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

www.ijprems.com
editor@ijprems.com

Vol. 03, Issue 12, December 2023, pp : 304-308

e-ISSN : 2583-1062

Impact Factor : 5.725

CANDIDATE ELIMINATION IN GRAPH MATCHING WITH MACHINE LEARNING

K. Priyadharshini[1]

[1]Computer Science, Fatima college, India.

ABSTRACT

Graph matching, a crucial approach in pattern recognition, involves comparing a graph representing an unknown pattern with a database of models. The complexity of this process increases when dealing with a large model graph database. Existing literature explores techniques to mitigate this complexity. This paper suggests a strategy of extracting basic graph features to eliminate potential candidate graphs from the database. The C4.5 algorithm is employed to identify the most influential set of features and construct a decision tree for effective candidate elimination. Experimental results validate the efficiency of this proposed method.

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

1. INTRODUCTION

Graph matching, a crucial approach in pattern recognition, involves comparing a graph representing an unknown pattern with a database of models. The complexity of this process increases when dealing with a large model graph database. Existing literature explores techniques to mitigate this complexity. This paper suggests a strategy of extracting basic graph features to eliminate potential candidate graphs from the database. The C4.5 algorithm is employed to identify the most influential set of features and construct a decision tree for effective candidate elimination. Experimental results validate the efficiency of this proposed method. In the realm of structural pattern recognition, graphs serve a pivotal role in representing patterns. Objects or their components are denoted by nodes, and relationships between these entities are conveyed through edges, enabling the explicit representation of structural connections. This leads to the recognition challenge of graph matching, where a graph extracted from an unknown input is compared with a database of model graphs for identification or classification [1,2,3]. Applications encompass character recognition [4,5], schematic diagram interpretation [6,7], shape analysis [8], and 3-D object recognition [9].

While graph matching is appealing for its universal representation, it poses computational challenges due to its exponential time and space complexity. Various indexing mechanisms have been proposed to tackle this complexity in large databases [3,12,10,11]. This paper introduces a novel approach leveraging machine learning techniques, specifically addressing the problem of graph isomorphism detection for simplicity. The proposed method revolves around using easily extractable features, such as the number of nodes or edges, their labeled counts, and edge incidences. Identical values of these features in the input graph and a candidate from the database are essential for isomorphism. Consequently, quick tests using these simple features can exclude certain candidates, reducing the set requiring expensive isomorphism tests. However, the abundance of simple features raises the question of which are most effective for rapid candidate exclusion. To address this, machine learning techniques, particularly the C4.5 algorithm [13], are applied. The subsequent section provides a brief introduction to C4.5, followed by its application to reduce the number of candidate graphs in a database. Section 4 presents experimental results, with conclusions drawn in Section 5.

2. INTRODUCTION

C4.5, a decision tree generation program [13], is chosen for its renowned capabilities in this domain. Operating on the divide and conquer paradigm, let's consider the set of training instances denoted as S, with classes C1, C2, C3, ..., Cn. Three scenarios arise:

- If S contains instances all belonging to a class C_j, the decision tree for S is a leaf specifically identifying C_j.
- If S is empty, the decision tree for S becomes a leaf. However, the associated class must be determined from sources beyond S, such as domain-specific background information or the overall majority class.

When **s** comprises instances spanning multiple classes, it undergoes refinement into subsets heading towards single-class collections. A test T, based on a single attribute with mutually exclusive outcomes O1, O2, O3, ..., On, is chosen. S is then partitioned into subsets S1, S2, S3, ..., S_l, where S_j contains instances with outcome On. The decision tree for **s** now incorporates a decision node identifying the test, with one branch for each possible outcome. This recursive process is applied to each subset, constructing decision trees tailored to their respective subsets of training instances.

The definition of the gain ratio used by C4.5 is defined below:

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

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

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

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

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

3. GRAPH CANDIDATE ELIMINATION USING C4.5:

Given an input graph g and a database with n model graphs 1,2,...,g1,g2,...,g n, where each gi represents a pattern class Ci, we aim to identify a graph g j in the database that is isomorphic to g. Formally, g and g j are isomorphic if a bijective mapping f exists from the nodes of g to the nodes of g j, preserving the structure of edges and all node and edge labels under f. In the context of a given input graph g and a database containing n model graphs g1, g2, ..., g n, each representing a pattern class Ci, our objective is to identify a graph g j within the database that is isomorphic to g. Graphs g and g j are considered isomorphic if there exists a bijective mapping f from the nodes of g to the nodes of g j, preserving the structure of edges and all node and edge labels. Our focus is on developing an efficient procedure to reduce the number of potential candidate graphs in the database.

The graph candidate elimination method comprises three stages. In the initial stage, we extract features from the graphs, including the number of vertices with a given label (feature-type 1), the number of incoming edges per vertex (feature-type 2), and the number of outgoing edges per vertex (feature-type 3).

The choice of these features is based on their ease of extraction, requiring only O(m^2) time and space for a graph with m vertices. Despite their simplicity, these features efficiently narrow down potential candidate graphs. For instance, if the input graph has three vertices label A, the search for isomorphic graphs in the database focuses solely on those with exactly three A-label vertices. Once extracted, these features are fed into C4.5. In the second stage, C4.5 constructs a decision tree using various combinations of the three feature types. The third stage involves determining candidate graphs matching the input graph.

This process entails extracting the input graph's features, traversing the decision tree, and either reaching a leaf node, indicating potential matches, or not reaching one, signal the absence of isomorphic graphs in the database.

The extraction cost for features from a graph with m vertices is O(m^2), traversing a decision tree of depth k costs O(k), and matching an input graph against c candidate graphs is O (c * m^2). For a database with n graphs, each of size m, finding a match for an input graph incurs a cost of O (c * m^2 + m^2 + k), where the decision tree has k levels, and each leaf in the tree is associated with a cluster of c graphs.

O (c · mm) + O(m2) + O(k) (1)

has to be compared to O (n · mm) (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.

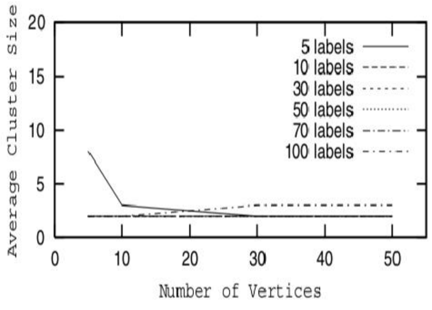
4. EXPERIMENTAL RESULTS

The efficiency of decision trees in our study is significantly influenced by two parameters: the cluster size (c) and the depth of the decision tree (k). Through various experiments conducted on a database of randomly generated graphs, we explored these parameters, considering factors such as the number of vertices, edges per graph, and the overall number of graphs in the database.

To maintain consistency, we omitted edge labels, assuming their influence would be similar to node labels. The experimental process involved generating a database of random graphs, extracting features, and utilizing C4.5 to construct decision trees for graph classification. In the final step, we analyzed the decision trees, recording average and largest cluster sizes, along with the depth of the constructed trees.

In the initial set of experiments, our focus was on the first category of features derived from graphs, specifically the frequency of various vertex labels. The decision tree used to classify the graphs in the database was constructed solely based on information from this feature type. We employed a straightforward histogram to capture the occurrence of each vertex label. Starting with only 5 labels, we progressively increased the count to 100. Simultaneously, we varied the number of vertices per graph from 5 to 50, and the edges ranged from 8 to 480.

This experiment involved a database of 1000 graphs, each having an identical number of nodes and edges. Figure 1 illustrates the average cluster size, revealing that as the number of vertices increases, the average cluster size tends to decrease.



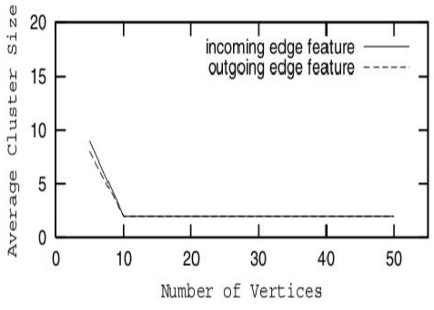
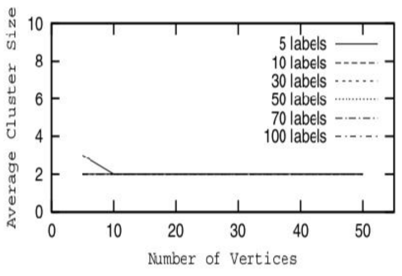


Fig. 1. (left) Using the vertex feature (1000 graphs).

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

In most experiments, the average cluster size consistently reached a value of 2. This implies that, on average, only 2 graphs out of 1000 are required for conducting a test on graph isomorphism. As anticipated, the cluster size diminishes with an increase in the number of vertices or labels. This expected outcome aligns with the notion that a higher number of vertices or labels per graph enhances graph distinctiveness. Consequently, C4.5 demonstrates the ability to generate improved classification trees with a reduced cluster size.

The second series of experiments explored vertex incidence frequency, considering both incoming and outgoing edges separately. The study involved increasing the number of vertices from 5 to 50, with edge variations ranging from 8 to 480. The database comprised 1000 instances, and no label information was utilized. Figure 2 illustrates the average cluster size for scenarios involving both incoming and outgoing edges. The findings parallel those observed in the first feature-type case, indicating that as the vertex count increases, the cluster size tends to decrease.



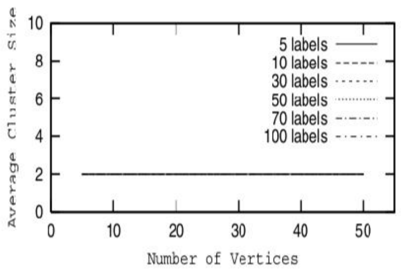
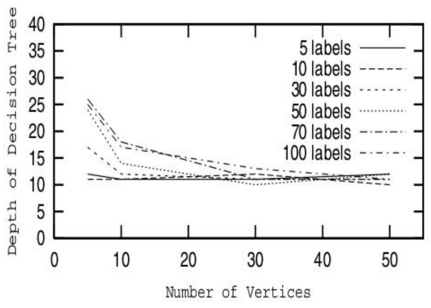


Fig. 3. (left) Using the feature combination 1-2 (1000 graphs).

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

In the third set of experiments, we employed pairs of features to construct a decision tree for classifying graphs in the database. Specifically, three pairs of features were utilized: (feature–type 1, feature–type 2), (feature–type 1, feature–type 3), and (feature–type 2, feature–type 3). Parameters consistent with feature–type 1 were used, involving 1000 graphs per database, 5-50 nodes per graph with uniform size across all graphs, and 5-100 vertex labels. The results from combining feature–type 1 and feature–type 2 are depicted in Fig. 3, revealing an average cluster size of 2 for most cases, except when dealing with graphs of size 5, where the average cluster size was 3.

These outcomes surpassed those obtained when any of the three individual features were employed for classification. The incorporation of two features provided additional information, enabling C4.5 to achieve more accurate graph classification.



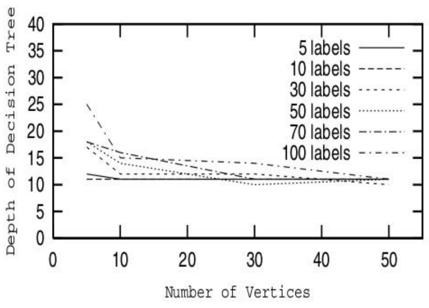
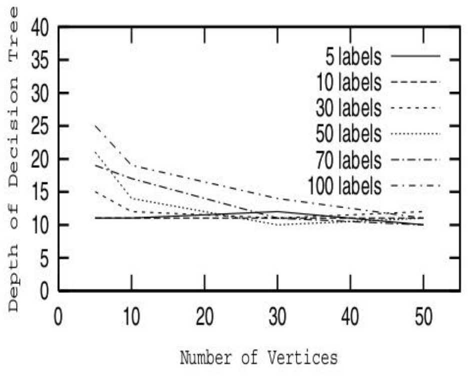


Fig: 5 **Fig: 6**

Results for the last two pairs of features, (feature–type 1, feature–type 3) and (feature–type 2, feature–type 3), exhibited similar outcomes. In both cases, the average cluster size was 2, and the results surpassed those achieved using a single feature for graph classification. In the fourth set of experiments, we combined all three types of features, and the results are presented in Fig. 4."



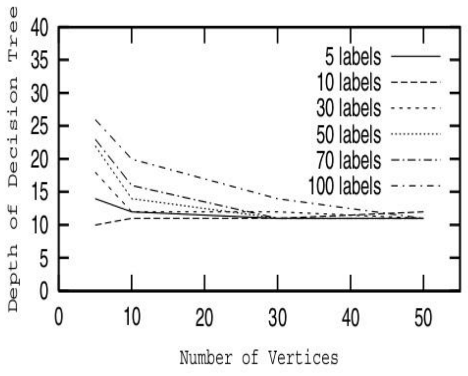


Fig :7 **Fig :8**

Figures 7,8, illustrate the outcomes derived from utilizing feature pairs. In each instance, a significant enhancement is evident when compared to outcomes employing feature–type 1. The tree depth converges more rapidly to an average value of 11, and even in the most challenging scenario (few vertices, numerous labels), the decision tree depth is merely 25. Combining features not only aids in diminishing average cluster size but also contributes to reducing tree depth. Additionally, the results indicate negligible differences among the three pairs of features. Greater enhancement can be achieved through the incorporation of additional feature types, as evidenced by the outcomes when employing feature-types 1–3 collectively (refer to Fig. 8). In this scenario, the tree depth converges more swiftly to an average of 11. Furthermore, with the utilization of more features, the tree depth in the most challenging scenario (few vertices, high number of labels) is below 25, marking a nearly 50% reduction compared to the case involving feature-type 1.

To provide tangible insights into the potential time savings facilitated by the method proposed in this paper, we conducted measurements on quantities in equations (1) and (2) using a set of graphs, each with 20 vertices. Employing Ullman's algorithm [14] on a standard workstation, matching an input graph with a database of 1000 graphs takes approximately 20 seconds. Meanwhile, traversing a decision tree of depth 11 consumes 0.3 seconds, and feature extraction requires 0.08 seconds. Excluding the offline time for decision tree construction, our method demands only 0.4 seconds compared to the 20 seconds required for a full search. Given the exponential complexity of graph matching, larger speedup gains can be anticipated for more extensive graphs.

5. CONCLUSION

This paper introduces an innovative graph matching approach utilizing decision trees, offering distinct advantages such as a significant reduction in the number of candidate graphs in the database and the requirement for easily extractable features. The method entails constructing a decision tree that categorizes graphs based on selected features.

To find a match for a given graph, one navigates down the decision tree. In the best-case scenario, the leaf node contains just one graph, while in the worst case, it may include several. The method's complexity is governed by three parameters: the cluster size associated with the leaf node, the graph size, and the depth of the decision tree.

We conducted numerous experiments to explore the average cluster size at the leaf node and decision tree depth using a 1000-graph database. The graphs had 5 to 50 vertices with 5 to 100 vertex labels. Results indicate that the average cluster size is influenced by the number of vertices and vertex labels; larger graphs and more labels result in smaller clusters. Combinations of two or three features yield better results in minimizing cluster size.

Our proposed method demonstrates significant computation time savings for graph matching, with fast feature extraction and decision tree traversal effectively narrowing down candidates for expensive isomorphism tests.

The approach is tailored for graph isomorphism and future research will explore its applicability to subgraph isomorphism and approximate graph matching. Additionally, investigating more complex features like vertex-edge chains is part of our future work to enhance classification accuracy.

6. REFERENCES

[1] L.G. Shapiro and R.M. Haralick. Structural descriptions and inexact matching. IEEE Trans. Pattern Analysis and Machine Intelligence.

[2] A. Sanfeliu and K.S. Fu. A distance measure between attributed relational graphs for pattern recognition. In IEEE Trans. Systems, Man, and Cybernetics.

[3] B. Messmer and H. Bunke. A new algorithm for error–tolerant subgraph isomorphism detection. In IEEE Trans. Pattern Analysis and Machine Intelligence.

[4] S.W. Lu, Y. Ren, and C.Y. Suen. Hierarchical attributed graph Represenation and recognition of handwritten Chinese characters. In Pattern Recognition.

[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.

[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.

[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.

[9] E.K. Wong. Model matching in robot vision by subgraph isomorphism. In Pattern Recognition, volume 25, no. 3, pages 287–304, 1992.

[10] L.G. Shapiro and R.M. Haralick. Organization of relational models for scene analysis. In IEEE Trans. Pattern Analysis and Machine Intelligence.

[11] K. Sengupta and K.L. Boyer. Organizing large structural model bases. In IEEE Trans. Pattern Analysis and Machine Intelligence.

[12] B. Messmer and H. Bunke. A decision tree approach to graph and subgraph isomorphism. In Pattern Recognition.

[13] J. R. Quinlan. C4.5 : Programs for machine learning. Morgan Kaufmann Publishers, 1993.