GRAPH MATCHING ALGORITHM-THROUGH ISOMORPHISM DETECTION

L. Ravi, K. Sujitha
Department of Computer Science, Sacred Heart College (Autonomous),
Tirupattur, Vellore, Tamil Nadu, India

Abstract- This Paper delivers a new algorithm to the problem of graph isomorphism detection. Basically this method is designed in such a way that, a model graph which is known prior is compared with an unknown graph called input graph. Now the problem to be solved is to find a graph isomorphism from the input graph which is given online with the one which is known as model graph. At dynamic time the input graph is compared with the entire model graph for which there exists a graph isomorphism from the input graph are detected. The time complexity depends on the number of input graphs given and size of the graphs. Furthermore it is independent of number of model graphs given and the number of edges in any of the graph.

Keywords- Graph matching, graph isomorphism, subgraph isomorphism.

I. INTRODUCTION

Graphs are powerful and universal data structure used in various sub fields of science and engineering. A graph consists of set of vertices and a set of edges. A graph may be directed or undirected. When graphs are used for representation, the problem arises in comparing two or more objects. The similarity between two graphs is called as isomorphism. Hence this paper deals with the problem of graph isomorphism detection. So a new approach for solving this graph isomorphism in polynomial time is under research. This research provides a similar small approach to solve the problem in polynomial time. In many applications a crucial operation is the comparison between two objects or between an object and a model to which the object could be related. When structured information is represented by graphs this comparison is performed using some form of graph matching.

Graph matching is the process of finding a correspondence between the nodes and the edges of two graphs that satisfies some (more or less stringent) constraints ensuring that similar substructures in one graph is mapped to similar substructures in the other. Although graph theory is one of the younger branches of mathematics, it is fundamental to a number of applied fields, including operations research and computer science. Here, an implementation of the paper work is based on the graph isomorphism on a directed graph.

Enumeration algorithm with refinement procedure is used to find the subgraph isomorphism techniques are specified in [1]. An Algorithm gives the near optimum solution to the weighted undirected graph matching problem is dealt in [2]. The monocular description technique which is used for any kind of images, including random dots also, two kinds of matching namely, local matching and global matching is explain at in [3]. The handle seal imprints under much less constraints are dealt in [4]. The matching algorithm for a large number of model objects are discusses in [5]. In [6] the efficiency of graph matching is improved through the reduced storage capability.[7] represents the new method for effective recognition of handwritten Chinese character. A systematic method for 3D model matching in robot vision by using subgraph matching techniques is explained in [8]. The proposed method checks geometric constraints between and current partial matching pairs and remaining possible pairs and significantly reduces the search space in [9]. Generalized Rete Networks
algorithm for matching is given in [10]. An incremental clustering system based on a new principle function to group patterns represented by attributed graphs is presented in [11]. The ordering of graph, match _label and constraints, the matching procedure of graph is illustrated in [12]. A new method that combines the Graph Matching approach with Rule-Based approaches from Machine Learning is specificer in [13]. The computational algorithm for the matching explained in [14]. The proposed method can be used to solve other planar graph problems including connectivity, diameter, girth, induced subgraph isomorphism, and shortest paths illustrated in [15]. [16] Illustrates an are association graph from two (rooted) trees, based on the graph-theoretic notions of connectivity and the distance matrix. The problem of comparing different objects to each other can be formulated as the search for correspondences between the attributed graphs representing the objects. Such correspondences can be established by subgraph isomorphism detection is discussed in [17]. An efficient subgraph isomorphism detection through the use of decomposition approach is explained in [18]. An efficient algorithm for inexact graph matching is illustrate in [19],[20] explains how raster images are converted in Region Adjacency Graphs structures. The problem of computing a perfect matching of minimum cost in an undirected weighted graph is given in [21]. An algorithm for graph isomorphism and subgraph isomorphism suited for dealing with large graphs is given in [22]. The algorithm exploits graph sparsity to improve computational efficiency is proposed in [23]. [24] explains an inexact graph-matching is a problem of potentially exponential complexity; the problem may be simplified by decomposing the graphs to be matched into smaller subgraphs. In [25] illustrates the utility of the resulting method for shape-analysis. A new filtering algorithm for the subgraph isomorphism problem that exploits the global structure of the graph to achieve a stronger partial consistency is dealt in [26].Learning graph matching is about designing efficient algorithms for approximately solving the quadratic assignment problem is deal in [27]. 

A novel graph indexing method to handle with the more general, inexact matching problem is proposed in [28].

Section1 deals with introduction. Basic definitions are explained in section2. Section3 discusses an approach of proposed graph matching algorithm and explanations associated with it. Results and discussions are explained in section4.Finally, in Section 5 deals with the conclusion and explains about further works and directions.

II. DEFINITIONS

Definition 1. Graph

A graph is usually denoted by G(V,E) or by G = (V,E). A graph consists of set of vertices (V) together with a set of edges (E). The number of vertices in a graph is usually denoted by n while the number of edges is usually denoted by m.

Definition 2. Graph Isomorphism

Two graphs, G1 and G2, are isomorphic if there is a one-to-one correspondence between the vertices of G1 and those of G2 with the property that the number of edges joining any two vertices of G1 is equal to the number of edges joining the corresponding vertices of G2.

Definition 3. Matching

Matching is defined as the comparison of two or more graphs, and then the resultant similarities are extracted as output. This is known graph isomorphism. Here the matching is done by comparing the model graph which is stored in a file and an input graph that is given at run time. A directed graph is taken for matching.

III. GRAPH MATCHING ALGORITHM

3.1 Over view of the method

The concept of Graph isomorphism detection has been applied in a variety of fields. For instance, a given graph G1 is isomorphic to a graph H of another given G2 if there exists a one-to one mapping of the nodes of G1 onto the
nodes of H such that all corresponding edge adjacencies are preserved. The problem is that of finding an efficient algorithm or approach for determining whether one given graph (pattern graph, G1) is an isomorphic Graph of another graph (model graph, G2). Here, all graphs are supposed to be connected graphs. Its difficulty can be seen easily from the fact that selecting out of the \( mn \) possibilities that arise in the combinatorial matching of \( n \) nodes in the smaller graph to \( m \) nodes in the larger graph, while preserving all the adjacencies. No efficient algorithm for this problem is known so far, and it was conjectured by many experts that no polynomial-time algorithm exists because of its Non Polynomial-completeness. Generally, for majority of practical applications, both pattern and model graphs are labeled graphs. Their label sets are known and their sizes also. In the extreme case of the label set size being one, a labeled graph becomes indistinguishable with an unlabeled graph, so that labeled Graph isomorphism problem contains unlabeled Graph isomorphism problem which is a well known Non Polynomial-Completeness problem. In fact, unlabeled Graph isomorphism problems are usually used on theoretical analysis since it can provide an up-bound of time complexity due to its Non Polynomial-completeness.

3.2 New Algorithm

The Graph Isomorphism problem is a frequent computation in many applications, where the search and recognition of a smaller graph from a larger graph is needed. It deals with two different graphs.

1. Model graph
2. Input graph.

Model graphs are known prior and with that model graph an input graph is to be compared. This algorithm provides a new approach to compare model graph with more than one input graphs. It compares with the model graph which is stored on a file prior with the input graphs given at dynamic run time. Comparison is done between the first input graphs, then after that the results obtained from the first comparison is compared with the second input graph. If any match exists, then that is termed as the final result.

Table I  NEW ALGORITHM

| Step 1 | Get the input for the First Graph |
| Step 2 | Get the no. of nodes |
| Step 3 | Then get the vertices and its path on a adjacency matrix . |

```java
for(i=1;i<=n;i++)
{
    for(j=1;j<=n;j++)
    {
        // PRINT i value and j value//
        graph1[i][j]=Integer.parseInt(br.readLine());
    }
}
```

| Step 4 | By default nodes pointing to the same node is kept zero. If that has to be entered manually, |

```java
for(i=1;i<=n;i++)
{
    for(j=1;j<=n;j++)
    {
        graph1[i][j]=0;
    }
}
```

| Step 5 | Then store first graph on to a file. Which acts as an model graph. |

```java
BufferedWriter ot = new BufferedWriter(new FileWriter("graph.txt"));
for(i=1;i<=n;i++)
```

Integrated Intelligent Research (IIR)
This graph matching algorithm provides the way to identify matches or similarities between two or more graphs. It solves the problem of Graph isomorphism which is one of the major challenging aspects. The new method is introduced in this algorithm where the nodes having same path in two model graphs are represented only once. And that value is compared with the input graph that is given online. This partially reduces the time complexity of an algorithm. And another method handled is, the nodes pointing to itself is marked zero by default. This reduces the time consumption while giving input.

Initially, first graph input is obtained and stored in an array. Then the path between nodes is obtained on an adjacency matrix graph1[i][j]. Graph1 is said to be a Model graph. Since a model graph is to be known prior, It is initially stored on a file on local system. Then similarly the second graph value is obtained and stored on graph2[i][j]. Then values or paths between edges are obtained. At last final comparison is done by retrieving the model graphs from file and compared with the input graphs that are given online. This algorithm compares with more than one input graph and the match is found.

3.3 Example

The following section explains an example of the proposed algorithm.

(1) Get the model graph as first graph

![Model Graph](image)
(2) Generate Adjacency Matrix for model Graph1

\[
\begin{bmatrix}
1 & 2 & 3 & 4 & 5 \\
1 & 0 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 1 & 0 & 1 \\
5 & 1 & 0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

(3) Get the input graph as graph 2

\[\text{Fig 2. Input Graph1}\]

(4) Generate Adjacency Matrix for input Graph1

\[
\begin{bmatrix}
1 & 2 & 3 & 4 \\
1 & 0 & 1 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 \\
3 & 1 & 0 & 0 & 0 \\
4 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

(5) Get the input graph2

\[\text{Fig 3. Input graph 2}\]

(6) Generate Adjacency Matrix for input Graph2

\[
\begin{bmatrix}
1 & 2 & 3 \\
1 & 0 & 1 & 0 \\
2 & 0 & 0 & 1 \\
3 & 0 & 0 & 0 \\
\end{bmatrix}
\]

(7) After comparisons, the matched graph is produced as output

Common nodes and path are:

1-2=1
3-1=6

\[\text{Fig 4. Output graph}\]

In first step the model graph gets input. The size of the model graph depends on the user wish. Then the two input graphs get as an input graph. The size of input graph also depends on user
wish. Then generate the adjacency matrix for model graph and input graphs. Here the Model Graph and input graph which has the value 0 and 1. The value 0 (Zero) denotes that there is no path or connection between two vertices.

Input is given as an adjacency matrix. For e.g. $1 \to 3 = 0$ denotes that there is no link between 1 and 3. $1 \to 2 = 1$ denotes that the connection exists between two edges. So these values are obtained in an array and stored on a file using file stream. Then for comparison, each value is fetched from a file compared with the rest of the paths. If two graphs has the same path, then that match is extracted and then similarly it is done for all the rest of the nodes.

Be default, the diagonal Elements of an adjacency matrix is 0 (zero), since a node cannot point to itself. The comparison is done after sorting all the elements in ascending order, then by taking (1, 2) pair, it checks all the resultant values in the adjacency matrix up to (5,4). Then the final match is extracted.

3.4 Computational Complexity Analysis:

In this section a detailed analysis of the run time complexity of the proposed algorithm for model graph and input graphs are discussed. The analysis will be based on the following quantities:

$L=$ the number of model graph

$n=$ number of vertices of a model graph

$m1=$ the number of vertices that are unique to each input graph

$m=$the total number of vertices in input graph

The computation steps that are performed can be estimated in terms of calls of the nodes or vertices and the number of calls to the procedure combine times the complexity of combine itself.

The best case for the algorithm arises when each of the vertices of the model graph is uniquely labeled. For a single model graph, there will be different vertices. It will be called $n$ times. Each call test m vertices of the input graph and consequently, a total number of $O(mn)$ steps. The number of matchings found by $O(m/n)$. There will $O(m)$ calls to the procedure to combine.

Comparing the graphs of the mapping takes $O(k)$ steps with $k \leq n$.

Thus, the total computational complexity for matching a single model graph against an input graphs is the best case bounded by

$O(mn+n^2 m/n)=O(mn)$

IV. RESULTS AND DISCUSSIONS

As mentioned on the abstract, this algorithm will basically search over the combination of matches for the similarities between two input graphs and a model graph. By "match", that is a mapping from single node of graph1 to a single node of model graph. Each step in the search algorithm will add a new match to the isomorphism as developed so far. So once if this match extends to all nodes of input graphs, then it’s the end point of searching algorithm. Else if that is not accomplished, then it has to backtrack and attempt different matches. So the same process is repeated until the maximum matches between two graphs are found.

4.1 Results

As a result, this research obtains all the matches between an input graph and the model graph that has stored on a file priory. The main modification of the algorithm is, the nodes are sorted in the increasing order before matching and all the nodes are placed on a structure of array. And it also performs mapping between unsorted nodes, so that it can sort it down at the final stage and return mappings between nodes. Thus a final comparison results are fetched as an output.

4.2 Discussions

At this point, this paper work tries to find much worse cases. It seems obvious that it’s going to have more backtracks if the graph is "highly automorphic", by which it means that large portions of the graph are isomorphic to other
portions. When it tried this, sure enough, there were lots and lots of backtracks. Then it realized that it ought to be worse when there was no isomorphism, so it hooked two opposite “corners” together with an edge in one graph which wasn't present in the second. To cancel out of this run after a long wait, because, it's pretty close to the worst case. Since the corners had lower degrees than the middle nodes, they were going to be examined after all the middle nodes had been checked for isomorphism, and since they were the only non-isomorphic part, it was going to try zillions of backups. To verify this, instead of one graph between these two nodes, it inserted several so their higher degree caused the algorithm to examine them first. This caused the runtime to be almost immediate.

One possibility which would have eliminated this particular problem is to have just done a comparison of node degrees. Since it is already sorting nodes by degree, this should be easy. It will also work for graphs where it has to establish a one to one mapping between nodes in the second graph and nodes with a larger degree in the first graph.

V. CONCLUSION AND FURTHER WORKS

In this research work the new algorithm for graph matching is proposed. The new algorithm is based on the idea of finding an isomorphism between a single model graph and the different input graphs. In that at the run time the detection of the isomorphism is found out from a model graph to some of input graphs.

The graph matching is performed with the labeled and directed graphs. Graph matching algorithm is best suited for a particular application which depends on the number and the size of the underlying graphs, the kind of labels that occur in the graphs, and the type of matching that is required. In brief, the main conclusions from this research are that,

(i) The speed of graph matching can be significantly depends on the size of the model graph and the input graphs.

(ii) The accuracy of graph matching can be significantly improved without decreasing the speed.

5.1 Further Works and Directions

The new algorithm should not be applied to problems dealing with unlabeled and undirected graphs with restricted connectivity. The algorithm has been presented in a general form without the explicit use of domain features or heuristics. Therefore, it can be regarded in a generic tool that is applicable to various tasks.

The new graph matching be implemented with the use of different model graphs and n number of input graphs. The complexities of the graph matching approaches make it very difficult to take into account all types of dependencies between the vertices and edges in the model graph and input graphs. At last, it is to be mentioned that there is a potential for parallelization inherent in the new approach, which may be used for a further improvements of the performance.

REFERENCES


