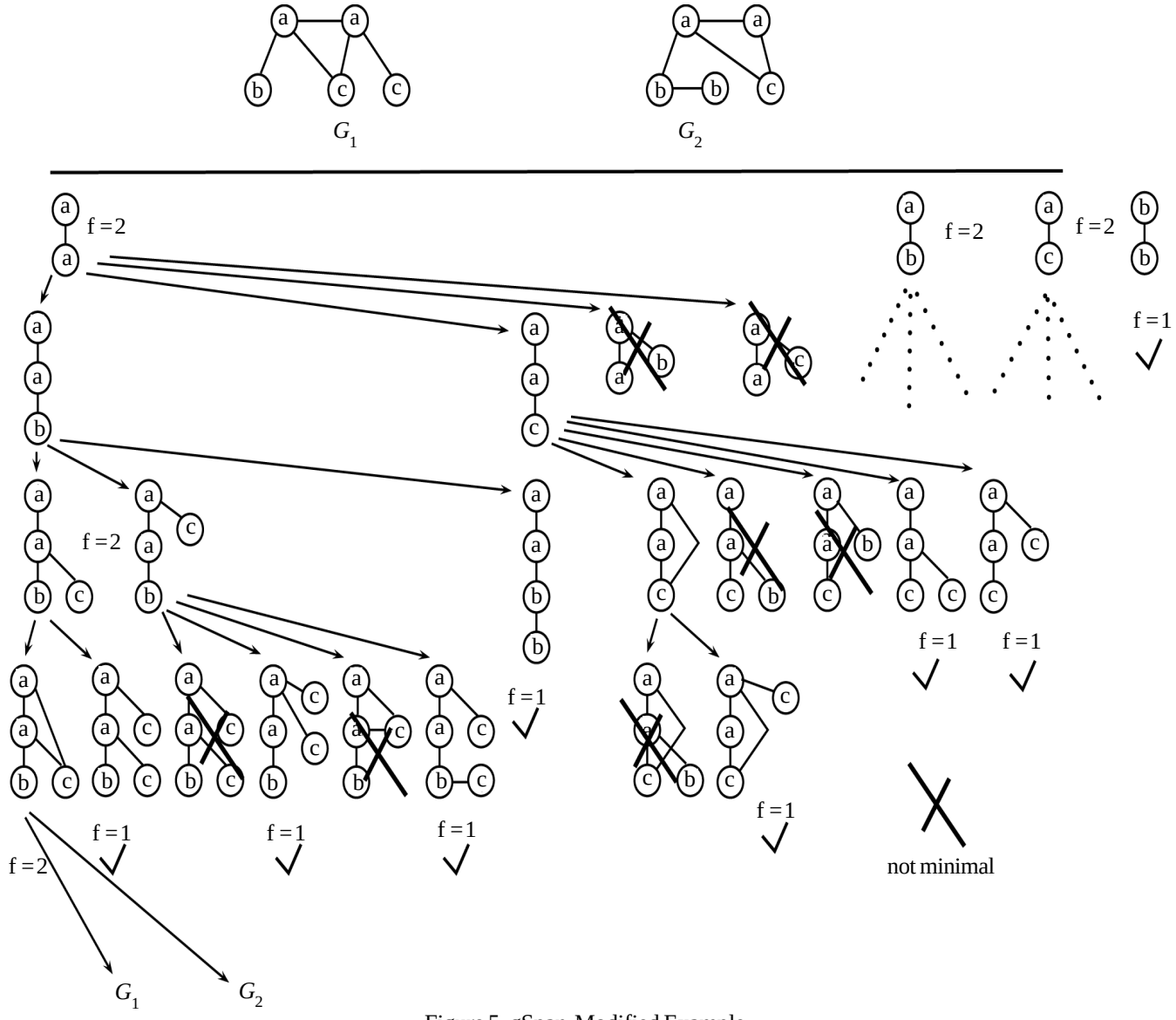

Figure 5. gSpan-Modified Example

4. Conclusion

In this paper, we introduced a new feature indexing for graph database and we proposed a new algorithm *gSpan- Modified* for extracting patterns with an exact frequency, based in effective modification of the algorithm gSpan. our algorithm guaranteed

firstly the extraction of with an exact frequency substructures. Second, each structure found is minimal, does not contain any subgraph with the same frequency. gSpan-Modified represents the best extraction method, actually, especially as gSpan is the most efficient approach for frequent subgraph mining. The use of gSpan-Modified for graph database indexing constitutes our immediate focus in our future research.

References

[1] Trinajstić, N. (1983). *Chemical graph theory*, ser. Chemical Graph Theory. CRC Press, 1 (1). [Online]. Available: http://books.google.tn/books?id=wuXvAAAAMAAJ.

[2] Cook, S. A. (1971). The complexity of theorem-proving procedures, *In*: Proceedings of the third annual ACM symposium on Theory of computing, ser. STOC '71. New York, NY, USA: ACM, p. 151– 158. [Online]. Available: http://doi.acm.org/10.1145/800157.805047.

[3] Giugno, R., Shasha, D. (2002). Graphgrep: A fast and universal method for querying graphs, *In*: ICPR (3), p. 112–115.

[4] Ferro, A., Giugno, R., Mongiovì, M., Pulvirenti, A., Skripin, D., Shasha, D. (2008). Graphfind: enhancing graph searching by low support data mining techniques, *BMC Bioinformatics*, 9 (S-4).

[5] Yan, X., Yu, P. S., Han, J. (2004). Graph indexing: a frequent structure-based approach, *In*: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, ser. SIGMOD '04. New York, NY, USA: ACM, p. 335–346. [Online]. Available: http://doi.acm.org/10.1145/1007568.1007607.

[6] Cheng, J., Ke, Y., Ng, W., Lu, A. (2007). Fg-index: towards verification-free query processing on graph databases, *In*: SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data. New York, NY, USA: ACM, 2007, p. 857–872. [Online]. Available: http://portal.acm.org/citation.cfm?id=1247480.1247574.

[7] Natale, R. D., Ferro, A., Giugno, R., Mongiovì, M., Pulvirenti, A., Shasha, D. (2010). Sing: Subgraph search in non-homogeneous graphs, *BMC Bioinformatics*, 11 (96).

[8] He, H., Singh, A. K. (2006). Closure-tree: An index structure for graph queries, *In*: ICDE, p. 38.

[9] Williams, D. W., Huan, J., Wang, W. (2007). Graph database indexing using structured graph decomposition, *Data Engineering, International Conference on*, 0, p. 976–985.

[10] Yan, X., Han, J. (2002). gspan: Graph-based substructure pattern mining, *In*: Proceedings of the 2002 IEEE International Conference on Data Mining, ser. ICDM '02. Washington, DC, USA: IEEE Computer Society, p. 721–. [Online]. Available: http://dl.acm.org/citation.cfm?id=844380.844811.

[11] Wörlein, M., Meinl, T., Fischer, I., Philippsen, M. (2005). A quantitative comparison of the subgraph miners mofa, gspan, ffsm, and gaston, *In*: Proceedings of the 9th European conference on Principles and Practice of Knowledge Discovery in Databases, ser. PKDD'05. Berlin, Heidelberg: Springer-Verlag, p. 392–403. [Online]. Available: http://dx.doi.org/10.1007/11564126_39.