INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor : 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 02, February 2024, pp : 509-514

GRAPH MATCHING: FAST CANDIDATE ELIMINATION USING MACHINE LEARNING TECHNIQUES

Anupriya B[1]

[1]M. sc., Department of Computer Science, Fatima College, Madurai, India.

ABSTRACT

Graph matching is an important class of methods in pattern recognition. Typically, a graph representing an unknown pattern is matched with a database of models. If the database of model graphs is large, an additional factor in induced into the overall complexity of the matching process. Various techniques for reducing the influence of this additional factor have been described in the literature. In this paper we propose to extract simple features from a graph and use them to eliminate candidate graphs from the database. The most powerful set of features and a decision tree useful for candidate elimination are found by means of the C4.5 algorithm, which was originally proposed for inductive learning of classifification rules. Experimental results are reported demonstrating that efficient candidate elimination can be achieved by the proposed procedure.

Keywords: Structural pattern recognition, graph matching, graph isomorphism, database retrieval, database indexing, machine learning, C4.5

1. INTRODUCTION

In structural pattern recognition, graphs play an important role for pattern representation. Typically, objects or parts of objects are represented by means of nodes,while information about relations between diffffferent objects or parts of objects is captured through edges. When graphs are used for pattern representation, the recognition problems turn into the task of graph matching. I.e., a graph extracted from an unknown input object is matched to a database of model graphs to recognize or classify the unknown input [1,2,3]. Application examples of graph matching include character recognition [4,5], schematic diagram interpretation[6,7], shape analysis [8] and 3-D object recognition [9].Structural pattern recognition by means of graph matching is attractive because graphs are a universal representation formalism.But on the other hand,graph matching is expensive from the computational complexity point of view.Matching two graphs with each other requires time and space exponential in the number of nodes involved. In this paper we propose a new approach based on machine learning techniques. For the purpose of simplicity, only the problem of graph isomorphism detection is considered. The main idea of the proposed approach is to use simple features, which can be efficiently extracted from a graph, to reduce the number of possible candidates in the database.

A potential problem with this approach is that there may exist a large number of simple features.Hence the questions arises which of these features are most suitable to rule out as many candidates from the database as quickly as possible. To find the best feature set, we propose the application of machine learning techniques in this paper.

C4.5 ALGORITHM:

C4.5 is a program that generates decision trees.The reason for selecting C4.5 for this work is because it is one of the best known programs of this kind. C4.5 is based on the divide and conquer paradigm described in the following.Let S be the set of training instances and let the classes be C1, C2 , C3 ,..., Cn.

There are 3 cases:

- S contains one or more instances which all belong to a class Cj . The decision tree for S is a leaf identifying Cj .
- S is empty. The decision tree for S is a leaf, but the class associated with the leaf must be determined from information other than S. For example,the class is chosen based on background information of the domain such as overall majority class.
- S contains instances that belong to a mixture of classes. In this case, S is refined into subsets of instances that are or seem to be heading towards single-class collection of cases. A test T is chosen, based on a single attribute, that has one or more mutually exclusive outcomes O1, O2, O3, ... , Ol and S is partitioned into subsets S1, S2, S3, ... , Sl where Sj contains all the instances in S that have outcome Oj in the chosen set.

In the particular application considered in this paper, each graph in the database corresponds to exactly one class Ci, and S = C1 ∪ C2 ∪... ∪ Cn is the set of prototype graphs in the database.

Partitioning at each level in the decision tree is achieved by finding the attribute that maximizes the information gained from choosing that attribute. C4.5 uses a normalized gain criterion to select which attribute should be used. The definition of the gain ratio used by c4.5 is defined below:

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor : 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 02, February 2024, pp : 509-514

$$\text{gain-ratio}(X) = \frac{\text{gain}(X)}{\text{split-info}(X)} \quad , \quad \text{where}$$

$$\text{split-info}(X) = \sum_{i=1}^{l} \frac{|T_i|}{|T|} \times \log_2 \left(\frac{|T_i|}{|T|} \right) \quad ,$$

$$\text{gain}(X) = \text{info}(T) - \text{info}_x(T) \quad ,$$

$$\text{info}(T) = \sum_{j=1}^{l} \frac{\text{freq}(C_j, T)}{|T|} \times \log_2 \left(\frac{\text{freq}(C_j, T)}{|T|} \right) \quad ,$$

$$\text{info}_x(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \text{info}(T_i)$$

and T is the set of instances, split into l outcomes based on the test X with Cj being some random class. The frequency ratio – freq(Cj , T)/|T| – represents the probability that given the case in which a training instance is taken at random from the training set T, that training instance belongs to class Cj .

Attribute values can be discrete or real-valued and C4.5 can handle cases involving unknown or imprecise values. C4.5 always attempts to develop the simplest decision tree that is representative of the data and when necessary, it can apply sophisticated pruning techniques to reduce the size of the tree.

Graph Candidate Elimination Using C4.5:

Given an input graph g and a database with n model graphs g1, g2,...,gn where each gi represents a pattern class Ci, we consider the problem of finding a graph gj in the database that is isomorphic to g. Formally, g and gj are isomorphic if there exists a bijective mapping f from the nodes of g to the nodes of gj such that the structure of the edges and all node and edge labels are preserved under f. As described in the introduction, we are interested in a procedure that efficiently reduces the number of potential candidate graphs in the database. Our method for graph candidate elimination has three stages. In the first stage we extract the features from the graphs. There are many features that can be potentially used, ranging from simple features such as the number of vertices or edges per graph to complex ones such as chains of vertices and edges with given labels. For our research we have selected the three types of features listed below

- number of vertices with a given label (feature–type 1).
- number of incoming edges per vertex (feature–type 2).
- number of outgoing edges per vertex (feature–type 3).

For a graph with m vertices, the extraction takes only O(m2) time and space. On the other hand, as will be shown later, these features are very efficient in ruling out potential candidate graphs. For example, if the input graph has three vertices of a given label A, then the search for isomorphic graphs can be restricted only to those graphs in the database that have exactly three vertices with the label A.

The second stage involves the use of C4.5 to build the decision tree based on any combination of the three types of features mentioned above.

The third stage is to determine which graphs can possibly match a given input graph. Obviously, a necessary condition for input graph g being isomorphic to model graph gj is that all features extracted from g have values identical to the corresponding features extracted from gj . The processing is as follows. First,we extract from the input graph the same features that were extracted from the graphs in the database. Then we use the values of the features extracted from the input graph to traverse the decision tree. There are only two possible outcomes.The first outcome is that we reach a leaf node. In this case the graphs associated with the leaf node are possible matches to the input graph. Each of these graphsis

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor : 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 02, February 2024, pp : 509-514

tested against the input for graph isomorphism using a conventional algorithm.The second outcome is that we do not reach a leaf node. In this case there are no graphs in the database that can be isomorphic to the input graph.

The cost to extract any of the features mentioned above from a graph with m vertices is O(m2). The cost to traverse a decision tree of depth k is O(k) while the cost to match a given input graph against a set of c candidate graphs is O(c · mm). Hence, for a database consisting of n graphs of size m each, the cost to find a match for an input graph is [O(c ·m+O(m2) +O(k)] given that the decision tree has k levels and the maximum number of graphs associated with a leaf in the decision tree is c. The graphs associated with a leaf node in the decision tree will be called a cluster in the following. We observe that the expression

$$O(c \cdot mm) + O(m2) + O(k) \quad (1)$$

has to be compared to

$$O(n \cdot mm) \quad (2)$$

which is the computational complexity of the straightforward approach, where we match the input graph to each element in the database. Because c<n or c << n, a significant speedup over the straightforward approach can be expected.

2. EXPERIMENTS RESULTS

The two parameters that are related to decision trees and have a significant impact on the overall efficiency of the method are the cluster size c and the depth of the decision tree k.

The experiments were conducted as follows. First a database of random graphs was generated. Then the features were extracted from the graphs and passed on to C4.5 which constructed a decision tree to classify the graphs in the database. In the final step of the experiment, the decision tree was analyzed and the average cluster and largest cluster size recorded along with the depth of the decision tree.

In the first set of experiments we focused on the first type of feature extracted from the graphs, i.e. the frequency of the various vertex labels in the graphs. Therefore the decision tree classifying the graphs in the database was built using information from only this type of feature. To extract the feature we used a simple histogram that recorded the occurrence of each given vertex label. We started with only 5 vertex labels and then gradually increased the number of labels to 100. We also varied the number of vertices per graph from 5 to 50. At the same time, the

number of edges was varied from 8 to 480. The number of graphs in the database was 1000 in this experiment. All graphs in the database had the same number of nodes and edges. The results indicate that as the number of vertices increases, the average cluster size decreases. In the majority of the experiments, the average cluster size converged to the value of 2 which means that in general only 2 graphs out

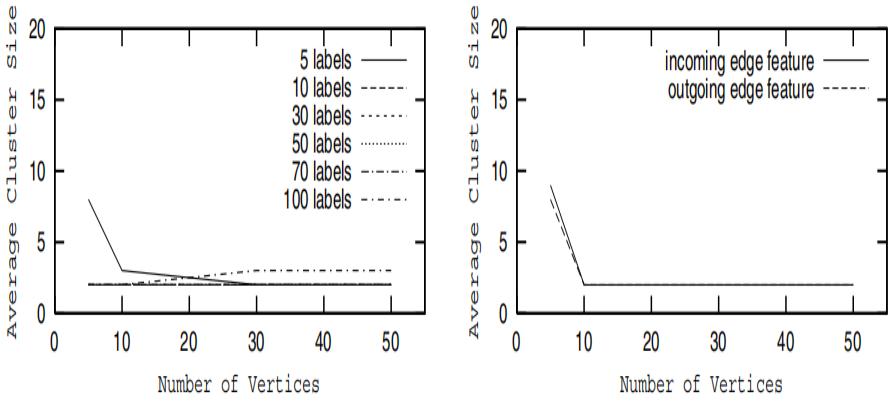

Fig. 1. (left) Using the vertex feature (1000 graphs).

Fig. 2. (right) Using the incoming/outgoing edge features (1000 graphs).

of a 1000 will be needed to conduct a test for graph isomorphism. The fact that the size of the cluster decreases as the number of vertices or labels increases was expected since the greater the number of vertices/labels per graph is, the more easily identifiable a graph becomes.

The second set of experiments were done using the frequency of edges incident per vertex. Both incoming and outgoing edges were considered in separate experiments. Again, the number of vertices was increased from 5 to 50. At the same time, the number of edges was varied from 8 to 480. The database's size was 1000. No label information was used in these experiments. Figure 2 shows the average cluster size for both the case in which the incoming and outgoing edges

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor : 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 02, February 2024, pp : 509-514

were used. The results are similar to those obtained from the case in which feature–type 1 was used, in the sense that the greater the number of vertices, the smaller the cluster size.

In the third set of experiments, we used pairs of features to build the decision tree classifying the graphs in the database. Three pairs of features were used: (feature–type 1,feature–type 2),(feature–type 1,feature–type 3)and (feature–type 2,feature–type 3). We used the same parameters as for feature–type 1: 1000 graphs per database, 5-50 nodes per graph with all graphs in the database having the same size, and 5-100 vertex labels. The results obtained from combining feature–type 1 and feature–type 2 are shown in Fig. 3. The average cluster size has value of 2 for all cases except when the graphs were of size 5 (the average cluster for this case was 3). These results are better than those in the case in which any of the three single features were used for classification.

In the fourth set of experiments, we combined all three types of features. The results obtained are shown in Fig. 4.

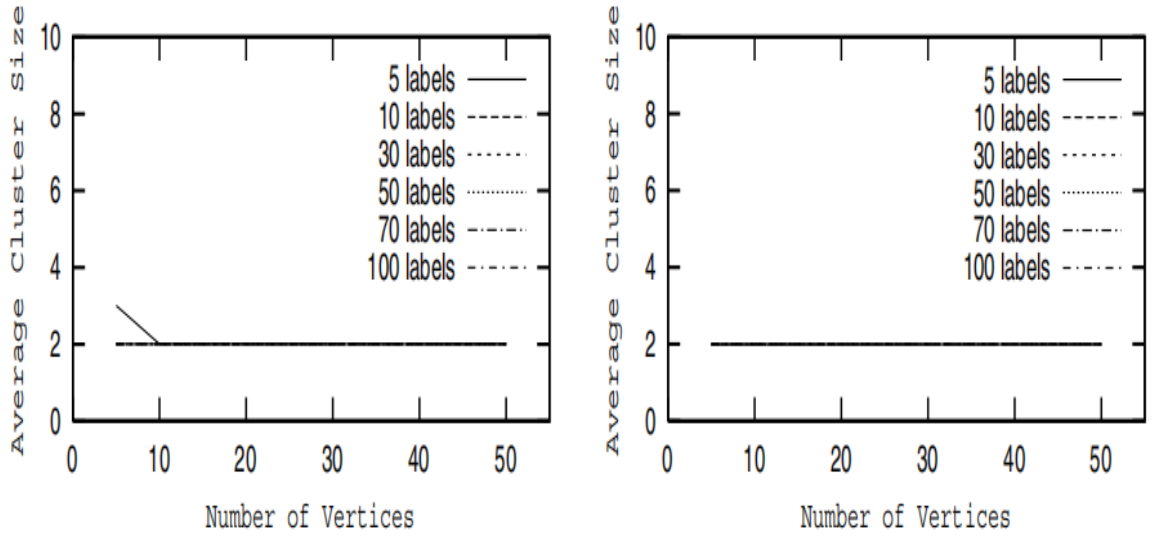

Fig. 3. (left) Using the feature combination 1-2 (1000 graphs).

Fig. 4. (right) Using the feature combination 1-2-3 (1000 graphs).

Note that this time the average cluster size had a constant value of 2 regardless of the size of the graphs. Therefore the results are better than those obtained for the cases in which single features or pairs of features were used by C4.5 to build the decision tree. This result matches our expectation since more information is

available and this leads to more accurate decision trees being built.

For each of the experiments described previously, we not only recorded the cluster size but also the depth of the decision tree. When feature–type 1 was used, the depth of the decision tree varied as shown in Fig. 5. When feature–type 2 and feature–type 3 were used, the depth of the decision trees was more constant and not influenced by the number of vertices in the graphs (see Fig. 6). The average decision tree depth is 11, which is the same as in the case of feature–type 1.

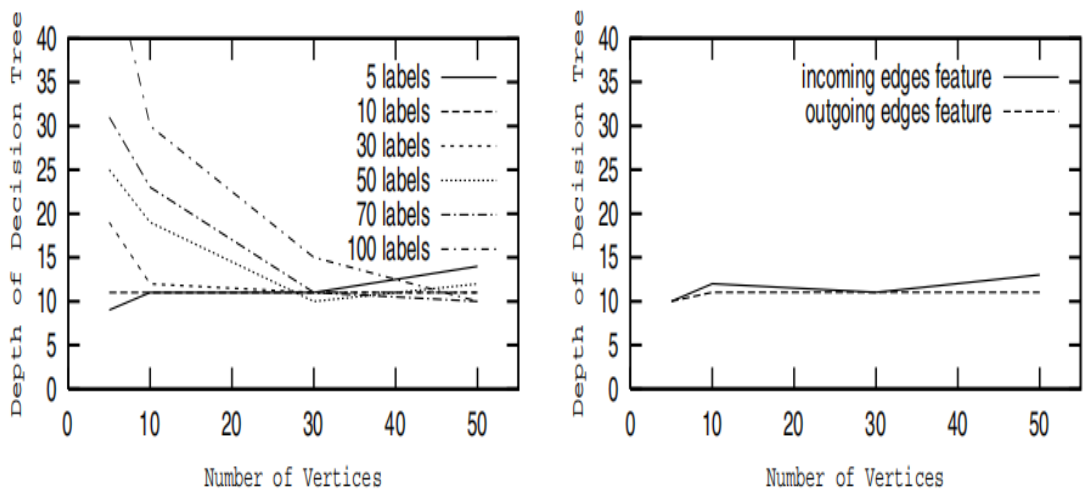

Fig. 5. (left) Using the vertex feature (1000 graphs).

Fig. 6. (right) Using the incoming/outgoing edge features (1000 graphs).

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

e-ISSN : 2583-1062

Impact Factor : 5.725

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 02, February 2024, pp : 509-514

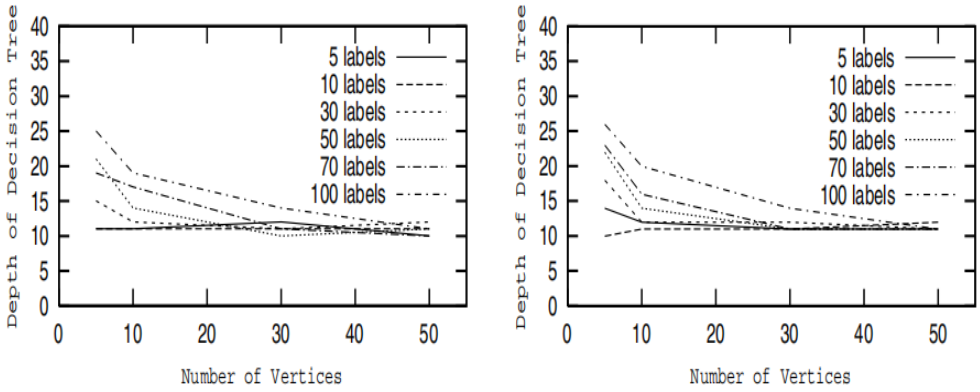

Fig. 7. (left) Using the feature combination 1-2 (1000 graphs).

Fig. 8. (right) Using the feature combination 1-3 (1000 graphs).

The results obtained from using pairs of features are shown in Figs. 7, 8 and 9. In all three cases there is a substantial improvement when compared with results obtained using feature–type 1. The depth of the tree converges faster to an average value of 11 and in the worst case (small number of vertices, high number of labels) the decision tree depth is only 25. The results also show that there were no substantial differences between the three pairs of features.

More improvement can be obtained by using more types of features. This is indicated by the results produced when using feature–types 1–3 together (see Fig. 10).

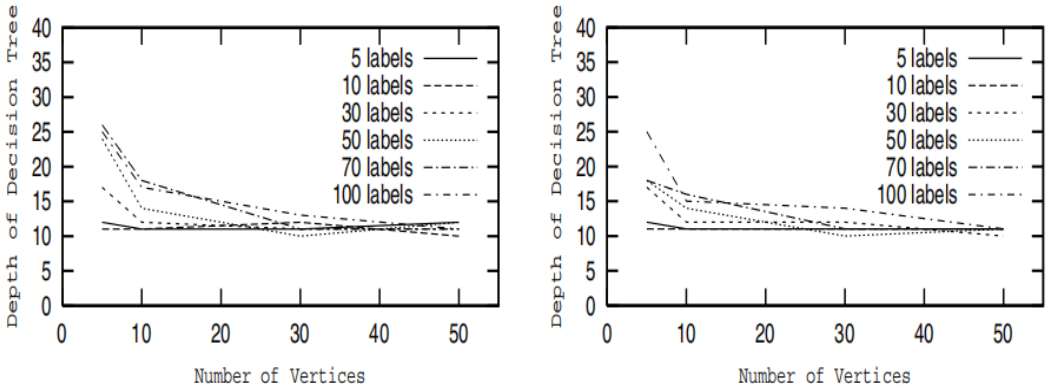

Fig. 9. (left) Using the feature combination 2-3 (1000 graphs).

Fig. 10. (right) Using the feature combination 1-2-3 (1000 graphs).

On a standard workstation, it takes about 20s to match an input graph with a database of 1000 graphs using an implementation of Ullman's algorithm [14]. Traversing a decision tree of depth 11 takes 0.3s, and feature extraction takes 0.08s. Thus if we neglect the time needed off-line for decision tree construction, only 0.4s are needed by our method as compared to 20s for a full search. Because of the exponential complexity of graph matching, a higher speedup can be expected for larger graphs.

3. CONCLUSION

In this paper we have presented a novel approach to graph matching that involves the use of decision trees. The method offers the advantages that it significantly reduces the number of candidate graphs in the database and it requires features that are easy to extract. It involves building a decision tree that classifies the graphs along selected features and, in order to find a match for a given graph one simply needs to go down the decision tree.

The higher the number of vertices per graph, the smaller is the average cluster size. Also the higher the number of vertex labels, the smaller is the average cluster size. Another way of keeping the size of the clusters minimal is to use combinations of features. Combinations of two and three features generally produce increasingly better results.

Future work will investigate if this method can be applied to the subgraph isomorphism and the approximate graph matching problems. Future work could also involve the use of more complex features for classification. The features used for the work described in this paper are simple and easy to extract. However, there are other types of features such as vertex-edge chains that could be used to build the decision trees. Such features are more complex and harder to extract from graphs, but they offer the benefit that they provide a more unique description of each graph in the database.

4. REFERENCES

[1] L.G. Shapiro and R.M. Haralick. Structural descriptions and inexact matching. IEEE Trans. Pattern Analysis and Machine Intelligence, volume 3, pages 504–519, 1981.

[2] A. Sanfeliu and K.S. Fu. A distance measure between attributed relational graphs for pattern recognition. In IEEE Trans. Systems, Man, and Cybernetics, volume 13, pages 353–363, 1983.

[3] B. Messmer and H. Bunke. A new algorithm for error–tolerant subgraph iso morphism detection. In IEEE Trans. Pattern Analysis and Machine Intelligence, volume 20, pages 493–505, 1998.

[4] S.W. Lu, Y. Ren, and C.Y. Suen. Hierarchical attributed graph represenation and recognition of handwritten Chinese characters. In Pattern Recognition, volume 24, pages 617–632, 1991.

[5] J. Rocha and T. Pavlidis. A shape analysis model with applications to a character recognition system. In IEEE Trans. Pattern Analysis and Machine Intelligence, volume 16, pages 393–404, 1994.

[6] S.W. Lee, J.H. Kim, and F.C.A Groen. Translation–, rotation–, and scale–invariant recognition of hand–drawn symbols in schematic diagrams. In Int'l J. Pattern Recognition and Artificial Intelligence, volume 4, no. 1, pages 1–15, 1990.

[7] H. Bunke, B. Messmer. Clustering and error–correcting matching of graphs for learning and recognition of symbols in engineering drawings. J. Hull and S. Taylor (eds.): Document Analysis Systems II, pages 102–117. World Scientific, 1998.

[8] A. Pearce, T. Caelli, and W.F. Bischof. Rule graphs for graph matching in pattern recognition. In Pattern Recognition, volume 27, no. 9, pages 1231–1246, 1994. lysis. In IEEE Trans. Pattern Analysis and Machine Intelligence, volume 3, pages 595–602, 1982.