Pushing the Boundaries of Combinatorial Graph Isomorphism Algorithms

Balaji N, Karthik Pai B H, Devidas, K Adithya Shastry, and Disha D N *Member IAENG*

***Abstract*—A computational problem well studied in the field of Complexity Theory is Graph Isomorphism. Color Refinement is a classical technique for testing the non-existence of isomorphism between two given graphs. Tinhofer's algorithm is an extension of the Color Refinement algorithm and the same technique does not succeed for all graphs. We characterize Tinhofer graphs algebraically. We propose and study in detail, a new graph hierarchy based on Tinhofer's algorithm which also prove for all the new graph classes in the hierarchy a sufficient condition. We look at the relations of the new graph classes with each other and also with the already known classes. We also facilitate an effective graph isomorphism algorithm for the lower classes in the hierarchy. Getting an algebraic characterization for the new hierarchy classes, understanding the exact location of other graphs in this hierarchy and extending the efficient algorithm for higher graph classes are identified as potential areas of future interest.**

***Index Terms*—Graph, Isomorphism, Color Refinement, Tinhofer Algorithm.**

I. Introduction

COMPLEXITY theory is a branch of mathematics that focuses on classifying problems based on their inherent hardness. In truth, problems are classified based on the resources of computation that are required for solving the problems. A computational resource is a resource utilized by a computational model, for example, a Turing Machine, to solve a computational problem. The most obvious examples of computational resources would be space and time. An algorithm time complexity is the time required by the algorithm to terminate, with the parameter being the size of the input string given.

An algorithm possesses Polynomial running time, when it is upper bounded with polynomial expression depending on the length of the string. Similarly, an algorithm possesses exponential running time, when the complexity is upper bounded with an expression by considering the size of input. Now, if an algorithm has an exponential running time, when the input string is even slightly long, the algorithm will take hundreds of years to terminate, that is as good as not having a solution.

A deterministic algorithm is one where, given any particular input, the output will always be the same, with the algorithm going through the same states each time it runs with the same input. Under Complexity group of decision problems which is resolved using polynomial time, with the help of deterministic Turing machine is called as P (P for polynomial). An algorithm of non-deterministic type is one that can exhibit different behaviors even when run on the same input.

Group of decision problems which is resolved with polynomial time, using Non-deterministic type Turing machine is called as NP (Polynomial of Non-deterministic type). Quite obviously, $P \subset NP$. But, the million-dollar question, whose answer has continued to elude computer scientist since the beginning of time, is whether $P = NP?$.

A. NP – Intermediate Problems

Several attempts have been made to understand these two complexity classes. A problem is named NP - Hard, if given a solution to that problem, we have a solution for all problems of NP. NP Complete problems are when it is in NP as well as in NP - Hard. The theorem of Cook-Levin states that the Boolean Satisfiability is NP - Complete [Cook (1971)]. Ladner's theorem defines, if $P \neq NP$, NP includes problems that are not in NP – Complete aswell as in P. This implies that $P = NP$ only when NPI is empty [Ladner (1975)].

Ladner, as proof of his theorem, constructs a class explicitly that is NPI. Whether such problems occur naturally remains to be answered. Schafer's Dichotomy Theorem provides us with conditions, such that when these conditions are fulfilled, Boolean Satisfiability problems will not be in NPI [Schaefer (1978)]. Graph Isomorphism, Factoring and Discrete Logarithm problems are good candidates to lie in the NPI Complexity class.

B. Graph Isomorphism and Color Refinement

Two graphs given, whether they are isomorphic, or is in NP can be called as Graph Isomorphism Problem. Whether or not it belongs in P, is still up for debate. Hence, we do not know the exact location of this problem in the Arithmetic Hierarchy. Unlikely the Graph Isomorphism is NP - Complete, since it has been shown that if this is the case, polynomial hierarchy will breakdown to the second level [Schning (1988)].

The attempts at solving Graph Isomorphism can be roughly categorized into two categories, combinatorial approaches and algebraic approaches. Color Refinement is a conventional method that is utilized to determine the existence of an isomorphism between two given graphs. Though

Manuscript received June 10, 2021; revised December 21, 2021. This work was supported in part by the NMAM Institute of Technology, Nitte under the Department of Information Science and Engineering.

Balaji N is an Assistant Professor of Department of Information Science and Engineering, NMAM Institute of Technology, Nitte, Karkala - 574 110, Udupi, Karnataka, India. e-mail: balaji.hiriyur@gmail.com

Karthik Pai B H is a Professor of Department of Information Science and Engineering, NMAM Institute of Technology, Nitte, Karkala - 574 110, Udupi, Karnataka, India. e-mail: karthikpai@nitte.edu.in

Devidas is an Assistant Professor of Department of Information Science and Engineering, NMAM Institute of Technology, Nitte, Karkala - 574 110, Udupi, Karnataka, India. e-mail: devidasbhat@nitte.edu.in

K Aditya Shastry ia an Associate Professor of Department of Information Science and Engineering, Nitte Meenakshi Institute of Technology, Benagluru, Karnataka, India. e-mail: adityashastry.k@nmit.ac.in

Disha D N is an Assistant Professor of Department of Artificial Intelligence and Machine Learning, NMAM Institute of Technology, Nitte, Karkala - 574 110, Udupi, Karnataka, India. e-mail: disha.dn@nitte.edu.in

this is a powerful technique, it does not succeed on all graphs. In this method, the vertices are classified into classes or colors based on necessary conditions for isomorphism. For example, if an isomorphism maps a vertex to another, they should have equal valences. Vertices are grouped iteratively until no further refinement is possible. Now, vertices can be mapped to one another only if they have the same color. This narrows down our search area.

Color Refinement Algorithm is considered as a combinatorial approach towards Graph Isomorphism. But this algorithm works only for some specific graphs. A study about the kind of graph is that Color Refinement work in Arvind et al., have given rise to a hierarchy of graph classes. The Recognition problem for any of these classes has been proven to be P - Hard [Arvind et al. (2015)].

C. Problem Statement

This article aims at exploring the hierarchy of graphs in a finer detail with respect to both complexity theory and structural restrictions on graphs. Till date, in the hierarchy, algorithms of polynomial time for determining the presence of isomorphism well known only up to the Tinhofer class [Arvind et al., (2015)]. What can be done for graphs outside the Tinhofer class is still under scrutiny. We shall propose a new hierarchy of graphs, based on Tinhofer's algorithm, based on our belief that this hierarchy will have a better chance of having polynomial time solutions.

Tinhofer graphs are the set of graphs for which the Tinhofer's algorithm works correctly. That is, when the graphs are isomorphic, the algorithm does end up saying that they are isomorphic (irrespective of the individualization that it does during the color refinement). The number of rounds of individualization that Tinhofer's algorithm will do is at most n (let this be called k). Let l be the least round (over all H') where the Tinhofer's algorithm goes wrong for a graph H. Thus, we can define, $LEVEL-k$ TINHOFER graphs to be the set of graphs where Tinhofer algorithm does not make an incorrect individualization up to level l.

This gives us a new hierarchy of graph classes based on Tinhofer's algorithm. We shall look at the connections with the earlier hierarchy. This new hierarchy encompasses the earlier one and also extends beyond. We shall look at the Color Refinement algorithm in the Directed Graphs setting and try to analyze the strengths and weaknesses of the algorithm.

II. Preliminaries

A. Graph Isomorphism and Automorphism

Let us first try to understand the meaning of the words 'Graph' and 'Isomorphism'. A graph can be defined by the collection of vertices and the edges between the vertices. Graphs can be used to model pairwise relations between objects. The objects are modelled as vertices and the relations are modelled by the edges between the vertices. Literal translation of the word isomorphism from ancient Greek is equal-form or equal-shape. Isomorphism is the structure preserving map amongst two objects of the same kind, which admits an inverse. Two objects can be considered to be isomorphic if they have equivalent structural properties. Isomorphism is studied widely in various fields of mathematics. For example, two programs written in the C language are called isomorphic if they produce the same outputs on all inputs. In algebra, group isomorphism is a bijective function amongst two groups that respects the given group properties.

Two graphs are isomorphic if the vertices of two graphs can be mapped in such a way that the edge relations between them are preserved. The given graphs say, H and G are called as isomorphic when there is a permutation p of V_n : for the edges $\{u, v\}$ represented as $E(H)$ only when $p(u)$, $p(v)$ is present in set of graph edges $E(G)$. A Graph Automorphism is a mapping from the vertices of a graph to itself in such a way that the resulting graph can be isomorphic with respect to the original graph. The set of automorphism includes a permutation group which is said to be as the graphs automorphism group.

The computational problem for ascertaining the two given graphs is isomorphic is named as graph isomorphism. It is a NP - class type of problem or NP Complete or in P is currently unknown. It may also be in the class NP - Intermediate if $P \neq NP$. It is clear that, if graph isomorphism belongs to NP – Complete, then the Polynomial hierarchy breakdowns to the next level.

Two graphs H and G with vertices $V = \{1, 2, \ldots, m\}$ are to be in isomorphic only when a permutation p of V_m such that $\{u, m\}$ is in the set of graph edges $E(H)$ and if and only when $p(u)$, $p(v)$ is in the set of graph edges $E(G)$. Graph automorphism is an isomorphism from a graph H to itself. A non-trivial automorphism is an automorphism which is not the identity mapping. A Graph Automorphism is when you map the vertices of a graph to itself such that the resulting graph is isomorphic to the original graph. The set of automorphism defines a permutation group which is called as the graphs automorphism group.

B. Group Theory

Group Theory studies the algebraic structure that are recognized as groups. These structures come up in various fields such as representation theory in physics and chemistry, public key cryptography and of course, Graph Isomorphism. The idea of groups has also been extended, giving rise to other algebraic structures like fields, rings, and vector spaces.

1) Definition-1: Group - It is a set H, which along with a binary operator '.' satisfies the below properties.

- **Closure:** $\forall\ x, y\ \in H$, $x \times y\ \in H$
- **Associativity:** $\forall\ a, b, c \in H\ (a.b).c = a.(b.c)$
- **Identity:** $e \in H$, such that $\forall c \in H, x.e = e.x$, the identity element is unique.
- **Inverse:** $\forall a \in H,\ a^{-1} \in H$ such that $a.a^{-1} = a^{-1}.a = e$

2) Definition-1: Group B with the subgroup A which forms a group with the identical group operation. A group homomorphism from a group H to G is defined as the mapping between $f : H \rightarrow G$ then $f(x.y) = f(x).f(y)$. Here, the identity of H gets mapped to the identity of G. The same is true for inverses $f(x^{-1}) = (f(x^{-1}))$. Two groups H, G are isomorphic if there is a homomorphism which is also a bijection.

3) Definition-3: Permutation of a set is a bijection on the set. The set of these permutations form group. It is called as symmetric group. S_m denotes the symmetric group on m elements. A group of permutations is a subgroup of S_m.

4) Theorem II-B.1: Cayley's Theorem says "Every group is isomorphic to a group of permutations."

5) Definition-4: Group H acts on set X, when there is a map from $H.X \to X$, and $(g, x) \to g.x$ such that

- $\forall h, x \in H,\ x \in X,\ h.(g.x) = (h.g).x$
- $\forall x \in X,\ e.x = x$

Now if we look at group theory in our context. A graph automorphism is permutation among the graph vertices. That means that we can look at the autmorphism group of a graph as a permutation group. Let us denote the autmorphism group of a graph H as $Aut(H) = X$. The permutation group X act on the vertex set $V(H)$, which also respects edges, giving rise to automorphism in the group.

6) Definition-5: If G is a permutation group stand-in on a set X, if $x \in X$ and the orbit of group of the element x, is subset of all images of x, under the permutations in G where $G_x = \{g(x)|g \in G\}$. A stabilizer of an element is said as a subgroup of G.

Now we shall prove a lemma that will be required in later parts of the article.

Lemma II-B.1: Let G is a permutation group action on set X. Suppose $x, y \in X$ belongs to the same group orbit, then the stabilizer subgroups G_x, G_y are conjugates.

Proof: Since x, y relates to the same orbit, $existsg \in G$ such a way $y = g(x)$ and let h stabilize x.

$$h(x) = x$$

$$h(g^{-1}(y)) = x$$

$$g(h(g^{-1}(y))) = g(x)$$

$$g.h.g^{-1}(y) = y$$

Thus, we have an injective homomorphism from $H \to H_x$ as follows - $h \in H_x, h \to ghg^{-1}$, whose inverse is $\forall h \in H$, $h \to g^{-1}hg$.

C. The Color Refinement Algorithm

The Color Refinement exploits the properties of graph automorphism group. Initially starting with coloring all the vertices of a graph by same color, two vertices are given different colors if they have differently colored neighbourhoods including multiplicities, until no further color changes can happen. Upon termination if two graphs produce different colored multi-set, then the graphs are declared as non-isomorphic. This is also called as the Naive Vertex Classification Algorithm.

Color refinement is a technique for Isomorphism of a Graph which has a step as shown below:

1) The Algorithm: Given graph G,
   a) Uniformly color every graph vertex, G with the same color.
   b) In every refinement step, suppose two vertices have the same color instead differently colored neighbourhood including multiplicities, then all these vertices get dissimilar colors.
   c) Apply this refinement step to the graph until the coloring is stable.
   d) Stable coloring can be reached in at most n refinement steps.
2) Application to Graph Isomorphism,
   a) Apply the Color Refinement algorithm to the disjoint union.
   b) Suppose, there exists a color c, with the stable coloring, the number of vertices of G with color c differs from the vertices in H, with color c, then the two graphs are not isomorphic.
3) Understanding the limitation: If the Color Refinement algorithm yields non-isomorphic, then the two graphs are non-isomorphic. However, two non-isomorphic graphs may produce the same stable coloring.
4) Equitable Partitions: Given a partition P for Graph G, its elements are specified as cells. Then, we call P as equitable.
   a) Let X be a cell, then $\forall u, v \in X$, $c(u)$ is the color of the vertex u.
   b) $\forall X \in P$, graph $G(X)$ is a regular.
   c) $\forall X, Y \in P$, graph $G(X, Y)$, called a bi-regular. Given a graph G, we need to apply the Color Refinement algorithm to G to get a stable coloring. Partition the vertex set $V(G)$ by means of their color, that is all vertices of the same color related to the same partition. Let X be a color partition for color c, trivially $\forall u, v \in cell()$, $X, c(u) = c(v)$, since we get a stable coloring $\forall u, v \in X, u, v$ has the same number of neighbors of each color, particularly of the color c. Hence $G(X)$ is regular.
   Also, let Y be another color partition of the color d. Since, all $u, v \in X$ have the same set of neighbors of the color d, and all $w, x \in Y$ have the similar number of neighbors of color c, $G(X, Y)$ is bi-regular. Thus, the Color Partitions are Equitable Partitions.
5) Graph Hierarchy based on Color Refinement: Despite its shortcomings, the Color Refinement algorithm remains a good tool in testing of isomorphism. The question, "For which pair of graphs does color refinement succeed in testing Graph Isomorphism", is the reason for the study of color refinement by considering all perspectives.
6) If the stable partition has only singleton classes, then the Color Refinement Algorithm will get ahead in testing isomorphism for the graph against all other graph. A graph G, can be called discrete if the Color Refinement algorithm, when applied on G, produces only singleton color classes. Graph G has a color refinement, suppose if it succeeds in distinguishing from all non-isomorphic graph H. A graph G, for which Color Refinement applies to can be called an amenable graph. Since, Color Refinement applies to all Discrete Graphs, all Discrete Graphs are Amenable Graphs. However, the complete graph $K3$ is Amenable but not Discrete.
   Tinhofer introduced the concept of fractional isomorphism for an approach of type linear programming to solve Graph Isomorphism. A graph is a compact when its polytope of fractional isomorphism is integral. A

graph is Compact when polytope in R^{n^2} consisting of the double stochastic matrices X such that $AX = XA$, where A is the adjacency matrix of G, has only integral extreme points. Graph G, is isomorphic to graph H, then polytope of the fractional isomorphism's from G to H is also integral, since we can also look at it as an automorphism from G to itself. Suppose G is not isomorphic to H, then we can say this polytope has no integral extreme point.

7) Tinhofer's Refinement Algorithm: Let the given graphs be G, H:
   a) Follow the Color-Refinement procedure on G, H. If the classification multi-set are different, output non-isomorphic.
   b) If all the color classes are singleton, then map the vertices according to colors, and declare the graphs as isomorphic.
   c) Pick a non-Singleton color class, and arbitrarily choose $u \in G$, $v \in H$, and assign with a new color. Go to step 1.

III. Literature Survey

A. Isomorphism Testing

The fields of chemistry, mathematics, and computing science, Graph isomorphism has a long history. GI is said to be in the NP complexity class; however, its exact location is unknown. It is still an open problem as to whether it lies in P or not and at the same time it is also unknown if it is an NP - complete. Some of the results on isomorphic graph are

- Number of Isomorphism - Mathon showed, counting set of graph isomorphisms is equivalent to solving GI problem [Mathon (1979)]. This add fuel to the statement that Graph Isomorphism Testing not an NP - complete, since it seems that counting counterparts of the NP - complete problems are even harder. However, this is also true for certain problems in P.
- Graph Automorphism versus Isomorphism - Mathon also showed that Graph Isomorphism Testing is equivalent to counting the non-trivial automorphism of a graph [Mathon (1979)]. It is interesting that GA reduces to GI, but the other way round reduction is not known.
- Polynomial Hierarchy fall – Suppose Graph Isomorphism is in NP - complete, polynomial hierarchy collapses to the second level ($\sum_2$) [Schning (1988)].
- Hardness Result - Toran shows that under DLOGTIME uniform AC^0 many-one reductions for the complexity classes, NL Graph Isomorphism is hard [Toran (2004)]. We shall have a look at his proof later, as it contains certain interesting ideas.

Since the graph isomorphism testing problem seems to be difficult when we consider all possible graphs, we can restrict the domain of the problem. For certain classes of graphs, it has been revealed that Graph Isomorphism testing can be performed in polynomial time.

- Planar Graphs [Datta et al. (2008)].
- With bounded genus Graphs [Miller (1983)].
- Graphs with bounded Eigen value multiplicities [babai1982isomorphism].
- Bounded Degree Graphs [Luks (1982)].
- Graphs with bounded tree widths [Bodlaender (1990)].

However, even restricting the domain to limited graphs does not work in all cases. In fact, for Bipartite Graphs, Chordal Graphs, graphs with bounded degeneracy or bounded expansion, isomorphism testing has been shown to be as hard as testing for general graphs.

B. Toran's Gadget

$DLOGTIME$ uniform, AC^0 many to one reductions of NL complexity classes. Toran proved, problem of graph isomorphism is hard, PL, for every Mod_kL. The key element is a graph gadget which proves that GI has required structure for encoding a modular addition gate. Utilizing the above fact, its possible to prove for any combinations $k \in N$ as well as AC^0 many to one reduction from the circuit problem value for addition mod k gates, is complete Mod_kL, to Graph Isomorphism.

Describe the reduction, the basic idea is to replace each Mod_kL gate with a gadget, and then build the entire graph.

1) The Gadget for a Gate: We assume that the gate takes in two inputs x, y and outputs $z = x+y\ (modulus)\ k$.
   a) **Vertices:** $\{x_i, y_i, z_i, u_{i,j} | 0 \leq i, j \leq (k-1)\}$
   b) **Edges:** $\{(x_i, u_{i,j}), (y_i, u_{i,j}), (u_{i,j}, z_{i+j}) | 0 \leq i, j \leq (k-1)\}$
   c) **Colors:** $X = \{x_i | 0 \leq i \leq (k-1)\}$.

Similarly, we define Y, U, Z and each of X, Y, U, Z are color classes with unique colors.

Lemma III-B.1 $\exists$ a unique automorphism ϕ that maps, $\forall 0 \leq a, b \leq (k-1)$,

- $\phi(x) = x_{i+a}$
- $\phi(y_j) = y_{i+b}$
- $\phi(z_i) = Z_{(i+a+b)}$

Theorem III-B.2 GI is hard for Mod_KL

Proof: We are supposed to develop a graph from the given circuit which contains Mod_kL gates. For every such gate we have the gadget as described above. Now, we shall look at the inter-gate connections. Say the output of gate g_i in Z_1 that is connected to the input x_2 of gate g_2, then we add edges (Z_i^1, x_i^2), $\forall 0 \leq i \leq k-1$.

For input variables $v^1, v^2, \ldots, v^n$ taking values $a_1, a_1, \ldots, a_n$, we have k nodes for each variable, $v_j^i | i \in [n], 0 \leq j \leq (k-1)$. By a straight forward induction, we get the circuit value equals b if and only if $\exists$ an automorphism that maps, $\forall i, v_j^i \rightarrow v_{(j \oplus a_i)}^i$ and for the output nodes $z_i \rightarrow z_{(i \oplus b)}$. Now what remains is to verify the existence of the automorphism. For that purpose, make two copies of G, and color z_0 in one $(G_b)^{'}$ and z_b in the order $(G_b)^{''}$ with the same color and then check for isomorphism. If we want to use Promise GI, then the input would be $(G_b^{'}, G_b^{''}), OR\ \cup_{i \neq b} (G_i^{'}, G_i^{''})$.

Theorem III-B.3: GI is hard for NL.

Proof: We know that Reachability (even for DAGs with fan-in less than 2) is NL - Complete. Since NL is closed under complementation, we shall reduce Not-reachability. Let $|V| = n$, now $\forall\ 0 < i < n+1$, transform G in C_i with addition modulo i gates. Now, output of the circuit map with $P\ mod\ i$ where P is combinations of paths from the beginning vertex to the last vertex, transform every C_i to graph G_{C_i} in which there is an automorphism mapping z_0^i to z_j^i when P is equal to $j\ modulus\ i$.

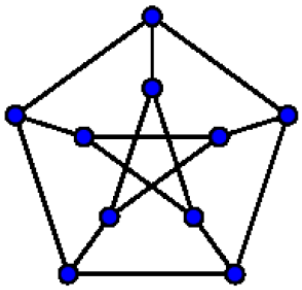

Fig. 1. The Petersen Graph

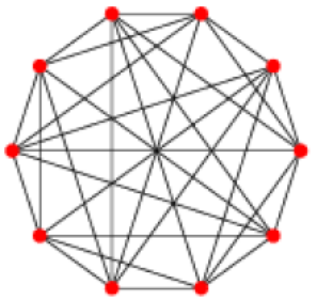

Fig. 2. Johnson Graph $J(n, 2)$.

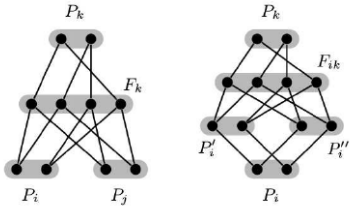

Fig. 3. CFI and IMP Gadgets.

C. Graph Isomorphism, Color Refinement and Compactness

This article explores the association amongst compactness and color refinement, studying the related algebraic and combinatorial properties of graph defined by Tinhofer [Arvind et al. (2015)]. They demonstrate that the graphs' corresponding classes create a hierarchy. We show that identifying every graph classes in P - hard.

1) The Hierarchy: **Theorem III-C.1:** Discrete $\subsetneq$ Amenable $\subsetneq$ Compact $\subsetneq$ Godsil $\subsetneq$ Tinhofer $\subsetneq$ Refinalble. We shall focus on Compact $\subsetneq$ Godsil $\subsetneq$ Tinhofer $\subsetneq$ Refinable.

2) Compact $\subsetneq$ Godsil: If G is compact then every equitable partition is an orbit partition.

Proof: Given a partition π with cells $C_1, C_2, \ldots, C_n$, the characteristic matrix P is a $|V| \times n$ matrix with $p_{ij} = 1$ if a vertex i belongs to the cell C_i and $p_{ij} = 0$ otherwise.

The directed graph with vertices $C_1, C_2, \ldots, C_r$ with b_{ij} arcs from C_i to C_j is called the quotient of G over π and is denoted by X/π. Now, since G is compact, we have $A(G)P = PA(G/\pi)$. Hence, we have $A(G/\pi) = (P^tP)^1P^tA(G)P$. The partition π is equitable if and only if $Q = P(P^tP)1P^t \in S(A)$. Since G is compact, $Q = \sum a_kR_k$, where $a_k \geq 0$ with $\sum a_k = 1$ and R_k are the permutation matrices that define automorphism of G.

Note that, ij - entry of Q is nonzero if and only if vertex i and j are in the same cell. But since $Q = \sum a_kR_k$, the ij - entry of Q is non-zero if and only if there is a permutation matrix in $S(A)$ whose ij - entry is non-zero, if and only if there is an automorphism of A taking vertex i to vertex j.

Hence, vertex i and vertex j are in the same cell if and only if there is an automorphism taking vertex i to vertex j. Thus, every cell of π is an orbit. Hence proved.

3) Godsil $\subsetneq$ Tinhofer: **Lemma III-B.3** Let G be a Godsil graph. Let H be isomorphic to G. After every Tinhofer color refinement step, there exists a color preserving automorphism from G to H.

Proof: We consider the base case first. Since G, H are isomorphic, and all vertices are uniformly colored, the base case is trivially true by the isomorphism ϕ. For the induction step, we first assume that until the i^{th} step, color preserving isomorphism ϕ_i exists and the partition is not discrete. Now, let the algorithm individualize $u \in V(G)$, $V \in V(H)$, if $\phi_i(u) = v, \phi_{i+1} = \phi_i$ Else $u, \phi_i^{-1}(v)$ are in the same color class of equitable partition.

Since G id Godsil, $\exists\alpha$, where α is an automorphism conserving the color partition, such that $(u) = \phi_i^{-1}(v)$, thus $\phi_{i+1} = \phi \bullet \alpha$. The Tinhofer algorithm terminates when singleton partitions are reached at some ϕ_k, Thus the algorithm successfully identifies G, H as isomorphic if G is Godsil, that means Godsil $\subsetneq$ Tinhofer. The Johnson Graph, $J(n, 2)$ is Tinhofer but not Godsil, thus the strict containment. Hence Proved.

4) Tinhofer $\subsetneq$ Refinable: Proof: If possible, let G be a Tinhofer graph that is not Refinable, thus $\exists\ u, v \in V(G)$ such that u, v both are in various orbits, but not separated by the CR partitioning. Hence, in the Tinhofer's algorithm applied on $G^{'}$, $G^{''}$, which are isomorphic copies of G, we will get different results based on the choice of vertices for individualization. Thus G cannot be a Tinhofer Graph.

D. Hardness Results on Graph Classes Testing membership

It is shown by Arvind et. al that membership testing in every class falls under P Hard [Arvind et al. (2015)].

Lemma III-D.1: Under uniform AC^0 reductions, problem of recognition for all classes presents in the hierarchy called a P - Hard.

Proof: We will reduce the Circuit Value Problem (CVP) to graph hierarchy. CVP is as follows:

Given a polynomial size circuit and the circuit input values, compute the circuit output. By De Morgen's laws, the negations can be brought down to the input gates, by doubling the circuit size. Thus, the rest of the circuit is monotone (containing and, or gates only). Since the input values of the gates are known, we can consider as constant value input gates. Thus, we shall consider monotone circuits with constant input gates. For every gate g_k in the circuit C, the graph G shall have a pair of vertices $\{P_k = a_k, b_k\}$. If g_k is a constant input gate value 1, then the vertices shall from singleton color classes, else P_k forms a color size of 2.

For every AND gate $g_k = g_i \wedge g_j$, we have the following: $P_k = \{a_k, b_k\}$, $F_k = \{c_k, d_k, e_k, f_k\}$ as color classes, with edge as follows:

- $(a_i, c_k), (c_k, a_j), (a_i, d_k), (d_k, a_j)$

- $(b_i, e_k), (e_k, b_j), (b_i, f_k), (f_k, b_j)$
- $(c_k, a_k), (a_k, f_k), (d_k, b_k), (b_k, e_k)$

and we shall refer to this as $CFI(P_k, P_i, P_j)$.

For every OR gate $g_k = g_i \vee g_j$, we have the following: $P_{i'}, P_{i''}, P_{j'}, P_{j''}, F_{ik}, F_{jk}, P_k$ and here $P_{i'}, P_{i''}$ are connected to P_i with parallel edges $(a_i, a_{i'}), (a_i, a_{i''})$ and $(a_j, a_{j'}), (a_j, a_{j''})$. $P_{i'}, P_{i''}, F_{ik}, P_k$ form a $CFI(P_k, P_{i'}, P_{i''})$, we shall call this $IMP(P_k, P_i)$ and similarly we also construct $IMP(P_k, P_j)$.

A few observations are noticed further. Firstly, after running CR on an AND gate g_k, a_k, b_k get different if and only if both pairs a_i, b_i and a_j, b_j have different colors. Secondly, after running CR on an AND gate g_k, a_k, b_k get different colors if and only if at least one of the pairs a_i, b_i and a_j, b_j have different colors. Thus, by induction P_k gets different colors if and only if g_k evaluates to 1.

Now, consider the final output gate P_l, join P_l to each of the zero valued input gates, if the circuit evaluates to 1, then P_1 have different colors and hence, all the zero valued inputs too. The 1 valued input already have different colors. Since the circuit is monotone, every vertex now forms a singleton color class. Now, we add an IMP (P_{l+1}, P_l) gadget at the top, note that if circuit evaluates to 1, the newly added gadget also gets partitioned into singleton classes, but if the circuit evaluates to 0, then the graph is no longer refinable. There is no automorphism in that maps $a_{l+1} \iff b_{l+1}$, but they belong to the same color class, thus the equitable partitioning after the CR algorithm and the automorphism orbit partitions are not the same. So G is not refinable. Hence, if $C = 0$, then G is non-refinable else if $C = 1$, then G is discrete. Thus all the classes in between have been proved to be P - Hard.

IV. A New Hierarchy of Graph Classes

A. Introduction

Arvind et. al have proposed the earlier mentioned Hierarchy of graph classes, for the purposes of isomorphism testing. Tinhofer graphs are a part of their hierarchy. Let us revisit Tinhofer graphs for a moment. Graph G is a Tinhofer graph, given graph $H \cong G$, for any sequence of individualizations, Tinhofer's algorithm declares them to be isomorphic. It is clear to us that, there exists at least one sequence of individualizations, that leads to "isomorphic". That is, for an isomorphism, if at every level we correctly guess a vertex and its image. Is there any way to relax Tinhofer's condition?.

Let p Tinhofer be the class of graphs for which Tinhofer's algorithm would work with a probability of at least p. Notice that in this setting, the Tinhofer graphs are exactly classified as 1.0-Tinhofer. Another extension to this line of thought is as follows. Say a graph is not Tinhofer, then up to how many minimum individualizations can the Tinhofer's algorithm run before it outputs the incorrect answer?

Let G be a graph with Cardinality n and $(0 \leq k \leq n)$ be an integer type. A graph is named k Tinhofer, if Tinhofer's algorithm does not go wrong for at least k steps, and there exists at least one sequence of individualizations where it outputs the wrong answer at the $(k+1)^{th}$ individualizations.

Let us explore this hierarchy further with some observations:

- Suppose a graph G is Tinhofer, G is k-Tinhofer for every k.
- Suppose a graph G is n-Tinhofer, then G is Tinhofer, thus $Tinhofer = n - Tinhofer$.
- $(n-1) - Tinhofer = n - Tinhofer$. Since there is only vertex left, individualising it is equivalent to not doing so.
- k - Tinhofer $\subseteq (k+1)$ - Tinhofer.
- 0-Tinhofer contains all graphs.

B. Algebraic Characterizations and Properties

Tinhofer's algorithm is a combinatorial approach towards the Graph Isomorphism problem. However, the Color Refinement algorithm paves way for our entry into the algebraic world. Now, we know that, for a refinable graph G, partition of equitable formed using the Color Refinement Algorithm is a partition of orbit for the underlying group of automorphism $Aut(G)$. For a Godsil graph H, all equitable partitions are orbits of some subgroups of the automorphism group $Aut(H)$. We also know that the Tinhofer class is in between the Godsil and Refinable classes. How does a Tinhofer graph look like, when viewed through the lens tinted by the color of algebra?.

Lemma IV-B.1: A graph G is Tinhofer, if and only if, for every vertex subset S, the stable partition obtained after individualizing the vertices of S coincides with the orbit partition of the point-wise stabilizer of S.

Proof: Let $|V| = G$, and G, a Tinhofer graph and S, an arbitrary subset of $V(G)$, $H(S) \leq Aut(G)$ be the point- wise stabilizer of S whose orbit partition is P_{H_S}. Let P_S be the stable partition obtained after running Tinhofer's algorithm on G while individualizing the vertices in the set S. Note that P_{H_S} is a refinement of P_S in both the graphs. If possible, let $\exists u, v \in V(G)$, that of the same color in P_S, but of different colors in P_{H_S}. Hence, there exists no automorphism that takes $u \to v$ and stabilizes S at the same time. Thus, if we personalise u in G and the copy of v, say $v^{'}$ in $G^{'}$, these will be non-isomorphism from $u \to v^{'}$. Hence the Tinhofer's algorithm will fail, and we get a contradiction.

Suppose G is a graph, then $\forall S \subseteq V(G)$, the stable partition after individualizing the vertices in S coincides with the orbit partition of the point wise stabilizer subgroup H_S that stabilizes the vertices in S. Now, no matter what vertices we choose for individualisation, we know that they belong to the same partition of orbit for a few subgroup of $Aut(G)$, which is the stabilizer subgroup H_S. Since the vertices belong to the same orbit, we know that there exists an isomorphism mapping them. Hence, the Tinhofer's algorithm will not fail. Now that we have an algebraic characterization of Tinhofer graphs, the natural question that comes to mind is whether this characterization can be extended to k-Tinhofer graphs. The extension arises from the fact that for a k-Tinhofer graph, it is not necessary that the characterization for Tinhofer graphs hold true for all the vertex subsets S. That means that a suitable relaxation on this condition will give us the required characterization. For a k-Tinhofer graphs, we want to take care of k individualizations. Hence, we relax the $\forall S \subseteq V(G)$ condition to $\forall S \subseteq V(G)$ where $|S| \leq k$.

Theorem IV-B.2: If for a given graph G, for every vertex subset S, $|S| \leq k$, the stable partition obtained after

individualizing the vertices of S collides with the partition of orbit for point-wise stabilizer of S, then G is k-Tinhofer. *Proof:* A very straight forward extension of the part 2 of the earlier proof works here. Theorem IV-B.2 only provides us with a enough condition for k-Tinhofer graphs. A graph G may be k-Tinhofer and still not satisfy the condition.

C. Refinability and the New Hierarchy

Graph G has refinable property when stable color partition obtained from Refinement of color accords with the partition of orbit for $Aut(G)$. We know from Arvind et. al paper that $Tinhofer \subsetneq Refinable$ Arvind et al. (2015)]. We shall now explore the relation between Refinable graphs and 1-Tinhofer graphs.

Theorem IV-C.1: Refinable $\subsetneq$ 1-Tinhofer
Proof: Assume refinable graph with many vertices say n. To show G is 1-Tinhofer, we have to show that no matter what individualization we make, Tinhofer's algorithm does not produce a wrong output after the first step. Let H be an isomorphic graph to G, after applying the Color Refinement Algorithm to both the graphs, say we get a partition Π_G, Π_H. Let $C_G \in \Pi_G$, $C_H \in \Pi_H$ be two color classes that have the same valency lists and the same cardinalitiy. Now let us individualize vertex $u_G \in C_G$ and $u_G \in C_G$ and $G \cong H$ via the mapping α and $\alpha(u_G) = v$, note that $v \in C_H$. Now, we know that G, H are refinable. Thus, the equitable partitions Π_G, Π_H are orbits of the automorphism groups of the graphs. Hence $\forall C \in \Pi_H$, $\forall (x, y) \in C$, $\exists a \in Aut(H)$, such that $a(x) = y$. Thus, $\exists a \in Aut(H)$, such that $a(v) = u_H$.

Now consider the following map $\beta = a \bullet \alpha$. Note that since α is an isomorphism and a an automorphism, β is also an isomorphism. Also $\beta(u_G) = (a \bullet \alpha) \bullet u_G = a \bullet (\alpha(u_G)) = a \bullet v = u_H$. Thus, there exists an isomorphism that maps u_G to u_H, since our choice of these vertices was arbitrary. $\forall u \in C_G, v \in C_H$, there occurs an isomorphism that takes u to v. Thus, for all choices of u_G, u_H individualizing the two vertices will lead us to equivalent partitions. Hence, the Tinhofer's algorithm won't go wrong for one step.

Theorem IV-C.2: A refinable graph G is 2-Tinhofer if for every vertex u, the stable partition obtained after individualizing u accords with the orbit partition of the point-wise stabilizer of u.

Proof: Assume G is a refinable graph on many vertices say n. With the above theorem, we know that G is 1 – Tinhofer. Let the graph H be isomorphic to G. Now, let $x \in V(G)$, $y \in V(H)$ be given vertices of the same color after application of the Color Refinement Algorithm, that are individualized in the first step. We know that there is an isomorphism a, from $G \to H$ such that $a(x) = y$. After individualization of x, y, we get new equitable partitions, that are the refinements of the earlier ones. We also know that the partitions are equivalent, or else Tinhofer's algorithm will declare the graphs as non-isomorphic and since G is 1Tinhofer, that will lead us to a contradiction. We know the new partitions coincide with the orbit partitions of the point-wise stabilizer sub-groups, that stabilize the individualized vertices. Assume A_x be stabilizer subgroup of $Aut(G)$, which stabilizes x. Note that, since x, y belongs to the similar orbit partition of the underlying automorphism group $a \bullet A_x \bullet a^{-1}$ is the stabilizer subgroup of $Aut(H)$, that stabilizes y. Now, for the second individualization, we will pick two vertices of the same color, from each of the graphs, say chose $x_1 \in V(G)$, $y_1 \in V(H)$, let $a(x_a) = y_2$ note that since an isomorphism takes x_1 to y_2 the color of the two vertices is same. Thus y_1, y_2 belongs to the same color partition in H. Hence, there exists an autmorphism b in the subgroup, that stabilizes y, such that $b(y_2) = y_1$.

D. Pushing the Boundaries

We have defined the hierarchy and understood its properties. We now provide a time algorithm of polynomial type for determining the existence of isomorphisms in the lower classes of the hierarchy. Let a graph G be $(n-k)$-Tinhofer, where k is a constant. We shall test for isomorphism between G and a given graph H.

1) If $V(G) \neq V(H)$, output is 0.
2) Apply Tinhofer's algorithm, until at most k vertices are not in singleton color classes.
3) Choose an arbitrary mapping between the remaining k vertices that respects the colors, and verify the isomorphism.
4) If the mapping is an isomorphism, output 'YES', else, go back to Step 3, and choose a different mapping.
5) If none of the mappings are isomorphism, output 'NO'.

Since G, is $(n-k)$-Tinhofer, Tinhofer's algorithm does not make an error when the number of vertices individualized are less than $(n-k)$. Thus, suppose an isomorphism exists between G, H, there exists a mapping in the remaining k vertices and this is isomorphism. Hence, the given graphs are non-isomorphic if none of the mappings is an isomorphism. There are at the most $k!$ mappings possible. Since k is a constant, the algorithm terminates in time which is equal to polynomial. Here we observe, the running complexity of the above algorithm will be less since we map vertices that only have same colors.

V. Other Contributions

A. Color Refinement Distance Lemma

The lemmas states that two vertices of the same color will have vertices with the same number of a color at any particular distance.

1) Definition-1: Neighbourhood of color c at a distance k of a vertex u. $N_u(k, c)$ = $\{v \in V \exists\ a\ k - length\ walk\ starting\ from\ u\ to\ v, v\ is\ closed\ to\ c\}$.

Lemma V-A.1: Assume u, v given vertices in color class X. It is the stable color partition obtained from a graph G, then $\forall$ (k, c), $N_u(k, c)$ = $N_v(k, c)$. *Proof:* Using induction with respect to distance. The base case $(k = 1)$ is true since the stable partition is an equitable partition. Assuming that the lemma is true for distance $k1$. We have $\forall c, |N_u(k-1, c)|$ = $N_v(k-1, c)$. Let S_u is vertices set in b which have neighbours of color c, S_v is defined similarly. For every vertex in S_u, we have a same-colored vertex in S_v, which will have a neighbour of color c. Hence proved.

B. Directed Color Refinement

In this segment we shall highlight the query of how does the color Refinement Algorithm work in the directed graphs setting. Let us first have a look directed graph. A graph is a named directed graph where every edge is directed from each of the vertices, Similarly, for vertex v, in-degree is equal to the count of edges incident on v, whereas, out-degree for vertex is count of edges which are directed from v. The in-neighbourhood of a vertex v is the set of vertices from which edges are incident on v. Similarly, the out-neighbourhood of the vertex is the set of vertices on which edges are incident from v. Now while applying Color Refinement to a directed graph, we are present with three options.

1) In-Color Refinement: Here, the basic distinguishing factor for two vertices to be assigned different colors based on vertex in-degree. We ignore vertex out-degree, and focus on the incoming edges only. Given a graph G,

- For G, uniformly color with the same to all vertices.
- In each refinement stage, when given vertices have the same color but it was colored differently in-neighbourhood including multiplicities, then these vertices get dissimilar colors.
- **Properties of Color Partition**
  - Every color class is monochromatic. Given any two vertices u, v from a partition class X, $c(u) = c(v)$.
  - Every color class is in-regular. Given a color partition X, then induced sub graph $G[X]$ is in- regular, that is given any vertices u, v, in-degrees are same.
  - For any given two color classes X, Y, the induced sub-graph $G[X, Y]$ creates a graph which is bipartite. Every vertex in X has in-degree which is same for the given Y, that is for any two vertices $u \in X$, the number of in-neighbours of u, v that belong to Y is equal.

2) Out-Color Refinement: Here, the basic distinguishing factor for two vertices to be assigned different colors is the vertices' out-degree. We ignore the in-degree of a vertex, and focus on the outgoing edges only. Given a graph G,

- Uniformly color all the vertices of the graph, G with the same color.
- In each further refinement, when given vertices have the same color but colored differently out-neighbourhood by considering multiplicities, these vertices get various colors.
- Apply the refinement step to the graph until the coloring is stable.
- **Properties of the Color Partition**
  - Every color class is monochromatic. Given any two vertices u, v from a partition class X, $c(u) = c(v)$.
  - Every color class is in-regular. Given a color partition X, then induced sub graph $G[X]$ is in regular, given any vertices say u, v, it has the same in-degree.
  - Any given color classes say X, Y, induced sub-graph $G[X, Y]$ forms a bipartite graph. All vertex present in X contains the same out-degree given Y.

3) InOut-Color Refinement: Here, the basic distinguishing factor for two vertices to be assigned different colors is the in-degree and the out-degree of the vertices. If either of these differ, the vertices shall be assigned different colors. Given a graph G,

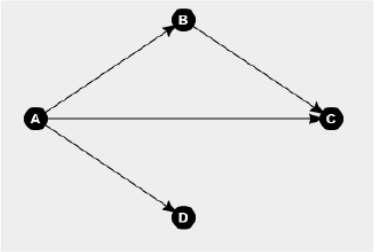

Fig. 4. Graph-1.

- Uniformly color the vertices of a graph, G with the same color.
- In each fine-tuning step, if two vertices have the same color but in a different way colored out-neighbourhood or in-neighbourhood including multiplicities, then these vertices get dissimilar colors.
- Apply the refinement step to the graph until the coloring is stable.
- **Properties of the Color Partition**
  - Every color class is monochromatic. Given any two vertices u, v from a partition class X, $c(u) = c(v)$.
  - Every color class is regular. Given a color partition X, then induced sub graph $G[X]$ is regular, that is given any two vertices u, v, the in-degrees and the out- degrees of u, v are same.
  - Given two color classes say X, Y, induced sub-graph $G[X, Y]$ forms bipartite graph. All vertex present in X has the same in-degree as well as out-degree by considering Y.

4) Relations between the Partitions: A natural question is that how are the three partitions related arises? While it intuitively seems that the in and out partitions should be equivalent. It turns out that this need not be the case always. We have found counter-examples. Consider the graph labelled as Graph-1 in figure-4.

The in-color refinement produces the partition $\{A\}$, $\{B, D\}$, $\{C\}$, where the out-color refinement produces the following partitions $\{A\}$, $\{B\}$, $\{C, D\}$. Define a partition intersections P_1, P_2 as follows. Given vertices u, v will belong to the same set when they are in the same classes in both the partitions. The in-out partition is a fine-tuning of the in-partition, out-partition and also of the intersection of the two. The intersection partition and the in-out partition need not be equivalent as seen from the example graph labelled as Graph 2. The in-partition would be $\{A\}$, $\{B, C\}$, $\{D, E, F\}$. The out-partition would be $\{A\}$, $\{B\}$, $\{C\}$, $\{D, E, F\}$ and their intersection would give precisely the out-partition whereas the in-out partition is each vertex in a different class.

VI. Conclusion

We have depicted and analysed a novel graph hierarchy based on Tinhofer's algorithm. The lowest class in this hierarchy is Tinhofer graphs, while the highest class spans all possible graphs, thus touching both ends of the spectrum. We have a time algorithm of polynomial type for testing

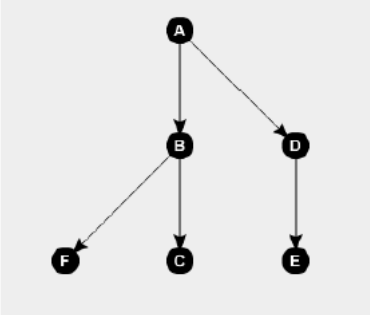

Fig. 5. Graph-2.

the existence of isomorphism for the lower classes in this hierarchy. We have proved an algebraic characterization for Tinhofer graphs, and extended it form a sufficient condition for k-Tinhofer graphs. Modifying the sufficient condition suitably to get a characterization for k-Tinhofer graphs would be the immediate sequel to this article.

Refinable graphs lie towards the top of the hierarchy, but below 1-Tinhofer. Refinable and $k - Tinhofer$ seem to be different classes that have intersecting points, for $k > 2$. As we gradually move down the hierarchy, i.e., increase k, at a point, all of them are refinable. The exact value of k, where this transition happens is still unknown. Finding examples for separating each of these classes is also something worth looking into.

The polynomial time algorithm has been presented for $(n - k)$ - Tinhofer graphs, where k is a constant. However, the algorithm's complexity depends upon the size of the color classes that are obtained after individualizations. Finding better upper bounds for this would amount to allowing non-constant values of k and still retaining the polynomial time complexity of the algorithm.

REFERENCES

[1] Arvind, V., J. Kobler, G. Rattan, and O. Verbitsky, On the power of color refinement. In Fundamentals of Computation Theory - 20th International Symposium, FCT 2015, Gdansk, Poland, August 17-19, 2015, Proceedings, 2015. URL: http://dx.doi.org/10.1007/978-3-319-22177-9_26.

[2] Bodlaender, H. L. (1990). Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, Journal of Algorithms, 11(4), 631–643.

[3] Cook, S. A., The complexity of theorem-proving procedures, In Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC '71. ACM, New York, NY, USA, 1971. URL: http://doi.acm.org/10.1145/800157.805047.

[4] Datta, S., N. Limaye, P. Nimbhorkar, T. Thierauf, and F. Wagner (2008), A logspace algorithm for canonization of planar graphs. CoRR, abs/0809.2319. URL: http://arxiv.org/abs/0809.2319.

[5] Ladner, R. E. (1975). On the structure of polynomial time reducibility. J, ACM, 22(1), 155–171, ISSN 0004-5411. URL: http://doi.acm.org/10.1145/321864.321877.

[6] Luks, E. M. (1982). Isomorphism of graphs of bounded valence can be tested in polynomial time, Journal of computer and system sciences, 25(1), 42–65.

[7] Mathon, R. (1979). A note on the graph isomorphism counting problem, Information Processing Letters, 8(3), 131–136.

[8] Miller, G. L. (1983). Isomorphism of k-contractible graphs. a generalization of bounded valence and bounded genus. Information and Control, 56(1), 1 – 20. ISSN 0019-9958. URL: http://www.sciencedirect.com/science/article/ pii/S0019995883800473.

[9] Schaefer, T. J., The complexity of satisfiability problems. In Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, STOC '78. ACM, New York, NY, USA, 1978. URL http://doi.acm.org/10.1145/800133.804350.

[10] Schning, U. (1988). Graph isomorphism is in the low hierarchy. Journal of Computer and System Sciences, 37(3), 312 – 323. ISSN 0022-0000. URL: http://www.sciencedirect.com/science/article/pii/0022000088900104

[11] Toran, J. (2004). On the hardness of graph isomorphism. SIAM Journal on Computing, 33(5), 1093–1108.