

Towards a Promising Edge Classification Algorithm for the Graph Isomorphism Problem

Islam A.T.F.Taj-Eddin

Faculty of Informatics and
Computer Science, The British
University in Egypt, Cairo,
Egypt.

Samir Abou El-Seoud

Faculty of Informatics and
Computer Science, The British
University in Egypt, Cairo,
Egypt.

Jihad M. AL-Ja'am

College of Engineering,
Dept. of Computer Science
and Engineering, Qatar
University, Qatar.

ABSTRACT

For over three decades the Graph Isomorphism (GI) problem has been extensively studied by many researchers in algorithms and complexity theory. To date, there is no formal proof to classify this problem to be in the class P or the class NP. In this paper, evidence had been proposed of the existing of polynomial time algorithm based on edge classification which can be used to prove that GI is rather in the class P.

General Terms

Computational Mathematics, Theoretical Informatics,
Theoretical Computer Science, Graph Algorithms,
Algorithms.

Keywords

Edge Classification, Graph Isomorphism, Polynomial
Algorithm, Graph Canonization.

Nomenclature

GI	Graph isomorphism
Gd	Directed Graph
DG	Directed unweighted graph with no self-loops and no multiple edges
SnD	Square n-partite directed acyclic graph
EBFS	Edge Breadth First Search

1. INTRODUCTION

The Graph Isomorphism (GI) problem consists of deciding whether two given graphs are identical even if they look different in their graphical or adjacency matrices representation. Formally, two graphs G_1 and G_2 are called *isomorphic* if there is a bijection function b from the vertex-set V_1 of G_1 to the vertex-set V_2 of G_2 such that the edge (v_i, v_j) belongs to G_1 if and only if the edge $(b(v_i), b(v_j))$ belongs to G_2 and b is an isomorphism [10]. The problem has many applications in different fields, such as in cheminformatics [15], graphical data mining [9] and electronic design verification [5]. Even though, the GI problem has remarkable properties in terms of its complexity structure, it is still not known whether it is in the class P, or the class NP (assuming that $P \neq NP$). However, there is some evidence to support that GI is not an NP-complete problem. In fact, if it is in NP then the polynomial hierarchy would collapse to the second level [17][6]. In addition, GI is also not known to be hard for P. In fact, the best known hardness results are still relatively weak [19][1][17].

More formally, two graphs G_1 and G_2 are *isomorphic* if they have the same number of vertices, the same number of edges, the same degree sequence for the vertices, and the same edges relations (i.e. paths) among vertices. The first three conditions

could be tested in a polynomial time. Without any loss of generality, the graphs G_1 and G_2 have the same number of vertices, the same number of edges and the same degree sequences. The fourth condition is relatively hard to be tested. It could be shown that there exists two graphs that satisfy the first three conditions but do not have isomorphic set of consistently generated trees (i.e. they do not satisfy the fourth condition) [17]. It is sufficient and enough to generate all possible paths of length at most n , where n is the number of vertices in both graphs G_1 and G_2 , to prove the isomorphism of G_1 and G_2 based on the isomorphism of all possible generated paths. This could be done in an exponential time at the worst case.

We assume that the graphs G_1 and G_2 are connected, unweighted, undirected and their vertices have no self-loops. In this paper, an undirected graph G will be represented by its corresponding Directed Graph (G_d), such that every undirected edge between the vertices v_1 and v_2 of G will be replaced by two directed edges in G_d . The first edge is from v_1 to v_2 while the second edge is from v_2 to v_1 . Figure 1 shows an example.

Note that the adjacency matrices of G and G_d are clearly identical. Then the undirected graphs G_1 and G_2 will be replaced by their corresponding directed graphs G_{d1} and G_{d2} . The isomorphism problem between G_1 and G_2 will be transformed to the problem of finding the isomorphism between G_{d1} and G_{d2} .

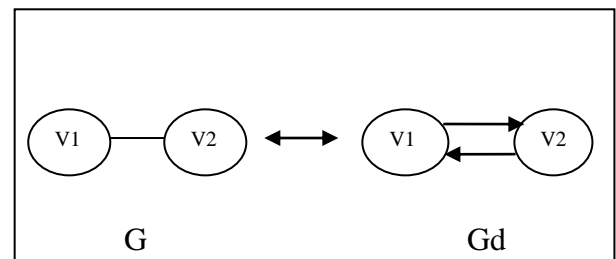


Fig. 1: The undirected graph G is represented by its corresponding directed graph G_d .

2. PREVIOUS WORKS

Unlike general graphs, it has been shown that there exists an efficient solution for the GI problem for special graphs with specific constraints on the number of vertices and edges (i.e. trees, planar graphs, permutation graphs, graphs of bounded degree) [16][14][8][20][13].

The graph canonization is the essence of many graph isomorphism algorithms. It is an open question whether there

is some canonizing function for graphs that can be computed in polynomial time, and if the two problems are polynomial-time equivalent [3][4][1][17][18][12].

Many attempts, of different nature, have been made to establish the GI complexity status. According to the current knowledge, there is no formal proof to solve the GI problem in a polynomial time. An attempt of using the edge classification algorithm to classify the GI problem was introduced in the late sixties [22][21]. Even though, it didn't provide a general formal proof, it succeeded however to classify GI for several types of graphs [2]. Note that, it has been proved in [7] that edge classification and n-tuples classification algorithms were unable to distinguish among all pairs of non isomorphic graphs. In [7], they considered graphs of $O(n)$ distinct vertices, with color class size of 4, and assumed a linear time canonical labeling algorithm. The resulted canonical form of the graph is isomorphic to its corresponding graph.

In [11], it has been mentioned that:

"...if f is an isomorphism between H and G itself, then any change in G must be reflected by a corresponding change in H , or else f will no longer be an isomorphism. In other words, proofs of NP-completeness seem to require a certain amount of redundancy in the target problem, a redundancy that GRAPH ISOMORPHISM lacks. Unfortunately, this lack of redundancy does not seem to be much of a help in designing a polynomial time algorithm for GRAPH ISOMORPHISM either, so perhaps it belongs to NPI..."

In this work, an edge classification algorithm that can be used to classify GI problem to be rather in the class P has been presented. The paper edge classification algorithm doesn't run against the proof given in [7]. In fact, the resulted canonical labeling graph, presented in this paper, contains redundant (not distinct) vertices. The element of redundancy will be used later, as a constraint, in the isomorphism testing without contradicting [7] (see figure 3 and theorem 2). The resulted canonical labeling graph is not isomorphic to its corresponding graph, yet it is a one-to-one and onto transformation.

3. GRAPH CANONIZATION

In this section, a vertex v of one graph will be chosen. The edges will be classified according to their distance from v . The classification procedure is presented as a (one-to-one and onto) transformation from the input graph (Gd) to an output graph denoted by SnD (the square n-partite directed acyclic graph) allowing the representation of paths of minimal length between the initial vertex v and all other vertices. The resulted graph (SnD) is not isomorphic to the original graph (Gd). Theorem 1 and its proof summarize the idea.

We assume that there exists a given starting vertex for any given graph Gd and for any directed unweighted graph with no self-loops and no multiple-edges (DG). Any criteria could be used to choose the starting vertex. Since $Gd \subset DG$, then the work will be done first with DG and later with Gd.

A square n-partite directed acyclic graph of $n \times n$ vertices (SnD), is an ordered finite set S of tuples S_j , where $S = \{S_j \mid 1 \leq j \leq n\}$ and n is the number of vertices in DG, such that the following four rules must be satisfied:

1. The tuple S_j , $1 \leq j \leq n$, is a finite collection of distinct n vertices, $S_j = \{v_i \mid 1 \leq i \leq n\}$.
2. Tuples S_j , $1 \leq j \leq n$, are equal (i.e. $S_1 = S_2 = S_3 = \dots = S_n$).
3. Vertices of S_j has a direct relation only with vertices of S_{j+1} , $1 \leq j \leq n-1$.

This direct relation is written as an ordered pair $(v_{xj}, v_{y(j+1)})$, with $1 \leq x, y \leq n$, $x \neq y$, (i.e. v_{xj} is vertex v_x that is a member of S_j and has a direct relation with $v_{y(j+1)}$ which is vertex v_y that is a member of S_{j+1}).

4. For any x and y , $1 \leq x, y \leq n$, if there exists a direct relation between v_x and v_y then there exists one and only one tuple $(v_{xi}, v_{y(i+1)})$, $1 \leq i \leq n-1$. This means that if there exists (v_{x3}, v_{y4}) then it is prohibited to have $(v_{xi}, v_{y(i+1)})$, such that $(1 \leq i \leq n-1 \text{ and } i \neq 3)$. (i.e. if (v_{x3}, v_{y4}) is true, then it is prohibited to have $(v_{x1}, v_{y2}), (v_{x2}, v_{y3}), (v_{x4}, v_{y5}), \dots, (v_{x(n-1)}, v_{yn})$).

3.1 Example 1

Assuming a square 3-partite SnD with a certain order as $S = \{S_1 = (v_1, v_3, v_2), S_2 = (v_1, v_3, v_2), S_3 = (v_1, v_3, v_2)\}$.

By rule 1 and 2, it could be written as $S = \{(v_{11}, v_{31}, v_{21}), (v_{12}, v_{32}, v_{22}), (v_{13}, v_{33}, v_{23})\}$, see figure 2. By rule 3, $(v_{11}, v_{22}), (v_{11}, v_{32})$ are allowed while $(v_{11}, v_{12}), (v_{11}, v_{23})$ are prohibited. By rule 4, if (v_{11}, v_{32}) does exist then (v_{12}, v_{33}) is prohibited.

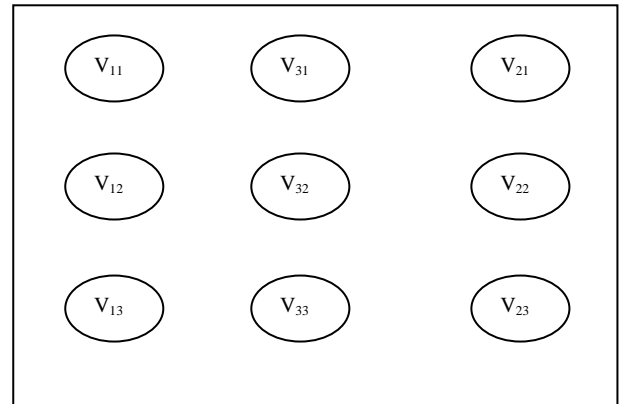


Fig. 2: The graph representation of $S = \{(v_{11}, v_{31}, v_{21}), (v_{12}, v_{32}, v_{22}), (v_{13}, v_{33}, v_{23})\}$.

3.2 Definition 1

Definition 1: Let G_s denotes the set of (v_x, DG) , v_x is the starting vertex of DG, $1 \leq x \leq n$, n is the number of vertices in DG. Let G_t denotes the set of (v_{x1}, SnD) , v_{x1} , which is the vertex v_x that is a member of S_1 of SnD, is the starting vertex of SnD. A canonical function for G_s is a one-to-one and onto function $f: G_s \rightarrow G_t$ from G_s to G_t , $f(v_x, DG) = (v_{x1}, SnD)$ and $f^{-1}(v_{x1}, SnD) = (v_x, DG)$, such that the (v_{x1}, SnD_i) is isomorphic to (v_{y1}, SnD_j) if and only if (v_x, DG_i) is isomorphic to (v_y, DG_j) .

Next, a one-to-one (f) and onto (f^{-1}) canonical functions that respect definition 1 will be proposed.

We define the function (f) as follows:

1. The input is a directed graph DG with a starting vertex v_x .
2. We have $f(v_x, DG) = (v_{x1}, SnD)$.
3. The output is SnD with a starting vertex v_x at tuple S_1 (i.e. v_{x1}).

We apply (f) as follows:

1. The input is (v_x, DG)
2. Apply $(EBFS(v_x, DG))$

Edge-Breadth-First-Search (EBFS), is similar in execution mechanism, correctness and time complexity to Breadth-First-Search (BFS) [10], but differs in the following:-

EBFS(v_x, DG)

1. The input is a directed graph DG with a starting vertex v_x .
2. The distance value (d) will be assigned to each edge rather than to each vertex. The distance value of the edge indicates how far is the edge on a path taken starting from the vertex v_x . Each edge traversed exactly once, and a distance value (d) is assigned to it.
3. The output is a directed graph DG with a starting vertex v_x and a distance value (d) for each edge, $1 \leq d \leq n$.
3. Create an Empty SnD, as stated at the definition of SnD n is the number of vertices in DG used in the previous steps. For each directed edge $e=(v_y, v_z)$ with distance (d) in DG, set a direct edge from v_y that belongs to set $S_{(d)}$ to v_z that belongs to set $S_{(d+1)}$ (i.e. $(v_{y(d)}, v_{z(d+1)})$).
4. Output SnD with starting vertex v_x at tuple S_1 (i.e. v_{x1}).

We conclude that $f(v_x, DG) = f(EBFS(v_x, DG)) = (v_{x1}, SnD)$.

We define $(f)^{-1}$ as follows:

1. The Input is SnD with a starting vertex v_x at tuple S_1 (i.e. v_{x1}).
2. $f^{-1}(v_{x1}, SnD) = (v_x, DG)$.
3. The output is a directed graph DG with starting vertex v_x .

Apply $(f)^{-1}$ as follows:

1. The input is SnD with starting vertex v_x at tuple S_1 (i.e. v_{x1}).

2. Create an empty DG with n vertices. For each directed edge $e=(v_{y(d)}, v_{z(d+1)})$, set a direct edge from v_y to v_z with one distance (d) for the edge in DG. Delete the value of (d) from each edge.
3. The output is a directed graph DG with a starting vertex v_x .

We conclude that $f^{-1}(v_{x1}, SnD) = f^{-1}(EBFS(v_x, DG)) = f^{-1}(v_x, DG) = (v_x, DG)$.

3.3 Theorem 1

Theorem 1: The function (f) is a one-to-one and onto function, such that:

1. $f(v_x, DG) = (v_{x1}, SnD)$.
2. $f^{-1}(v_{x1}, SnD) = (v_x, DG)$.

Proof:

1. In order for (f) not to be one-to-one, the following equations must be true:-
 $f(v_x, DG) = f(EBFS(v_x, DG)) = (v_{x1}, SnD1)$
and $f(v_x, DG) = f(EBFS(v_x, DG)) = (v_{x1}, SnD2)$,

Assume that $SnD1 \neq SnD2$, there must exist a directed edge $e=(v_y, v_z)$ with distance (d) in DG, such that there exists a direct edge from v_y at set $S_{(d)}$ to v_z at set $S_{(d+1)}$ (i.e. $(v_{y(d)}, v_{z(d+1)})$) at SnD1 and a different direct edge from v_y at set $S_{(d)}$ to v_z at set $S_{(d+1)}$ (i.e. $(v_{y(d)}, v_{z(d+1)})$) at SnD2.

Since the edge $e=(v_y, v_z)$ has one and only one value for (d), the edge $e=(v_y, v_z)$ with distance (d) at SnD1 and $e=(v_y, v_z)$ with distance (d) at SnD2 will be equal in distance (d) and position. The same could be said about all other edges, which means $SnD1 = SnD2$ (contradiction).

2. In order for f^{-1} not to be onto, the following equations must be true:- $f^{-1}(v_{x1}, SnD) = f^{-1}(EBFS(v_x, DG1)) = (v_x, DG1)$ and $f^{-1}(v_{x1}, SnD) = f^{-1}(EBFS(v_x, DG2)) = (v_x, DG2)$

Assume that $DG1 \neq DG2$, there must exist a directed edge $e=(v_{y(d)}, v_{z(d+1)})$, such that there exists a direct edge from v_y to v_z with distance (d) at DG1 and a different direct edge from v_y to v_z with distance (d) at DG2. Since the edge $e=(v_{y(d)}, v_{z(d+1)})$ is corresponding to one and only one edge $e=(v_y, v_z)$ with distance (d), the edge $e=(v_y, v_z)$ with distance (d) at DG1 will be equal with the edge $e=(v_y, v_z)$ with distance (d) at DG2. The same could be said about all other edges, which means $DG1 = DG2$ (contradiction).

4. THE RELATIONSHIP BETWEEN Gd1, Gd2 AND SnD1, SnD2

From now on and without any loss of generality, the paper will only deal with the Gd that represents a connected unweighted and undirected graph G.

As stated earlier, if and only if there is an algorithm to discover the isomorphism between SnD1 and SnD2 that will enforce that Gd1 is isomorphic to Gd2 and vice versa. (i.e. $G1 \leftrightarrow Gd1 \leftrightarrow SnD1$, and $G2 \leftrightarrow Gd2 \leftrightarrow SnD2$).

One reason behind the believe of the existence of a polynomial time algorithm that can discover the isomorphism

between SnD1 and SnD2 lies in the fact that SnD graph is sharing some common characteristics with planar graphs. An efficient polynomial time algorithm that can discover isomorphism between planar graphs was given in [16][14][8][20][13]. By definition, SnD1 does not have K_5 . By definition, SnD1 has $V=n^2$ vertices and maximum of $E=n^2-n$ edges. $E = n^2-n \leq (3V^2 - 6)$, for $n^2 \geq 3$, and $E = n^2-n \leq (2V^2 - 4)$, for $n^2 > 3$ [16][14][8][20][13]. In spite of these common characteristics with planar graphs, the graph SnD considered here is **not** a planar graph.

Before trying to discover a polynomial time algorithm for isomorphism between SnD1 and SnD2, a proof should be given for the case of any two distinct graphs Gd1 and Gd2 it holds that SnD1 and SnD2 differ at least in one characteristic of one metric. A definition of SnD metric will be given followed by a theorem and its proof.

4.1 Definition 2

Definition 2: define the **metric** $sg(v_i, k_i)$ as the ordered finite tuple $(v_{i1}, v_{i2}, v_{i3}, \dots, v_{in})$, for $1 \leq i \leq n$ that belongs to SnD, k_i , $1 \leq k_i \leq n$, is the location of v_i within all tuples S. It is allowed for a whole metric to swap with another whole metric only.

There exist some **characteristics** on metrics $sg(v_i, k_i)$, for $1 \leq i \leq n$. These characteristics are:

in-degree (v_{iy}) is the number of edges coming from any other vertex to vertex v_{iy} ,

out-degree (v_{iy}) is the number of edges coming into any other vertex from vertex v_{iy} ,

from-to (v_{iy}) is the ordered tuple of location(s) k_j , such that an edge is coming from vertex (v_{iy}) , which belongs to the metric $sg(v_i, k_i)$, going to vertex (v_{jk}) which belongs to the metric $sg(v_j, k_j)$ (i.e. $(v_{iy}) \rightarrow (v_{jk})$).

4.2 Example 2

Assuming a square 3-partite SnD is:

$S = \{S1=(v_1, v_3, v_2), S2=(v_1, v_3, v_2), S3=(v_1, v_3, v_2)\}$, that could be written as $S = \{(v_{11}, v_{31}, v_{21}), (v_{12}, v_{32}, v_{22}), (v_{13}, v_{33}, v_{23})\}$.

Assuming the starting vertex is v_{11} , see figure 3. The characteristics on metric $sg(v_i, k_i)$ of figure 3 are as follows:

metric $sg(v_1, k_1=1)=(v_{11}, v_{12}, v_{13})$:

in-degree $(v_{11})=0$,
out-degree $(v_{11})=2$,
from-to $(v_{11})=(k_3=2, k_2=3)$,

in-degree $(v_{12})=0$,
out-degree $(v_{12})=0$,
from-to $(v_{12})=\emptyset$,

in-degree $(v_{13})=2$,
out-degree $(v_{13})=0$,
from-to $(v_{13})=\emptyset$.

metric $sg(v_2, k_2=3)=(v_{21}, v_{22}, v_{23})$:

in-degree $(v_{21})=0$,
out-degree $(v_{21})=0$,
from-to $(v_{21})=\emptyset$,

in-degree $(v_{22})=1$,
out-degree $(v_{22})=2$,
from-to $(v_{22})=(k_1=1, k_3=2)$,

in-degree $(v_{23})=1$,
out-degree $(v_{23})=0$,
from-to $(v_{23})=\emptyset$.

metric $sg(v_3, k_3=2)=(v_{31}, v_{32}, v_{33})$:

in-degree $(v_{31})=0$,
out-degree $(v_{31})=0$,
from-to $(v_{31})=\emptyset$,

in-degree $(v_{32})=1$,
out-degree $(v_{32})=2$,
from-to $(v_{32})=(k_1=1, k_2=3)$,

in-degree $(v_{33})=1$,
out-degree $(v_{33})=0$,
from-to $(v_{33})=\emptyset$.

4.3 Theorem 2

Theorem 2: Given the set of all metrics computed for SnD1 of Gd1 and SnD2 of Gd2, it holds that if and only if Gd1 and Gd2 are distinct then SnD1 and SnD2 will differ at least in one characteristic of one metric.

Proof:

1. (If) Gd1 and Gd2 are distinct then SnD1 and SnD2 will differ at least in one characteristic of one metric.
 - 1.1. It is sufficient and enough to generate all possible paths of length at most n, beginning from a certain vertex v_x , where n is the number of vertices in both graphs Gd1 and Gd2, to prove the isomorphism of Gd1 and Gd2 based on the isomorphism of all possible generated paths. That could be done in exponential time at worst case analysis.
 - 1.2. For every path, the initial vertex v_x , will be v_{x1} , the second vertex v_y in the path will be v_{y2} , ... etc. It could be shown that all paths of designated SnD is a subset of the generated all possible paths. From now on and without any lose of generality the proof will deal with that subset of the generated all possible paths (i.e. all paths of designated SnD).

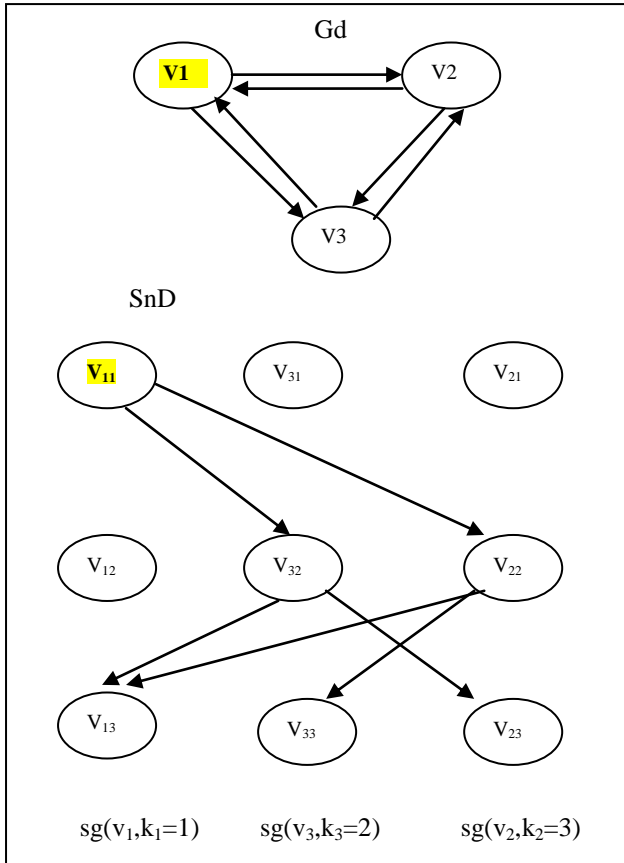


Fig. 3: This is the resulted SnD, with the starting vertex v_{11} , from the corresponding directed graph Gd, with the starting vertex v_1 , such that $Gd = \{(v_1, v_2), (v_2, v_1), (v_1, v_3), (v_2, v_3), (v_3, v_1), (v_3, v_2)\}$, k_i is the location of the metric $sg(v_i, k_i) = (v_{11}, v_{12}, v_{13})$, $1 \leq k_i \leq 3$.

- 1.3. If Gd1 and Gd2 are distinct then:
 - 1.3.1. Either at least one path, of the subset of the generated all possible paths, has different length at Gd1 than Gd2. For that case, it could be shown that at least one metric at SnD1 has different characteristic (**in-degree**(v_{iy}), **out-degree**(v_{iy}), **from-to**(v_{iy})) than SnD2;
 - 1.3.2. Or all paths have the same length, but at least one path, of that subset of the generated all possible paths at Gd1, has different relations (i.e. no bijection) than any paths at Gd2.

For that case, metrics at SnD1 and SnD2 will have the same (**in-degree**(v_{iy}), **out-degree**(v_{iy})) characteristics, but at least one metric at SnD1 will be different in its (**from-to**(v_{iy})) characteristic than any metric at SnD2.

Because since the paths length are equal in both Gd1 and Gd2, then (**in-degree**(v_{iy}), **out-degree**(v_{iy}))

characteristics in SnD1 and SnD2 will not be enough to distinguish between them, but from the definition of isomorphism no bijection means there exist at least one edge between two vertices in Gd1 that has no match in Gd2, that case will manifest itself at the (**from-to**(v_{iy})) characteristic which catch that relation between vertices of metrics in SnD1.

From the definition of isomorphism, the (**from-to**(v_{iy})) characteristic of one metric in SnD1 will be different than any (**from-to**(v_{iy})) characteristic of any metric in SnD2.

2. (Only if) SnD1 and SnD2 differ at least in one characteristic of one metric then Gd1 and Gd2 are distinct. That could be deduced directly from f^{-1} of Theorem 1.

5. CONCLUSION AND FURTHER WORK

The GI problem is extremely relevant in theoretical computer science. Many attempts, of different nature, have been made to establish its complexity status. Among these attempts, the classification of tuples of vertices is an important one. An edge classification algorithm that can be used to prove that GI problem is rather in the class P will be presented. The proposed algorithm chooses a vertex v of one graph and classifies the edges according to their distance from that vertex v . The classification procedure is presented as a transformation from the input graph Gd to a directed graph SnD allowing the representation of paths of minimal length between the initial vertex and all other vertices. The transformed graph SnD is not isomorphic to the original graph Gd. The proposed approach introduces the element of redundancy that does not contradict with the work given in [7].

As further work, a polynomial time algorithm that shows SnD1 is isomorphic to SnD2 is worth the trial to be found. Based on Theorem 2, if and only if Gd1 and Gd2 are distinct then SnD1 and SnD2 will differ at least in one characteristic of one metric. One reason behind the believe of the existence of a polynomial time algorithm that can discover isomorphism between SnD1 and SnD2 lies in the fact that SnD graph sharing some common characteristics with planar graphs. In spite of these characteristics, SnD is not a planar graph.

6. ACKNOWLEDGMENTS

The authors would like to thank Dr. Khaled Nagati, and all of the academic staff of the Faculty of Informatics and Computer Science at the BUE for their valuable remarks and suggestions.

7. REFERENCES

- [1] Arvind V., and Toran J. 2005. Isomorphism Testing: Perspective and Open Problems, Bulletin of the European Association for Theoretical Computer Science, Computational Complexity Column, (June, 2005), Number 86.

- [2] Babai L. 1996. Automorphism groups, isomorphism, reconstruction, Handbook of Combinatorics (eds. R. L. Graham, M. Grottschel and L. Lovasz), Chapter 27, North-Holland, Amsterdam, (1996), pages 1447-1540.
- [3] Babai L. and Kucera L. 1979. Canonical labeling of graphs in average linear time. Proc. 20th Annual IEEE Symposium on Foundations of Computer Science, 1979, 39–46.
- [4] Babai L. and Luks M. 1983. Canonical labeling of graphs. In proceedings of the fifteenth annual ACM symposium on theory of computing, 1983, page 171-183.
- [5] Baird H.S., Cho Y.E. 1975. An artwork design verification system, Proceedings of the 12th Design Automation Conference. IEEE Press., (1975) , pp. 414-420.
- [6] Boppana R., Hastad J. and Zachos S. 1987. Does co-NP have short interactive proofs?, Information Processing Letters 25(2), (1987), pages 127-132.
- [7] Cai J., Furer M., and Immerman N. 1992. An optimal lower bound on the number of variables for graph identification. Combinatorica, 12:389-410, (1992).
- [8] Colbourn C.J. 1981. On testing isomorphism of permutation graphs, 1981, Networks 11: 13–21, doi:10.1002/net.3230110103.
- [9] Cook D. J. and Holder L. B. 2007. Mining Graph Data, John Wiley & Sons, 2007.
- [10] Cormen T. H., Leiserson C. E., Rivest R. L. and Stein C. 2009. Introduction to algorithms, 3rd edition, the MIT press, 2009.
- [11] Gary M. R. and Johnson D. S. 2000. Computers and Intractability a Guide to the Theory of NP-Completeness, W.H. FREEMAN AND COMPANY, NY, 1979, pp. 155-156.
- [12] Gurevich Y. 1997. From Invariants to Canonization, The Bulletin of European Association for Theory of Computer Science, 1997, Number 63.
- [13] Hopcroft J. and Tarjan R. E. 1974. Efficient planarity testing. J. ACM, (1974), 21(4):549–568.
- [14] Hopcroft J. and Wong J. 1974. Linear time algorithm for isomorphism of planar graphs, Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, 1974, pp. 172–184, doi:10.1145/800119.803896.
- [15] Irmiger C-A M 2005. Graph Matching: Filtering Databases of Graphs Using Machine Learning, Aka, 2005.
- [16] Kelly P.J. 1957. A congruence theorem for trees, Pacific J. Math., 7, 1957, pp. 961–968.
- [17] Kobler J., Schoning U., and Toran J. 1993. The graph isomorphism problem: Its structural complexity. Birkhauser, 1993.
- [18] McKay B. D. 1981. Practical Graph Isomorphism. Congr. Numerantium ,(1981), 30, 45-87. (Nauty User's Guide, Version 2.2 (beta 6); (2003) (<http://cs.anu.edu.au/people/bdm/nauty/>)).
- [19] Toran J. 2000. On the hardness of graph isomorphism. FOCS, 2000, 180–186.
- [20] Toran J. and Wagner F. 2009. The complexity of planar graph isomorphism, Bulletin of the European Association for Theoretical Computer Science, Computational Complexity Column, (February 2009), Number 97.
- [21] Weisfeiler Boris ed. 1976. On Construction and Identification of Graphs, Lecture Notes in Mathematics 558, Springer, (1976).
- [22] Weisfeiler B. and A.A. Lehman 1968. A Reduction of a Graph to a Canonical Form and an Algebra Arising during this Reduction, (in Russian), Nauchno-Tekhnicheskaya Informatsia, Seriya 2, 9 (1968), 12-16.