The Complexity of Modular Graph Automorphism

V. Arvind[1], R. Beigel[2⋆], and A. Lozano[3⋆⋆]

[1] Institute of Mathematical Sciences, Chennai 600113, India
arvind@imsc.ernet.in
[2] Department of EE&CS, Lehigh University, Bethlehem PA 18015-3084, U.S.A.
beigel@eecs.lehigh.edu
[3] Universitat Politècnica de Catalunya, 08034 Barcelona, Catalonia, E.U.
antoni@lsi.upc.es

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 $\mathrm{Mod}_k\mathrm{GA}$ which consist, for each $k > 1$, of deciding whether the number of automorphisms of a graph is divisible by k. The $\mathrm{Mod}_k\mathrm{GA}$ problems all turn out to be intermediate in difficulty between Graph Automorphism and Graph Isomorphism.
We define an appropriate search version of $\mathrm{Mod}_k\mathrm{GA}$ and design an algorithm that polynomial-time reduces the $\mathrm{Mod}_k\mathrm{GA}$ search problem to the decision problem. Combining this algorithm with an IP protocol, we obtain a randomized polynomial-time checker for $\mathrm{Mod}_k\mathrm{GA}$, 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 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

⋆ Supported in part by NSF grants CCR–8958528 and CCR–9415410, and NASA grant NAG 52895. Work done while on sabbatical from Yale University and affiliated with the Human–Computer Interaction Laboratory.
⋆⋆ Supported in part by the E.U. through ESPRIT project (no. LTR 20244, ALCOM-IT) and by DGICYT (Koala project, with no. PB95-0787), and CICYT (no. TIC97-1475-CE). Work completed while visiting the first author at the Institute of Mathematical Sciences.

to GA (see [9] for these and related results). However, in [11] 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 [4] 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 [4, 5]. One of the first program checkers in [4] 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 [10] 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 has also a nonadaptive checker, and the bottleneck here is that search does not reduce to decision for GI using 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 $\mathrm{Mod}_k\mathrm{GA} = \{G : |Aut(G)| \equiv 0 \pmod{k}\}$.

We show in Theorems 3 and 4 that for any $k > 1$, GA $\leq_m^p$ Mod_kGA $\leq_m^p$ GI; thus the Mod_kGA problems are intermediate in difficulty between GA and GI. It is an open question whether any of the Mod_kGA problems is polynomial-time equivalent to GA or GI. We conjecture that Mod_kGA is *not* polynomial-time equivalent to GA or GI, for any $k > 1$. An evidence that some of the Mod_kGA problems could be actually harder than GA is our observation that Tournament Isomorphism (graph isomorphism for tournament graphs) is many-one reducible to Mod_2GA. This follows from the fact that the automorphism group of any tournament is of odd size [9], 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_kGA problems are located between GA and GI. In Section 4, we show that search is polynomial-time Turing equivalent to decision for Mod_kGA, and in Section 5 we use this result in combination with an IP protocol for $\overline{\mathrm{Mod}_p\mathrm{GA}}$ to obtain an efficient program checker for Mod_kGA. Notice that although both GA and GI have program checkers ([10] and [4] resp.) and Mod_kGA is intermediate in complexity, it does not necessarily imply that Mod_kGA has a program checker [4].

In this extended abstract we omit several proofs due to lack of space[4].

[4] A fuller version is available at `http://www.imsc.ernet.in/~arvind/modga.ps.gz`

2 Preliminaries

In this paper by a graph we mean a finite directed graph[5] (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 positions in X and Y. It is not hard to see that in $G_{[X]}$ vertices of X are pointwise fixed in any automorphism[6]. 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 easily 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 $\mathrm{Path}(G_1, \ldots, G_n)$ is obtained by taking one copy of each of the graphs $G_1, \ldots, G_n$ and, for $1 \leq i \leq n$, 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, $1 \leq i \leq n$, labeled with a single label l. The graph $\mathrm{Cycle}(G_1, \ldots, G_n)$ is obtained by taking one copy of each of the graphs $G_1, \ldots, G_n$ and, for $1 \leq i \leq n$, attaching all the vertices of G_i to v_i.

In both $\mathrm{Path}(G_1, \ldots, G_n)$ and $\mathrm{Cycle}(G_1, \ldots, G_n)$, 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 $\mathrm{Path}(G_1, \ldots, G_n)$ $(\mathrm{Cycle}(G_1, \ldots, G_n))$ when restricted to $\{v_1, v_2, \ldots, v_n\}$ is an automorphism of P_n (C_n) This means that an automorphism of $\mathrm{Path}(G_1, \ldots, G_n)$ cannot permute the copies of $G_1, \ldots, G_n$, while an automorphism of $\mathrm{Cycle}(G_1, \ldots, G_n)$ 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]. 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

[5] In this paper we consider the problems GI, GA, and $\mathrm{Mod}_k\mathrm{GA}$ on directed graphs. However, all results of this paper hold for these problems on undirected graphs as well.

[6] 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. [9].

NP-complete problems. In particular, we recall that for GI, search is polynomial-time Turing reducible to decision [12] whereas for GA a stronger result holds: search is *nonadaptively* polynomial-time reducible to decision [10]. Finally, we recall from [6, 10] that GI is both a d-cylinder and a c-cylinder[7] [3].

3 Locating the Mod_kGA Problems

We show in this section that $\mathrm{Mod}_k\mathrm{GA}$ is located between GA and GI, for all $k > 1$.

Theorem 3. *For all* $k > 1$, $\mathrm{GA} \leq_m^p \mathrm{Mod}_k\mathrm{GA}$.

Proof. Given a graph G, we define for every i, j with $1 \leq i < j \leq n$, the graph $H_{i,j} = \mathrm{Cycle}(G_{[i]}, G_{[j]}, \ldots, G_{[j]})$ 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 $\mathrm{Mod}_k\mathrm{GA}$.

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 $\mathrm{Cycle}(G_{[i]}, G_{[j]}, \ldots, G_{[j]})$ 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 $H_{i,j} \in \mathrm{Mod}_k\mathrm{GA}$. Since $|Aut(H)| = \prod_{1 \leq i < j \leq n} |Aut(H_{i,j})|$, it follows that $H \in \mathrm{Mod}_k\mathrm{GA}$.

For the converse, assume that $H \in \mathrm{Mod}_k\mathrm{GA}$. 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 $H_{i,j} = \mathrm{Cycle}(G_{[i]}, G_{[j]}, \ldots, G_{[j]})$, 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 [11] has shown that $|Aut(G)|$ is polynomial-time computable with GI as oracle. From this it easily follows that $\mathrm{Mod}_k\mathrm{GA} \leq_T^p \mathrm{GI}$. In the next theorem, we strengthen this to a $\leq_m^p$-reduction using some permutation group theory.

Theorem 4. *For all* $k > 1$, $\mathrm{Mod}_k\mathrm{GA} \leq_m^p \mathrm{GI}$.

We need a couple of definitions and group-theoretic lemmas before we prove Theorem 4. Let A be a subgroup of S_n and let $[n]$ denote the set $\{1, 2, \ldots, n\}$. A

[7] Elsewhere in the literature, e.g. [9], these properties are called OR and AND functions respectively.

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. An obvious useful property is that $|A^X| \leq |A|$, for all A-invariant sets X. The proofs of the following two group-theoretic lemmas are omitted.

Lemma 5. *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.*

Lemma 6. *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 $a_i b_j = a_j b_i$, for $1 \leq i, j \leq t$. Then $|\langle X \rangle|$ divides the order of the group $\langle \{a_i b_i : 1 \leq i \leq t\} \rangle$.*

Proof of Theorem 4. First, we argue that it suffices to show that $\mathrm{Mod}_{p^l}\mathrm{GA} \leq_m^p \mathrm{GI}$ for all prime p and $l > 0$. To see this, let $\prod_{1 \leq j \leq r} p_j^{l_j}$ be the prime factorization of k. Clearly, a graph $G \in \mathrm{Mod}_k\mathrm{GA}$ iff $G \in \bigcap_{1 \leq j \leq r} \mathrm{Mod}_{p_j^{l_j}}\mathrm{GA}$. Thus, if $\mathrm{Mod}_{p_j^{l_j}}\mathrm{GA} \leq_m^p \mathrm{GI}$ for $1 \leq j \leq r$, it follows that $\mathrm{Mod}_k\mathrm{GA} \leq_m^p \mathrm{GI}$, 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 $\mathrm{Mod}_{p^l}\mathrm{GA}$ if and only if there exist a set $X \subseteq [n]$ with $|X| \leq p^l(\log p^l)$ and a subgroup $K = \{a_1, a_2, \ldots, a_{p^l}\}$ of S_X such that each $a_i \in K$ is a partial automorphism of G.*

Proof. Let $G \in \mathrm{Mod}_{p^l}\mathrm{GA}$ 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 5 there is an A-invariant set $X \subseteq [n]$ with $|X| \leq p^l(\log p^l)$, 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 $a_i b_i \in Aut(G)$. It is not hard to see that Lemma 6 can be applied to the elements $\{a_i\}_{1 \leq i \leq p^l}$ and $\{b_i\}_{1 \leq i \leq p^l}$. 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 [9]). We will give a truth-table reduction from $\mathrm{Mod}_{p^l}\mathrm{GA}$ 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 $\mathrm{Mod}_{p^l}\mathrm{GA}$ is $\leq_m^p$-reducible to GI. We describe below the said reduction of $\mathrm{Mod}_{p^l}\mathrm{GA}$ to B as a logical expression, which is easily seen to describe a disjunction-of-conjunctions kind of truth-table reduction:

$$G \in \mathrm{Mod}_{p^l}\mathrm{GA} \iff (\exists\, X \subseteq [n] : |X| \leq p^l \log p^l)$$
$$(\exists \text{ subgroup } K < S_X : |K| = p^l)(\forall a \in K)[(G, a) \in B] \quad \square$$

4 Computing Solutions for $\mathrm{Mod}_k\mathrm{GA}$ Instances

The goal of this section is to design a polynomial-time algorithm that reduces the search problem for $\mathrm{Mod}_k\mathrm{GA}$ to the decision problem. Consider $\mathrm{Mod}_k\mathrm{GA}$ 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 $\mathrm{Mod}_k\mathrm{GA}$ is a collection of m subgroups $\{A_1, A_2, \ldots, A_m\}$ of $Aut(G)$ where A_i is of order $p_i^{e_i}$, and is listed as a set of permutations. We consider such a witness as a solution for G for the $\mathrm{Mod}_k\mathrm{GA}$ search problem and we design a polynomial-time algorithm that computes this witness for any given instance of $\mathrm{Mod}_k\mathrm{GA}$ with oracle access to the $\mathrm{Mod}_k\mathrm{GA}$ decision problem. The next theorem is the main result of this section.

Theorem 7. *There is a polynomial-time algorithm $\mathcal{A}_k$ with $\mathrm{Mod}_p\mathrm{GA}$ as oracle such that given a graph $G \in \mathrm{Mod}_{p^k}\mathrm{GA}$ as input the algorithm $\mathcal{A}_k$ lists out the elements of a p^k-subgroup of $Aut(G)$.*

The proof of Theorem 7, which is a lengthy proof by induction on k and which exploits the structure of p-groups, is omitted from this extended abstract. In order to give a flavor of the sort of ideas involved in the proof of Theorem 7 we discuss in detail the base case, $k = 1$, in the following lemma.

Lemma 8. *There is a polynomial-time algorithm $\mathcal{A}_1$ with $\mathrm{Mod}_p\mathrm{GA}$ as oracle such that given a graph $G \in \mathrm{Mod}_p\mathrm{GA}$ 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 $G \in \mathrm{Mod}_p\mathrm{GA}$.

Algorithm $\mathcal{A}_1$

Input G;
if $G \notin \mathrm{Mod}_p\mathrm{GA}$ **then stop**;
$X = \emptyset$;
for $i := 1$ **to** $|V(G)|$ **do**
    if $G_{[X \cup \{i\}]} \in \mathrm{Mod}_p\mathrm{GA}$ **then** $X := X \cup \{i\}$;
$S := V(G) - X$; $\mathcal{C} := \emptyset$; $G' := G$; $G'' := G$;
for each p-cycle $C \subseteq S - \bigcup_{D \in \mathcal{C}} D$ **do**
    if $Cycle(G'_{[C]}, G''_{[r(C)]}, \ldots, G''_{[r(C)]}) \in \mathrm{Mod}_p\mathrm{GA}$ **then**

(* There are $p-1$ copies of $G''_{[r(C)]}$ in the above Cycle definition *)
$G' := G'_{[C]}$; $G'' := G''_{[r(C)]}$; $\mathcal{C} := \mathcal{C} \cup \{C\}$;
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 $G \in \mathrm{Mod}_p\mathrm{GA}$ as input and computes the graph $G_{[X]} \in \mathrm{Mod}_p\mathrm{GA}$ with X as its set of fixed points (such that no more points can be fixed preserving membership in $\mathrm{Mod}_p\mathrm{GA}$). It remains to show that there is an order-p automorphism with $\mathcal{C}$ as its collection of p-cycles and X as its fixed-point set. We will prove this as an invariant of the second for-loop in the algorithm. 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 $\mathrm{Mod}_p\mathrm{GA}$. 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. This completes the proof. $\square$

Notice that an immediate consequence of Theorem 7 is that search is polynomial-time Turing reducible to decision for $\mathrm{Mod}_k\mathrm{GA}$. Another consequence of Theorem 7 is that $\mathrm{Mod}_{p^k}\mathrm{GA}$ is polynomial-time Turing reducible to $\mathrm{Mod}_p\mathrm{GA}$, for any prime p.

5 A Program Checker for $\mathbf{Mod}_k\mathrm{GA}$

The goal of this section is to show that for each $k > 1$ the decision problem $\mathrm{Mod}_k\mathrm{GA}$ has a program checker in the sense of [4]. We first recall the definition of program checkers.

Definition 9. [4] 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 [7].

Definition 10. 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 $x \in \Sigma^*$:

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

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

The design of our checker for Mod_kGA is based on the following theorem [4].

Theorem 11. [4] *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.*

Notice that Theorem 7 already gives an IP protocol for Mod_kGA with the prover polynomial-time Turing reducible to Mod_kGA. Thus, it suffices to design an IP protocol for $\overline{\text{Mod}_k\text{GA}}$ with requisite properties. The overall IP protocol for $\overline{\text{Mod}_k\text{GA}}$ is based on the following IP protocol for $\overline{\text{Mod}_p\text{GA}}$, for prime p.

Lemma 12. *For any prime p, there is an IP protocol for $\overline{\text{Mod}_p\text{GA}}$ in which the honest prover is polynomial-time Turing reducible to* Mod_pGA.

Proof. We rewrite the definition of $\overline{\text{Mod}_p\text{GA}}$ as follows: $\overline{\text{Mod}_p\text{GA}} = \{G : G$ 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);
Let $Y := [n] - \{i : i \in C\}$;
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 **Prover**
else send $G' = \psi \circ C(G)$ to **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** the Verifier **accepts else** the 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 (proof omitted). 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.*

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 $1/2$. The error probability can be made exponentially small (say 2^{-n}) in the above protocol by repeating the protocol [8] (in parallel or sequentially). We now describe the IP protocol for $\overline{\mathrm{Mod}_p\mathrm{GA}}$.

IP Protocol for $\overline{\mathrm{Mod}_p\mathrm{GA}}$

Input G; (* G has n nodes *)
for each p-cycle $C \in S_n$ **do**
if the IP protocol for L rejects (G, C) **then**
Reject (and stop);
Accept

It is easy to see that this IP protocol accepts $G \in \overline{\mathrm{Mod}_p\mathrm{GA}}$ with probability 1 and rejects $G \in \mathrm{Mod}_p\mathrm{GA}$ with probability at least $(1 - 2^{-n})^{qn^p}$ which is larger than $1/2$. The following claim completes the proof of the lemma.

[8] With some modifications we can easily get constant round IP protocols.

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

Proof of Claim B. First we observe that in bounding the complexity of the honest prover we are concerned about inputs $G \in \overline{\mathrm{Mod}_p\mathrm{GA}}$. More precisely, we must show that there is a polynomial-time algorithm with $\mathrm{Mod}_p\mathrm{GA}$ as oracle that can simulate the honest prover correctly for inputs $G \in \overline{\mathrm{Mod}_p\mathrm{GA}}$. 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 $\{(G, C) : C$ is a p-cycle in $S_n\}$, where $G \in \overline{\mathrm{Mod}_p\mathrm{GA}}$. 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 $G \in \overline{\mathrm{Mod}_p\mathrm{GA}}$ 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 8 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 $G \notin \mathrm{Mod}_p\mathrm{GA}$ and $G' \notin \mathrm{Mod}_p\mathrm{GA}$, 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 $\mathrm{Mod}_p\mathrm{GA}$. $\square$

Theorem 13. *For each* $k > 1$, $\mathrm{Mod}_k\mathrm{GA}$ *has a 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 [4], by Lemma 12, it suffices to show that $\mathrm{Mod}_k\mathrm{GA} \equiv_T^p \mathrm{Mod}_{p_1}\mathrm{GA} \oplus \cdots \oplus \mathrm{Mod}_{p_i}\mathrm{GA}$. Observe that for any graph G, $G \in \mathrm{Mod}_k\mathrm{GA} \iff (\forall i \le m)[G \in \mathrm{Mod}_{p_i^{e_i}}\mathrm{GA}]$. By Theorem 7, $\mathrm{Mod}_{p_i^{e_i}}\mathrm{GA} \equiv_T^p \mathrm{Mod}_{p_i}\mathrm{GA}$ for each i. Therefore, $\mathrm{Mod}_k\mathrm{GA} \le_T^p \mathrm{Mod}_{p_1}\mathrm{GA} \oplus \cdots \oplus \mathrm{Mod}_{p_i}\mathrm{GA}$. It is easy to prove that $\mathrm{Mod}_{p_i}\mathrm{GA} \le_T^p \mathrm{Mod}_k\mathrm{GA}$ for each i. Therefore, $\mathrm{Mod}_{p_1}\mathrm{GA} \oplus \cdots \oplus \mathrm{Mod}_{p_i}\mathrm{GA} \le_T^p \mathrm{Mod}_k\mathrm{GA}$ as well. $\square$

6 Concluding Remarks

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

Indeed, our initial motivation in studying the $\mathrm{Mod}_k\mathrm{GA}$ 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 $\mathrm{Mod}_p\mathrm{GA}$.

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. R. Beigel, M. Kummer, and F. Stephan. Approximable sets, *Information and Computation* **120**(2):304–314, 1995.
4. M. Blum and S. Kannan. Designing programs that check their work, *Journal of the ACM,* **43**:269–291, 1995.
5. M. Blum, M. M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems, *J. Comput. Syst. Sci.* **47**:73–83, 1993.
6. R. Chang. On the structure of NP computations under boolean operators. Ph. D. Thesis, Cornell University, 1991.
7. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing* **18**:186–208, 1989.
8. F. Harary. *Graph Theory*, Addison Wesley, Reading, 1969.
9. J. Köbler, U. Schöning, and J. Torán. *The graph isomorphism problem: its structural complexity*, Birkhäuser, Boston, 1993.
10. 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.
11. R. Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, **8**:131–132, 1979.
12. C. P. Schnorr. On self-transformable combinatorial problems. *Math. Programming Study*, **14**:95–103, 1982.

This article was processed using the LaTeX macro package with LLNCS style