A Note on Decision versus Search for Graph Automorphism*

M. Agrawal†
Department of Computer Science & Engineering
Indian Institute of Technology
Kanpur, India
email: ma@iitk.ernet.in

V. Arvind
Institute of Mathematical Sciences
Madras 600113, India
email: arvind@imsc.ernet.in

Abstract

We show that for any graph G, k non-trivial automorphisms of G—if as many exist—can be computed in time $|G|^{O(\log k)}$ with nonadaptive queries to GA, the decision problem for Graph Automorphism. As a consequence we show that some problems related to GA are actually polynomial-time truth-table equivalent to GA. One of these results provides an answer to an open question of Lubiw [Lu81].

---

*A preliminary version was presented at the Computational Complexity'96 conference

†Part of the work done while working with SPIC Science Foundation, Madras, India and while visiting Universität Ulm, Germany in 1995-96.

1 Introduction

The Graph Isomorphism problem (GI)—of testing if two graphs are isomorphic—and the Graph Automorphism problem (GA)—of testing if a graph has a non-trivial automorphism—are well-studied problems in the class NP. Much of the research interest in these problems is due to the fact that they are neither known to be in P nor have they been shown to be NP-complete. In fact they are in the class NP $\cap$ coAM and hence cannot be NP-complete unless PH collapses to Σ_2^p [Sch88].

The subject of this note is the relative complexity of decision vs. search for Graph Automorphism. For NP-complete problems decision and search are equivalent as there are *self-reducible* NP-complete sets. The decision vs. search question can be nicely formalized using the notion of *self-computability* due to Balcázar [B89]. Let $A \in$ NP and $R \in$ P be the polynomial-time binary relation defining solutions for A. Let $Sol_R(x)$ denote the set of solutions for $x \in A$. The set A is said to have *self-computable* solutions if there is a deterministic polynomial-time oracle machine that on input $x \in A$ outputs a string $w \in Sol_R(x)$ using A as oracle. Let $Prefix(A) = \{\langle x, z\rangle \mid x \in A$ and z is a prefix of some $w \in Sol_R(x)\}$. If $Prefix(A)$ is Turing reducible[1] to A then A has self-computable solutions [B89]. Since $Prefix(A)$ and A are many-one equivalent for any NP-complete set A, it follows that NP-complete sets have self-computable solutions. Since GI and $Prefix(GI)$ (suitably defined) are many-one equivalent [KST92], GI also has self-computable solutions.[2]

In general, if $Prefix(A)$ is reducible to A, there is an apparently stronger equivalence between search and decision for A than implied by self-computability. If $Prefix(A)$ is reducible to A then for every $x \in A$ the lexicographically smallest and largest solutions in $Sol_R(x)$ can be computed in polynomial time using A as oracle. In fact, given $x \in A$ and any proper subset X of $Sol_R(x)$ as input, there is a polynomial (in $||X||$) time algorithm that uses A as oracle and outputs $w \in Sol_R(x) - X$. For GI an even stronger property holds: since the counting problem #GI is equivalent to GI [Ma79], given an instance x of GI and a natural number i, the lexicographically ith solution for x (if it exists) can be computed in time polynomial in $|x|$ using GI as oracle.

The decision vs. search question for Graph Automorphism (GA) has interesting peculiarities. On the one hand GA has self-computable solutions: the lexicographically smallest nontrivial automorphism of a graph can be computed in polynomial time with even *nonadaptive* queries to GA [LT92]. On the other hand, the set $Prefix(GA)$ is apparently harder than GA. It is shown in [LT92] that $Prefix(GA)$ is many-one equivalent to GI, and GI is not known to be reducible to GA. In particular, it is shown in [LT92] that computing the lexicographically largest automorphism of a graph is equivalent to GI.

In a different line of research Lubiw [Lu81] studies several variations of Graph Isomorphism. It turns out that some variations are equivalent to GI, while others are NP-complete. In [Lu81] Lubiw also makes a detailed study of some variations of GI which are directed related to the complexity of computing solutions for instances of GI and GA. In particular, she defines the following interesting decision problem which relates GI to GA: Given two graphs G_1 and G_2 and k distinct isomorphisms between G_1 and G_2, is there yet another isomorphism between G_1 and G_2?

[1] All reductions and reducibilities discussed in this paper are polynomial-time computable.
[2] This was already known since GI is self-reducible [Schn82].

As already observed, it is not known if GI is reducible to GA (although GA is many-one reducible to GI [LT92]). Lubiw observes that the above problem, when k is allowed to vary with the input, is many-one equivalent to GI. The interesting case is when k is a fixed parameter that is not part of the input. Here, she shows that for $k = 0$ this decision problem is essentially GI, and for $k = 1$ it is equivalent to GA. She leaves the complexity of the problem as an open question for larger values of k (see also [KST92]).

In this note we show that the lexicographically first k automorphisms of a graph G—or all automorphisms if there are fewer than k—can be computed in time $|G|^{O(\log k)}$ with nonadaptive queries to GA. Thus, for fixed k, the first k automorphisms of G—if they exist—can be computed in polynomial time with nonadaptive queries to GA.

As a consequence, it follows that for each fixed parameter k, the corresponding problem of Lubiw remains truth-table equivalent to GA.

Another corollary of our result is that the sets

$\mathrm{GA}_{k\text{-}rough} = \{G \mid$ the number of automorphisms of G has at most k prime factors (with multiplicities)$\}$, and

$\mathrm{GI}_{k\text{-}rough} = \{\langle G_1, G_2\rangle \mid G_1, G_2 \in \mathrm{GA}_{k\text{-}rough}$ and G_1 is isomorphic to $G_2\}$ for each fixed $k > 0$, are truth-table equivalent to GA.

2 Preliminaries

We consider directed labeled graphs in this paper. This is no loss of generality since we polynomial-time many-one equivalent versions of the problems GI and GA irrespective of whether we consider directed or undirected, labeled or unlabeled graphs [KST92].

The vertex set of a graph G is denoted $V(G)$ and the edge set $E(G)$. For a graph G, let $Aut(G)$ denote the automorphism group of G and let $id \in Aut(G)$ denote the identity automorphism. Let $\pi \in Aut(G)$. A vertex i of G is said to be a *fixpoint* of π if $\pi(i) = i$ (and π is said to fix i). Let $X \subseteq V(G)$. The *pointwise stabilizer* of X is the set $Stab(X) = \{\pi \in Aut(G) \mid \forall i \in X, i \text{ is a fixpoint of } \pi\}$. $Stab(X)$ is clearly a subgroup of $Aut(G)$.

For a graph G and a subset $X = \{i_1, i_2, \ldots, i_k\} \subseteq V$ let $G_{[X]}$ denote the graph obtained from G by *labeling* vertex i_1 with color c_1, vertex i_2 with color c_2, …, vertex i_k with color c_k. This labeling of vertices with colors has the effect of distinguishing the labeled vertex from the rest of the vertices of the graph. As described for example in [KST92], labeling vertex i of a graph G with a distinct color c_i can be effected by attaching a special graph-theoretic gadget of size $O(|V(G)|)$ to vertex i. Let this new graph with vertex i colored c_i be G'. It turns out that every $\phi \in Aut(G')$ fixes i and also fixes every other node in the gadget attached to i. Thus $Aut(G')$ is isomorphic to the subgroup of $Aut(G)$ that fixes i. Furthermore, given any automorphism of G' the corresponding automorphism of G can be easily constructed and vice-versa. The following proposition, which appears implicitly in [Ma79] (also in [LT92, KST92]), summarizes this property.

Proposition 2.1 [Ma79] *Let G be a labeled graph and $X \subseteq V(G)$. $Stab(X)$ is isomorphic to the automorphism group $Aut(G_{[X]})$. Furthermore, given any element of $Aut(G_{[X]})$ the corresponding element of $Stab(X)$ can be easily computed and vice-versa.*

By abuse of notation we sometimes identify $Aut(G_{[X]})$ with $Stab(X)$.

The *union* graph,

$G \mathring{\cup} H$, of two graphs G and H is the graph obtained by first making their vertex sets disjoint by renaming, and then taking the union of their vertex and edge sets [Har69].[3]

The following construction that we describe first appeared in [Ma79] (also see [Hof82, KST92] for other applications of this construction).

Let G be a graph with vertex set $V = \{1, 2, \ldots, n\}$, and $i \in V$. Let $I = \{i_1, \ldots, i_t\}$ be a list of t distinct vertices from $\{i, i+1, \ldots, n\}$. Similarly, let $J = \{j_1, \ldots, j_t\}$ be another list of t distinct vertices from $\{i, i+1, \ldots, n\}$. We term such lists as *ordered* subsets.

Let $G^{(i)}$ denote the graph $G_{[1,2,\ldots,i]}$. We define a new graph $G^{(i-1)}_{[I]} \mathring{\cup} G^{(i-1)}_{[J]}$, such that for every $j : 1 \leq j < i$, the vertex j of the subgraph $G^{(i-1)}_{[I]}$ has the *same color label* as the vertex j of the subgraph $G^{(i-1)}_{[J]}$. Furthermore, for every $r : 1 \leq r \leq t$, the vertex $i_r \in I$ of the subgraph $G^{(i-1)}_{[I]}$ has the same color label as the vertex j_r of the subgraph $G^{(i-1)}_{[J]}$.

Observe that the graph $G^{(i-1)}_{[I]} \mathring{\cup} G^{(i-1)}_{[J]}$ can have the following two kinds of automorphisms. In the first case, an automorphism π can map the subgraph $G^{(i-1)}_{[I]}$ to itself and the subgraph $G^{(i-1)}_{[J]}$ to itself. For such an automorphism π, $\pi(x) = x$ for every vertex $x \in \{1, 2, \ldots, i-1, i_1, i_2, \ldots, i_t\}$ of the subgraph $G^{(i-1)}_{[I]}$. Similarly, $\pi(y) = y$ for every vertex $y \in \{1, 2, \ldots, i-1, j_1, j_2, \ldots, j_t\}$ of the subgraph $G^{(i-1)}_{[J]}$. In the other case, an automorphism π can map the subgraph $G^{(i-1)}_{[I]}$ to the subgraph $G^{(i-1)}_{[J]}$ and vice-versa.

The following crucial proposition relating the automorphisms of $G^{(i-1)}_{[I]} \mathring{\cup} G^{(i-1)}_{[J]}$ to the automorphisms of $G^{(i-1)}$ is essentially from [Ma79]. For applications of this property refer to [LT92, KST92].

Proposition 2.2 [Ma79] *The set of automorphisms of $G^{(i-1)}_{[I]} \mathring{\cup} G^{(i-1)}_{[J]}$ that map the subgraph $G^{(i-1)}_{[I]}$ to the subgraph $G^{(i-1)}_{[J]}$ and vice-versa is in 1-1 correspondence with the set of automorphisms of $G^{(i-1)}$ which maps vertex $i_r \in I$ to vertex $j_r \in J$ for $1 \leq r \leq t$. Furthermore, given an automorphism of $G^{(i-1)}_{[I]} \mathring{\cup} G^{(i-1)}_{[J]}$ that maps the subgraph $G^{(i-1)}_{[I]}$ to the subgraph $G^{(i-1)}_{[J]}$, it is easy to compute in polynomial time the corresponding automorphism of $G^{(i-1)}$.*

The basic complexity-theoretic concepts used in this paper like many-one and truth-table reducibilities, can be found in a standard textbook, for example [BDG88].

3 The result

For any graph G, we define $Auto(k, G)$ as follows. If G has at least k automorphisms, then $Auto(k, G)$ is defined as the list $(id, \pi_1, \ldots, \pi_{k-1})$ where π_1, $\ldots$, π_{k-1} are the lexicographically first $k-1$ distinct non-trivial automorphisms of G; if G has j non-trivial automorphisms with $j < k-1$, then $Auto(k, G)$ is defined as $(id, \pi_1, \ldots, \pi_j)$ which is the list of all automorphisms of G.

[3] Our notation is at variance with Harary's who uses '$\cup$' rather than '$\mathring{\cup}$' to denote the union of graphs. We do so because we frequently use '$\cup$' to denote the union of sets of graphs.

Theorem 3.1 *$Auto(k, G)$ is computable in time $|G|^{O(\log k)}$ with nonadaptive queries to* GA.

Proof. Let G be the given graph with n vertices and $t = \lceil \log k \rceil$. We assume, w.l.o.g, that G is connected (if not, we work with $\overline{G}$ which is connected and has the same automorphism group as G). We describe the algorithm in different components; the overall algorithm can be obtained easily by composing these components. The algorithm that computes $Auto(k, G)$ has first a querying phase where it makes $|G|^{O(\log k)}$ parallel queries to GA. The rest of the algorithm analyzes the answers to these queries and computes the function $Auto(k, G)$ in $|G|^{O(\log k)}$ time.

Let G be any graph on n vertices. For every i, $1 \leq i \leq n$, and for every pair of *ordered* subsets I, J of $\{i, i+1, \ldots, n\}$ such that $0 \leq |I| = |J| \leq 2t + 2$, let
$S_{i,I,J} = \{G^{(i-1)}_{[I]} \mathring{\cup} G^{(i-1)}_{[J]}\} \cup \{G^{(i-1)}_{[I,l]} \mathring{\cup} G^{(i-1)}_{[J,m]} \mid i < l, m \leq n, l \notin I, m \notin J\}$.[4]

Querying Phase:

Input G; (*G is a labeled graph with n vertices*)
for $i : 1 \leq i \leq n$ **do**
For every pair of *ordered* subsets I, J of $\{i, i+1, \ldots, n\}$ such that $0 \leq |I| = |J| \leq 2t+2$ and for each graph in $S_{i,I,J}$ query oracle GA in parallel and collect answers
endfor;
for $i : 1 \leq i \leq n$ **do**
In parallel query oracle GA for $G^{(i)}$ and
collect answers
endfor;

Notice that for a graph G on n vertices there are at most $n \cdot (2t+3) \cdot n^{4t+4} \cdot (n^2+1) \leq n^{4t+10}$ graphs which are queried in the Querying Phase. We now explain how the function $Auto(k, G)$ can be computed from the answers to these queries. Call the vertex i of the graph G *free* if there is an automorphism π of the graph $G^{(i-1)}$ such that $\pi(i) \neq i$. It is easy to see that G has nontrivial automorphisms iff it has free vertices.

Our first aim is to compute as many free vertices of G as possible using the answers obtained in the Querying Phase. The largest free vertex of G (call it j_1) is easy to compute: j_1 is the largest i such that $G^{(i-1)} \in$ GA and $G^{(i)} \notin$ GA. Observe that $G^{(i-1)}$ and $G^{(i)}$ are queries already made to GA for all i in the Querying Phase.

Now suppose we have computed the r largest free vertices $j_1, \ldots, j_r$, with $j_1 > \cdots > j_r$. Define the set
$T_i^r = \{G^{(i-1)}_{[I]} \mathring{\cup} G^{(i-1)}_{[J]} \mid I$ and J are ordered sets such that $1 \leq i < j_r, r+1 \leq |I| = |J| \leq 2r+2, \{i, j_r, \cdots, j_1\} \subseteq I \cap J$, i occurs as the first vertex in I and not as the first vertex in $J\}$.

The following claim is a crucial property of the sets T_i^r.

Claim 3.1.1 *Let $j_1, \ldots, j_r$, with $j_1 > \cdots > j_r$, be the r largest free vertices of G, and suppose no $j : i < j < j_r$ is a free vertex. If some graph $G^{(i-1)}_{[I]} \mathring{\cup} G^{(i-1)}_{[J]} \in T_i^r$ has a nontrivial automorphism, then i is the $(r+1)^{th}$ free vertex.*

[4]The subscript $[I, l]$ stands for the ordered set $I \cup \{l\}$, and $[J, m]$ is similarly defined.

Proof of Claim 3.1.1. Consider a graph $G_{[I]}^{(i-1)} \mathring{\cup} G_{[J]}^{(i-1)}$ in T_i^r that has a non-trivial automorphism ψ. If ψ maps subgraph $G_{[I]}^{(i-1)}$ to itself then it must also map subgraph $G_{[J]}^{(i-1)}$ to itself. Since the set I contains $\{i, j_r, \ldots, j_1\}$, the restriction of ψ to $G_{[I]}^{(i-1)}$ yields an automorphism of $G^{(i-1)}$ in which the vertex i is a fixed point and also j_r, ..., j_1 are all fixpoints. This, in turn, yields an automorphism of G in which all free vertices of G are fixpoints, which must therefore be id. Similarly, we can argue that the restriction of ψ to $G_{[J]}^{(i-1)}$ is also id. This contradicts our assumption that ψ is non-trivial. Therefore ψ must map vertices of $G_{[I]}^{(i-1)}$ to $G_{[J]}^{(i-1)}$ and vice-versa. Since the first vertex of J is different from i it follows that i is a free vertex (the $(r+1)^{th}$ free vertex). Notice that from Proposition 2.2 the corresponding nontrivial automorphism of $G^{(i-1)}$ is easy to compute from ψ. □

Claim 3.1.2 *Vertex i is the $(r+1)^{th}$ free vertex of G iff $T_i^r \cap \mathrm{GA} \neq \emptyset$ and for every j, $i < j < j_r$, $T_j^r \cap \mathrm{GA} = \emptyset$.*

Proof of Claim 3.1.2. $(\Rightarrow)$ Since i is the $(r+1)^{th}$ free vertex of G, $T_j^r \cap \mathrm{GA} = \emptyset$ for every $i < j < j_r$. Let $\pi \in Aut(G^{(i-1)})$ with $\pi(i) = i'$, $i' \neq i$. Let $K = \{i, j_r, \ldots, j_1\} \cup \pi^{-1}(\{i, j_r, \ldots, j_1\})$. Order the vertices in K such that i is the first one and let this ordered set be I. Let $J = \pi(I)$. Clearly, the first vertex in J is $i' \neq i$. The graph $G_{[I]}^{(i-1)} \mathring{\cup} G_{[J]}^{(i-1)}$ belongs to T_i^r and has a non-trivial automorphism that maps vertices of the subgraph $G_{[I]}^{(i-1)}$ to vertices of subgraph $G_{[J]}^{(i-1)}$ according to π and vertices of $G_{[J]}^{(i-1)}$ to vertices of $G_{[I]}^{(i-1)}$ according to π^{-1}.
$(\Leftarrow)$ By the forward implication, since $T_j^r \cap \mathrm{GA} = \emptyset$, vertex j cannot be a free vertex for $i < j < j_r$. Suppose that vertex i is also not a free vertex. Consider a graph $G_{[I]}^{(i-1)} \mathring{\cup} G_{[J]}^{(i-1)}$ in T_i^r that has a non-trivial automorphism ϕ. From the proof of Claim 3.1.1 it is clear that ϕ must map the subgraph $G_{[I]}^{(i-1)}$ to $G_{[J]}^{(i-1)}$ and vice-versa. Since i is the first vertex in I but not in J, ϕ yields an automorphism of $G^{(i-1)}$ in which i is not a fixpoint. This contradicts our assumption that i is not free. □

Let $s := min\{t$, the number of free vertices of $G\}$. The following algorithm computes the largest s free vertices of G using the answers from the Querying Phase. Notice that the graphs in T_i^r, for $r \leq t$, have already been queried for membership in GA in the Querying Phase. The working of the algorithm should be clear from the above claim.

Computing free vertices:

Compute largest i such that $G^{(i-1)} \in \mathrm{GA}$ and let $j_1 := i$; (*j_1 *is the largest free vertex* *)
$r := 1$; $j := j_1 - 1$;
while $j > 0$ **and** $r \leq t$ **do**
  if $T_j^r \cap \mathrm{GA} \neq \emptyset$ **then**
    $r := r + 1$; $j_r := j$;
  endif
  $j := j - 1$;
endwhile;

Let these computed free vertices of G be $j_1, j_2, \ldots, j_s$, $s \le t$. We now compute the set of vertices to which j_r can be mapped by an automorphism of $G^{(j_r-1)}$. I.e., $orb(j_r) := \{j' \in V \mid j' \ne j_r$, and there is an automorphism of $G^{(j_r-1)}$ that maps j_r to $j'\}$, for each j_r, $1 \le r \le s$. The following claim characterizes $orb(j_r)$, for each j_r, $1 \le r \le s$.

Claim 3.1.3 *For each j_r, $1 \le r \le s$, $orb(j_r) = \{j' \mid G^{(j_r-1)}_{[I]} \mathring{\cup} G^{(j_r-1)}_{[J]} \in T^{r-1}_{j_r} \cap \mathrm{GA}$ for some I and J and j' is the first vertex in $J\}$.*

Proof of Claim 3.1.3. Let I and J be ordered subsets with j_r as the first element of I and j' as the first element of J. By definition both j_r and j' have the same color label. It follows that $G^{(j_r-1)}_{[I]} \mathring{\cup} G^{(j_r-1)}_{[J]} \in T^{r-1}_{j_r} \cap \mathrm{GA}$ for some such I and J iff there is an automorphism of $G^{(j_r-1)}$ that maps j_r to j'. □

Thus, for each j_r, $1 \le r \le s$, $orb(j_r)$ can be computed using the answers to the queries in the set $T^{r-1}_{j_r}$.

Claim 3.1.4 *Any graph $H = G^{(j_r-1)}_{[I]} \mathring{\cup} G^{(j_r-1)}_{[J]} \in T^{r-1}_{j_r} \cap \mathrm{GA}$ has exactly one non-trivial automorphism.*

Proof of Claim 3.1.4. Since H is in GA, it has at least one non-trivial automorphism. Let π_1 and π_2 be two non-trivial automorphisms of H. By Claim 3.1.1, both π_1 and π_2 map the subgraph $G^{(j_r-1)}_{[I]}$ to $G^{(j_r-1)}_{[J]}$ and vice-versa. Therefore, the automorphism $\pi_1\pi_2^{-1}$ maps the vertices of the subgraph $G^{(j_r-1)}_{[I]}$ to itself (and the subgraph $G^{(j_r-1)}_{[J]}$ to itself). From the proof of Claim 3.1.1 it follows that $\pi_1\pi_2^{-1}$ is id. Therefore, $\pi_1 = \pi_2$. □

Claim 3.1.5 *Let $H = G^{(j_r-1)}_{[I]} \mathring{\cup} G^{(j_r-1)}_{[J]} \in T^{r-1}_{j_r} \cap \mathrm{GA}$ be any graph and ϕ_H be its unique non-trivial automorphism. For vertices $l, m : j_r < l, m \le n$, $l \notin I$ and $m \notin J$, ϕ_H maps l to m iff the graph $G^{(j_r-1)}_{[I,l]} \mathring{\cup} G^{(j_r-1)}_{[J,m]} \in \mathrm{GA}$.*

Proof of Claim 3.1.5. From the proof of Claim 3.1.1 it follows that any nontrivial automorphism of $G^{(j_r-1)}_{[I,l]} \mathring{\cup} G^{(j_r-1)}_{[J,m]}$ maps the subgraph $G^{(j_r-1)}_{[I,l]}$ to $G^{(j_r-1)}_{[J,m]}$ and vice-versa. Therefore, $G^{(j_r-1)}_{[I,l]} \mathring{\cup} G^{(j_r-1)}_{[J,m]} \in \mathrm{GA}$ iff ϕ must map l to m. The claim follows. □

From the above claim it is clear that ϕ_H is easy to compute from the answers to the queries $\{G^{(j_r-1)}_{[I,l]} \mathring{\cup} G^{(j_r-1)}_{[J,m]} \mid j_r < l, m \le n, l \notin I, m \notin J\}$ made in the Querying Phase to the GA oracle.

From Proposition 2.2 it follows that we can easily compute for each vertex $j' \in orb(j_r)$ an automorphism of G mapping j_r to j'. Let this set of computed automorphisms be denoted as $Maps(j_r)$. Note that $|Maps(j_r)| = |orb(j_r)|$, for each j_r and $1 \le r \le s$.

Since $Maps(j_r) \cup \{id\}$ is a set of distinct coset representatives of the subgroup $Aut(G^{(j_r)})$ of $Aut(G^{(j_r-1)})$, it follows from elementary group theory that the set $\{\Pi_{1 \le r \le s}\psi_r \mid \psi_r \in Maps(j_r) \cup \{id\}\}$ of automorphisms of G is precisely the *entire* subgroup $Aut(G^{(j_s-1)})$ of $Aut(G)$. Moreover, $|Aut(G^{(j_s-1)})| = \Pi_{1 \le r \le s}(1 + |orb(j_r)|)$. We have computed the automorphisms in the entire subgroup $Aut(G^{(j_s-1)})$.

If $s < t$, then the whole of $Aut(G)$ is computed from the answers to the queries made to GA in the Querying Phase. In particular, $Auto(k, G)$ is computed. If $s = t$, then $\Pi_{1\leq r\leq t}(1+|orb(j_r)|)$ automorphisms of G are computed. Since $|orb(j_r)| \geq 1$ for $1 \leq r \leq t$, and $t = \lceil \log k \rceil$, it follows in this case also that $Auto(k, G)$ is computed from the answers to the queries in the Querying Phase.

It is easy to see that the time required to compute $Auto(k, G)$ is bounded by a fixed polynomial in the number of queries made in the Querying Phase which is $n^{O(\log k)}$. It is also easy to see that the lexicographically smallest k automorphisms are computed.

■

We now show as a consequence of Theorem 3.1 that some interesting problems related to GI and GA are truth-table equivalent to GA.

Corollary 3.2 *The following problems are truth-table equivalent to* GA*:*

1. $\mathrm{GI}_k = \{(G, H, \pi_1, \pi_2, \ldots, \pi_k) \mid \pi_1, \pi_2, \ldots, \pi_k$ *are* k *different isomorphisms between* G *and* H *and there exists another isomorphism between* G *and* H *different from these*$\}$, *for any* $k > 0$.[5]

2. $\mathrm{GA}_k = \{G \mid$ *the number of non-trivial automorphisms of* G *is at least* $k\}$, *for any* $k > 0$.

3. $\mathrm{GA}_{prime} = \{G \mid$ *the number of automorphisms of* G *is a prime number*$\}$.

Proof. We first show that GA is many-one reducible to GI_k and GA_k. Let G be an instance of GA (assume w.l.o.g. that G is connected and $|V(G)| > k$). Let the graph H be the *directed* cycle with k vertices. Notice that $Aut(H)$ is a cyclic group of order k. Let ψ be a generator of $Aut(H)$. Now, consider the graph $G \mathring{\cup} H$. Since $G \mathring{\cup} H$ has exactly two connected components, any automorphism of $G \mathring{\cup} H$ either maps $V(G)$ into itself and $V(H)$ into itself, or it maps $V(G)$ into $V(H)$ and $V(H)$ into $V(G)$. The latter case is not possible since $|V(G)| > k = |V(H)|$. Therefore, any automorphism of $G \mathring{\cup} H$ is an automorphism of the subgraph G when restricted to G and an automorphism of the subgraph H when restricted to H. For $1 \leq i \leq k$, let π_i be defined as the automorphism of $G \mathring{\cup} H$ such that $\pi_i = id$ when restricted to G and $\pi_i = \psi^{i-1}$ when restricted to H. Consider the mapping from GA to GI_k defined as $G \mapsto (G \mathring{\cup} H, G \mathring{\cup} H, \pi_1, \ldots, \pi_k)$. Clearly, $\pi_1, \ldots, \pi_k$ are isomorphisms from $G \mathring{\cup} H$ to $G \mathring{\cup} H$. Moreover $G \mathring{\cup} H$ has other isomorphisms iff G has a *nontrivial* automorphism. Thus the above mapping is a reduction from GA to GI_k. Similarly, it is easy to see that $G \mapsto G \mathring{\cup} H$ is a reduction from GA to GA_k.

In [LT92] it is shown that GA is truth-table reducible to UniqueGA (the language consisting of graphs that have a unique nontrivial automorphism). Observe that UniqueGA $\subseteq$ GA_{prime}. Now, it is easy to see that the reduction described in [LT92] is in fact also a truth-table reduction from GA to GA_{prime}.

The truth-table reduction from GA_k to GA is obvious from Theorem 3.1. The truth-table reduction generates the polynomially many nonadaptive queries to GA required to compute $Auto(k, G)$ (as in Theorem 3.1). Next, $Auto(k, G)$ is computed (using the

[5] Recall that this is Lubiw's open question mentioned in the introduction.

algorithm in Theorem 3.1 with the answers to the queries made to GA). The truth-table evaluates to true iff G has at least k distinct automorphisms.

The reduction from GI_k to GA is as follows. Let $(G, H, \pi_1, \ldots, \pi_k)$ be an instance of GI_k. Again, the truth-table reduction generates the polynomially many queries to GA required to compute $Auto(k+1, G)$. If any one of $\{\pi_1, \pi_2, \ldots, \pi_k\}$ is not an isomorphism from G to H then the truth-table evaluates to false regardless of the queries. Otherwise, $Auto(k+1, G)$ is computed. If there are $k+1$ automorphisms of G then the truth-table evaluates to true (since it implies that there are at least $k+1$ isomorphisms from G to H) otherwise it evaluates to false.

Finally, we describe a truth-table reduction from GA_{prime} to GA. Let G be any graph such that $|Aut(G)|$ is a prime number. Since $|Aut(G)| = \Pi_{1 \le i \le n} |Aut(G^{(i-1)})|/|Aut(G^{(i)})|$ it follows that there is exactly one vertex i such that $|Aut(G^{(i-1)})|/|Aut(G^{(i)})| > 1$. Thus G has exactly one free vertex. Now, let G be any instance of GA_{prime}. The truth-table reduction first makes polynomially many nonadaptive queries (as explained in Theorem 3.1) and using these query answers computes in polynomial time the largest two free vertices of G (if they exist). If $|Aut(G)|$ is a prime number, then, in fact, there is exactly one free vertex of the graph G. So, if G has no free vertices or more than one free vertex then the truth-table evaluates to false. Otherwise, compute all the automorphisms of G (since there is only one free vertex, we can do this in polynomial time as explained in Theorem 3.1) and accept if G has a prime number of automorphisms. Note that it is easy to test $|Aut(G)|$ for primality in polynomial time because if G has a unique free vertex it holds that $|Aut(G)| \le n$, which means $|Aut(G)|$ is logarithmic in the input size. ■

We can generalize the corollary for GA_{prime} to a somewhat larger language. Call a positive integer n *k-rough* if the prime factorization of n is $p_1^{e_1} p_2^{e_2} \ldots . p_r^{e_r}$ such that $\sum_{1 \le i \le r} e_i \le k$. A rough positive integer has very few prime factors (even including the multiplicities).[6]

Consider the decision problem: $\mathrm{GA}_{k\text{-}rough} = \{G \mid$ the number of automorphisms of G is a k-rough integer$\}$.

Clearly, from the arguments in the above proof for GA_{prime}, it follows that if an instance G is in $\mathrm{GA}_{k\text{-}rough}$ then G has at most k free vertices. Along similar lines as the proof for GA_{prime} we can easily show the following corollary.

Corollary 3.3 *1. For any $k > 0$, $\mathrm{GA}_{k\text{-}rough} = \{G \mid$ the number of automorphisms of G is a k-rough integer$\}$ is truth-table equivalent to* GA.

2. For any $k > 0$, $\mathrm{GI}_{k\text{-}rough} = \{\langle G_1, G_2 \rangle \mid G_1, G_2 \in \mathrm{GA}_{k\text{-}rough}$ and G_1 is isomorphic to $G_2\}$ is truth-table equivalent to GA.

Proof Sketch. To see that $\mathrm{GA}_{k\text{-}rough}$ is truth-table reducible to GA we first compute $k+1$ (if they exist) free vertices of a given instance G of $\mathrm{GA}_{k\text{-}rough}$ with parallel queries to GA. If $G \in \mathrm{GA}_{k\text{-}rough}$, then there will exist at most k free vertices. If j is a free vertex then it contributes a factor $|Aut(G^{(j)})|/|Aut(G^{(j+1)})|$ (whose value is at most n) to $|Aut(G)|$. Since $|Aut(G^{(j)})|/|Aut(G^{(j+1)})|$ is logarithmic in the input size, it can be easily factorized. Thus we can compute a prime factorization for $|Aut(G)|$. We can then check that $|Aut(G)|$ is k-rough.

[6] We choose to call these integers rough because in number theory a positive integer is called *smooth* if it has only small prime factors. Intuitively, smoothness is just the opposite of what we term as roughness.

To see that GA is truth-table reducible to $\mathrm{GA}_{k\text{-}rough}$, let G be an instance of GA. Now, it is easy to construct a graph H such that $|Aut(H)| = 2^k$ (H will be of size polynomial in k, which is constant). Notice that $|Aut(G \mathring{\cup} H)| = 2^k|Aut(G)|$. Thus $G \in$ GA iff $G \mathring{\cup} H \notin \mathrm{GA}_{k\text{-}rough}$.

We proceed to the second part. Let $\langle G_1, G_2\rangle$ be an instance of $\mathrm{GI}_{k\text{-}rough}$ such that $|Aut(G_1)| = |Aut(G_2)| = M$. It is known [KST92] that if $\langle G_1, G_2\rangle \in$ GI then $|Aut(G_1 \mathring{\cup} G_2)| = 2M^2$ and if $\langle G_1, G_2\rangle \notin$ GI then $|Aut(G_1 \mathring{\cup} G_2)| = M^2$. In order to see that $\mathrm{GI}_{k\text{-}rough}$ is truth-table reducible to GA, we can compute $k+1$ free vertices of G_1 (or all of them if fewer exist) and $k+1$ free vertices of G_2 (or all of them if fewer exist) and $2k+2$ free vertices of $G_1 \mathring{\cup} G_2$ (or all of them if fewer exist). Now, it can be checked in polynomial time from the computed free vertices of G_1 and G_2, whether $G_1, G_2 \in \mathrm{GA}_{k\text{-}rough}$. If $G_1, G_2 \in \mathrm{GA}_{k\text{-}rough}$, then, using the algorithm explained in Theorem 3.1, from the corresponding query answers obtained we can compute $|Aut(G_1)|$ and $|Aut(G_2)|$ and verify that $|Aut(G_1)| = |Aut(G_2)|$, which is, say M.

Now, $\langle G_1, G_2\rangle \notin$ GI iff $|Aut(G_1 \mathring{\cup} G_2)| = M^2$. Since $|Aut(G_1 \mathring{\cup} G_2)|$ is M^2 or $2M^2$, in any case $|Aut(G_1 \mathring{\cup} G_2)|$ is $2k+1$-rough. This can be easily verified from the $2k+2$ or fewer free vertices of $G_1 \mathring{\cup} G_2$ that have already been computed. Now, since $|Aut(G_1 \mathring{\cup} G_2)|$ is $2k+1$-rough we can actually compute $|Aut(G_1 \mathring{\cup} G_2)|$ exactly from the query answers and therefore check that the value is M^2. Since k is a constant, as a consequence of Theorem 3.1, this entire computation can be carried out in polynomial time with only nonadaptive queries to GA.

To see that GA is truth-table reducible to $\mathrm{GI}_{k\text{-}rough}$, we use the result from [KST92] that GA is truth-table equivalent to UGI (for Unique Graph Isomorphism whose 'yes' instances $\langle G_1, G_2\rangle$ have a unique isomorphism). It suffices for us to show that UGI is truth-table reducible to $\mathrm{GI}_{k\text{-}rough}$. Let $\langle G_1, G_2\rangle$ be an instance of UGI. If $\langle G_1, G_2\rangle$ is a 'yes' instance then clearly both G_1 and G_2 are rigid graphs. As said earlier it is easy to construct a graph gadget H such that $|Aut(H)| = 2^k$.

Then, $|Aut(G_1 \mathring{\cup} H)| = 2^k$ and $|Aut(G_2 \mathring{\cup} H)| = 2^k$ iff both G_1 and G_2 are rigid. Thus $G_1 \mathring{\cup} H$ and $G_2 \mathring{\cup} H$ are in $\mathrm{GA}_{k\text{-}rough}$ iff both G_1 and G_2 are rigid. Furthermore, notice that it can be ensured easily that $G_1 \mathring{\cup} H$ and $G_2 \mathring{\cup} H$ are isomorphic iff G_1 and G_2 are isomorphic. Finally, it is easy to see that $\langle G_1, G_2\rangle \in$ UGI iff $\langle G_1 \mathring{\cup} H, G_2 \mathring{\cup} H\rangle \in \mathrm{GI}_{k\text{-}rough}$. This completes the proof. ■

Finally, we mention an interesting consequence concerning program checking for the problems considered in this paper. The definitions and fundamental results can be found in [BK95].

It is known that GI is checkable [BK95]. It follows from the results of [LT92] (also see [KFM93]) that GA is *nonadaptively* checkable. I.e. the program checker needs to ask just one round of parallel queries to a purported program for GA in order to check it. However, it is an open question if GI is nonadaptively checkable.

The following theorem is a nonadaptive version of Beigel's trick for program checking [BK95]. We omit the proof of this theorem since it is essentially the same as that for Beigel's trick.

Theorem 3.4 *(Beigel's trick for nonadaptive checkers) Let π_1 and π_2 be two decision* *problems that are truth-table equivalent. The problem π_1 is nonadaptively checkable iff π_2 is nonadaptively checkable.*

As a consequence of the above theorem and the fact that GA is nonadaptively checkable it follows that the problems considered in Corollaries 3.2 and 3.3 in this paper are nonadaptively checkable.

Corollary 3.5 *For any $k > 0$, GI_k, GA_k, $\mathrm{GA}_{k\text{-}rough}$, $\mathrm{GI}_{k\text{-}rough}$ are nonadaptively checkable. The problem GA_{prime} is also nonadaptively checkable.*

Acknowledgements. We are grateful to the referees for providing useful comments.

References

[B89] J. L. Balcázar. Self-reducibility structures and solutions to NP problems. *Revista Matemática,* Universidad Complutense de Madrid, vol. 2, no. 2/3 (1989), 175-184.

[BDG88] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1988.

[BK95] M. Blum and S. Kannan. Designing programs that check their work. *Journal of the ACM,* pages 269-291, Vol. 43, No. 1, 1995.

[KFM93] L. Fortnow, S. Kannan, and S. Mahaney. Graph automorphism is nonadaptively checkable. Manuscript, 1993.

[LT92] A.Lozano and J. Torán. On the nonuniform complexity of the graph isomorphism problem. In *Proceedings of the Structure in Complexity Theory Conference*, pages 118–129, 1992.

[Har69] F. Harary. *Graph Theory.* Addison-Wesley, Reading, 1969.

[Hof82] C. M. Hoffmann. *Group-Theoretic Algorithms and Graph Isomorphism. Lecture Notes in Computer Science*, vol. 136, Springer-Verlag, 1982.

[KST92] J. Köbler, U. Schöning, and J. Torán. *Graph isomorphism: its structural complexity.* Birkhäuser, Boston, 1992.

[Lu81] A. Lubiw. Some NP-complete problems similar to Graph Isomorphism. *SIAM Journal on Computing*, 10:11–21, 1981.

[Ma79] R. Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8:131–132, 1979.

[Schn82] C. P. Schnorr. On self-transformable combinatorial problems. *Math. Programming Study*, 14:95–103, 1982.

[Sch88] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37:312–323, 1988.