Structure-Based Similarity Search with Graph Histograms *

Apostolos N. Papadopoulos and Yannis Manolopoulos
Data Engineering Lab.
Department of Informatics, Aristotle University
Thessaloniki 54006, Greece
email: $\{$*apostol,yannis*$\}$*@delab.csd.auth.gr*

Abstract

Objects like road networks, CAD/CAM components, electrical or electronic circuits, molecules, can be represented as graphs, in many modern applications. In this paper, we propose an efficient and effective graph manipulation technique that can be used in graph-based similarity search. Given a query graph $G_q(V,E)$, we would like to determine fast which are the graphs in the database that are similar to $G_q(V,E)$, with respect to a similarity measure. First, we study the similarity measure between two graphs. Then, we discuss graph representation techniques by means of multi-dimensional vectors. It is shown that no false dismissals are introduced by using the vector representation. Finally we illustrate some representative queries that can be handled by our approach, and present experimental results, based on the proposed graph similarity algorithm. The results show that considerable savings are obtained with respect to computational effort and I/O operations, in comparison to conventional searching techniques.

1 Introduction

Many modern applications handle objects that can be represented as graphs. Transportation applications need to manipulate road networks, CAD/CAM applications require the organization of electrical or electronic components, pattern recognition and computer vision applications require the classification of an unknown object, chemistry and molecular biology applications require the manipulation of molecules. In the aforementioned applications, as well as in much more, the objects are structural in nature and therefore can be considered as graphs. A graph $G(V,E)$ is composed of a set V of vertices or nodes and a set E of edges or lines. An edge connects two vertices. An example is illustrated in Figure 1.

* Supported by TMR CHOROCHRONOS (FMRX-CT96-0056)

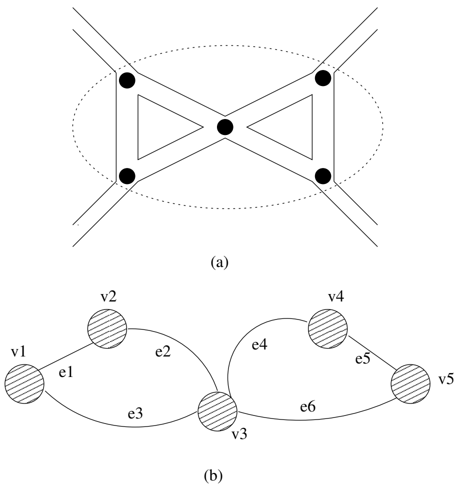

Figure 1. (a) Road network, (b) The graph representation.

In graph databases, two similarity search operations can be applied: whole-graph search, and sub-graph search. In the former, given a query graph $G_q(V,E)$ we require all similar graphs, assuming that each graph is an atomic object in the database. On the other hand, in sub-graph searching, given a query graph $G_q(V,E)$ we are interested also in identifying graphs that have sub-graphs similar to $G_q(V,E)$. In this paper, we focus on whole-graph search, since it is simpler than sub-graph search, and leads to more elegant solutions.

A number of graph classes have been identified and studied thoroughly in the literature [4]. Among them there are *simple graphs*, *pseudo-graphs* (with loops), *multi-graphs* (two or more edges connecting a pair of vertices), *directed graphs* (the edges have an orientation), *weighted graphs* (there is a weight associated with each edge). We assume that the underlying graph database consists of a number of *simple graphs*, *pseudo-graphs* and *multi-graphs*. Each graph may be *connected* or composed of a number of *connected components*.

The rest of the work is organized as follows: Section 2 presents related work and the motivation behind this work. Section 3 illustrates some graph similarity concepts. In Section 4, the vector representation of graphs is discussed and explained in detail. Section 5 presents similarity queries and experimental results. Finally, Section 6 concludes the paper and motivates for further research.

2 Related Research and Motivation

The problem of similarity between structured objects has been studied in the domain of structural pattern recognition and pattern analysis. In [10] the graph distance measures are grouped into two categories:

- *Feature-Based Distances:* a set of features is extracted from the structural representation, and these features are used as an n-d vector where the Euclidean distance can be applied.

- *Cost-Based Distances:* the distance between two objects measures the number of modifications required in order to transform the first object to the second.

Tsai and Fu worked on exploiting error-correcting codes of attributed relational graphs to express similarities [11, 12], in order to perform pattern analysis and syntactic pattern recognition. The concept of inexact graph matching has been also used by Bunke and Allermann in [3] and applied to structural pattern recognition. Sanfeliu and Fu [10] define a distance between attributed relational graphs (ARGs), based on a descriptive graph grammar. A recent discussion on discovering structural similarities can be found in [5].

The common characteristic of the aforementioned research efforts is that the emphasis is given in how fast the distance of two structured objects can be determined. Although this is extremely useful when the distance calculation is executed often, our viewpoint is database-oriented. We are interested in efficient algorithms and access methods exploitation, towards similarity search in large databases of structural objects.

The approach followed here is general and can be applied to every application that manipulates structural objects and requires structure-based similarity retrieval. Our primary objectives are:

- The extraction of representative features from the graph objects, which have a clear meaning to the user.

- The definition of a meaningful cost-based distance between graphs (which satisfies the metric space properties) and the calculation of this distance by means of the feature vectors.

- The study of structure-based similarity query processing algorithms by means of an experimental evaluation procedure.

The capability of searching the database by means of similarity-based algorithms is very advantageous. From one viewpoint, allows the retrieval of objects that are similar to a given query object. From another viewpoint, offers flexibility in query processing, since some objects may be subject to distortion and therefore the exact-match query is not sufficient for users' needs. Moreover, some data mining techniques are based on structural similarity concepts [5]. In conclusion, the efficient processing of structure-based similarity queries in large databases is very important towards satisfying the demands of modern applications.

3 Similarity Concepts

In some application domains, the concept of similarity is intuitive. For example, two color images are similar if they have a resemblance with respect to the color distributions, or two persons are similar if the share some common characteristics and differ in some others but not significantly. On the other hand, two entities are the same, if we can not distinguish between them. For example, we can not distinguish between two copies of the same image, or among a collection of identical pieces of paper.

The concept of similarity or dissimilarity is expressed by means of a distance function. If $\mathcal{D}$ is a domain, $dist : \mathcal{D} \times \mathcal{D} \to \mathcal{R}^+$ is a mapping function, and $x \in \mathcal{D}$, $y \in \mathcal{D}$ are two objects, then the similarity between them can be expressed by $dist(x, y)$. Very often, the function $dist$ is a metric, satisfying the three fundamental properties (metric space properties), non-negativity, symmetry and triangular inequality.

3.1 Graph Distance

Assume now that we are given two graph objects $G_1 \in \mathcal{G}$ and $G_2 \in \mathcal{G}$, where $\mathcal{G}$ is the graph domain. The problem is to define a similarity measure, in order to distinguish between them. A similar issue is to determine if the graphs are identical or not. In order to continue, we need to introduce the concept of *graph isomorphism*.

Definition 1
Two graphs $G_1(V_1, E_1) \in \mathcal{G}$ and $G_2(V_2, E_2) \in \mathcal{G}$ are *isomorphic* if there is a one-to-one correspondence between the nodes such that adjacency is preserved. More formally, G_1 and G_2 are isomorphic if there is a one-to-one function ϕ from V_1 onto V_2 such that $v_i v_j \in E_1$ if and only if $\phi(v_i)\phi(v_j) \in E_2$.

Although graph isomorphism is a concept that helps in graph comparison, it can be used only to determine if two graphs are isomorphic or not. However, we would like to have a distance function $dist_G(G_1, G_2)$, $G_1, G_2 \in \mathcal{G}$, expressing the degree of similarity between G_1 and G_2. The similarity between G_1 and G_2 can be expressed as the minimum number of primitive operations (structural modifications) that need to be applied to G_1, in order to be as close to G_2 as possible.

In Figure 2 three graphs G_1, G_2 and G_3 are illustrated. It is evident, that G_1 is "closer" to G_2 than to G_3. This is because we need to add just one edge to G_1 in order to get G_2, whereas we need one extra edge and one extra vertex in order to get G_3.

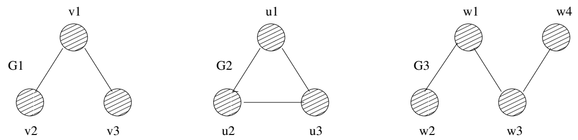

Figure 2. Similarity among graphs: G_1 is more similar to G_2 than G_3.

The primitive operations that we propose to perform on a graph are: vertex insertion, vertex deletion and vertex update. During vertex insertion, a new vertex v is added to the set of V of vertices. Vertex deletion signals the removal of a vertex from the set V of vertices. Finally, vertex update is the operation performed on a vertex v in order to insert (remove) an incident edge to (from) v. Using these primitive operations on the graphs of Figure 2, two primitive operations are required on G_1 to be matched with G_2 (two vertex updates), whereas we need three operations on G_1 to be matched with G_3 (one vertex insertion and two vertex updates). Therefore, under this similarity framework, we conclude that G_1 is more similar to G_2 than to G_3.

3.2 Distance Calculation

The problem arising is how can we calculate the value $dist_G(G_1, G_2)$, which expresses the similarity between the two graph objects G_1, G_2. In this respect, we introduce the concept of *graph histogram*. Given a graph $G(V, E)$, its histogram is constructed by calculating the degree of each vertex of the graph. Then, each vertex corresponds to a different histogram bin. If we sort the histogram bins in decreasing order, we obtain the *sorted graph histogram*. Also, we increment the histogram values by one (this will be justified later).

Lemma 1
Given a graph $G(V, E)$ and a vertex $v \in V$ with $deg(v) = 0$, the corresponding graph histogram bin has a value of 1.

Lemma 2
Given two graphs $G_1 \in \mathcal{G}$ and $G_2 \in \mathcal{G}$ with $histo(G_1)$ and $histo(G_2)$ two graph histograms, then $L_1(histo(G_1), histo(G_2))$ gives the number of primitive operations (not necessarily the minimum) required to transform G_1 in order to have the same order, size and degree sequence with G_2. More formally:

$$\boxed{L_1(histo(G_1), histo(G_2)) \geq dist_G(G_1, G_2)} \quad (1)$$

Lemma 2 suggests that determining the number of primitive operations that need to be applied to G_1 in order to have the same order, size and degree sequence with G_2, corresponds to the Manhattan distance of two arbitrary graph histograms of G_1 and G_2. However, this does not guarantee that the number of primitive operations is the minimum.

Theorem 1
Given two graphs $G_1 \in \mathcal{G}$ and $G_2 \in \mathcal{G}$ with $histo_s(G_1)$ and $histo_s(G_2)$ the corresponding sorted graph histograms, then $L_1(histo_s(G_1), histo_s(G_2))$ gives the minimum number of primitive operations required to transform G_1, in order to have the same order, size and degree sequence with G_2. More formally:

$$\boxed{L_1(histo_s(G_1), histo_s(G_2)) = dist_G(G_1, G_2)} \quad (2)$$

4 The Vector Representation of Graphs

If the size of the database is relatively small, then sequential search can be applied directly to the sorted graph histograms in order to determine similar objects. However, this technique causes performance degradation for large or very large databases. Therefore, additional auxiliary mechanisms need to be exploited. In this respect, the focus is to apply efficient indexing techniques towards fast similarity search in large graph databases. One possible solution is to use the sorted histograms to build a multi-dimensional index structure. Although the tools are available, there are two main disadvantages with this approach:

1. The order of the graphs is not constant. This means that very small graphs (5-10 vertices) as well as larger graphs (100-200 vertices) may be stored in the database. This causes a major problem in selecting the dimensionality of the index.

2. By selecting a sufficiently large dimensionality (e.g. 256 dimensions) the *dimensionality curse* is introduced, deteriorating the performance of similarity search [6].

In order to overcome the aforementioned difficulties, we propose to apply a transformation to the histograms and map each graph object to a vector, with reduced dimensionality. The maximum number of vertices is assumed to be known. Henceforth, we assume that a graph can have at most v vertices. Therefore, each graph can be considered as a vector in the v-d space. As stated previously, our goal is to reduce this dimensionality, in order to provide efficient query processing techniques.

Let $SH_1 = (x_1, x_2, ...x_v)$ and $SH_2 = (y_1, y_2, ..., y_v)$ be two sorted graph histograms. If a graph has less than v vertices, then zeros are padded to the end of the corresponding sorted histogram. As the following theorem states, if we combine neighboring histogram bins together, the distance is lower-bounded.

Theorem 2
If SH_1 and SH_2 are two sorted histograms with n bins each, and f is a positive integer number, such that n is divisible by f, then the following holds:

$$\boxed{L_1(SH_1, SH_2) \geq L_1(V_1, V_2)} \quad (3)$$

where:

$$V_1 = \left(\sum_{j=1}^{f} x_j, \sum_{j=f+1}^{2f} x_j, ..., \sum_{j=n-f}^{n} x_j \right)$$

$$V_2 = \left(\sum_{j=1}^{f} y_j, \sum_{j=f+1}^{2f} y_j, ..., \sum_{j=n-f}^{n} y_j \right)$$

Theorem 2 guarantees that during search in the feature space, there will be no false dismissals, and therefore no answer will be missed. However, false alarms are introduced which can be eliminated during the refinement step.

5 Performance Evaluation

After the transformation process from the histogram space to the feature space is completed, a multidimensional access method (e.g. R-tree) can be exploited in order to speed-up searching. The motivation behind the exploitation of an access method lies in the queries we like to perform against the underlying data. We focus on the following queries, that are characterized as *graph similarity queries*:

- *Range Query*: given a query graph $G_q(V_q, E_q)$ and a tolerance $\epsilon \geq 0$, we ask for all graph objects $G_x(V_x, E_x)$ in the database such that the distance from G_q to G_x is less than or equal to ϵ.
- *Nearest-Neighbor Query*: given a query graph $G_q(V_q, E_q)$ and an integer k, we ask for the k graph objects $G_x(V_x, E_x)$ which are closer to G_q than the other database objects.
- *Isomorphic Query*: given a query graph $G_q(V_q, E_q)$ we ask for all graphs in the database which are isomorphic to G_q.

We illustrate experimental results for the similarity range query only, due to lack of available space, and study the performance of similarity range queries with different parameter values. The results presented are average values of 100 queries, and each point query is selected from the database. The database is composed of 50000 random graphs. Each graph has between 10 and 256 vertices, and between 50 and 2000 edges. The disk page size is set to 4KBytes and the disk cluster size is set to 4 pages.

We assume that an R*-tree [1] is used to index the derived graph vectors. Other access methods could be used as well (e.g. X-tree [2]).

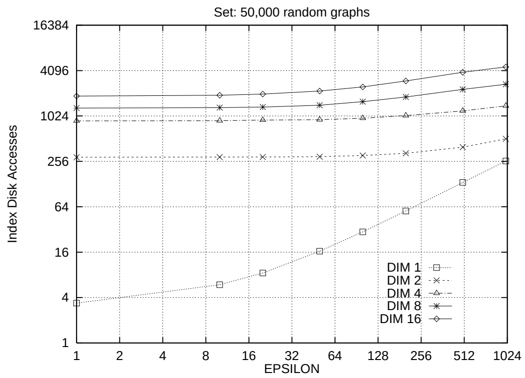

Figure 3. Number of disk accesses vs. ϵ.

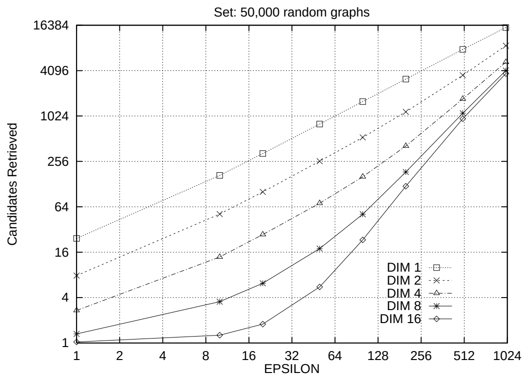

Figure 4. Number of candidate objects vs. ϵ.

Figure 3 illustrates the number of disk accesses vs. the value of the distance ϵ. We observe that by increasing the dimensionality of the histograms, more disk accesses are introduced. Using 1-D histograms, is more efficient than using 16-D histograms. This was anticipated, because more space is needed by the index, and more dead space is introduced increasing the number of disk accesses. Figure

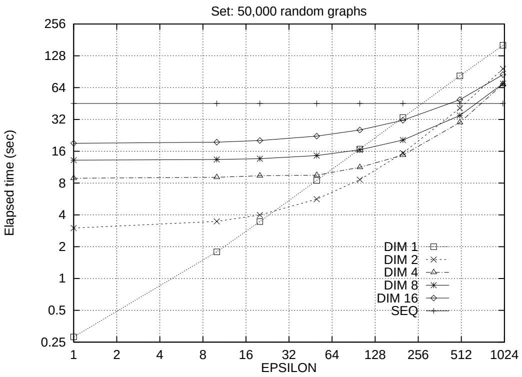

Figure 5. Total running time vs. ϵ.

4 shows the number of candidate objects retrieved during processing. For 1-D histograms we obtain a large number of candidates, whereas using 16-D histograms the number of candidates is considerably smaller. These candidates are the objects that satisfy the query, plus the false drops that are introduced. By increasing the number of dimensions, the number of objects is decreasing because there are less false drops. Figure 5 depicts the total elapsed time for similarity range queries, for dimensions 1 to 16. The cost of sequential search is also included. We observe that for small query ranges ($\epsilon < 16$), using 1-D histograms is sufficient. However, as the ϵ value increases, more dimensions are needed in order to guarantee efficiency. Using 2-D or 4-D histograms we obtain satisfactory results, for a realistic range of ϵ values. Recall that for very large values of ϵ the concept of similarity is degenerated, since many of the output graphs may not be so similar with respect to the input graph. The reason is that the ϵ value represents the number of structural modifications that need to be applied.

6 Concluding Remarks

In this paper, we studied similarity query processing in structural databases, where a large collection of graph objects is manipulated. The results of the experiment series have shown that the improvement in the search is considerable, in comparison to the sequential search of the database. Moreover, the similarity measures defined have a clear meaning to the user and therefore can be used for similarity search introducing only false alarms and never false dismissals. Future research may include: (a) the consideration of other graph classes, like weighted, directed and attributed relational graphs, (b) the investigation of subgraph similarity searching, which is a very challenging and interesting problem for large databases, and (c) he application and support of user-defined transformation mechanisms, where for example an edge insertion may be considered more costly than a vertex insertion.

References

[1] N. Beckmann, H.P. Kriegel and B. Seeger: "The R*-tree: an Efficient and Robust Method for Points and Rectangles", *Proceedings of the 1990 ACM SIGMOD Conference*, pp.322-331, Atlantic City, NJ, 1990.

[2] S. Berchtold, D. Keim and H.-P. Kriegel: "The X-tree: An Index Structure for High-Dimensional Data", *Proceedings of the 22nd VLDB Conference*, Bombay, India, 1996.

[3] H. Bunke and G. Allermann: "Inexact Graph Matching for Structural Pattern Recognition", *Pattern Recognition Letters*, vol.1, no.4, pp.245-253, 1983.

[4] G. Chartrand and O. Oellermann: *"Applied and Algorithmic Graph Theory"*, McGraw-Hill, 1993.

[5] S. Djoko, D.J. Cook and L.B. Holder: "An Empirical Study of Domain Knowledge and its Benefits to Structural Discovery", *IEEE Transactions on Knowledge and Data Engineering*, vol.9, no.4, July/August, 1997.

[6] C. Faloutsos: *"Searching Multimedia Databases by Content"*, Kluwer Academic Publishers, 1997

[7] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele and P. Yanker: "Query by Image and Video Content: the QBIC System", *IEEE Computer*, vol.28, no.9, pp.23-32, September 1995.

[8] A. Guttman: "R-trees: a Dynamic Index Structure for Spatial Searching", *Proceedings of the 1984 ACM SIGMOD Conference*, pp.47-57, Boston, MA, 1984.

[9] N. Roussopoulos, S. Kelley and F. Vincent: "Nearest Neighbor Queries", *Proceedings of the 1995 ACM SIGMOD Conference*, pp.71-79, San Jose, CA, 1995.

[10] A. Sanfeliu and K.-S. Fu: "A Distance Measure between Attributed Relational Graphs for Pattern Recognition", *IEEE Transactions on Systems, Man and Cybernetics*, vol.13, pp.353-362, 1983.

[11] W.H. Tsai and K.-S. Fu: "Error-Correcting Isomorphism of Attributed Relational Graphs for Pattern Analysis", *IEEE Transactions on Systems, Man and Cybernetics*, vol.9, pp.757-768, 1979.

[12] W.H. Tsai and K.-S. Fu: "Subgraph Error-Correcting Isomorphism for Syntactic Pattern Recognition", *IEEE Transactions on Systems, Man and Cybernetics*, vol.13, pp.48-62, 1983.