The Complexity of Modular Graph Automorphism

V. Arvind [*]
Institute of Mathematical Sciences

R. Beigel [†]
University of Illinois at Chicago

A. Lozano [‡]
Universitat Politècnica de Catalunya

Abstract

Motivated by the question of the relative complexities of the Graph Isomorphism and the Graph Automorphism problems, we define and study the modular graph automorphism problems. These are the decision problems mod_k-GA which consist, for each $k > 1$, of deciding whether the number of automorphisms of a graph is divisible by k. The mod_k-GA problems all turn out to be intermediate in difficulty between Graph Automorphism and Graph Isomorphism.

We define an appropriate search version of mod_k-GA and design an algorithm that polynomial-time reduces the mod_k-GA search problem to the decision problem. Combining this algorithm with an IP protocol, we obtain a randomized polynomial-time checker for mod_k-GA, for all $k > 1$.

1 Introduction

The Graph Isomorphism problem (GI) consists of determining whether two graphs are isomorphic. It is well known that GI is in NP, but despite decades of study by mathematicians and computer scientists, it is not known whether GI is in P or whether GI is NP-complete. Many researchers conjecture that GI's complexity lies somewhere between P and NP-complete. Related to GI are several other decision problems (some graph-theoretic and others group-theoretic in nature) that are similarly not known to be in P or NP-complete. One such problem which is closely related to GI is Graph Automorphism (GA): Deciding

[*]Madras 600113, India (email: arvind@imsc.ernet.in)

[†]Research performed at the University of Maryland and Yale University. Address: Dept. of Electrical Engineering and Computer Science, University of Illinois at Chicago, 851 S Morgan St, Chicago IL 60607-7053, USA. Supported in part by the National Science Foundation under grants CCR-8958528, CCR-9700417, and CCR-9796317 and by NASA under grant NAG 52895. Email: `beigel@eecs.uic.edu`. Web: `http://www.eecs.uic.edu/~beigel/`

[‡]L.S.I. Dept., Campus Nord, Mòdul C6, Jordi Girona 1–3, E08034-Barcelona, Catalonia, E. U. (email: antoni@lsi.upc.es). Supported in part by the E.U. through ESPRIT project (no. LTR 20244, ALCOM-IT) and by DGICYT (Koala project, no. PB95-0787), and CICYT (no. TIC97-1475-CE). Work completed while visiting the first author at the Institute of Mathematical Sciences.

whether a graph has a nontrivial automorphism. Regarding the relative complexities of GA and GI, it is known that GA is polynomial-time many-one reducible to GI. On the other hand, GI is not known to be even polynomial-time Turing reducible to GA (see [10] for these and related results). However, in [12] it is shown that GI is polynomial-time reducible to the problem of computing the number of automorphisms of a graph.

The notion of program checking was introduced by Blum and Kannan [3] as an algorithmic alternative to program verification. Since then the design of efficient checkers for various computational problems has rapidly grown into a discipline of algorithm design [3, 4]. One of the first program checkers in [3] was a randomized polynomial-time checker for GI. It is an outstanding open question in the area if NP-complete problems have efficient program checkers. This can be construed as another evidence that GI is not NP-complete. Later, in [11] it was shown that GA has a *nonadaptive* checker. In other words, the checker can make all its queries to the program in parallel, hence enabling it to be fast in parallel (in NC, to be precise). It is an open question whether GI too has a nonadaptive checker, and the apparent bottleneck here is that the search problem for GI is not known to be polynomial-time truth-table reducible to the decision problem for GI (i.e. the reduction is nonadaptive: it uses only parallel queries).

Thus, a natural next step in investigating the relationship between GI and GA is to consider exactly how much we need to know about the number of automorphisms of a graph in order to solve the Graph Isomorphism problem. This motivates us to define and study modular graph automorphism problems. Let $Aut(G)$ denote the automorphism group of the graph G.

Definition 1 *For any k, let mod_k-*GA $= \{G : |Aut(G)| \equiv 0 \pmod{k}\}$.

We show in Theorems 4 and 5 that for any $k > 1$, GA $\leq_m^p$ mod_k-GA $\leq_m^p$ GI; thus the mod_k-GA problems are intermediate in difficulty between GA and GI. It is an open question whether any of the mod_k-GA problems is polynomial-time equivalent to GA or GI. We conjecture that mod_k-GA is *not* polynomial-time equivalent to GA or GI, for any $k > 1$. An evidence that some of the mod_k-GA problems could be actually harder than GA is our observation that Tournament Isomorphism (graph isomorphism for tournament graphs) is many-one reducible to mod_2-GA. This follows from the fact that the automorphism group of any tournament is of odd size [10], which in turn implies that two tournaments are isomorphic iff the automorphism group of their disjoint union contains an order-two permutation (which must switch the two graphs).

The layout of the paper is as follows. Section 2 contains the preliminaries. In Section 3, we prove that the mod_k-GA problems are located between GA and GI. In Section 4, we show that search is polynomial-time Turing equivalent to decision for mod_k-GA, and in Section 5 we use this result in combination with an IP protocol for $\overline{mod_p\text{-GA}}$ to obtain an efficient program checker for mod_k-GA. Notice that although both GA and GI have program checkers (shown in [11] and [3] resp.) and mod_k-GA is intermediate in complexity, it does not necessarily imply that mod_k-GA has a program checker [3].

2 Preliminaries

In this paper by a graph we mean a finite directed graph[1] (see for example [8] or any other standard text on graph theory for basic definitions). For a graph G, let $V(G)$ denote its vertex set and $E(G)$ denote its edge set. A permutation π on the vertex set $V(G)$ of a graph G is an automorphism of G if $(u, v) \in E(G) \iff (\pi(u), \pi(v)) \in E(G)$. The set of automorphisms $Aut(G)$, of a graph G, is a subgroup of the permutation group on $V(G)$. The identity automorphism of any graph will be denoted by id.

Let X be a list of vertices in $V(G)$ for a given graph G. By $G_{[X]}$ we mean the graph G with distinct labels attached to the vertices in X. Given two lists of vertices $X, Y \subseteq V(G)$, the graphs $G_{[X]}$ and $G_{[Y]}$ have the same labels in vertices occupying the same relative positions in X and Y. It is not hard to see that in $G_{[X]}$ vertices of X are pointwise fixed in any automorphism[2]. Thus $Aut(G_{[X]})$ is isomorphic to the subgroup of $Aut(G)$ which pointwise fixes the vertices in X. Furthermore, given an automorphism of $Aut(G_{[X]})$ the corresponding automorphism of $Aut(G)$ can be efficiently (i.e. in polynomial time) constructed.

Definition 2 *Let $G_1, \ldots, G_n$ be n graphs.*

- *Let P_n be a directed simple path of n new vertices $v_1, v_2, \ldots, v_n$, where each vertex v_i is labeled with a single label l. The graph Path($G_1, \ldots, G_n$) is obtained by taking one copy of each of the graphs $G_1, \ldots, G_n$ and, for $1 \le i \le n$, attaching all the vertices of G_i to v_i.*
- *Let C_n denote the directed simple cycle on n new vertices $v_1, v_2, \ldots, v_n$, with each vertex v_i, $1 \le i \le n$, labeled with a single label l. The graph Cycle($G_1, \ldots, G_n$) is obtained by taking one copy of each of the graphs $G_1, \ldots, G_n$ and, for $1 \le i \le n$, attaching all the vertices of G_i to v_i.*

In both Path($G_1, \ldots, G_n$) and Cycle($G_1, \ldots, G_n$), since the new vertices $v_1, v_2, \ldots, v_n$ are labeled with l, any automorphism of these graphs must map the set $\{v_1, v_2, \ldots, v_n\}$ onto itself. Consequently, any automorphism of Path($G_1, \ldots, G_n$) (Cycle($G_1, \ldots, G_n$)) when restricted to $\{v_1, v_2, \ldots, v_n\}$ is an automorphism of P_n (C_n) This means that an automorphism of Path($G_1, \ldots, G_n$) cannot permute the copies of $G_1, \ldots, G_n$, while an automorphism of Cycle($G_1, \ldots, G_n$) can permute them but only along the cycle C_n.

The reducibilities discussed in this paper are the standard polynomial-time Turing and many-one reducibilities. Formal definitions of these and other standard notions in complexity theory can be found in [2, 1].

We finish this section with some complexity-theoretic concepts which will be used later. A set $A \subseteq \Sigma^*$ is a *d-cylinder* if there is an FP function OR that takes a list of strings $x_1, x_2, \ldots, x_m$ as argument and produces a string y such that

$$\mathrm{OR}(x_1, x_2, \ldots, x_m) = y \in A \iff \exists i \;:\; 1 \le i \le m \;:\; x_i \in A$$

[1]In this paper we consider the problems GI, GA, and mod_k-GA on directed graphs. However, all results of this paper hold for these problems on undirected graphs as well.

[2]Each label can be implemented with a graph gadget like a long path such that the overall size of the graph is still polynomially bounded. See, e.g. [10].

Similarly, a set $A \subseteq \Sigma^*$ is a *c-cylinder* if there is an FP function AND that takes a list of strings $x_1, x_2, \ldots, x_m$ as argument and produces a string y such that

$$\text{AND}(x_1, x_2, \ldots, x_m) = y \in A \iff \forall i \ : \ 1 \leq i \leq m \ : \ x_i \in A$$

Now, we recall that GI satisfies both properties[3].

Proposition 3 [5, 11] GI *is a d-cylinder and a c-cylinder.*

The relative complexity of decision and search for NP problems is well studied [2, 1]. For instance, it is known that search and decision are polynomial-time Turing equivalent for all NP-complete problems. In particular, we recall that for GI, search is polynomial-time Turing reducible to decision [13] whereas for GA a stronger result holds: search is *nonadaptively* polynomial-time reducible to decision [11].

3 Locating the mod_k-GA Problems

We show in this section that mod_k-GA is located between GA and GI, for all $k > 1$.

Theorem 4 *For all $k > 1$,* GA $\leq_m^p$ mod_k-GA.

Proof. Given a graph G, we define for every i, j with $1 \leq i < j \leq n$, the graph $H_{i,j} = \text{Cycle}(G_{[\{i\}]}, G_{[\{j\}]}, \ldots, G_{[\{j\}]})$ which contains one copy of $G_{[\{i\}]}$ and $k-1$ copies of $G_{[\{j\}]}$. Further, let H be obtained by applying the Path operator to all the graphs $H_{i,j}$ with $1 \leq i < j \leq n$. We claim that G has a nontrivial automorphism if and only if H is in mod_k-GA.

Suppose that G has a nontrivial automorphism φ. There exist two vertices i and j such that $\varphi(i) = j$. Notice that $H_{i,j}$ has the following nontrivial automorphism α that cyclically permutes the k graphs in $\text{Cycle}(G_{[\{i\}]}, G_{[\{j\}]}, \ldots, G_{[\{j\}]})$ as follows. The automorphism α maps the first graph $G_{[\{i\}]}$ to $G_{[\{j\}]}$ by φ. It maps each of the first $k-2$ copies of $G_{[\{j\}]}$ to the next copy of $G_{[\{j\}]}$ by the identity automorphism. Finally, α maps the last copy of $G_{[\{j\}]}$ back to $G_{[\{i\}]}$ by the automorphism φ^{-1}.

The order of α is k since the vertices in $H_{i,j}$ are moved in a cyclic way through the different k subgraphs. In fact, the permutation α is a product of a bunch of k-cycles. Thus $H_{i,j} \in mod_k$-GA. Since $|Aut(H)| = \prod_{1 \leq i < j \leq n} |Aut(H_{i,j})|$, it follows that $H \in mod_k$-GA.

For the converse, assume that $H \in mod_k$-GA. Then, H has a nontrivial automorphism, say, α. Notice that α must induce an automorphism β in one of its subgraphs $H_{i,j}$. Since $H_{i,j} = \text{Cycle}(G_{[\{i\}]}, G_{[\{j\}]}, \ldots, G_{[\{j\}]})$, there are two possibilities: either β induces a nontrivial automorphism of $G_{[\{i\}]}$ or $G_{[\{j\}]}$, or else β maps the copy of $G_{[\{i\}]}$ to some copy of $G_{[\{j\}]}$. In either case, it is clear that we get a nontrivial automorphism of G. ■

Mathon [12] has shown that $|Aut(G)|$ is polynomial-time computable with GI as oracle. From this it easily follows that mod_k-GA $\leq_T^p$ GI. In the next theorem, we strengthen this to a $\leq_m^p$-reduction using some permutation group theory.

[3]Elsewhere in the literature, e.g. [10], these properties are called OR and AND functions respectively.

Theorem 5 *For all $k > 1$, mod_k-GA $\leq_m^p$ GI.*

We need a couple of definitions and group-theoretic lemmas before we prove Theorem 5. Let A be a subgroup of S_n and let $[n]$ denote the set $\{1, 2, \ldots, n\}$. A subset $X \subseteq [n]$ is *A-invariant* if $g(X) = X$ for all $g \in A$. If $X \subseteq [n]$ is A-invariant then consider the action of A *restricted* to X. This gives rise to a subgroup of the symmetric group S_X, which we denote by A^X. A useful property that is obvious is that $|A^X| \leq |A|$, for all A-invariant sets X.

Lemma 6 *Let A be a subgroup of S_n s.t. $|A| = m$. Then there exists an A-invariant subset $X \subseteq [n]$ with $|X| \leq m \log m$, such that A is isomorphic to A^X.*

Proof. Consider the following procedure for constructing the set X:

```
$X \leftarrow \emptyset$;
**while** $\exists i \notin X : |A^X| < |A^{X \cup A(i)}|$ **do**
/* $A(i)$ denotes the orbit of $i$ under $A$ */
{
    Pick such an $i$;
    $X \leftarrow X \cup A(i)$
}
```

First we claim that, as a loop invariant, X is always an A-invariant subset of $[n]$. To see this, notice that it holds at the beginning when X is empty, and if X is A-invariant then so is $X \cup A(i)$ since we are including an entire A-orbit in the set.

Next, suppose X is A-invariant and $i \notin X$ is some index. Consider the mapping φ from $A^{X \cup A(i)}$ to A^X which maps an element of $A^{X \cup A(i)}$ to its restriction to X. Since X is A-invariant, it is easy to verify that φ is a surjective homomorphism from $A^{X \cup A(i)}$ to A^X. It follows that $|A^X|$ divides $|A^{X \cup A(i)}|$. Suppose now, at some stage of the while loop, i is an index such that $|A^X| < |A^{X \cup A(i)}|$. Then it must hold that $2|A^X| \leq |A^{X \cup A(i)}|$. Thus we have argued that every time X increases by including an orbit $A(i)$ in it, the size of the group A^X increases by at least a factor of 2. Thus the assignment $X \leftarrow X \cup A(i)$ is executed at most $\log m$ times, implying also that the procedure must stop. Since the size of any orbit $A(i)$ is bounded by $|A|$, it follows that the procedure stops with an A-invariant set X such that $|X| \leq m \log m$. Let X be the set computed when the while-loop is exited. To complete the proof we must show that A^X is isomorphic to A. Consider the canonical surjective homomorphism ψ from A to A^X, which maps a given element of A to its corresponding restriction to X. To show that this homomorphism is an isomorphism we only need to argue that $Ker(\psi)$ is (id). Suppose $g \in Ker(\psi)$ is a nontrivial element. Then there is $i \notin X$ such that $g(i) \neq i$. This in turn implies that the surjective homomorphism φ from $A^{X \cup A(i)}$ to A^X which maps an element of $A^{X \cup A(i)}$ to its restriction to X, has a nontrivial kernel with $g \in Ker(\varphi)$. Consequently, $|A^X| < |A^{X \cup A(i)}|$. Thus, both X and i satisfy the while-loop condition contradicting the fact that the while loop has terminated. This completes the proof of this lemma. ∎

Lemma 7 *Let A be a finite group. Let $X = \{a_1, a_2, \ldots, a_t\}$ and $Y = \{b_1, b_2, \ldots, b_t\}$ be two subsets of A such that $\langle X\rangle \cap \langle Y\rangle = (\mathrm{id})$ and $a_ib_j = b_ja_i$, for $1 \le i, j \le t$. Then $|\langle X\rangle|$ divides the order of the group $\langle\{a_ib_i : 1 \le i \le t\}\rangle$.*

Proof. Let H denote the subgroup of A generated by $\{a_ib_i \mid 1 \le i \le t\}$, K denote the subgroup of A generated by $\{a_i \mid 1 \le i \le t\}$, and L denote the subgroup of A generated by $\{b_i \mid 1 \le i \le t\}$. Notice that, since $a_ib_j = b_ja_i$, for $1 \le i, j \le t$, we have $KL = LK$ and therefore the set KL is actually a subgroup of A. Next, notice that, by definition of H, any $x \in H$ can be written as a product of elements from the generator set $\{a_ib_i \mid 1 \le i \le t\}$. Using $a_ib_j = b_ja_i$, for $1 \le i, j \le t$ as a rewrite rule, this product of generators expressing x can be rewritten as ay, where $a \in K$ and $y \in L$. It follows that $H \subseteq KL$. Consider the following map ψ from the group H to the group K defined as follows:

$$\forall x \in H \ : \ \psi(x) = a \text{ where } x = ay, \text{ with } a \in K \text{ and } y \in L$$

We claim that ψ is a well-defined surjective homomorphism from H to K. We first show that ψ is well-defined. Suppose there are two distinct elements $a, a' \in K$ such that $x = ay = a'y'$ for elements $y, y' \in L$. This implies, by cancelation laws, that $a^{-1}a' = yy'^{-1}$, which belongs to both K and L. Since $K \cap L = (\mathrm{id})$, we have $a = a'$. Thus ψ is well-defined. To see that ψ is a homomorphism is routine: we can easily check that $\psi(xx') = \psi(x)\psi(x')$ and that $\psi(x^{-1}) = (\psi(x))^{-1}$ hold using the rewrite rules $a_ib_j = b_ja_i$, for $1 \le i, j \le t$. To see that ψ is surjective, let $a \in K$ be any element. We can express a as a product $\Pi_{1\le r\le m} a_{i_r}$ for indices $i_r \in [t]$. Consider the element $x = \Pi_{1\le r\le m} a_{i_r}b_{i_r} \in H$. It is easy to see that $\psi(x) = a$.

Thus by the fundamental theorem of homomorphisms it follows that $H/Ker(\psi)$ is isomorphic to K. Therefore, $|H/Ker(\psi)| = |K|$. It follows that $|K|$ divides $|H|$ which proves the lemma. ■

Proof of Theorem 5

First, we argue that it suffices to show that mod_{p^l}-GA $\le^p_m$ GI for all prime p and $l > 0$. To see this, let $\prod_{1\le j\le r} p_j^{l_j}$ be the prime factorization of k. Clearly, a graph $G \in mod_k$-GA iff $G \in \bigcap_{1\le j\le r} mod_{p_j^{l_j}}$-GA. Thus, if $mod_{p_j^{l_j}}$-GA $\le^p_m$ GI for $1 \le j \le r$, it follows that mod_k-GA $\le^p_m$ GI, since GI is a c-cylinder.

We first prove a useful group-theoretic claim. Let G be a graph on n vertices and f be a partial permutation on $[n]$ (i.e. f is defined on a subset of the domain $[n]$ and can be extended to a permutation in S_n). Then we call f a *partial automorphism* of G if f can be extended to an automorphism of G.

Claim. *Let p be a fixed prime and $l > 0$. A graph G on n vertices is in mod_{p^l}-GA if and only if there exist a set $X \subseteq [n]$ with $|X| \le p^l(\log p^l)$ and a subgroup $K = \{a_1, a_2, \ldots, a_{p^l}\}$ of S_X such that each $a_i \in K$ is a partial automorphism of G.*

Proof. Let $G \in mod_{p^l}$-GA be an n vertex graph. Since p^l divides $|Aut(G)|$, by Sylow's theorem $Aut(G)$ has a subgroup A of size p^l. By Lemma 6 there is an A-invariant set $X \subseteq [n]$ with $|X| \le p^l(\log p^l)$, such that A^X is isomorphic to A. Let $A^X = \{a_1, a_2, \ldots, a_{p^l}\}$. Furthermore, it also follows that A^X is a subgroup of S_X where each $a_i \in A^X$ is a partial automorphism of G. Conversely, suppose there is $X \subseteq [n]$ with $|X| \leq p^l(\log p^l)$ and a subgroup $K = \{a_1, a_2, \ldots, a_{p^l}\}$ of S_X where each $a_i \in K$ is a partial automorphism of G. Then for each i with $1 \leq i \leq p^l$, there is a $b_i \in S_{[n]-X}$ such that $a_i b_i \in Aut(G)$. We can now apply Lemma 7 to the elements $\{a_i\}_{1 \leq i \leq p^l}$ and $\{b_i\}_{1 \leq i \leq p^l}$, since the required conditions are fulfilled. Consequently, $|\langle\{a_i b_i : 1 \leq i \leq p^l\}\rangle|$ is divisible by p^l. Since $\langle\{a_i b_i : 1 \leq i \leq p^l\}\rangle$ is a subgroup of $Aut(G)$, it follows that p^l divides $|Aut(G)|$. ■

Now, note that the language $B = \{(G, f) : f$ is a partial automorphism of the graph $G\}$ is $\leq_m^p$-reducible to GI (for details see [10]). We will give a truth-table reduction from mod_{p^l}-GA to B, where the truth-table is a disjunction of conjunctions. Since the language B is $\leq_m^p$-reducible to GI and since GI is both a c-cylinder and a d-cylinder, it follows that mod_{p^l}-GA is $\leq_m^p$-reducible to GI. We describe below the said reduction of mod_{p^l}-GA to B as a logical expression, which is easily seen to describe a disjunction-of-conjunctions kind of truth-table reduction:

$$\begin{aligned} G \in mod_{p^l}\text{-GA} \iff & (\exists\, X \subseteq [n] : |X| \leq p^l \log p^l) \\ & (\exists \text{ subgroup } K < S_X : |K| = p^l)(\forall a \in K)[(G, a) \in B] \end{aligned}$$

This completes the proof of Theorem 5. ■

4 Computing Solutions for mod_k-GA Instances

The goal of this section is to design a polynomial-time algorithm that reduces the search problem for mod_k-GA to the decision problem. Consider mod_k-GA for an arbitrary $k > 1$. Notice that if the prime factorization of k is $\prod_{1 \leq i \leq m} p_i^{e_i}$, then the natural NP witness of the membership of a graph G in mod_k-GA is a collection of m subgroups $\{A_1, A_2, \ldots, A_m\}$ of $Aut(G)$ where, for each i, A_i is of order p_i^{ei}, and A_i is listed as a set of permutations. We consider such a witness as a solution for G for the mod_k-GA search problem and we design a polynomial-time algorithm that computes this witness for any given instance of mod_k-GA with oracle access to the mod_k-GA decision problem.

In the following lemma we introduce one of the two last graph gadgets which we will need in order to prove the main result of this section.

Lemma 8 *Given t graphs $G_1, G_2, \ldots, G_t$, each with n nodes, we can construct in polynomial time a new graph $Paste(G_1, G_2, \ldots, G_t)$ such that the following properties hold.*

1. *A permutation $\psi \in S_n$ is an automorphism of $Paste(G_1, G_2, \ldots, G_t)$ iff there is a permutation $\pi \in \bigcap_{1 \leq i \leq t} Aut(G_i)$ such that ψ restricted to G_i is π, for $1 \leq i \leq t$.*

2. *Let p be a prime. $Paste(G_1, G_2, \ldots, G_t)$ has an automorphism of order p iff there is an order-p permutation $\pi \in \bigcap_{1 \leq i \leq t} Aut(G_i)$.*

3. *Given $\psi \in Aut(Paste(G_1, G_2, \ldots, G_t))$ we can in polynomial time construct the corresponding $\pi \in \bigcap_{1 \leq i \leq t} Aut(G_i)$.*

Proof. Notice that parts 2 and 3 of the lemma are both direct consequences of part 1. Thus it suffices to prove the first part. The graph $Paste(G_1, G_2, \ldots, G_t)$ basically consists of one copy of each of $G_1, G_2, \ldots, G_t$. Furthermore, for $1 \leq i \leq t$ we color the nodes of the copy of G_i using color C_i. This forces every automorphism of the new graph to map the copy of G_i to itself. Next, we use n distinct labeling nodes L_j, $1 \leq j \leq n$, as follows: from the jth node of every graph G_i put a long path of some fixed length N to node L_j. This ensures that for any automorphism of $Paste(G_1, G_2, \ldots, G_t)$, if node j_1 is mapped to node j_2 in G_i then j_1 gets mapped to the node j_2 also for each $G_{i'}$, $i' \neq i$. This construction guarantees the following: given an automorphism ψ of $Paste(G_1, G_2, \ldots, G_t)$, there is a permutation $\pi \in \bigcap_{1 \leq i \leq t} Aut(G_i)$ such that ψ restricted to G_i is π, for $1 \leq i \leq t$. This proves the lemma. ∎

Before we proceed we need to recall a definition.

Definition 9 [9] *Let $\pi \in S_n$ be a permutation. The* cycle graph *of π is the directed graph $G = ([n], E)$, where $(i, j) \in E$ iff $\pi(i) = j$.*

We next recall a lemma from [9].

Lemma 10 [9] *If G is the cycle graph of $\pi \in S_n$ then $Aut(G)$ is precisely the set of all permutations in S_n that commute with π.*

The second graph gadget needed is the following.

Lemma 11 *Let G be a graph on n nodes and $S = \{g_1, g_2, \ldots, g_t\} \subseteq S_n$ be a set of permutations. Further let $\mathcal{C} = \{C_1, C_2, \ldots, C_s\} \subseteq S_n$ be a set of pairwise disjoint cycles, p be a fixed prime, and τ be a permutation on $[t]$. Then we can compute in time polynomial in n a graph $Comb(\tau, G, S, \mathcal{C}, p)$ such that $Comb(\tau, G, S, \mathcal{C}, p) \in mod_p$-GA iff one of the following holds.*

1. *G has a nontrivial automorphism π of order p such that $\pi g_i \pi^{-1} = g_i$, for $1 \leq i \leq t$, and such that $\pi(x) = x$ for all $x \in \bigcup_{1 \leq i \leq s} C_i$.*
2. *G has a nontrivial automorphism π such that $C_1, C_2, \ldots, C_s$ are cycles of π and such that $\pi g_i \pi^{-1} = g_{\tau(i)}$, for $1 \leq i \leq t$.*

Proof. Let the composition $C_1 C_2 \cdots C_s$ of the cycles of $\mathcal{C}$ be denoted by $\psi \in S_n$. Further, let G' denote the graph obtained from G by coloring each node $x \in \bigcup_{1 \leq i \leq s} C_i$ with a distinct color n_x. Similarly, let G'' denote the graph obtained from G by coloring each node $\psi(x) \in \bigcup_{1 \leq i \leq s} C_i$ with the color n_x, for each x (where n_x is used to color node x in G').

Now, let $H = Paste(G', G_1, G_2, \ldots, G_t)$, where G_i is the cycle graph of g_i, for $1 \leq i \leq t$. Similarly, let $K = Paste(G'', G_{\tau(1)}, G_{\tau(2)}, \ldots, G_{\tau(t)})$. Finally, we put one copy of H and $p-1$ copies of K together to build the graph $Cycle(H, K, \ldots, K)$ (in which we have $p-1$ copies of K). This graph $Cycle(H, K, \ldots, K)$ is defined to be $Comb(\tau, G, S, \mathcal{C}, p)$.

Suppose $Comb(\tau, G, S, \mathcal{C}, p) \in mod_p$-GA. Now, suppose the first of the above two properties does not hold for G. We will prove that the second property must hold. Let ρ be an order-p automorphism of the graph $Comb(\tau, G, S, \mathcal{C}, p)$. Since the first property does not hold, notice that the order-p automorphism ρ of the graph $Cycle(H, K, \ldots, K)$ cannot map H to itself and each copy of K to itself. Therefore, since p is prime, ρ must permute the p graphs in the list $(H, K, \ldots, K)$ by a p-cycle. More precisely, ρ can be seen as a p-tuple $(\rho_1, \rho_2, \ldots, \rho_p)$ of permutations $\rho_i \in S_n$, $1 \leq i \leq p$, where ρ_1 maps the copy of H into a copy of K, and the permutations $\rho_2,\ldots,\rho_{p-1}$ map a copy of K into a fresh copy of K, and finally ρ_p maps a copy of K back to H. Recall that $H = Paste(G', G_1, G_2, \ldots, G_t)$ and $K = Paste(G'', G_{\tau(1)}, G_{\tau(2)}, \ldots, G_{\tau(t)})$, and observe that ρ_1, which maps H to K, is forced due to the color labels to map G' to G'' and to map G_i to $G_{\tau(i)}$ for each i, $1 \leq i \leq t$. Thus, ρ_1 is an automorphism of G that has $C_1, C_2, \ldots, C_s$ as its cycles, and $g_i \rho_1 g_i^{-1} = g_{\tau(i)}$, for $1 \leq i \leq t$ as claimed by the second property.

For the converse implication, suppose the first property holds. Let π be an order-p automorphism of G satisfying the first property. Consider the permutation ν of the nodes of the graph $Comb(\tau, G, S, \mathcal{C}, p)$, where the copy of H and each copy of K is mapped to itself under π. Clearly, ν is an order-p automorphism of $Comb(\tau, G, S, \mathcal{C}, p)$. Next, suppose that the first property fails and the second property holds. Again, let π denote the automorphism of G satisfying the second property. Consider the permutation μ of the nodes of $Comb(\tau, G, S, \mathcal{C}, p)$, which maps the copy of H into the first copy of K according to π, and then successively maps the first $p-2$ copies of K by the identity permutation into the corresponding next copy of K in the cyclic order, and finally maps the last copy of K to H according to π^{-1}. Observe that the permutation μ is in fact a product of disjoint p-cycles: the p-cycles are the orbits of vertices of H. It follows that μ is an order-p automorphism of $Comb(\tau, G, S, \mathcal{C}, p)$. ■

The next theorem is the main result of this section. Its proof draws on group-theoretic results concerning p-groups.

Theorem 12 *For any prime p, there is a polynomial-time algorithm $\mathcal{A}_k$ with mod_p-*GA *as oracle such that given a graph $G \in mod_{p^k}$-*GA *as input, the algorithm $\mathcal{A}_k$ lists out the elements of an order-p^k subgroup of $Aut(G)$.*

We will prove Theorem 12 by induction on k. We first take care of the base case (when $k = 1$) in the following lemma.

Lemma 13 *For any prime p, there is a polynomial-time algorithm $\mathcal{A}_1$ with mod_p-*GA *as oracle such that given a graph $G \in mod_p$-*GA *as input, the algorithm $\mathcal{A}_1$ outputs a cyclic group of order p contained in $Aut(G)$.*

Proof. For any list of vertices $X = \{i_1, \ldots, i_m\}$, let $r(X)$ be a right shift of X, this is $r(X) = \{i_m, i_1, \ldots, i_{m-1}\}$. Consider the following algorithm, which computes an order-p automorphism of an input graph $G \in mod_p$-GA.

Algorithm $\mathcal{A}_1$:

```
**input** $G$;
**if** $G \notin mod_p$-GA **then stop**;
$X \leftarrow \emptyset$;
```

```
for $i = 1$ to $|V(G)|$ do
    if $G_{[X \cup \{i\}]} \in mod_p$-GA then $X \leftarrow X \cup \{i\}$;
$S \leftarrow V(G) - X$;
$\mathcal{C} \leftarrow \emptyset$;
$G' \leftarrow G_{[X]}$; $G'' \leftarrow G_{[X]}$;
for each $p$-cycle $C \subseteq S - \bigcup_{D \in \mathcal{C}} D$ do
    if $Cycle(G'_{[C]}, G''_{[r(C)]}, \ldots, G''_{[r(C)]}) \in mod_p$-GA then
    /* There are $p-1$ copies of $G''_{[r(C)]}$ in the above Cycle definition */
    {
        $G' \leftarrow G'_{[C]}$; $G'' \leftarrow G''_{[r(C)]}$;
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$
    };
output the order-$p$ automorphism consisting of $p$-cycles $\mathcal{C}$ and fixed-point set $X$
```

We now prove the correctness of the above algorithm. Notice that the first for-loop takes $G \in mod_p$-GA as input and computes the graph $G_{[X]} \in mod_p$-GA with X as its set of fixed points (such that no more points can be fixed preserving membership in mod_p-GA). We have to show that when the algorithm stops it outputs an order-p automorphism which has $\mathcal{C}$ as its collection of p-cycles and X as its fixed-point set. To begin with, notice that any order-p automorphism with X as its fixed-point set is a product of disjoint p-cycles and 1-cycles (corresponding to elements of X).

We will prove this by showing as loop invariant that at each stage there is an order-p automorphism of G that contains $\mathcal{C}$ among its p-cycle set and contains X in its fixed point set. Clearly, before the loop is entered, there is an order-p automorphism of $G_{[X]}$ with $\mathcal{C} = \emptyset$ as subset of its p-cycle set. Suppose this property holds at the beginning of some iteration of the for-loop. Suppose in the next iteration a new p-cycle C gets included in $\mathcal{C}$. We have to show that there is an order-p automorphism of G with X as fixed-point set and such that $\mathcal{C} \cup \{C\}$ is contained in its p-cycle set. Consider $Cycle(G'_{[C]}, G''_{[r(C)]}, \ldots, G''_{[r(C)]})$, which is in mod_p-GA. Notice that the corresponding order-p automorphism ψ of $Cycle(G'_{[C]}, G''_{[r(C)]}, \ldots, G''_{[r(C)]})$ cannot map the copy of $G'_{[C]}$ to itself since $G'_{[C]}$ cannot have order-p automorphisms (because it forces G to have order-p automorphisms with $X \cup C$ as fixed points). Thus ψ must map the p graphs in $Cycle(G'_{[C]}, G''_{[r(C)]}, \ldots, G''_{[r(C)]})$ by a p-cyclic rotation. In particular, it implies that ψ maps $G'_{[C]}$ to some copy of $G''_{[r(C)]}$. Hence, ψ restricted to the nodes of G yields an automorphism ϕ of G with X as fixed-point set and such that $\mathcal{C} \cup \{C\}$ is contained in the p-cycle set of ϕ.

By induction, it follows that when the loop is exited we have an order-p automorphism which is completely specified: $\mathcal{C}$ is its collection of p-cycles and X is the fixed point set. ■

Proof of Theorem 12

We will prove the theorem by induction on k. Notice that the base case for $k = 1$ is proven in Lemma 13. More precisely, the induction hypothesis is the following:
Suppose that we have a polynomial time algorithm $\mathcal{A}_{k-1}$ with oracle mod_p-GA that computes an order-p^{k-1} subgroup of $Aut(G)$ given a graph $G \in mod_{p^{k-1}}$-GA as input.

We now prove the induction step by designing a polynomial-time algorithm with oracle mod_p-GA that, given as input a graph $G \in mod_{p^k}$-GA, computes the elements of an order-p^k

subgroup of $Aut(G)$. The induction hypothesis gives us the algorithm $\mathcal{A}_{k-1}$ using which we can compute in polynomial time an order-p^{k-1} subgroup of $Aut(G)$ (call it S_{k-1}). Let

$$S_{k-1} = \{g_1, g_2, \ldots, g_{p^{k-1}}\}$$

We first recall the following result from the theory of p-groups (for instance, it is well-known consequence of Sylow's theorems) [7]:

Proposition 14 *If A is a finite group such that p^k divides $|A|$ for some prime p, then for every subgroup H of A such that $|H| = p^{k-1}$ there exists a subgroup K of A such that $|K| = p^k$ and H is a normal subgroup of K.*

Let now G be a graph such that $G \in mod_{p^k}$-GA. Furthermore, let S_{k-1} be a subgroup of $Aut(G)$ such that $|S_{k-1}| = p^{k-1}$. An immediate consequence of this proposition is the following fact.

Fact 15 *There is a subgroup S_k of $Aut(G)$ of order p^k such that S_{k-1} is a normal subgroup of S_k.*

The quotient group S_k/S_{k-1} has p elements, more explicitly we can write it as

$$S_k/S_{k-1} = \{S_{k-1}g, \ldots, S_{k-1}g^p = S_{k-1}\}, \text{ for some } g \in S_k$$

As a first step to designing the required algorithm for computing S_k we prove the following claim.

Claim. $G \in mod_{p^k}$-GA *iff there exists $g \in Aut(G)$ such that the following hold*

1. $g \notin S_{k-1}$.

2. $o(g) = p^l$ *for some* $l \leq k$.

3. $S_{k-1}g = gS_{k-1}$

Proof. Clearly the forward direction of the claim is the fact stated above. To prove the reverse implication suppose there exists $g \in Aut(G)$ satisfying the above three conditions. Consider the group H generated by the set $S_{k-1} \cup \{g\}$. Since $S_{k-1}g = gS_{k-1}$ it follows that S_{k-1} is a normal subgroup of H. Notice that the quotient group H/S_{k-1} is the cyclic group generated by $S_{k-1}g$ and therefore its order is a power of p (more precisely, it is p^j for some $j \leq l$). Since p must divide $|H/S_{k-1}|$ it follows that p^k divides $|H|$ and thus it also divides $|Aut(G)|$ proving the claim. ∎

Observation 16 *Notice that if we compute an element g described in the above claim, we can compute (in polynomial time, by brute-force listing) the subgroup H generated by the set $S_{k-1} \cup \{g\}$. Applying Proposition 14 we know that there is a subgroup S_k of H such that $|S_k| = p^k$ and S_{k-1} is normal in S_k. Since H has at most p^{2k} elements, we can do a brute-force search for S_k in polynomial time.*

It remains to show how, given an input graph $G \in mod_{p^k}$-GA and S_{k-1}, we can compute with a mod_p-GA oracle an element g satisfying the properties of the above claim. Let

$$S_{k-1} = \{g_1, g_2, \ldots, g_{p^{k-1}}\}$$

For $1 \leq i \leq p^{k-1}$ let $F_i = \{j \in [n] : g_i(j) = j\}$ and $M_i = [n] - F_i$.

For simplicity we explain the rest of the algorithm in two phases. In the first phase of the algorithm we check if there is an automorphism of order p^l, for $l \leq k$, that fixes some $x_i \in M_i$ for all $i \in [p^k]$.

Phase 1 of Algorithm $\mathcal{A}_k$:

```
for each choice $\{x_i \in M_i\}_{1 \leq i \leq p^{k-1}}$ do
    for each $p$-cycle $P \in S_n$ disjoint from $\{x_i\}_{1 \leq i \leq p^{k-1}}$ do
    {
        $\mathcal{C} \leftarrow \{(x_1), (x_2), \ldots, (x_{p^{k-1}}), P\}$;
        /* Strictly speaking, in $\mathcal{C}$ there are no repetitions of the 1-cycles */
        for each permutation $\tau$ of $[p^{k-1}]$ do
            if $Comb(\tau, G, S_{k-1}, \mathcal{C}, p) \in mod_p$-GA then
            {
                Use Algorithm $\mathcal{A}_1$ of Lemma 13 to compute an
                order-$p$ automorphism $\nu$ of $Comb(\tau, G, S_{k-1}, \mathcal{C}, p)$;
                Applying Lemma 11, from $\nu$ we compute an
                order-$p$ automorphism $\pi$ of $G$;
            }
    }
```

To see the correctness of Phase 1 let $Comb(\tau, G, S_{k-1}, \mathcal{C}, p) = Cycle(H, K, \ldots, K)$ as in Lemma 11, with $p-1$ copies of K, where H and K are appropriately defined. Suppose ν maps H to itself and each copy of K to itself. Then ν projected to H gives an order-p automorphism of G that fixes all points in $\{x_i\}_{1 \leq i \leq p^{k-1}}$ as well as all points in P. On the other hand, if ν cyclically rotates the p graphs $(H, K, \ldots, K)$ then by Lemma 11 we get an automorphism ψ of G that has a p-cycle P and fixes all points in $\{x_i\}_{1 \leq i \leq p^{k-1}}$. We can easily compute the order $o(\psi) = pr$. If we choose $\pi = \psi^r$ we get the desired order-p automorphism of G.

Thus, in either case if Phase 1 succeeds it outputs an order-p automorphism $\pi \notin S_{k-1}$ such that $\pi S_{k-1} = S_{k-1}\pi$. Given this element π we can compute an order-p^k subgroup of $Aut(G)$ (which contains S_{k-1}) by a brute-force search for it in the group generated by $S_{k-1} \cup \{\pi\}$.

The algorithm goes to the second phase if the first phase does not succeed. In the second phase of the algorithm we check if there is an automorphism of G of order p^l, for some $l \leq k$, that differs from all automorphisms in S_{k-1}. In this phase, the correctness relies on the fact that Phase 1 has not succeeded.

At this point we introduce some notation. Let M denote $M_1 \times \cdots \times M_{p^{k-1}}$. Let $C_1, \ldots, C_t$ be a collection of t cycles such that for $1 \leq i \leq t$ $|C_i| = p^{e_i}$ and $0 \leq e_i \leq k$. We say that the above collection $C_1, \ldots, C_t$ of cycles is *good* w. r. t. $S \subseteq [p^{k-1}]$, $(l_1, l_2, \ldots, l_{p^{k-1}}) \in M$, and $\tilde{f} = \{f_j \in F_j : j \in S\}$ if

$$\tilde{f} \subseteq \cup_{i=1}^{t} C_i, \text{ and } l_j \in \cup_{i=1}^{t} C_i \text{ for } 1 \leq j \leq p^{k-1}$$

Phase 2 of Algorithm $\mathcal{A}_k$:

```
for each $S \subseteq [p^{k-1}]$ do
for each tuple $\tilde{l} = (l_1, l_2, \ldots, l_{p^{k-1}}) \in M$ do
for each set $\tilde{f} = \{f_j \in F_j : j \in S\}$ do
for $t = 1$ to $p^{k-1}$ do
for each cycle collection $\{C_1, \ldots, C_t\}$ that is good w. r. t. $S$, $\tilde{l}$, and $\tilde{f}$ do
{
    For each $j \in S$ check that $C_1 C_2 \cdots C_t(l_j) = g_j(l_j)$;
    For each $j \notin S$ check that $C_1 C_2 \cdots C_t(l_j) \neq g_j(l_j)$;
    For each $j \in S$ check that $C_1 C_2 \cdots C_t$ does not fix $f_j$;
    if all the above three checks succeed then
    {
        $\mathcal{C} \leftarrow \{C_1, \ldots, C_t\}$;
        for each permutation $\tau \in S_{p^{k-1}}$ do
        if $Comb(\tau, G, S_{k-1}, \mathcal{C}, p) \in mod_p$-GA then
        {
    /* At this point there is $g \in Aut(G)$ of order $p^l$ that satisfies conditions of the Claim */
            Construct such an automorphism $g$ by adaptively querying $mod_p$-GA for
            $Comb(\tau, G, S_{k-1}, \{C\} \cup \mathcal{C}, p)$
            for different cycles $C$ of size $p^\alpha$ for $\alpha \leq k$, and
            including $C$ in $\mathcal{C}$ if $Comb(\tau, G, S_{k-1}, \{C\} \cup \mathcal{C}, p) \in mod_p$-GA;
            if the above construction succeeds then
                output a desired automorphism $g$ of $G$ and stop
        }
    }
}
```

To see the correctness, we use the fact that the algorithm enters Phase 2 only if Phase 1 is completed unsuccessfully. Now, if $Comb(\tau, G, S_{k-1}, \mathcal{C}, p) \in mod_p$-GA it is not possible that the witnessing order-p automorphism of $Comb(\tau, G, S_{k-1}, \mathcal{C}, p) = Cycle(H, K, \ldots, K)$ maps the copy of H and each copy of K to themselves. Otherwise we would have an order-p^l (for some $l \leq k$) automorphism of G that commutes with S_{k-1} and fixes each point in a collection $l_i \in M_i$, $1 \leq i \leq p^{k-1}$, contradicting Phase 1's failure. Thus, it follows that the order-p automorphism of $Comb(\tau, G, S_{k-1}, \mathcal{C}, p)$ must cyclically permute the copy of H and $p-1$ copies of K. Hence $C_1, \ldots, C_t$ are cycles of the corresponding order-p^l automorphism of G, which is computed in the last step of the algorithm. Observe that the three checks made in Phase 2 guarantee that the sought for automorphism with $C_1, \ldots, C_t$ as a subset of its cycles is not in S_{k-1}. Now, it is not hard to see that if p^k divides $|Aut(G)|$ then, an element g promised by the Claim is computed either in Phase 1 or in Phase 2.

We can compute an order-p^k subgroup of $Aut(G)$ (which contains S_{k-1}) by a brute-force search for it in the group generated by $S_{k-1} \cup \{g\}$. ■

Notice the following immediate consequence of Theorem 12. Interestingly, it is analogous to the well-known result that Mod_pP and Mod_{p^k}P are identical. However, technically the proof of Theorem 12 is very different in nature.

Corollary 17 *For any prime p and any $k > 0$, mod_p-GA and mod_{p^k}-GA are polynomial-time Turing equivalent.*

Another consequence of Theorem 12 is that search is polynomial-time Turing reducible to decision for mod_k-GA, for a search problem such as the one defined at the beginning of this section.

Corollary 18 *For each $k > 1$, search is polynomial-time Turing reducible to decision for mod_k-GA.*

5 A Program Checker for mod_k-GA

The goal of this section is to show that for each $k > 1$ the decision problem mod_k-GA has a program checker in the sense of [3]. We first recall the definition of program checkers.

Definition 19 [3] *A program checker C_A for a decision problem A is a (probabilistic) algorithm that for any program P (supposedly for A) that halts on all instances, for any instance x_0 of A, and for any positive integer k (the security parameter) presented in unary:*

1. *If P is a correct program, that is, if $P(x) = A(x)$ for all instances x, then with probability $\geq 1 - 2^{-k}$, $C_A(x_0, P, k)$=Correct.*
2. *If $P(x_0) \neq A(x_0)$ then with probability $\geq 1 - 2^{-k}$, $C_A(x_0, P, k)$=Incorrect.*

The probability is computed over the sequences of coin flips that C_A could have tossed. Also C_A is allowed to make queries to the program P on some instances.

Before we proceed we also need the definition of IP protocols which was first introduced in [6].

Definition 20 *An interactive proof system consists of a prover-verifier pair $P \leftrightarrow V$. The verifier V is a probabilistic polynomial time machine and the prover P is, in general, a machine of unlimited computational power which shares the input tape and a communication tape with V.*

$P \leftrightarrow V$ is an interactive (i.e. IP) protocol for a language L, if for every $x \in \Sigma^*$:

$$x \in L \;\rightarrow\; \text{Prob}[P \textit{ makes } V \textit{ accept}\,] > 3/4,$$

$$x \notin L \;\rightarrow\; \forall \textit{ provers } P' : \text{Prob}[P' \textit{ makes } V \textit{ accept}\,] < 1/4,$$

The design of our checker for mod_k-GA is based on the following theorem [3].

Theorem 21 [3] *If a decision problem A and its complement have both interactive proof systems, in each of which the honest prover can be simulated in polynomial time with queries to A, then A has a polynomial-time program checker.*

We will first provide a program checker for mod_p-GA, for any prime p. Notice that Lemma 13 already gives an IP protocol for mod_p-GA with the prover polynomial-time Turing reducible to mod_p-GA. Thus, it suffices to design an IP protocol for $\overline{mod_p\text{-GA}}$ with the requisite properties.

Lemma 22 *For any prime p, there is an IP protocol for $\overline{mod_p\text{-GA}}$ in which the honest prover is polynomial-time Turing reducible to mod_p-GA.*

Proof. We rewrite the definition of $\overline{mod_p\text{-GA}}$ as follows: $\overline{mod_p\text{-GA}} = \{G : G$ has no automorphism with a p-cycle$\}$. Given an input graph G, the aim is to design an IP protocol which accepts G with high probability if G has *no automorphism* with a p-cycle, and which rejects G with high probability otherwise. Notice that since the prime p is a constant, the total number of p-cycles in S_n is bounded by qn^p, where q is a constant. We will build the desired IP protocol from an IP protocol for the following related language $L = \{(G, C) : |V(G)| = n,\ C \in S_n$ is a p-cycle and G has no automorphism with C as one of its cycles$\}$.

2-round IP Protocol for L:

```
input $(G, C)$;
$Y \leftarrow [n] - \{i : i \in C\}$;
1. Verifier:
    Pick a permutation $\psi \in S_Y$ uniformly at random;
    Pick a random bit $b \in \{0, 1\}$;
    if $b = 0$ then
        send $G' = \psi(G)$ to the Prover
    else
        send $G' = \psi \circ C(G)$ to the Prover
2. Prover:
    if there exists permutation $\pi \in S_Y$ such that $\pi(G) = G'$ then
        send back a bit $c = 0$
    else
        send back a bit $c = 1$
    if $c = b$ then
        Verifier accepts
    else
        Verifier rejects
```

We first show that if the prover is honest then the protocol accepts an input $(G, C) \in L$ with probability 1. Suppose b took the value 0 and the graph $\psi(G) = G'$ was sent to the prover. Then clearly, the prover will find a permutation, namely ψ, such that $\psi(G) = G'$ and send back $c = 0$ leading to the acceptance of the input. Next, suppose b took the value 1. In that case we claim that there does not exist any permutation $\pi \in S_Y$ such that $\pi(G) = G'$. Suppose there exists such a π. Then, since $\pi(G) = \psi \circ C(G)$, it follows that $(\pi)^{-1}\psi \circ C$ is in $Aut(G)$, which contradicts the assumption that $(G, C) \in L$. In this case the prover will send back $c = 1$ and the verifier will again accept.

Now, to prove the soundness of the protocol, we must show that for an input $(G, C) \notin L$, the verifier will reject the input with probability at least 1/2, for any prover. We first need the following claim. In the sequel we use X to denote the set $\{i : i \in C\}$ and Y to denote $[n] - X$.

Claim A. *If G has an automorphism τ with C as one of its cycles then the random graphs $\psi(G)$ and $\psi \circ C(G)$ are identically distributed, where ψ is picked uniformly at random from S_Y.*

Proof. Let $\tau = \rho \circ C$, where $\rho \in S_Y$. From $\rho \circ C(G) = G$ it is not hard to see that for any graph H

$$\exists \alpha \in S_Y[\alpha(G) = H] \iff \exists \beta \in S_Y[\beta \circ C(G) = H]$$

Thus for any graph H

$$Prob_\alpha[\alpha(G) = H] \ = \ 0 \iff Prob_\beta[\beta \circ C(G) = H] \ = \ 0$$

where α and β are picked uniformly at random from S_Y.

Now, since $\alpha(G) = H$ iff $\alpha\rho \circ C(G) = H$, it is straightforward to derive that the set of permutations $\{\beta \in S_Y : \beta \circ C(G) = H\}$ is precisely $\varphi Aut(G_{[X]})\rho$ which is of size $|Aut(G_{[X]})|$. Therefore,

$$Prob_\beta[\beta \circ C(G) = H] \ = \ |Aut(G_{[X]})|/(n-p)!$$

where β is picked uniformly at random from S_Y. ■

It follows from Claim A that if $(G, C) \notin L$ the prover cannot distinguish between whether G' came from the case $b = 0$ or from $b = 1$. In fact, whether $b = 0$ or $b = 1$ the prover will find a $\pi \in S_Y$ such that $\pi(G) = G'$. Therefore, the bit c that is sent back by any (even cheating) prover can agree with b with probability at most 1/2. Consequently, the verifier will reject an input $(G, C) \notin L$ with probability at least 1/2. We now describe the IP protocol for $\overline{mod_p}$-GA.

IP Protocol for $\overline{mod_p}$-GA:

```
input $G$; /* $G$ has $n$ nodes */
$bool \leftarrow true$;
for each $p$-cycle $C \in S_n$ do
    if the IP protocol for $L$ rejects $(G, C)$ then
        $bool \leftarrow false$;
if $bool$=true then Verifier accepts else Verifier rejects
```

Since $G \in \overline{mod_p\text{-GA}}$ iff $(G, C) \in L$ for every p-cycle C, and since the IP protocol for L has one-sided error it easily follows that the above IP protocol accepts $G \in \overline{mod_p\text{-GA}}$ with probability 1 and rejects $G \in mod_p$-GA with probability at least 1/2. The error probability can be made exponentially small (say 2^{-n}) in the above protocol by repeating the protocol [4] (in parallel or sequentially).

The following claim completes the proof of the lemma.

Claim B. *There is an honest prover that is polynomial-time Turing reducible to* mod_p-GA *for the above IP protocol for* $\overline{mod_p\text{-GA}}$.

Proof. First we observe that in bounding the complexity of the honest prover we are concerned about inputs $G \in \overline{mod_p\text{-GA}}$. More precisely, we must show that there is a polynomial-time algorithm with mod_p-GA as oracle that can simulate the honest prover correctly for inputs $G \in \overline{mod_p\text{-GA}}$. Notice that the honest prover of the overall IP protocol must actually simulate the honest prover of the IP protocol for L for each input in the set $\{(G, C) : C \text{ is a } p\text{-cycle in } S_n\}$, where $G \in \overline{mod_p\text{-GA}}$. The honest prover in the protocol for L is supposed to try and compute a permutation $\pi \in S_Y$ such that $\pi(G) = G'$. We have already argued in the correctness proof that for $G \in \overline{mod_p\text{-GA}}$ such a permutation π exists if and only if the outcome of b is 0 and $G' = \psi(G)$ for the random permutation $\psi \in S_Y$. The honest prover constructs the graph $G'' = Cycle(G_{[X]}, G'_{[X]}, \ldots, G'_{[X]})$, with $p-1$ copies of $G'_{[X]}$. Using algorithm $\mathcal{A}_1$ of Lemma 13 the honest prover computes an automorphism of G'' of order p if it exists. Notice that if there is a permutation $\pi \in S_Y$ such that $\pi(G) = G'$ then there is a permutation π' such that $\pi'(G_{[X]}) = G'_{[X]}$. Hence we can find an order-p automorphism of G'' which cyclically permutes the p graphs in G'', by mapping the copy of $G_{[X]}$ to the first copy of $G'_{[X]}$ by π', and each of the first $p-2$ copies of $G'_{[X]}$ are mapped to the next copy of $G'_{[X]}$ by the identity permutation, and finally, the last copy of $G'_{[X]}$ is mapped back to $G_{[X]}$ by π'^{-1}. It is easy to see that this is an automorphism of G'' of order p. Conversely, suppose that G'' has an order-p automorphism τ computed by the honest prover. Since $G \notin mod_p$-GA and $G' \notin mod_p$-GA, the p graphs defining G'' must be rotated in some p-cyclic order by the automorphism τ. It follows that the copy of $G_{[X]}$ is mapped by τ to some copy of $G'_{[X]}$. Let π' be the projection of τ to these two copies. We have $\pi'(G_{[X]}) = G'_{[X]}$. From π' we can easily recover a permutation $\pi \in S_Y$ such that $\pi(G) = G'$. Thus the honest prover finds an order-p automorphism τ of G'' iff there exists $\pi \in S_Y$ such that $\pi(G) = G'$, and moreover, from such a τ the corresponding π is easily computed. Hence, the honest prover is polynomial-time Turing reducible to mod_p-GA. ■

■

We can now conclude that, for any prime p, mod_p-GA has an efficient program checker.

Theorem 23 *For any prime p,* mod_p-GA *has a polynomial-time program checker.*

Proof. Note that from Lemma 13 we get an IP protocol for mod_p-GA with the prover polynomial-time Turing reducible to mod_p-GA and that by Lemma 22 an IP protocol with

[4]With some modifications we can easily get a constant round IP protocol.

requisite properties exists for $\overline{mod_p\text{-GA}}$. Now, Theorem 21 proves the existence of an efficient checker for mod_k-GA. ■

Now it is easy to provide a checker for any mod_k-GA problem.

Theorem 24 *For each $k > 1$, mod_k-GA has a polynomial-time program checker.*

Proof. Let $\prod_{1 \le i \le m} p_i^{e_i}$ be the prime factorization of k. Because the class of checkable sets is obviously closed under join and under Turing equivalence [3], by Theorem 23 it suffices to show that $mod_k\text{-GA} \equiv_T^p mod_{p_1}\text{-GA} \oplus \cdots \oplus mod_{p_m}\text{-GA}$. Observe that a graph G belongs to mod_k-GA if and only if $(\forall i \le m)[G \in mod_{p_i^{e_i}}\text{-GA}]$. Since, by Corollary 17, $mod_{p_i^{e_i}}\text{-GA} \equiv_T^p mod_{p_i}\text{-GA}$ for each i, we have $mod_k\text{-GA} \le_T^p mod_{p_1}\text{-GA} \oplus \cdots \oplus mod_{p_m}\text{-GA}$. It is easy to prove that $mod_{p_i}\text{-GA} \le_T^p mod_k\text{-GA}$ for each i. Therefore, $mod_{p_1}\text{-GA} \oplus \cdots \oplus mod_{p_m}\text{-GA} \le_T^p mod_k\text{-GA}$ as well. ■

6 Concluding Remarks

In this paper we define modular graph automorphism problems (mod_k-GA) and locate them between GA and GI. We also design an efficient program checker for mod_k-GA based on an algorithm that reduces search to decision for mod_k-GA and an IP protocol for $\overline{mod_k\text{-GA}}$. The bottleneck in making our checker nonadaptive is essentially the following: can search be reduced to decision via parallel queries for mod_p-GA, for prime p?

Indeed, our initial motivation in studying the mod_k-GA problems was to understand the difference between GI and GA by introducing problems of intermediate difficulty. In this context, a challenging question is whether search reduces to decision via parallel queries for GI (hence yielding nonadaptive checkers for GI). We believe that as a first step this question must be answered for mod_p-GA.

References

[1] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II.* Springer–Verlag, 1990.

[2] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer–Verlag, second edition, 1995.

[3] M. Blum and S. Kannan. Designing programs that check their work, *Journal of the ACM,* **43**:269–291, 1995.

[4] M. Blum, M. M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems, *J. Comput. Syst. Sci.* **47**:73–83, 1993.

[5] R. Chang. On the structure of NP computations under boolean operators. Ph. D. Thesis, Cornell University, 1991.

[6] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing* **18**:186–208, 1989.

[7] M. Hall. *The Theory of Groups*, Macmillan, New York, 1959.

[8] F. Harary. *Graph Theory*, Addison Wesley, Reading, 1969.

[9] C. Hoffman. Subcomplete generalizations of graph isomorphism. *Journal of Computer and System Sciences,* **25**:332–359, 1982.

[10] J. Köbler, U. Schöning, and J. Torán. *The graph isomorphism problem: its structural complexity*, Birkhäuser, Boston, 1993.

[11] A. Lozano and J. Torán. On the nonuniform complexity of the graph isomorphism problem. In *Proceedings of the 7th Structure in Complexity Theory Conference*, pp. 118–129, 1992.

[12] R. Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, **8**:131–132, 1979.

[13] C. P. Schnorr. On self-transformable combinatorial problems. *Math. Programming Study*, **14**:95–103, 1982.