Solvable Group Isomorphism is (almost) in NP ∩ coNP

V. Arvind
Institute of Mathematical Sciences
C. I. T. Campus
Chennai 600 113, India
arvind@imsc.res.in

Jacobo Torán
Theoretische Informatik
Universität Ulm
D-89069 Ulm, Germany
toran@informatik.uni-ulm.de

November 27, 2003

Abstract

The Group Isomorphism problem consists in deciding whether two input groups G_1 and G_2 given by their multiplication tables are isomorphic. We first give a 2-round Arthur-Merlin protocol for the Group Non-Isomorphism problem such that on input groups (G_1, G_2) of size n, Arthur uses $O(\log^6 n)$ random bits and Merlin uses $O(\log^2 n)$ nondeterministic bits. We derandomize this protocol for the case of solvable groups showing the following two results:

(a) When the input groups are solvable, we give a uniform NP machine for Group Non-Isomorphism, that works correctly on all but $2^{\text{polylog}(n)}$ inputs of any length n. Furthermore, this NP machine is always correct when the input groups are nonisomorphic. The NP machine is obtained by an unconditional derandomization of the AM protocol, and the derandomization is done using the Goldreich and Wigderson method [12] of extracting randomness from the input.

(b) Using the Nisan-Wigderson generator we get another derandomization of the above AM protocol under the assumption that EXP $\not\subseteq$ i.o−PSPACE. Thus, EXP $\not\subseteq$ i.o−PSPACE implies that Group Isomorphism for solvable groups is in NP ∩ coNP.

Finally, we use the above AM protocol to show that Group Isomorphism (for arbitrary groups) cannot be complete for the limited nondeterminism class NP$(\log^2 n)$ unless the coNP-complete problem $\overline{\text{CLIQUE}}$ has polynomial-size proofs that can be checked in subexponential time with a polynomial-size advice. We also show that the hardness of Group Isomorphism for the parameterized class W[1] would have a similar unlikely consequence for $\overline{\text{CLIQUE}}$.

1 Introduction

The Group Isomorphism problem, GROUP-ISO is stated as follows: given two groups G_1 and G_2 of order n as input given by their multiplication tables (also known as their Cayley tables), test whether they are isomorphic groups. Recall that an isomorphism between G_1 and G_2 is a bijection φ between the groups such that for every pair $i, j \in G_1$, $\varphi(ij) = \varphi(i)\varphi(j)$.[1] This is a natural problem and its computational complexity has already been studied for nearly three decades. Groups of order n have generator sets of size bounded by $\log n$ and because of this fact an isomorphism algorithm running in time $n^{\log n + O(1)}$ can be obtained by computing a generator set of size $\log n$ in G_1, mapping this set in every possible way to a set of elements in G_2. The map has to be

[1] For convenience we represent the group operation by concatenation in both groups.

extended to the entire group G_1 using the multiplication table and it has then to be checked that an isomorphism is defined. This algorithm is attributed to Tarjan in [18]. A stronger result showing that GROUP-ISO can be solved in space $O(\log^2 n)$ was given in [16]. Observe that Tarjan's algorithm can in fact be converted into a polynomial time nondeterministic procedure for GROUP-ISO that uses only $\log^2 n$ nondeterministic bits, by guessing the mapping from the generator in G_1 to G_2 instead of testing all possible 1-1 mappings.

For abelian groups it is known that GROUP-ISO can be solved in polynomial time. Also efficient algorithms for special classes of nilpotent groups can be found in [13]. However, no deterministic polynomial time algorithm for the problem is known. Indeed, from a computational complexity perspective, it is not known if GROUP-ISO is in NP $\cap$ coNP, even even for special classes of groups like solvable groups or even nilpotent groups.

In this paper we make progress on this question (mainly for solvable groups as input). Our results are summarized below.

1. We present a new Arthur-Merlin protocol for GROUP-NONISO with the property that on input groups of size n, Arthur uses $O(\log^6 n)$ random bits and Merlin uses only $O(\log^2 n)$ guess bits. The protocol works for arbitrary groups and is carefully tailored so it can be derandomized in the case of solvable groups.

2. We apply the powerful derandomization method of Goldreich and Wigderson [12] to derandomize the above protocol, by extracting randomness from the input itself. We exploit the fact that finite solvable groups have short presentations to show that this derandomization is incorrect for at most $2^{\log^{O(1)} n}$ inputs of length n.

3. Finally, under the assumption EXP $\not\subseteq$ i.o$-$PSPACE we use the Nisan-Wigderson generator [19] to completely derandomize the AM protocol for GROUP-NONISO, implying that GROUP-ISO is in NP $\cap$ coNP under the assumption.

We give some background to this line of research. Arthur-Merlin games were introduced by Babai in [5] as a randomized version of NP. It turns out that several important group problems, including the Graph Isomorphism problem, are in either NP $\cap$ coAM or AM $\cap$ coAM. With the development of derandomization techniques, building on Nisan and Wigderson's research [19], several results on derandomizing AM under a hardness assumption were shown. In [2], applying the methods of Nisan and Wigderson [19], an average-case hardness assumption is used to construct a pseudorandom number generator that suffices for derandomizing AM to NP. This result was improved to worst-case non-uniform hardness assumptions in [15] and MiVi99. Finally Lu gave in [17] a uniform worst-case assumption for the derandomization of AM.

GROUP-ISO is known to be polynomial-time reducible to Graph Isomorphism and it appears to be an easier problem since GROUP-ISO has an $n^{O(\log n)}$ time algorithm. In our approach to put GROUP-ISO in NP $\cap$ coNP, we basically proceed along similar lines as the above-mentioned research on Graph Isomorphism. However, there are two crucial results for GROUP-ISO, that allow us to derive new derandomization results for GROUP-ISO (not applicable to Graph Isomorphism). On the one hand, we design an AM protocol for GROUP-NONISO in which Arthur uses only a polylogarithmic number of random bits and Merlin uses a polylogarithmic number of guess bits. On the other hand, we exploit the fact that the inputs (for solvable groups) admit short encodings in a certain precise sense. We focus upon the case of solvable groups. It is known that solvable groups have *short presentations* [7]. It turns out that from such a presentation for a solvable group G, a group isomorphic to G can be easily recomputed (in time polynomial in $|G|$).

In Section 4 we use a derandomization method due to Goldreich and Wigderson [12]. The idea here is to extract randomness from the input string (using an extractor with suitable parameters). Notice that there is a penalty incurred in this approach: the derandomization procedure might be incorrect on the small fraction of input instances from which not enough randomness can be extracted. We use this method to show that the AM protocol for GROUP-NONISO can be transformed into a polynomial time nondeterministic algorithm that correctly solves the problem on all but at most $2^{\log^{O(1)} n}$ of the groups of order n. The fact that solvable groups have short presentations as described plays a crucial role. Also, it is important to mention that this result does not rely on any unproven hardness assumptions. Such a derandomization is not known for the AM protocol for Graph Isomorphism.

In Section 5, we further analyze the AM protocol for GROUP-NONISO and observe that the crucial verification part that Arthur has to do for solvable groups of order n, can in fact be done in polylogarithmic space in n. Using this fact we apply the Nisan-Wigderson [19] generator to derandomize the AM protocol for GROUP-NONISO under the plausible assumption EXP $\not\subseteq$ i.o$-$PSPACE.[2] This hardness assumption is weaker than the assumption used in [17], which is the only known uniform hardness assumption for derandomizing the class AM.

We observe here that our derandomization results cannot be used for arbitrary AM protocols which use polylog(n) random bits and guess bits. Our results exploit properties that are specific to the Group Isomorphism problem.

In Section 7 we briefly outline how these results can be extended to arbitrary finite groups assuming an unproven conjecture on short presentations for finite groups [7].

In Section 6 we apply the AM protocol result to show that Group Isomorphism cannot be complete for the limited nondeterminism class NP($\log^2 n$) unless the coNP-complete problem $\overline{\text{CLIQUE}}$ has polynomial-size proofs that can be checked in subexponential time with a polynomial-size advice. We also show that the hardness of Group Isomorphism for the parameterized class W[1] would have a similar unlikely consequence for $\overline{\text{CLIQUE}}$. These results are for general groups.

Most of the concepts used in the paper are defined when required. For the definition of standard complexity classes we refer the reader to books in the area like [4, 21].

2 AM protocol for Group Nonisomorphism

Denote by AM($r(n), s(n)$) the class of languages accepted by 2-round AM protocols, with error probability 1/4, in which Arthur uses $O(r(n))$ random bits and Merlin uses $O(s(n))$ nondeterministic bits. Formally, a language L is in AM($r(n), s(n)$) if there is a set $B \in$ P such that for all x, $|x| = n$,

$$\begin{aligned} x \in A &\Rightarrow \text{Prob}_{w \in_R \{0,1\}^{r'(n)}}[\exists y, |y| = s'(n) : \langle x, y, w\rangle \in B] \geq 3/4, \\ x \notin A &\Rightarrow \text{Prob}_{w \in_R \{0,1\}^{r'(n)}}[\forall y, |y| = s'(n) : \langle x, y, w\rangle \in B] \leq 1/4, \end{aligned}$$

where r' and s' are functions in $O(r(n))$ and $O(s(n))$ respectively. Notice that the above definition is equivalent to the definition in terms of 2-round Arthur-Merlin protocols. Indeed, the standard AM class is AM($n^{O(1)}, n^{O(1)}$).

We present a two-round AM protocol for Group Nonisomorphism that has constant success probability, and Arthur uses $O(\log^6 n)$ random bits and Merlin uses $O(\log^2 n)$ nondeterministic bits. Thus Group Nonisomorphism is in AM($\log^6 n, \log^2 n$).

[2] A language L is in i.o$-$PSPACE if there is a PSPACE machine that is correct on L for infinitely many input lengths.

Let G be a group with n elements. A sequence of k group elements $X = (g_1, \ldots, g_k)$ is called a *cube generating k-sequence* for G if

$$G = \{g_1^{\epsilon_1} g_2^{\epsilon_2} \cdots g_k^{\epsilon_k} \mid \epsilon_i \in \{0,1\}\}.$$

The set $\{g_1^{\epsilon_1} g_2^{\epsilon_2} \cdots g_k^{\epsilon_k} \mid \epsilon_i \in \{0,1\}\}$ is the *cube* $\mathrm{Cube}(X)$ generated by the sequence X. Erdös and Renyi [9] proved the following important theorem about the probability that $\mathrm{Cube}(X) = G$, for a k-sequence X chosen uniformly at random from G^k.

Theorem 2.1 [9] *Let G be a finite group with n elements. For $k \geq \log n + \log\log n + 2\log 1/\delta + 5$,*

$$\mathrm{Prob}_{X \in_R G^k}[X \textit{ is a cube generating sequence for } G] > 1 - \delta.$$

For G with n elements we choose $k = 4\log n$ and obtain the following useful corollary.

Corollary 2.2 *Let G be a finite group with n elements and $k = 4\log n$. Then*

$$\mathrm{Prob}_{X \in_R G^k}[u \textit{ is a cube generating sequence for } G] > 1 - 1/n.$$

In particular, for $k = 4\log n$, more than $(1 - 1/n)n^{4\log n}$ sequences in G^k are cube generating sequences for G.

Now, we make the following easy observation.

Proposition 2.3 *Let G be a group with n elements and S be a set of n elements. Let $\phi : G \to S$ be a 1-1 onto function. Then there is a group H (defined on the set S) such that ϕ is an isomorphism from G to H. Furthermore, given ϕ and the Cayley table for G, the Cayley table for H can be computed in polynomial time.*

Proof: It is easy to see that the group multiplication for H defined as follows will suffice:

$$x \cdot y = \phi(\phi^{-1}(x)\phi^{-1}(y)).$$

∎

Now, let G be a group with n elements and $X = (g_1, g_2, \ldots, g_k)$ be a cube generating sequence for G. There is a natural 1-1 mapping $\pi_X : G \to \{0,1\}^k$ that is defined by the cube generating sequence X. The mapping π_X is defined as follows: $\pi_X(g) = (\epsilon_1, \epsilon_2, \ldots, \epsilon_k)$, where $(\epsilon_1, \epsilon_2, \ldots, \epsilon_k)$ is the lexicographically first k-tuple in $\{0,1\}^k$ such that $g = g_1^{\epsilon_1} g_2^{\epsilon_2} \cdots g_k^{\epsilon_k}$. Clearly π_X is an injective mapping and, given the Cayley table for G as input, π_X can be computed in polynomial time.

Let $S = \pi_X(G)$. Applying the algorithm described in Proposition 2.3 we can construct the Cayley table of a group H isomorphic to G, where the isomorphism is given by the mapping π_X. We summarize this procedure in the following lemma.

Lemma 2.4 *Let G be a group with n elements given by its Cayley table, and X is a cube generating k-sequence for G. There is a polynomial (in n) time procedure $\mathcal{B}$ that takes as input the pair (X, G) and outputs a pair (Y, H), where H is the Cayley table of a group defined on the set $\pi_X(G) \subseteq \{0,1\}^k$ and $Y = \pi_X(X)$ is a cube generating sequence for H.*

Proof: It suffices to note that given π_X, we can apply the algorithm in Proposition 2.3 and compute the group H as a Cayley table in polynomial time. Furthermore, since G is isomorphic to H, it is easy to see that the image of X under the isomorphism will be a cube generating sequence for H. ∎

The following proposition is an important property of $\mathcal{B}$.

Proposition 2.5 *Let G_1 and G_2 be groups of order n and ϕ be an isomorphism from G_1 to G_2. Let X be a cube generating k-sequence of G_1. Then $\mathcal{B}(X, G_1) = (Y, H)$ implies $\mathcal{B}(\phi(X), G_2) = (Y, H)$.*

Proof: Let $X = (g_1, \ldots, g_k)$. Clearly, $\phi(X) = (\phi(g_1), \ldots, \phi(g_k))$ is a cube generating k-sequence for G_2. To prove the proposition, it suffices to observe that for every $g \in G_1$, $g = g_1^{\epsilon_1} g_2^{\epsilon_2} \cdots g_k^{\epsilon_k}$ if and only if $\phi(g) = \phi(g_1)^{\epsilon_1} \phi(g_2)^{\epsilon_2} \cdots \phi(g_k)^{\epsilon_k}$. Thus, $\pi_X(g) = (\epsilon_1, \ldots, \epsilon_k)$ if and only if $\pi_{\phi(X)}(\phi(g)) = (\epsilon_1, \ldots, \epsilon_k)$. ∎

Now, for a group G with n elements we define the following set.

$$\begin{aligned} C(G) \;=\; & \{(S, H, \psi) \mid \text{ there is a cube generating } 4\log n\text{-sequence } X \subset G \text{ such that} \\ & \mathcal{B}(X, G) = (S, H) \text{ and } \psi \in Aut(H)\}. \end{aligned}$$

Lemma 2.6 *If G_1 and G_2 are isomorphic finite groups then $C(G_1) = C(G_2)$ and if G_1 and G_2 are nonisomorphic then $C(G_1) \cap C(G_2) = \emptyset$.*

Proof: Suppose G_1 and G_2 are isomorphic and ϕ is an isomorphism from G_1 to G_2. Let $(S, H, \psi) \in C(G_1)$ and $\mathcal{B}(X, G_1) = (S, H)$. By Proposition 2.5 we have $\mathcal{B}(\phi(X), G_2) = (S, H)$. Thus, $(S, H, \psi) \in C(G_2)$. It follows that $C(G_1) \subseteq C(G_2)$. By a similar argument, $C(G_2) \subseteq C(G_1)$, and hence $C(G_1) = C(G_2)$.

On the other hand, suppose (S, H, ψ) is $C(G_1) \cap C(G_2)$. Then clearly it follows that $G_1 \cong H$ and $G_2 \cong H$, which implies that $G_1 \cong G_2$. ∎

Let $C(G_1, G_2) = C(G_1) \cup C(G_2)$. We estimate the size of $C(G_1, G_2)$ in the two cases: $G_1 \cong G_2$ and $G_1 \ncong G_2$.

Lemma 2.7 *For a group G with n elements*

$$(1 - 1/n) n^{4 \log n} \leq |C(G)| \leq n^{4 \log n}.$$

Proof: Let $k = 4 \log n$ and $N_k(G) \subseteq G^k$ be the collection of cube generating k-sequences for the group G. By Theorem 2.1 and Corollary 2.2 we have

$$(1 - 1/n) n^{4 \log n} \leq |N_k(G)| \leq n^{4 \log n}$$

by the choice of $k = 4 \log n$.

It suffices to show that $|N_k(G)| = |C(G)|$. We consider the following natural action of the automorphisms of G on cube generating k-sequences in G: $\rho \in Aut(G)$ maps a cube generating k-sequence $X = (g_1, \ldots, g_k)$ to the cube generating k-sequence $\rho(X) = (\rho(g_1), \ldots, \rho(g_k))$. Now, let X_1 and X_2 be two cube generating k-sequences of G such that $\mathcal{B}(X_1, G) = \mathcal{B}(X_2, G) = (S, H)$. Then $\pi_{X_2}^{-1} \pi_{X_1}$ is an automorphism of G that maps X_1 to X_2. On the other hand, suppose ρ is an

automorphism of G. Let $\mathcal{B}(X_1, G) = (S, H)$, where X_1 is a cube generating k-sequence of G. Then, by Proposition 2.5, $\mathcal{B}(\rho(X_1), G) = (S, H)$. Thus, for two cube generating k-sequences X_1 and X_2 of G, $\mathcal{B}(X_1, G) = \mathcal{B}(X_2, G) = (S, H)$ if and only if there is an automorphism ρ of G that maps the sequence X_1 to the sequence X_2. Notice that only the identity automorphism of G fixes a cube generating k-sequence. Hence, it follows that for each $(S, H, \psi) \in C(G)$ there are exactly $|Aut(G)|$ generating k-sequences X such that $\mathcal{B}(X, G) = (S, H)$. But $(S, H, \psi) \in C(G)$ implies H and G are isomorphic and therefore $|Aut(G)| = |Aut(H)|$. Putting it together, it follows that $|C(G)|$ is exactly the number of cube generating k-sequences of G. I.e. $|N_k(G)| = |C(G)|$. ∎

We have the immediate corollary which is crucial to the AM protocol.

Corollary 2.8 *Let G_1 and G_2 be groups of n elements. For $n > 4$ we have:*

1. *$G_1 \cong G_2$ implies $|C(G_1, G_2)| \leq n^{4 \log n}$.*
2. *$G_1 \ncong G_2$ implies $|C(G_1, G_2)| > 1.5 n^{4 \log n}$.*

Proof: It directly follows from the previous lemma that if G_1 and G_2 are isomorphic groups of n elements then $(1 - 1/n)n^{4 \log n} \leq |C(G_1, G_2)| \leq n^{4 \log n}$ and if G_1 and G_2 are nonisomorphic then $(1 - 1/n) \cdot 2 \cdot n^{4 \log n} \leq |C(G_1, G_2)| \leq 2 \cdot n^{4 \log n}$. The corollary follows as $(2 - 2/n) > 1.5$ for $n > 4$. ∎

In the sequel let m denote $n^{4 \log n}$. Let the set X be the 7-fold Cartesian product $C(G_1, G_2)^7$ of $C(G_1, G_2)$ with itself. As we have shown above, if $G_1 \cong G_2$ then $|X| \leq m^7$ and if they are not isomorphic then $|X| \geq 1.5^7 \cdot m^7$. Although $|X| = n^{O(\log n)}$, the elements in X have polynomial in n length. We can assume that each element $x \in X$ is encoded as a positive integer $\text{num}(x)$, so that in the AM protocol (which we are about to describe) we can use fingerprints obtained from Chinese remaindering in order to restrict the number of random bits needed. Let p be a random prime number of $\log^3 n$ bits. Since $|X| = n^{O(\log n)}$, with probability more than $1 - 2^{-\log^2 n}$ the following event succeeds: $\text{num}(x)$ $(mod\ p) \neq \text{num}(y)$ $(mod p)$ for every pair $x, y \in X$ such that $x \neq y$. Call p a *good* prime if this event succeeds for p. For a good prime, let X_p denote the set $\{\text{num}(x)\ (mod p) \mid x \in X\}$. Then $|X_p| = |X|$ for good primes p. Notice that elements of X_p can be represented as bit strings of length $t = \log^3 n$. In the first part of the protocol Arthur selects a random prime number p of $\log^3 n$ bits. This can be done by sampling random numbers for the given length and using the primality test algorithm [1]. With a sample of size $\text{O}(\log^3 n)$, Arthur has a constant success probability of picking a random prime number. Thus, $O(\log^6 n)$ random bits are needed for this first task.

We now recall a version of Sipser's hashing lemma [23].

Lemma 2.9 *Let X be a nonempty subset of Σ^t that does not contain 0^n. Let S be a random variable denoting the number of strings in X mapped to 0^k by a uniformly chosen random linear transformation h over $\mathbb{F}_2$, $h : \Sigma^t \to \Sigma^k$. Then we have $E(S) = 2^{-k} \cdot |X|$ and $Var(S) = 2^{-k}(1 - 2^{-k}) \cdot |X|$.*

Let $k = \log(4m^7)$ and let $h : \Sigma^t \to \Sigma^k$ be a random linear transformation as in the above lemma, where $X_p \subseteq \Sigma^t$. We are interested in the event that there is an $x \in X_p$ such that $h(x) = 0^k$. By the above lemma, if $G_1 \ncong G_2$ then $E(S) \geq 4$ and $E(S) \leq 1/4$ otherwise. Using Chebyshev's inequality it now follows that if $G_1 \cong G_2$ then

$$\text{Prob}_h[\exists x \in X_p \ : \ h(x) = 0^k] \leq 1/4$$

and if $G_1 \ncong G_2$ we have

$$\text{Prob}_h[\exists x \in X_p \ : \ h(x) = 0^k] \geq 3/4.$$

We now describe the $\text{AM}(\log^6 n, \log^2 n)$ protocol below.

Arthur Randomly sample numbers of $\log^3 n$ bits until a prime number p is found. If after $5 \log n$ trials no prime number has been found, then reject the input. Otherwise pick a random $\mathbb{F}_2$-linear function $h : \Sigma^t \to \Sigma^k$, where $t = \log^3 n$ and $k = \log(4m^7)$. Send p and h to Merlin.

Merlin Sends back two 7-tuples $\langle x_1, \ldots, x_7 \rangle$ and $\langle i_1, \ldots, i_7 \rangle$. Where for $j = 1, \ldots, 7$, $i_j \in \{1, 2\}$ and $x_j = (S_j, T_j, \psi_j)$, where S_j is a sequence of elements of $\{0,1\}^{4 \log n}$ of length $4 \log n$, T_j is a cube generating $4 \log n$-sequence for the group G_{i_j}, and $\psi_j : S_j \to \{0,1\}^{4 \log n}$ is a 1-1 mapping.

Arthur For each $j = 1, \ldots, 7$, Arthur does the following verification, all in polynomial time: Let $x_j = (S_j, T_j, \psi_j)$. Computes $\mathcal{B}(T_j, G_{i_j}) = (S, H_j)$ and verifies that $S = S_j$. Then using the group multiplication of H_j, Arthur extends ψ_j to all of H_j and verifies that it is an automorphism of H_j. Now, let $y_j = (S_j, H_j, \psi_j)$ for $1 \leq j \leq 7$ and let $y = \langle y_1, \ldots, y_7 \rangle$ which is an element of X. Verify that $h(\text{num}(y) \ (mod p)) = 0^k$ and if so, accept the pair (G_1, G_2) as nonisomorphic.

The analysis given before the protocol proves that the AM protocol accepts nonisomorphic pairs with probability $1/2 + \epsilon$ and rejects isomorphic pairs with probability $1/2 + \epsilon$, for some constant $\epsilon > 0$. Furthermore, notice that Arthur uses $O(\log^6 n)$ random bits and Merlin uses $O(\log^2 n)$ nondeterministic bits. The error probability can be bounded by $1/4$ which only a constant factor increase in the random bits of Arthur and nondeterministic bits of Merlin. We have thus proved the following theorem.

Theorem 2.10 *There is a 2-round AM protocol with error probability* $1/4$ *for Group Nonisomorphism in which Arthur uses* $O(\log^6 n)$ *random bits and Merlin uses* $O(\log^2 n)$ *nondeterministic bits. Hence, the problem is in* $\text{AM}(\log^6 n, \log^2 n)$.

Remark. We briefly explain how to efficiently compute $\text{num}(y)(mod p)$, given $y = \langle y_1, \ldots, y_7 \rangle \in X$, where $y_j = (S_j, H_j, \psi_j)$ for $1 \leq j \leq 7$.

Consider y_j. Each T_j is contained in $\{0,1\}^{4 \log n}$ and has $4 \log n$ elements. It can be encoded as a binary string of length $(4 \log n)^2$. Similarly, each H_j is a list of n^2 triples from $\{0,1\}^{4 \log n} \times \{0,1\}^{4 \log n} \times \{0,1\}^{4 \log n}$ and can be encoded as a binary string of length $12n^2 \log n$. Likewise, each ϕ_j consists of $4 \log n$ pairs of strings from $\{0,1\}^{4 \log n}$, and can be encoded as a binary string of length $2(4 \log n)^2$. The entire encoding is a concatenation of the encodings of the y_j, prefixed by a 1 to give us $\text{num}(y)$. The length of $\text{num}(y)$ is $7(12n^2 \log n + 3(4 \log^2 n)) + 1$. Furthermore, it is easy to see that given y as input we can compute $\text{num}(y)(mod p)$ polynomial time and polylog(n) space.

We note here, without proof, that it is also possible to give a 2-round IP protocol for Group Nonisomorphism which is somewhat more efficient in the number of random bits used by the verifier.

3 Short Presentations for Solvable Groups

The goal in the next sections is to derandomize the AM protocol for the case of solvable groups. We will present two derandomization results. A key ingredient for these results is the representation of the groups is a succinct way in terms of generators and relations. To this end we need to develop some group theory, including some simple algorithms.

We recall some group-theoretic definitions. Let A and B be two groups. We write $B < A$ to denote that B is a subgroup of A. The subgroup B of A is said to be normal if for all a in A, $aB = Ba$, and we write $B \lhd A$ to denote that B is a normal subgroup of A. A group is simple if it does not contain any nontrivial normal subgroups.

A *normal series* of a group A is a finite sequence of subgroups $A_0, \ldots, A_k$ such that

$$I = A_0 \lhd A_1 \lhd \ldots \lhd A_k = A.$$

A normal series is a *composition series* if each factor group A_{i+1}/A_i is a simple group (i.e. it has no nontrivial normal subgroup).

A group A is solvable if it has a normal series such that each normal factor A_{i+1}/A_i is abelian. Equivalently, each factor in the composition series for G will be cyclic.

Let G be a solvable group with n elements given by its Cayley table. First, in time polynomial in n we can obtain a composition series for G:

$$G = G_k \rhd G_{k-1} \rhd \cdots \rhd G_1 = \{1\},$$

where G_{i+1}/G_i is a cyclic group of prime order, say p_i, for $i = 1, 2, \ldots, k-1$. Thus, $n = \prod_{i=1}^{k-1} p_i$. To obtain such a composition series in polynomial time, it suffices to note that we can find a normal subgroup N of G in polynomial time: this is done by taking the normal closure of $x \in G$ for different elements $x \in G$ until one of them actually gives us a nontrivial normal subgroup. Note, that the normal closure of each $x \in G \setminus \{1\}$ gives the whole of G if and only if G is already simple (which would mean G is cyclic since we are dealing with solvable groups). Now, the normal series $G \rhd N \rhd \{1\}$ can be refined by recursively applying the same step to N and G/N. The overall algorithm will be polynomial time (in n). This would also test G for solvability, because if G is not solvable then at least one of the factor groups G_{i+1}/G_i in the composition series will be nonabelian.

Coming back to the solvable case, the algorithm will also give us a generator $a_{i+1}G_i$ for G_{i+1}/G_i for $1 \leq i \leq k-1$.

Notice that every element of G is uniquely expressible as $a_k^{l_k} a_{k-1}^{l_{k-1}} \ldots a_2^{l_2}$, where $0 \leq l_j \leq p_j - 1$ for each j. Thus, we can rename the elements of G using such products $a_k^{l_k} a_{k-1}^{l_{k-1}} \ldots a_2^{l_2}$ and rewrite the Cayley table using these products.

Definition 3.1 A presentation of a group G with n elements is a pair (X, R) where

X is a generating set for G,

R is a set of words over $G \cup G^{-1}$ such that $w \in R$ defines the equation $w = 1$ and

(X, R) defines the group G in the sense that there is an algorithm that on input a presentation for a group G computes the Cayley table of a group G' isomorphic to G.

For a constant $c > 0$ we say that a presentation (X, R) is c-short if its length $|(X, R)|$ is at most $\log^c n$.

We now show that in time polynomial in n we can convert the composition series for G into a short presentation for G with generating set $\{a_2, \ldots, a_k\} = X$. This is done inductively, by obtaining it for G_i for increasing values of i starting from $i = 1$. The base case is trivial for $i = 1$. Thus, it clearly suffices to show that if (X_i, R_i) is a short presentation for G_i, then in time polynomial in n we will obtain a short presentation (X_{i+1}, R_{i+1}) for G_{i+1}.

We let $X_{i+1} = X_i \cup \{a_{i+1}\}$. In order to define R_{i+1} we only need to use $a_{i+1}G_i = G_i a_{i+1}$, which follows from the normality of G_i in G_{i+1}. In particular, this will give rise to the following set of relations.

$$a_{i+1}^{p_{i+1}} = u_{i+1} \quad , \quad \text{where } u_{i+1} \in G_i. \tag{1}$$

$$a_j a_{i+1} = a_{i+1} w_{i+1,j} \quad : \quad 1 \leq j \leq i, w_{i+1,j} \in G_i. \tag{2}$$

$$\tag{3}$$

Notice that here u_{i+1} and $w_{i+1,j}$ are words of the form $a_i^{l_i} a_{i-1}^{l_{i-1}} \ldots a_2^{l_2}$, where $0 \leq l_j \leq p_j - 1$ for each j, defining elements of G_i, as explained.

We define R_{i+1} as the union of the above relations and R_i and claim that (X_{i+1}, R_{i+1}) defines the group G_{i+1}. To see this it suffices to note that R_{i+1} completely specifies the Cayley table for G, in terms of the renamed elements $a_{i+1}^{l_{i+1}} a_i^{l_i} \ldots a_2^{l_2}$. This is because, the relations in R_{i+1} will clearly allow arbitrary words over X_{i+1} to be rewritten in the form $a_{i+1}^{l_{i+1}} a_i^{l_i} \ldots a_2^{l_2}$, where $0 \leq l_j \leq p_j - 1$ for each j. Thus we have the following theorem.

Theorem 3.2 *Let G be solvable group with n elements, where G is given by its Cayley table, and let $G = G_k \rhd G_{k-1} \rhd \cdots \rhd G_1 = \{1\}$ be a composition series for G, where G_{i+1}/G_i is a cyclic group of prime order p_{i+1} generated by $a_{i+1}G_i$ for each i. Then there is a polynomial (in n) time algorithm that inductively computes a short presentation (X, R) for G, described above, which includes a short presentation (X_i, R_i) for each group G_i in the composition series. Moreover, the size $|(X, R)|$ of the short presentation is $c \log^c n$, where c is a fixed constant independent of the group.*

We now describe an algorithm that on input a presentation (X, R) for a group G, constructed as above, efficiently computes the Cayley table of a group G' isomorphic to G. Let G be a solvable group with n elements given by a short presentation (X, R) obtained from G by applying the algorithm of Theorem 3.2. Consider $X = \{a_k, a_{k-1}, \ldots, a_2\}$ as an ordered list. Then the elements of G' will consist of words of the form $a_k^{l_k} a_{k-1}^{l_{k-1}} \ldots a_2^{l_2}$, where $0 \leq l_j \leq p_j - 1$ for each j. Notice that the primes p_j are already part of the presentation. We now describe how to obtain the Cayley table for G' from (X, R). Since X is an ordered list, we can efficiently obtain from (X, R) the presentations (X_i, R_i) for $2 \leq i \leq k$, where $(X_k, R_k) = (X, R)$, and the generating list $X_i = \{a_i, \ldots, a_2\}$. Let G'_i be the group defined by (X_i, R_i) for $2 \leq i \leq k$ (note that $G'_1 = \{1\}$). We will obtain the Cayley table for G_i inductively, for increasing values of i. It suffices to show how to obtain the Cayley table for G'_{i+1} from (X_{i+1}, R_{i+1}), given the Cayley table for G'_i. Notice that elements of G'_{i+1} are $a_{i+1}^{l_{i+1}} a_i^{l_i} \ldots a_2^{l_2}$, where $0 \leq l_j \leq p_j - 1$ for each j. Thus, we only need to express the product of each pair of elements $a_{i+1}^{l_{i+1}} a_i^{l_i} \ldots a_2^{l_2}$ and $a_{i+1}^{m_{i+1}} a_i^{m_i} \ldots a_2^{m_2}$ from G'_{i+1} again as an element of G'_{i+1}. To this end, write $a_{i+1}^{l_{i+1}} a_i^{l_i} \ldots a_2^{l_2}$ as $a_{i+1}^{l_{i+1}} u$ and $a_{i+1}^{m_{i+1}} a_i^{m_i} \ldots a_2^{m_2}$ as $a_{i+1}^{m_{i+1}} v$, where $u, v \in G'_i$. Now, by repeatedly applying the relations defined in Equation 1, in time polynomial in n we can rewrite $a_{i+1}^{l_{i+1}} u a_{i+1}$ as

$$a_{i+1}^{l_{i+1}} u \cdot a_{i+1} = a_{i+1}^{l_{i+1}+1} v_1 v_2 \ldots v_r = a_{i+1}^{l_{i+1}+1} w, \tag{4}$$

where the v_j's are elements of G'_i and $r = \sum_{t=2}^{i+1} l_t$, and $v_1 v_2 \ldots v_r = w \in G'_i$ obtained in polynomial time by using the multiplication of G'_i. It is clear, that on m_{i+1} applications of the rewrite rule given by Equation 4, we will obtain an element of the form $a_{i+1}^{l_{i+1}+m_{i+1}} w_0$, where $w_0 \in G'_i$. Finally, we can further use Equation 1 to reduce $a_{i+1}^{l_{i+1}+m_{i+1}} w_0$ to $a_{i+1}^{l} w$, where $0 \leq l \leq p_{i+1} - 1$. Clearly, the computation of this entry of the Cayley table requires polynomial in n time. Proceeding thus, the Cayley table for G' can be obtained in polynomial time. It is clear that G is isomorphic to G'. We summarize the above observation.

Theorem 3.3 *Let (X, R) be a short presentation as obtained by Theorem 3.2, for a solvable group G with n elements. There is an algorithm with running time bounded by a polynomial in n that constructs from (X, R) the Cayley table of a group G' that is isomorphic to G.*

The polynomial time algorithm of Theorem 3.2 can be modified to a polylog(n) space bounded algorithm that takes (X, R) (obtained from a solvable group G of order n) as input and outputs the Cayley table of a group G' isomorphic to G. This will be needed in the second derandomization result. For this we first note that the algorithm of Theorem 3.3 is essentially a breadth-first search algorithm: we need to compute and store the table for G'_i, which is used to build the table for G'_{i+1}. Thus, we get a polynomial time algorithm that requires $O(n^{O(1)})$ space. However, instead of storing the table for G'_i we can recursively compute the product of two words $u, v \in G'_i$ when required. Notice that for the recursive computation what is put on the stack, for each recursive call, is of polylog(n) size and the depth of the recursion is bounded by $\log n$ (which bounds the length of the composition series). Thus, the overall space required by this modified algorithm (which is depth-first) is polylog(n). Observe that the time taken by this algorithm is not polynomial anymore. We summarize this observation in the following theorem.

Theorem 3.4 *Let (X, R) be a short presentation as obtained by Theorem 3.2, for a solvable group G with n elements. There is a space-bounded algorithm that requires space $\log^{O(1)} n$, and on input (X, R) constructs the Cayley table of a group G' that is isomorphic to G.*

4 Derandomization without Assumptions

Throughout this section we assume that the input instances (G_1, G_2) for GROUP-ISO are such that G_1 and G_2 are solvable groups. Given an instance (G_1, G_2), by applying the algorithm of Theorem 3.2 and then the algorithm of Theorem 3.3 to both G_1 and G_2 we can obtain a new pair of groups (G'_1, G'_2) such that $G_1 \cong G_2$ if and only if $G'_1 \cong G'_2$. We call such an instance (G'_1, G'_2) a *reduced instance* of GROUP-ISO. The key observation of this section is that there is a constant c such that the number of reduced instances (G'_1, G'_2) for pairs of graphs with n elements is bounded by $2^{\log^c n}$. This is immediate from the bound on the size of short presentations for solvable groups given in Theorem 3.2.

Lemma 4.1 *The number of reduced instances (G'_1, G'_2) is bounded by $2^{\log^c n}$ for a fixed constant $c > 0$, where G'_1 and G'_2 are groups with n elements.*

For the derandomization of the AM protocol for GROUP-NONISO we give an easy generalization of a theorem from the Goldreich and Wigderson paper [12, Theorem 3] for a nondeterministic setting. The idea is to try and derandomize certain advice-taking randomized algorithms by extracting randomness from the input. It can be proved almost exactly as [12, Theorem 3].

Theorem 4.2 *Let M be an advice-taking* NP *machine for a problem Π, where the length of the advice is bounded by $\log^c m$ for some constant c, for inputs $x \in \{0,1\}^m$. Suppose it holds that at least $2/3$ fraction of the $\log^c m$-bit advice strings are good advice strings. More precisely*

$$\text{Prob}_{w \in \{0,1\}^{\log^c m}}[\forall x \in \{0,1\}^m \text{ it holds that } M(x,w) \text{ is correct}] \geq 2/3.$$

Then for every $\epsilon > 1$, there is an NP machine M' for Π that is incorrect on at most $2^{\log^{c\epsilon} m}$ inputs $x \in \{0,1\}^m$.

In order to be able to use this result we have to transform the AM protocol for GROUP-NONISO of Section 2 into an advice taking NP machine (with short advice) for the problem. The standard amplification of the success probability of the AM protocol would not work since the resulting advice string would be of polynomial length. We show how the AM protocol can be modified in order to avoid this problem.

Fix a standard encoding of an instance (G_1, G_2) of groups of with n elements by a binary string of length $m = Cn^2\lceil \log n \rceil$, where C is some fixed constant. Furthermore, we can assume that both this encoding and its inverse are computable in time polynomial in n. In this section m stands for this number. We can assume that the AM protocol Section 2 takes as input a string $x \in \{0,1\}^m$ and first checks if it encodes an instance (G_1, G_2) of solvable groups and rejects if it does not. We can think of the binary strings x as the input to the AM protocol.

The AM protocol of Section 2 is modified as follows: on input $x \in \{0,1\}^m$, first Arthur decodes x to get (G_1, G_2) and checks that G_1 and G_2 are solvable groups. Then, applying the algorithms of Theorems 3.2 and 3.3 in succession, Arthur converts (G_1, G_2) to a reduced instance (G_1', G_2'). Now, Arthur starts the AM protocol *for the reduced instance* (G_1', G_2'). Merlin is also supposed to compute (G_1', G_2') and execute his part of the protocol for (G_1', G_2'). Observe that by Lemma 4.1 there are only $2^{\log^c n}$ reduced instances for a fixed constant $c > 0$. Notice that effectively the original AM protocol is now being applied only to reduced instances. By standard methods of amplifying the success probability of the AM protocol, we can convert the AM protocol to a ($\log^{O(1)} n$ size) advice taking NP machine M. We summarize the above observation as a theorem.

Theorem 4.3 *There is an ($\log^{(O(1)} n$ size) advice-taking* NP *machine M for* GROUP-NONISO *such that that for inputs $x \in \{0,1\}^m$ the following holds:*

$$\text{Prob}_{w \in \{0,1\}^{\log^c n}}[\forall x \in \{0,1\}^m \text{ it holds that } M(x,w) \text{ is correct}] \geq 2/3.$$

The AM protocol can be transformed using well-known techniques into a one-sided error protocol for GROUP-NONISO such that when the input groups are nonisomorphic, the protocol accepts with probability 1, where the protocol still uses only a polylogarithmic number of random bits. Consequently, the advice-taking NP machine M defined above also has only one-sided error.

Now, applying Theorems 4.3 and 4.2 we immediately have the following consequence for GROUP-ISO in the case of solvable groups.

Theorem 4.4 *For some constant $c > 1$ there is an* NP $\cap$ coNP *machine M for* GROUP-ISO *for solvable groups that is incorrect on at most $2^{\log^c m}$ inputs $x \in \{0,1\}^m$ for every m.*

The proof follows by combining the standard NP machine for GROUP-ISO with the NP machine M' for GROUP-NONISO given by Theorem 4.2. Observe that M' may be incorrect only when its input is a pair of isomorphic groups.

5 Derandomization under the Assumption EXP $\not\subseteq$ i.o−PSPACE

In the previous section we proved, unconditionally, that there is an NP $\cap$ coNP machine M for GROUP-ISO for solvable groups that is incorrect on only a few inputs. In this section we apply the Nisan-Wigderson pseudorandom generator construction to prove that GROUP-ISO for solvable groups is in NP $\cap$ coNP assuming EXP $\not\subseteq$ i.o−PSPACE/poly[3]. Since EXP $\subseteq$ i.o−PSPACE/poly implies EXP $\subseteq$ i.o−PSPACE the result holds also under the uniform hardness assumption EXP $\not\subseteq$ i.o−PSPACE.

Theorem 5.1 *If* EXP $\not\subseteq$ i.o−PSPACE/poly *then* GROUP-ISO *for solvable groups is in* NP$\cap$coNP.

Proof:

Notice that in order to derandomize the AM protocol for GROUP-NONISO, it suffices to build a pseudorandom generator that stretches $O(\log n)$ random bits to $O(\log^6 n)$ random bits, such that the pseudorandom string of $O(\log^6 n)$ bits cannot be distinguished from a truly random string of the same length by the protocol.

We start by carefully examining the AM protocol for GROUP-NONISO. Similar to the proof of Theorem 4.4, in the protocol Arthur begins by converting the input instance (G_1, G_2) to a reduced instance (G_1', G_2') (Merlin is expected to execute his steps of the protocol for (G_1', G_2')). This is Phase I of the protocol. In Phase II of the protocol Arthur picks a random hash function h and a random prime p, of suitable size as explained in the protocol, using $O(\log^6 n)$ random bits and sends these to Merlin.

The remainder of the protocol is Phase III: Merlin sends back a string x of length $O(\log^2 n)$ and Arthur then performs a polynomial (in n) time deterministic computation. This final computation of Arthur consists of two parts. The first part converts the string x to a string $y = f(x)$ that is of length $O(n^2 \log n)$. It then converts y to the natural number num(y) and computes a t bit string $z = (\text{num}(y))(mod\ p)$. Then Arthur verifies that $h(z) = 0^k$. We recall from the protocol description that t, k, the sizes of h and p are all polylog(n) bits.

Next, we will restructure Phase III of Arthur's computation as an oracle computation with running time polynomial in $\log n$. This is the crucial part of the proof.

For that purpose we define the following language L. Let (Y_1, R_1) and (Y_2, R_2) be the short presentations corresponding to the solvable groups G_1' and G_2' respectively, computed in Phase I by Arthur, as explained in Theorems 3.2 and 3.3.

$$
\begin{aligned}
L \;=\; \{((Y_1,R_1),(Y_2,R_2),h,p) \mid\; & \exists x=(x_1,\ldots,x_7) \;:\; x_j=(X_j,b_j,T_j,\phi_j)\ ,\ \mathcal{B}(X_j,G_{b_j})=(T_j,H_j),\\
& \text{each } w\in X_j \text{ is expressed } using Y_{b_j} \text{ and } \phi_j \;:\; T_j\to\{0,1\}^{4\log n}\\
& \text{extends to an automorphism of } H_j\ ,\ 1\le j\le 7\ ,\quad \text{and } h(\text{num}(y)(mod p))=0^k\\
& \text{where } y=((T_1,H_1,\phi_1),\ldots,(T_7,H_7,\phi_7))\}
\end{aligned}
$$

Observe that the languages L is designed so that Phase III of the AM protocol can be replaced with a single query to L, where (Y_1, R_1) and (Y_2, R_2) are the short presentations computed for G_1 and G_2 respectively. Thus the entire AM protocol can be replaced by the following three steps:

1. On input G_1 and G_2, compute their short presentations (Y_1, R_1) and (Y_2, R_2).

[3] A language L is in i.o−PSPACE/poly if there is a PSPACE machine that takes polynomial-size advice and is correct on L for infinitely many input lengths.

2. Uniformly at random pick h and p using $O(\log^6 n)$ random bits.

3. Query L for $((Y_1, R_1), (Y_2, R_2), h, p)$ and accept if it is in L.

It is clear that on all inputs (G_1, G_2) the above computation has the same acceptance probability as the AM protocol.

Furthermore, a crucial property of the language L is the following.

Claim 1. Checking if $((Y_1, R_1), (Y_2, R_2), h, p)$ is in L can be done in polylog(n) space, where n is the size of the inputs groups.

To prove the claim we outline a polylog(n) space algorithm by explaining its main aspects:

- In the definition of L notice that the guess $x = (x_1, \ldots, x_7)$ is of length $O(\log^2 n)$. The polylog(n) space algorithm will exhaustively search for x by cycling through all strings of length $O(\log^2 n)$. Observe that for an $x = (x_1, \ldots, x_7)$, where $x_j = (X_j, b_j, T_j, \phi_j)$, each $w \in X_j$ is expressed in terms of the generators in Y_j. From Section 3 we know that such an expression is of polylog(n) size.

- For an $x = (x_1, \ldots, x_7)$, where $x_j = (X_j, b_j, T_j, \phi_j)$, we need to simulate algorithm $\mathcal{B}$ in polylog(n) space to compute $\mathcal{B}(X_j, G'_{b_j}) = (T_j, H_j)$.

  The difficulty is in computing the table for the n element group H_j (isomorphic to G'_{b_j}). However, notice that we are interested in computing the final value num$(y)(mod\ p)$, where $y = ((T_1, H_1, \phi_1), \ldots, (T_7, H_7, \phi_7))$. From the definition of num(y) (see remark at the end of Section 2), it is easy to see that in order to compute num$(y)(mod\ p)$, it suffices to generate the string y from left to right, bit by bit. In turn, it implies that it suffices to generate the entries of the Cayley table for H_j one by one, for each $H_j, 1 \leq j \leq 7$. We do not need to store the entire table for the H_j's.

  In order to generate the entries for H_j, we exploit the fact that T_j is a cube-generating sequence for H_j. In particular:

  (a) We can identify elements of H_j as strings in $\{0,1\}^{4 \log n}$, as explained in Section 2, by cycling through the products in the cube generated by T_j. This can be done in polylog(n) space.

  (b) To find the entries of the Cayley table for H_j we need to use the Cayley table for G'_{b_j}. Although we do not explicitly have the table for G'_{b_j}, we have its short presentation (Y_{b_j}, R_{b_j}). Now, we can apply Theorem 3.4 in order to multiply elements of G'_{b_j} using just the presentation (Y_{b_j}, R_{b_j}).

  Then, as described above, in polylog space the entries of the table for H_j can be generated. As we obtain the entries for H_j in a lexicographic order, we are effectively getting the tuple y from left to right. We can convert it into the bits of num(y) and incrementally compute num$(y)(mod p)$ with the following standard method: for the k-bit prefix z of num(y) if we have computed $z(mod p)$ and we have the $k+1^{st}$ bit of num(y), then $2z + 1(mod p)$ is the $k+1$-bit prefix of num(y) modulo p. Continuing thus, we will finally compute num$(y)(mod\ p)$, and the algorithm now simply has to check if $h(\text{num}(y)(mod\ p)) = 0^k$, which can be done in polylog(n) space.

Now, we are ready to describe the derandomization. We shall use the Nisan-Wigderson pseudorandom generator [19], that stretches $O(\log n)$ random bits to $O(\log^6 n)$ random bits using an

EXP complete language as the hard function. We recall the construction and the key properties of the generator.

Let r, l, m, k be positive integers. A collection $D = (D_1, \ldots, D_r)$ of sets $D_i \subseteq \{1, \ldots, l\}$ is called a (r, l, m, k)-*design* if $\|D_i\| = m$ for all i, and for all $i \neq j$, $\|D_i \cap D_j\| \leq k$. Using D we get from a boolean function $g : \{0,1\}^m \to \{0,1\}$ a sequence of boolean functions $g_i : \{0,1\}^l \to \{0,1\}$, $i = 1, \ldots, r$, defined as $g_i(s_1, \ldots, s_l) = g(s_{i_1}, \ldots, s_{i_m})$ where $D_i = \{i_1, \ldots, i_m\}$. By concatenating the values of these functions we get a function $g_D : \{0,1\}^l \to \{0,1\}^r$ where $g_D(s) = g_1(s) \ldots g_r(s)$. Nisan and Wigderson show [19, Lemma 2.4] that the output of g_D looks random to a small deterministic circuit, provided g is hard to approximate by deterministic circuits of a certain size (in other words, the hardness of g implies that the pseudorandom generator g_D is secure against small circuits). The following makes this more precise.

For a set A let $\mathcal{CIR}^A(n, s)$ stand for the set of n-input boolean functions that can be computed by deterministic circuits of size at most s, having besides the normal gates oracle gates evaluating the characteristic function of A.

For an oracle A, a boolean function $g : \{0,1\}^m \to \{0,1\}$ is said to be a $\mathcal{CIR}^A(n, r(n))$-hard function if

$$\frac{1}{2} - \frac{1}{r(n)} < \frac{\|\{x \in \{0,1\}^n \mid f(x) = g(x)\}\|}{2^n} < \frac{1}{2} + \frac{1}{r(n)}$$

holds for all functions $g \in \mathcal{CIR}^A(n, r(n))$.
Let $r : \mathbb{N} \to \mathbb{R}^+$ and let L be any language. L is said to be $\mathcal{CIR}^A(r)$-*hard* if for all but finitely many n, the n-ary boolean function $L^{=n}$ is $\mathcal{CIR}^A(n, r(n))$-hard.

We state a crucial lemma due to Nisan and Wigderson [19], in a form used in [2].

Lemma 5.2 [19] *Let D be a* (r, l, m, k)-design *and let $g : \{0,1\}^m \to \{0,1\}$ be an $\mathcal{CIR}^A(m, r^2 + r2^k)$-hard function (for some oracle A). Then the function g_D has the property that for every r-input circuit c of size at most r^2,*

$$\left| \mathrm{Prob}_{y \in_R \{0,1\}^r}[c^A(y) = 1] - \mathrm{Prob}_{s \in_R \{0,1\}^l}[c^A(g_D(s)) = 1] \right| \leq 1/r.$$

Choose $r = O(\log^6 n)$, $l = \log n$, $m = \sqrt{\log n}$, and $k = \log \log n$. By [19, Lemma 2.5] we know that there is an (r, l, m, k)-design, call it D, for these values, that is computable in space $k = \log \log n$, and hence in time polynomial in $\log n$.

Our goal is to derandomize the AM protocol using the pseudorandom generator g_D that stretches an $O(\log n)$ bit random seed to $O(\log^6 n)$ pseudorandom bits. From our preceding discussion about the AM protocol it is clear that it is Phase III of the protocol that uses the random bits h and p. Furthermore, the computation of Phase III can be simulated by a polylog(n) size circuit *with oracle* L that takes as input the short presentations (Y_1, R_1) and (Y_2, R_2) and the random bits h and p.

Now, let g be the characteristic function of an EXP-complete language in E. Furthermore, let the language L defined above be the oracle A of Lemma 5.2. As already explained, Phase III's computation can be carried out by a polylog(n) size circuit with L as oracle.

We claim that the derandomization is correct on all but finitely many inputs.

Suppose not. In particular, suppose the derandomization of the AM protocol fails for some input pair (G_1, G_2) of solvable groups. Let (Y_1, R_1) and (Y_2, R_2) be their short presentations computed in Phase I. As a consequence of the failure of the derandomization, it follows that the polylog(n) size

circuit with oracle L for Phase III with input fixed as (Y_1, R_1) and (Y_2, R_2) is a distinguisher circuit that distinguishes between the output of g_D and the truly random bits. Applying Yao's method as explained in [19], we can convert the distinguisher into a next bit predictor, and finally obtain a polylog(n) size circuit with oracle L that computes g correctly on a $1/2 + 1/\log^{O(1)} n$ fraction of $O(\log n)$ size inputs. Notice here that we are using the fact that g_D can be computed in time polynomial in $\log n$.

By applying the methods of [6], we can apply Yao's XOR lemma and the fact that EXP has random-self reducible complete sets to conclude that an EXP-complete set can be correctly computed on all $\log n$ size inputs[4] by a polylog(n) size circuit with oracle L. This is true for infinitely many input lengths $\log n$ (since we assumed that the derandomization fails for infinitely many inputs). But that implies EXP $\subseteq$ i.o$-$PSPACE/poly, contradicting the hardness assumption. This completes the proof. ∎

Balcázar proved in [3] that EXP $\subseteq$ PSPACE/poly implies the inclusion of EXP in uniform PSPACE. The same holds for the case of i.o. classes, that is:

Lemma 5.3 *If* EXP $\subseteq$ i.o$-$PSPACE/poly *then* EXP $\subseteq$ i.o$-$PSPACE.

From this lemma and Theorem 5.1 the uniform derandomization result follows.

Theorem 5.4 *If* EXP $\not\subseteq$ i.o$-$PSPACE *then* GROUP-ISO *for solvable groups is in* NP $\cap$ coNP.

6 Limited Nondeterminism

In this section we apply our AM protocol for GROUP-NONISO to show that GROUP-ISO cannot be complete for the limited nondeterminism class NP($\log^2 n$) unless the coNP-complete problem $\overline{\text{CLIQUE}}$ has nonuniform subexponential size proofs. We also study the parameterized complexity of the problem with the size of generating sets of the groups as parameter. We show that the hardness of Group Isomorphism for the parameterized class W[1] would also imply an unexpected upper bound for the complexity of the clique problem.

Complexity subclasses of NP that arise when the number of nondeterministic bits is bounded have been defined in the literature in different contexts (see [11] for a survey). In particular Kintala and Fischer introduced in [14] subclasses on NP with a polylogarithmic number of nondeterministic bits. Let us denote by NP($\log^k n$) the subclass of NP in which for an input of size n only $O(\log^k n)$ nondeterministic steps are allowed. As we have seen in the introduction, GROUP-ISO is contained in NP($\log^2 n$). Papadimitriou and Yannakakis ask in [22] whether this problem is in fact complete for NP($\log^2 n$). In this section we give some evidence suggesting that this is not the case. We show that if GROUP-ISO is complete for NP($\log^2 n$) then the Clique problem is contained in the class coNTIME[$2^{o(n)}$]/*poly*.

We will consider the following version of the general clique problem CLIQUE = $\{G \mid G$ has n vertices and a clique of size $n/2\}$. Notice that CLIQUE is NP-complete.

Theorem 6.1 *If* GROUP-ISO *is many-one complete for NP*($\log^2 n$) *then* CLIQUE *is in the class* coNTIME[$n^{O(1)}, 2^{O(\log n\sqrt{n})}$]/poly. *I.e. for inputs of length n,* $\overline{\text{CLIQUE}}$ *has polynomial-size proofs which can be verified in* $2^{O(\log n\sqrt{n})}$ *time with a polynomial-size advice.*

[4] There will be change in input length which we can ignore here without affecting the results.

Proof: Consider the problem log-CLIQUE $= \{(G,k) \mid G$ has n vertices, $k \leq \log n$ and G has a clique of size $k\}$. log-CLIQUE is clearly in NP($\log^2 n$). If GROUP-ISO is complete for this class, then there is a many-one polynomial time reduction from log-CLIQUE to GROUP-ISO and by the results of the previous section the complement of the problem, log-$\overline{\text{CLIQUE}} \in \text{AM}(\log^6, log^2 n)$.

We will now apply an idea of Feige and Kilian [10]: Let G be a graph with n nodes and for simplicity assume that $n/2$ is a perfect square. We can convert in time $n^{O(\sqrt{n})}$ the instance G of CLIQUE to a pair (G', l') in the following way: G' is a graph with at most $\binom{n}{\sqrt{n/2}}$ nodes. Each node of G' corresponds to a subset of $V(G)$ of size $\sqrt{n/2}$, and there is an edge between two nodes of G' if and only if their corresponding sets of nodes in G are disjoint and form a clique of size $2\sqrt{n/2}$. Clearly G' has a clique of size $\sqrt{n/2}$ if and only if $G \in$ CLIQUE. We set l' to $\sqrt{n/2}$. Let N be the number of vertices in G'. $N = \binom{n}{\sqrt{n/2}}$. l' is smaller than $\log N$ and therefore (G', l') is an instance of the log-CLIQUEproblem. The AM protocol on input (G', l') will use $O(\log^6 N)$ random bits and $O(\log^2 N)$ nondeterministic bits. Also observe that the final deterministic computation done by Arthur after the communication rounds is of polynomial time in N. This implies the following claim.

Claim 6.2 *There is an* AM$(n^{O(1)}, n^{O(1)})$ *protocol for the coNP complete problem* $\overline{\text{CLIQUE}}$*, where the final deterministic computation done by Arthur after the communication rounds is of time* $2^{O(\log n\sqrt{n})}$*. Furthermore, the protocol has error probability bounded by* 1/4*.*

Applying the standard method of repeating the protocol $n^{O(1)}$ times and taking a majority vote, we get an AM$(n^{O(1)}, n^{O(1)})$ protocol for $\overline{\text{CLIQUE}}$ where the final deterministic computation done by Arthur is still of time $2^{O(\log n\sqrt{n})}$ and the error probability is now bounded by $2^{-n^{O(1)}}$. We can derandomize the protocol by fixing for each length n the random bits to an $n^{O(1)}$ size advice string. Thus, we have shown that $\overline{\text{CLIQUE}}$ is in NTIME$[n^{O(1)}, 2^{O(\log n\sqrt{n})}]$/poly. This completes the proof. ∎

6.1 *Parameterized complexity setting*

A different perspective to GROUP-ISO is given by the parameterized complexity setting introduced by Downey and Fellows [8], which is a useful and rich paradigm for classifying problems. Parameterized complexity deals with problems where the instances are pairs (x, k) where the parameter k is usually a positive integer that measures the "size" of a solution to the input x. The parameterized decision problem now is to test if there is a solution of size at most k (and the search problem is to find such a solution). Algorithms with time bounds of the form $f(k)n^{O(1)}$, for arbitrary functions f, are considered efficient. Problems with such algorithms are *fixed parameter tractable*. Analogous to NP-completeness, there is a theory of hardness [8] which classifies parameterized problems. Let Π and Π' be two parameterized problems. A parameterized many-one reductions from Π to Π' maps instance (x, k) of Π to instances (x', k') of Π', such that the running time of the reduction is bounded by $f(k)|x|^{O(1)}$ and $k' \leq f(k)$ for an arbitrary function f. Recall from [8] that W[P] is the class of parameterized problems reducible via parameterized reductions to the weight-k circuit satisfiability problem. The weight-k circuit value problem is: given a boolean circuit with n input gates, is there an input of hamming weight k accepted by the circuit. The $k-$CLIQUE problem is the parameterized clique problem with instances (G, k) and the question is whether G has a clique of size k. The class W[1] can be defined as parameterized problems that are many-one reducible via parameterized reductions to the $k-$CLIQUE problem. The $k-$CLIQUE problem is W[1]-complete and is considered unlikely to be fixed parameter tractable [8]. Notice that W[1] $\subseteq$ W[P].

We consider the following natural parameterized version of group isomorphism k-GROUP-ISO: let G_1 and G_2 be groups with n elements each given by their Cayley tables and generating sets S_1 and S_2 of size k each. The problem is to test if the groups are isomorphic (and, if so, to find an isomorphism).

As noted in the introduction, this parameterization makes sense for GROUP-ISO because there are natural classes of groups with small generating sets. E.g. every simple group of size n has a generating sets with just two elements.

There is an easy $n^{O(k)}$ algorithm for k-GROUP-ISO which puts it in the class W[P]. But, no algorithm with running time of the form $f(k)n^{O(1)}$ is known for $k-$GROUP-ISO. Thus, we do not know if it is fixed parameter tractable. We are interested in the question of whether k-GROUP-ISO is complete for $W[P]$. We show that probably this is not the case since the problem does not even seem to be hard for $W[1]$. We now provide evidence of this fact by showing that this assumption implies a nonuniform nondeterministic subexponential time algorithm for $\overline{\text{CLIQUE}}$.

Theorem 6.3 *If* $k-$GROUP-ISO *is* W[1]*-hard then* $\overline{\text{CLIQUE}}$ *is in* NTIME$[n^{O(1)}, 2^{o(n)}]$/poly. *I.e. for inputs of length* n, $\overline{\text{CLIQUE}}$ *has polynomial-size proofs which can be verified in* $2^{o(n)}$ *time with a polynomial-size advice.*

Proof: Suppose $k-$GROUP-ISO is W[1]-hard. Then there is a parameterized reduction from k-CLIQUE to $k-$GROUP-ISO. By combining the reduction with the AM$(O(\log^6 n), O(\log^2 n))$ protocol for GROUP-NONISO, we get an AM$(O(f(k)\log^6 n), O(f(k)\log^2 n))$ protocol for k-$\overline{\text{CLIQUE}}$ (the complement of k-CLIQUE), in which the final deterministic computation done by Arthur after the communication rounds takes time $f(k)n^{O(1)}$, where f is some monotonically increasing function. We can assume w.l.o.g. that $f(k) \geq k$ for all k. Now, define the function $f^{-1}(n)$ to be the largest positive integer m such that $f(m) \leq n$. Observe that $f^{-1}(n)$ tends to ∞ monotonically with n.

Recall that CLIQUE $= \{G \mid G$ has n vertices and a clique of size $n/2\}$. For an instance G of CLIQUE, set $l' = f^{-1}(n/2)$. Let $\alpha(n)$ denote $\frac{1}{2f^{-1}(n/2)}$. Notice that $\alpha(n)$ tends to 0 monotonically as n increases.

We again apply the Feige-Kilian construction [10]: in time $\binom{n}{O(n\alpha(n))}$ we can convert G to a pair (G', l'), where G' is a graph with at most $\binom{n}{n\alpha(n)}$ nodes, and each node of G' is a subset of $V(G)$ of size $n\alpha(n)$ that induces a clique in G. Furthermore, two nodes u and v in G' are adjacent if their union (as subsets of $V(G)$) forms a clique in G. Clearly, G has an $n/2$-clique if and only if G' has an l'-clique. Now, consider (G', l') as an instance of the parameterized clique problem. Notice that the above AM protocol on input (G', l') will use $O(f(k)\log^6 n) = O(f(l')\log^6 N) = O(n^7 \log^6 n)$ random bits and $O(f(k)\log^2 n) = O(f(l')\log^2 N) = O(n^3 \log^2 n)$ nondeterministic bits. Also observe that the final deterministic computation done by Arthur after the communication rounds is of time $\binom{n}{O(n\alpha(n))}$. Then the running time for the final deterministic computation of Arthur can be bounded by $2^{n \cdot H(c\cdot\alpha(n))}$ where c is a constant and H denotes the entropy function. Observe that $H(c\cdot\alpha(n)) \to 0$ as $n \to \infty$, since $\alpha(n)$ tends to 0 as $n \to \infty$. Thus, $2^{n\cdot H(c\cdot\alpha(n))}$ is $2^{o(n)}$. Putting it together, we have shown the following claim.

Claim 6.4 *Let* G *be an instance of* $\overline{\text{CLIQUE}}$. *There is an* AM$(n^{O(1)}, n^{O(1)})$ *protocol where the final deterministic computation done by Arthur after the communication rounds is of time* $2^{o(n)}$. *Furthermore, the protocol has error probability bounded by* 1/4.

Now, as done in the proof of Theorem 6.1, by applying the standard method of repeating the protocol $n^{O(1)}$ times and taking a majority vote, we get an AM$(n^{O(1)}, n^{O(1)})$ protocol for

$\overline{\text{CLIQUE}}$ where the final deterministic computation done by Arthur is still of time $2^{o(n)}$ and the error probability is now bounded by $2^{-n^{O(1)}}$. We can derandomize the protocol by fixing the random bits to an $n^{O(1)}$ size advice string. Thus, we have shown that $\overline{\text{CLIQUE}}$ is in NTIME$[n^{O(1)}, 2^{o(n)}]$/poly. This completes the proof. ∎

7 Concluding Remarks

As the main result of this paper, we have studied the group isomorphism problem for the case of solvable groups. We briefly here outline how our derandomization results could be generalized to arbitrary groups. First, we observe that it suffices to have analogues of Theorems 3.2, 3.3, and 3.4 for general finite groups. With these theorems, we can proceed as in Sections 4 and 5 to obtain the same derandomization results for general groups. Notice that the analogue of Theorem 3.2 for general finite groups is an effective version of the *short presentation conjecture*: namely, that a short presentation for every finite group G can be computed in time polynomial in $|G|$. In fact, a stronger effective version is conjectured in [7, Conjecture 3]. We recall that the short presentation conjecture (see [7, Conjecture 1] for details) states that every finite group has a short presentation. Regarding analogues for Theorems 3.3 and 3.4, it is possible to show that if analogues of these theorems are true for all finite simple groups, then they hold for all finite groups.

References

[1] M. Agrawal, N. Kayal and N. Saxena, PRIMES is in P. Preprint, August 4, 2002, `http://www.cse.iitk.ac.in/users/manindra/`.

[2] V. Arvind and J. Köbler, On resource bounded measure and pseudorandomness, *Proc. 17th FSTT Conference* Lecture Notes in Computer Science 1346 Springer Verlag, 235–249, (1997).

[3] J. L. Balcázar, Self-Reducibility, *Journal of Computer and System Sciences* 41-3, 367–388, (1990).

[4] J. L. Balcázar, J. Díaz, J. Gabarró, *Structural Complexity I*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1989.

[5] L. Babai, Trading group theory for randomness, *Proc. 17th ACM Symposium on Theory of Computing*, 421–429, 1985.

[6] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson, BPP has subexponential simulations unless EXPTIME has publishable proofs, *Computational Complexity,* 3, pp. 307–318, 1993.

[7] L. Babai, A.J. Goodman, W. Kantor, E. Luks, P.P. Pálfy, Short presentation for finite groups, in *Journal of Algebra*, 194, 79–112, 1997.

[8] R.G. Downey and M.R. Fellows *Parameterized Complexity,* Springer Verlag 1992.

[9] P. Erdös, A. Rényi, Probabilistic Methods in group theory, *Jour. Analyse Mathématique*, vol. 14, (1965), 127–138.

[10] U. Feige, J. Kilian, On Limited versus Polynomial Nondeterminism, *Chicago Journal of Theoretical Computer Science*, March (1997).

[11] J. Goldsmith, M. Levy and M. Mundhenk, Limited nondeterminism in *SIGACT news*, june 1996.

[12] O. Goldreich, A. Wigderson, Derandomizing that is rarely wrong from short advice that is typically good, in *Proc. 6th RANDOM workshop* Lecture Notes in Computer Science 2483, 2002, 209–223.

[13] Garzon and Zalcstein, On Isomorphism Testing of a Class of 2-Nilpotent Groups, in *Journal of Computer and System Sciences,* vol. 42(2), 237–248, 1991.

[14] C. Kintala and P. Fischer, Refining nondeterminism in relativized polynomial time computations, *SIAM J. on Computing*, 9, 1980, 46–53.

[15] A. Klivans and D. van Melkebeek, Graph Isomorphism has subexponential size provers unless the polynomial time hierarchy collapses. *In Proc. 31st ACM STOC*, 1999, 659–667.

[16] R.J. Lipton, L. Snyder, Y. Zalcstein The complexity of word and isomorphism problems for finite groups. Johns Hopkins University 1976.

[17] C.-J. Lu, Derandomizing Arthur-Merlin games under uniform assumptions, in *Journal of Computational Complexity*, 10 2001, 247–259.

[18] G.L. Miller, On the $n^{\log n}$ isomorphism technique, in *Proc. 10th ACM Symposium on the Theory of Computing*, 1978, 51–58.

[19] N. Nisan and A. Wigderson, Hardness vs randomness, in *Journal of Computer and System Sciences*, 49:149–167, 1994.

[20] P.B. Miltersen, N. Vinodchandran, Derandomizing Arthur-Merlin games using hitting sets, in *Proc. 40th IEEE Symposium on Foundations of Computer Science*, 1999, 71–80.

[21] C. Papadimitriou, *Computational Complexity*, Addison Wesley, 1994.

[22] C. Papadimitriou, M. Yannakakis On limited nondeterminism and the complexity of the VC dimension. In *Journal of Computer and System Sciences*, 53(2): 161-170, 1996.

[23] M. Sipser,A complexity theoretic approach to randomness. In *Proc. 15th ACM Symp. Theory of Computer Science* 1983, 330–335.