Graph Isomorphism in View of Decomposition Technique

Krishnendu Basuli, Sonali Das

Abstract: Graph Isomorphism is an open problem in graph theory. A large number of graphs have the polynomial time algorithms for detection. But in general there are specific algorithms for certain range of vertices. Our try to design a generic algorithm. In this paper we proposed an algorithm of $O(n^3)$ complexity. The decomposition techniques have been adopted as well as reconstruction conjecture in reverse for taking decision of whether the two graphs are isomorphic or not. The ideology is inherited from DNA isolation in molecular biology [Chomczynski et. al].

Keywords: Graph, Isomorphism, reconstruction, Conjecture, graph decomposition, DNA isolation.

I. INTRODUCTION:

Two graphs $G = (V,E)$ and $G' = (V',E')$ be graphs. $G$ and $G'$ are said to be isomorphic if there exist a pair of functions $f:V \rightarrow V'$ and $g:E \rightarrow E'$ such that $f$ associates each element in $V$ with exactly one element in $V'$ and vice versa; $g$ associates each element in $E$ with exactly one element in $E'$ and vice versa, and for each $v \in V$ and each $e \in E$, if $v$ is an endpoint of the edge $e$, then $f(v)$ is an endpoint of the edge $g(e)$[Garey 1999].

It is not always easy to establish if two graphs are isomorphic or not. An exception is the case where the graphs are simple. In this case, we just need to check if there is a bijection $f:V_1 \rightarrow V_2$ which preserves adjacent vertices (i.e. if $v_1, v_2$ are adjacent in first graph, then $f(v_1), f(v_2)$ must be adjacent in second graph).

Graph that have same fundamental shape and Identical connections (but maybe drawn differently) are said to be isomorphic.Isomorphism is only for a finite graph, not for an infinite graph. Isomorphism is easy for simple graph. For some specific graph for which have specific algorithm in polynomial complexity these areIsomorphism of bipartite graphs, labeled graphs, and comparability graphs, directed graphs, regular graphs, perfect graphs, chordal graphs. Special cases are easies Isomorphism of polynomial time algorithm for graphs of bounded degree, Planner graphs, Trees, Bounded tree width and Isomorphism of non-deterministic polynomial time for except random graphs and interval graphs [Miller, 1977], [Hopcroft and Wong, 1974], [Booth, 1978] [Babai, 1977][Agrawal, M. and Arvind, V. 1996] [Arvind V., Kurur P.P.2002] [Fortin S.1996] [Harary F., 1982] [Lauri J1992] [Rosa A.1967] [Toran J. 2000].

The graph isomorphism question simply asks when two graphs are really the same graphs in disguise because there’s a one to one correspondence (an ‘isomorphism’) between their nodes are connected. The problem is easy to state, but tricky to solve, since even small graphs can be made to look very different just by moving their nodes around.

Some basic constraints for two isomorphic graphs $G_1$ and $G_2$ are as follows[Deo, 2006]:

1. Same number of vertices of two given graphs $G_1$ and $G_2$.
2. Same number of edges of two given graphs $G_1$ and $G_2$.
3. Same number of Total Degrees of two given graphs $G_1$ and $G_2$.
4. Same number of component of two given graphs $G_1$ and $G_2$.
5. Same number of Adjacency degree with each vertex $V$ and its neighbor vertices of two given graphs $G_1$ and $G_2$.
6. Same number of characters of each vertex deleted sub-graph of two given graphs $G_1$ and $G_2$.

To isolate a specific gene, one often begins by constructing a DNA library. By passing several steps it becomes isolated. Such way after passing several checking constraints at each steps we are reaching a decision whether two graphs are isomorphic or not.

Revised Manuscript Received on December 15, 2019.

*Corresponding Author:
Krishnendu Basuli, Dept. Of Computer Science, West Bengal State University, Barasat, Kolkata-126, West Bengal, India.
krishnendu.basuli@gmail.com

Sonali Das, Department of Laboratory Medicine, Calcutta School of Tropical Medicine, 108, C. R. Avenue, Kolkata 700 073, India, sonalidasstm@gmail.com.
II. PROPOSED FLOW CHART AND ALGORITHM:

Figure-Flow Chart

In above flow chart some special words are used, that’s words meaning are as follows

1. Vertex1: Total number of vertex of first graph.
2. Vertex2: Total number of vertex of second graph.
3. Edge1: Total number of edges of first graph.
4. Edge2: Total number of edges of second graph.
5. Degree1: Total number of degree of first graph.
6. Degree2: Total number of degree of second graph.
7. Component1: Total number of component of first graph.
8. Component2: Total number of component of second graph.
9. Adjacency_Degree1: Total number of adjacency degree for each vertex and its neighbor vertex of first graph.
10. Adjacency_Degree2: Total number of adjacency degree for each vertex and its neighbor vertex of second graph.
11. Vertex_Adjacency-Degree1: Delete Each Vertex and create a sub-graph, now calculate total number of adjacency degree for each vertex and its neighbor vertex of first graph.
12. Vertex_Adjacency-Degree2: Delete Each Vertex and create a sub-graph, now calculate total number of adjacency degree for each vertex and its neighbor vertex of second graph.

III. PROPOSED ALGORITHM:

Step 1: Start
Step 2: Read Adjacency Matrix of First graph and second graph.
Step 3: Calculate total number of Vertices of First graph and store into Vertex1.
Step 4: Calculate total number of Vertices of second graph and store into Vertex2.
Step 5: if Vertex1 is equal to Vertex2 go to next step, otherwise go to step 22.
Step 6: Calculate total number of Edges of First graph and store into Edge1.
Step 7: Calculate total number of edges of second graph and store into Edge2.
Step 8: If Edge1 is equal to Edge2 go to next step, otherwise go to step 22.
Step 9: Calculate total number of Degrees of First graph and store into Degree1.
Step 10: Calculate total number of Degrees of Second graph and store into Degree2.
Step 11: If Degree1 is equal to Degree2 go to next step, otherwise go to step 22.
Step 12: Calculate total number of components of First graph and store into Component1.
Step 13: Calculate total number of components of Second graph and store into Component2.
Step 14: If Component1 is equal to Component2 go to next step, otherwise go to step 22.
Step 15: Calculate total number of Adjacency degree of each vertex with its neighbor's adjacency degree of First graph and store into Adjacency_Degree1 [1, 2, 3, ………].
Step 16: Calculate total number of Adjacency degree of each vertex with its neighbor's adjacency degree of Second graph and store into Adjacency_Degree2 [1, 2, 3, ………].
Step 17: If Adjacency_Degree1 [1, 2, 3, ………] is equal to Adjacency_Degree2 [1, 2, 3, ………] go to next step, otherwise go to step 22.
Step 18: Calculate each vertex deleted sub-graph total number of Adjacency degree of each vertex with its neighbor's adjacency degree of First graph and store into Vertex_Adjacency_Degree1 [1, 2, 3, ………].
Step 19: Calculate each vertex deleted sub-graph total number of Adjacency degree of each vertex with its neighbor's adjacency degree of Second graph and store into Vertex_Adjacency_Degree2 [1, 2, 3, ………].

Step 20: If Vertex_Adjacency_Degree1 [1, 2, 3, ………] is equal to Vertex_Adjacency_Degree2 [1, 2, 3, ………] go to next step, otherwise go to step 22.
Step 21: Display Two Graph are Isomorphism and go to Step 23.
Step 22: Display Two Graph are Not Isomorphism.
Step 23: Exit.

IV. COMPLEXITY OF THE PROPOSED ALGORITHM:

For this proposed algorithm, there are three major steps to perform-

1. Component checking
2. Adjacency integer sequence for the one vertex deleted sub-graph.
3. And checking whether the both input graphs matrix after competition step Adjacency integer sequence for the one vertex deleted sub graphs.

(i) The complexity of the step 1 of above mention step is - O(n^2)
(ii) The completion of the step 2 is about O(n^3)
(iii) And the final step of step 3 is O(n^2)

The total complexity of this whole proposed algorithm is about - O(n^2) + O(n^3) + O(n^2) ≈ O(n^3)

EXAMPLES:- I:

Step 1: Adjacency Matrix of First graph and Second graph

Step 2: Total Number of Vertex1 = 5
Step 3: Total Number of Vertex2 = 5
Step 4: Total Number of Edge1 = 6
Step 5: Total Number of Edge2 = 6
Step 6: Total Number of Degree1 = 6
Step 7: Total Number of Degree2 = 6
Step 8: Total Number of Component1 = 1
Step 9: Total Number of Component2 = 1

Step 10: Adjacency_degree1 = {11, 11, 10, 8, and 4}
Step 11: Adjacency_degree1 = {11, 11, 10, 8, and 4}
Step 12: Vertex_Deleted_Sub-graph_Adjacency_Degree1

| A1 | A2 | A3 | A4 | A5 | A6 |
|----|----|----|----|----|----|
| A1 | 0  | 1  | 0  | 0  | 1  |
| A2 | 1  | 0  | 1  | 0  | 1  |
| A3 | 0  | 1  | 0  | 1  | 1  |
| A4 | 0  | 0  | 1  | 0  | 0  |
| A5 | 1  | 1  | 1  | 0  | 0  |

| B1 | B2 | B3 | B4 | B5 |
|----|----|----|----|----|
| B1 | 0  | 1  | 1  | 1  | 0  |
| B2 | 1  | 0  | 1  | 0  | 0  |
| B3 | 1  | 1  | 1  | 0  | 0  |
| B4 | 1  | 0  | 1  | 0  | 1  |
| B5 | 0  | 0  | 0  | 1  | 0  |
Graph Isomorphism in View of Decomposition Technique

Step 1: Adjacency Matrix of First graph and Second graph

Step 2: Total Number of Vertex1 = 6
Step 3: Total Number of Vertex2 = 6
Step 4: Total Number of Edge1=5
Step 5: Total Number of Edge2=5
Step 6: Total Number of Degree1=10
Step 7: Total Number of Degree2=10

V. APPLICATION OF GRAPH ISOMORPHISM:

1. In computer science, Graph isomorphism is some problems related to object-oriented pertinent for example in the state space generating problem and symmetry check determine whether two states are symmetric.
2. In cheminformatics and in mathematical chemistry, graph isomorphism testing is used to identify a chemical compound within a chemical database. Also, in organic mathematical chemistry graph isomorphism testing is useful for generation of molecular graphs and for computer synthesis.
3. In electronic design automation graph isomorphism is the basis of the Layout Versus Schematic (LVS) circuit design step, which is a verification whether the electric circuits represented by a circuit schematic and an integrated circuit layout are the same.
4. In detection of component parts in CAD images of mechanical drawings, the authors transform this problem into sub-graph Isomorphism problem for planner graph and provide an SGI algorithm for this class of graph.

EXAMPLES:- II:

Step 13: Vertex_Deleted_Sub-graph_Adjacency_Degree2

Step 1: Adjacency Matrix of First graph and Second graph

Step 2: Total Number of Vertex1 = 6
Step 3: Total Number of Vertex2 = 6
Step 4: Total Number of Edge1=5
Step 5: Total Number of Edge2=5
Step 6: Total Number of Degree1=10
Step 7: Total Number of Degree2=10

VI. CONCLUSION:

We have done for one level of vertex deleted subgraphs. But we can not established the theorem for that. But we have applied our algorithm on benchmark graphs for testing. And we get outstanding result. If we continue up to n-level i.e. First label and Second label and so on. Up to are isolated vertex label than it is confirmed that it should give the decision. But it may increase cost up to n (no. of vertices) label. So we try to prove it as theorem.

REFERENCES:
1. Agrawal M. and Arvind V. (1996), A note on decision versus search for graph automorphism. Information and Computation, 131:179-189.
2. Arvind V., Kurur P.P.(2002), Graph Isomorphism is in SP P. Electronic Colloquium on Computational Complexity, Report No.37, (2002).
3. Babai, L., Erdos, P., and Selkow, S. M. (1980), Random
graph isomorphism. SIAM J. Comput., 9:628-635.
4. Booth, K. S. (1978). Isomorphism testing for graphs, semigroups, and finite automata are polynomially equivalent problems. SIAM J. Comput., 7:273-279.
5. Chomczyński P, Mackey K, Drew R, Wifinger W. DNAzol: a reagent for the rapid isolation of genomic DNA. Biotechniques. 1997;22(3): 550-553. [PubMed].
6. Deo N.(2006). Graph theory with Applications to Engineering and Computer Science, PHI, India, 2006.
7. Fortin S.(1996), The Graph Isomorphism Problem, Technical Report TR 96-20, Department of Computing Science, The University of Alberta, Canada, (1996).
8. Garey M., Johnson D.(1999). Computers and Intractability A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, 1999.
9. Harary F.(1982). Graph Theory, Reading(Mass.), Addison-Wesley. (1969) [12] Kuc’era L., Kombinatorické algoritmy, Praha, SNTL, (1982)
10. Hopcroft, J. and Wong, J. (1974). Linear time algorithms for isomorphism of planar graphs. Proc. 6th Annual ACM Symp. Theory of Computing, pages 172-184.
11. Lauri J(1992), Vertex-deleted and Edge-deleted Subgraphs, Collected Papers, The University of Malta, (1992).
12. Miller Grey L. (1977), Graph Isomorphism, General Remarks. In Proceedings of Ninth Annual ACM Symposium on Theory of Computing, pp. 143-510, May 1977.
13. Rosa A.(1967). On certain valuations of the vertices of a graph. Theory of Graphs, Int. Symp. Rome July 1966. Gordon and Breach, Pp. 349-355.
14. Toran J. (2000). On the hardness of graph isomorphism. Proc. 41st Annual Symposium on Foundations of Computer Science, California, pages 180-186.

AUTHORS PROFILE

Dr. Basuli, is an Assistant Professor of Dept. of Computer Science, West Bengal State University. His research interest is graph theory and combinatorics. He published various papers on different graph theory open problems. He completed his Ph.D work from Calcutta University, India.

Dr. Das, is Research Associate in the department of Laboratory Medicine, Calcutta School Of Tropical Medicine. Her research interest is on molecular biology and parasitology. She has various publication in this field. She completed her Ph.D from Calcutta University, India.