Minimum Circuit Size, Graph Isomorphism, and Related Problems[*]

Eric Allender[†1], Joshua A. Grochow[‡2], Dieter van Melkebeek[§3], Cristopher Moore[4], and Andrew Morgan[¶5]

1 **Rutgers University, Piscataway, NJ, USA**
`allender@cs.rutgers.edu`

2 **University of Colorado at Boulder, Boulder, CO, USA**
`joshua.grochow@colorado.edu`

3 **University of Wisconsin–Madison, Madison, WI, USA**
`dieter@cs.wisc.edu`

4 **Santa Fe Institute, Santa Fe, NM, USA**
`moore@santafe.edu`

5 **University of Wisconsin–Madison, Madison, WI, USA**
`amorgan@cs.wisc.edu`

Abstract

We study the computational power of deciding whether a given truth-table can be described by a circuit of a given size (the Minimum Circuit Size Problem, or MCSP for short), and of the variant denoted MKTP where circuit size is replaced by a polynomially-related Kolmogorov measure. All prior reductions from supposedly-intractable problems to MCSP / MKTP hinged on the power of MCSP / MKTP to distinguish random distributions from distributions produced by hardness-based pseudorandom generator constructions. We develop a fundamentally different approach inspired by the well-known interactive proof system for the complement of Graph Isomorphism (GI). It yields a randomized reduction with zero-sided error from GI to MKTP. We generalize the result and show that GI can be replaced by any isomorphism problem for which the underlying group satisfies some elementary properties. Instantiations include Linear Code Equivalence, Permutation Group Conjugacy, and Matrix Subspace Conjugacy. Along the way we develop encodings of isomorphism classes that are efficiently decodable and achieve compression that is at or near the information-theoretic optimum; those encodings may be of independent interest.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.2 Modes of Computation, F.1.3 Complexity Measures and Classes

Keywords and phrases Reductions between NP-intermediate problems, Graph Isomorphism, Minimum Circuit Size Problem, time-bounded Kolmogorov complexity

Digital Object Identifier 10.4230/LIPIcs.ITCS.2018.20

[*] A full version of this paper is available as [3], `https://arxiv.org/abs/1710.09806`.
[†] EA acknowledges the support of National Science Foundation grant CCF-1555409.
[‡] JAG was supported by an Omidyar Fellowship from the Santa Fe Institute and National Science Foundation grant DMS-1620484.
[§] DvM acknowledges the support of National Science Foundation grant CCF-1319822.
[¶] AM acknowledges the support of National Science Foundation grant CCF-1319822.

9th Innovations in Theoretical Computer Science Conference (ITCS 2018).
Editor: Anna R. Karlin; Article No. 20; pp. 20:1–20:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Finding a circuit of minimum size that computes a given Boolean function constitutes the overarching goal in nonuniform complexity theory. It defines an interesting computational problem in its own right, the complexity of which depends on the way the Boolean function is specified. A generic and natural, albeit verbose, way to specify a Boolean function is via its truth-table. The corresponding decision problem is known as the Minimum Circuit Size Problem (MCSP): Given a truth-table and a threshold θ, does there exist a Boolean circuit of size at most θ that computes the Boolean function specified by the truth-table? The interest in MCSP dates back to the dawn of theoretical computer science [30]. It continues today partly due to the fundamental nature of the problem, and partly because of the work on natural proofs and the connections between pseudorandomness and computational hardness.

A closely related problem from Kolmogorov complexity theory is the Minimum KT Problem (MKTP), which deals with compression in the form of efficient programs instead of circuits. Rather than asking if the input has a small circuit when interpreted as the truth-table of a Boolean function, MKTP asks if the input has a small program that produces each individual bit of the input quickly. To be more specific, let us fix a universal Turing machine U. We consider descriptions of the input string x in the form of a program d such that, for every bit position i, U on input d and i outputs the i-th bit of x in T steps. The KT cost of such a description is defined as $|d| + T$, i.e., the bit-length of the program plus the running time. The KT complexity of x, denoted $\mathrm{KT}(x)$, is the minimum KT cost of a description of x. $\mathrm{KT}(x)$ is polynomially related to the circuit complexity of x when viewed as a truth-table (see Section 4.1 for a more formal treatment). On input a string x and an integer θ, MKTP asks whether $\mathrm{KT}(x) \leq \theta$.

Both MCSP and MKTP are in NP but are not known to be in P or NP-complete. As such, they are two prominent candidates for NP-intermediate status. Others include factoring integers, discrete log over prime fields, graph isomorphism (GI), and a number of similar isomorphism problems.

Whereas NP-complete problems all reduce one to another, even under fairly simple reductions, less is known about the relative difficulty of presumed NP-intermediate problems. Regarding MCSP and MKTP, factoring integers and discrete log over prime fields are known to reduce to both under randomized reductions with zero-sided error [1, 27]. Recently, Allender and Das [2] showed that GI and all of SZK (Statistical Zero Knowledge) reduce to both under randomized reductions with bounded error.

Those reductions and, in fact, *all* prior reductions of supposedly-intractable problems to MCSP / MKTP proceed along the same well-trodden path. Namely, MCSP / MKTP is used as an efficient statistical test to distinguish random distributions from pseudorandom distributions, where the pseudorandom distribution arises from a hardness-based pseudorandom generator construction. In particular, [20] employs the construction based on the hardness of factoring Blum integers, [1, 2, 5, 27] use the construction from [16] based on the existence of one-way functions, and [1, 9] make use of the Nisan-Wigderson construction [24]. The property that MCSP / MKTP breaks the construction implies that the underlying hardness assumption fails relative to MCSP / MKTP, and thus that the supposedly hard problem reduces to MCSP / MKTP.

1.1 Contributions

The main conceptual contribution of our paper is a fundamentally different way of constructing reductions to MKTP based on a novel use of known interactive proof systems. Our approach applies to GI and a broad class of isomorphism problems. A common framework for those isomorphism problems is another conceptual contribution. In terms of results, our new approach allows us to eliminate the errors in the recent reductions from GI to MKTP, and more generally to establish *zero-sided error* randomized reductions to MKTP from many isomorphism problems within our framework. These include Linear Code Equivalence, Matrix Subspace Conjugacy, and Permutation Group Conjugacy (see Section 3.1 for the definitions). The technical contributions mainly consist of encodings of isomorphism classes that are efficiently decodable and achieve compression that is at or near the information-theoretic optimum.

We note that our techniques remain of interest even in light of the recent quasi-polynomial-time algorithm for GI [6]. For one, GI is still not known to be in P, and Group Isomorphism stands as a significant obstacle to this (as stated at the end of [6]). More importantly, our techniques also apply to the other isomorphism problems mentioned above, for which the current best algorithms are still exponential.

Let us also provide some evidence that our approach for constructing reductions to MKTP differs in an important way from the existing ones. We claim that the existing approach can only yield zero-sided error reductions to MKTP from problems that are in $\mathsf{NP} \cap \mathsf{coNP}$, a class that neither GI nor any of the other isomorphism problems mentioned above are known to reside in. The reason for the claim is that the underlying hardness assumptions are fundamentally average-case, which implies that the reduction can have both false positives and false negatives. For example, in the papers employing the construction from [16], MKTP is used in a subroutine to invert a polynomial-time-computable function, and the subroutine may fail to find an inverse. Given a reliable but imperfect subroutine, the traditional way to eliminate false positives is to use the subroutine for constructing an efficiently verifiable membership witness, and only accept after verifying its validity. As such, the existence of a traditional reduction without false positives from a language L to MKTP implies that $L \in \mathsf{NP}$. Similarly, a traditional reduction from L to MKTP without false negatives is only possible if $L \in \mathsf{coNP}$, and zero-sided error is only possible if $L \in \mathsf{NP} \cap \mathsf{coNP}$.

Instead of using the oracle for MKTP in the *construction* of a candidate witness and then verifying the validity of the candidate without the oracle, we use the power of the oracle in the *verification* process. This obviates the need for the language L to be in $\mathsf{NP} \cap \mathsf{coNP}$ in the case of reductions with zero-sided error.

1.2 Organization

In Section 2 we develop our technique for $L = \mathrm{GI}$ in an informal way, and in Section 3 we extend it to a broad class of isomorphism problems. In Section 4 we rigorously develop our result for GI. A formal treatment for other isomorphism problems is deferred to the full version [3]. Section 5 presents the information-theoretically optimal encodings that we use for our result for GI. In Section 6 we suggest directions for further research.

2 Main Idea for Graph Isomorphism

Recall that an instance of GI consists of a pair (G_0, G_1) of graphs on the vertex set $[n]$, and the question is whether $G_0 \equiv G_1$, i.e., whether there exists a permutation $\pi \in S_n$ such that $G_1 = \pi(G_0)$, where $\pi(G_0)$ denotes the result of applying the permutation π to the vertices of G_0. In order to develop a zero-sided error algorithm for GI, it suffices to develop one without false negatives. This is because the false positives can subsequently be eliminated using the known search-to-decision reduction for GI [22].

The crux for obtaining a reduction without false negatives from GI to MKTP is a witness system for the complement $\overline{\mathrm{GI}}$ inspired by the well-known two-round interactive proof system for $\overline{\mathrm{GI}}$ [11]. Consider the distribution $R_G(\pi) \doteq \pi(G)$ where $\pi \in S_n$ is chosen uniformly at random. By the Orbit–Stabilizer Theorem, for any fixed G, R_G is uniform over a set of size $N \doteq n!/|\operatorname{Aut}(G)|$ and thus has entropy $s = \log(N)$, where $\operatorname{Aut}(G) \doteq \{\pi \in S_n : \pi(G) = G\}$ denotes the set of automorphisms of G. For ease of exposition, let us assume that $|\operatorname{Aut}(G_0)| = |\operatorname{Aut}(G_1)|$ (which is actually the hardest case for GI), so both R_{G_0} and R_{G_1} have the same entropy s. Consider picking $r \in \{0,1\}$ uniformly at random, and setting $G = G_r$. If $(G_0, G_1) \in \mathrm{GI}$, the distributions R_{G_0}, R_{G_1}, and R_G are all identical, and therefore R_G also has entropy s. On the other hand, if $(G_0, G_1) \notin \mathrm{GI}$, the entropy of R_G equals $s+1$. The extra bit of information corresponds to the fact that in the nonisomorphic case each sample of R_G reveals the value of r that was used, whereas that bit gets lost in the reduction in the isomorphic case.

The difference in entropy suggests that a typical sample of R_G can be compressed more in the isomorphic case than in the nonisomorphic case. If we can compute some threshold such that $\mathrm{KT}(R_G)$ *never* exceeds the threshold in the isomorphic case, and exceeds it with nonnegligible probability in the nonisomorphic case, we have the witness system for $\overline{\mathrm{GI}}$ that we aimed for: Take a sample from R_G, and use the oracle for MKTP to check that it cannot be compressed at or below the threshold. The entropy difference of 1 may be too small to discern, but we can amplify the difference by taking multiple samples and concatenating them. Thus, we end up with a randomized mapping reduction of the following form, where t denotes the number of samples and θ the threshold:

$$\begin{array}{l}\text{Pick } r \doteq r_1 \dots r_t \in \{0,1\}^t \text{ and } \pi_i \in S_n \text{ for } i \in [t] \text{, each uniformly at random.} \\ \text{Output } (y, \theta) \text{ where } y \doteq y_1 \dots y_t \text{ and } y_i \doteq \pi_i(G_{r_i}).\end{array} \tag{1}$$

2.1 Rigid Case

We need to analyze how to set the threshold θ and argue correctness for a value of t that is polynomially bounded. In order to do so, let us first consider the case where the graphs G_0 and G_1 are *rigid*, i.e., they have no nontrivial automorphisms, or equivalently, $s = \log(n!)$.

- If $G_0 \not\equiv G_1$, the string y contains all of the information about the random string r and the t random permutations $\pi_1, \dots, \pi_t$, which amounts to $ts + t = t(s+1)$ bits of information. This implies that y has KT-complexity close to $t(s+1)$ with high probability.
- If $G_0 \equiv G_1$, then we can efficiently produce each bit of y from the adjacency matrix representation of G_0 (n^2 bits) and the function table of permutations $\tau_i \in S_n$ (for $i \in [t]$) such that $y_i \doteq \pi_i(G_{r_i}) = \tau_i(G_0)$. Moreover, the set of all permutations S_n allows an efficiently decodable indexing, i.e., there exists an efficient algorithm that takes an index $k \in [n!]$ and outputs the function table of the k-th permutation in S_n according to some ordering. An example of such an indexing is the Lehmer code (see, e.g., [21, pp. 12-13] for specifics). This shows that

  $$\mathrm{KT}(y) \leq t\lceil s \rceil + (n + \log(t))^c \tag{2}$$

  for some constant c, where the first term represents the cost of the t indices of $\lceil s \rceil$ bits each, and the second term represents the cost of the n^2 bits for the adjacency matrix of G_0 and the polynomial running time of the decoding process.

If we ignore the difference between s and $\lceil s \rceil$, the right-hand side of (2) becomes $ts + n^c$, which is closer to ts than to $t(s+1)$ for t any sufficiently large polynomial in n, say $t = n^{c+1}$. Thus, setting θ halfway between ts and $t(s+1)$, i.e., $\theta \doteq t(s + \frac{1}{2})$, ensures that $\mathrm{KT}(y) > \theta$

holds with high probability if $G_0 \not\equiv G_1$, and never holds if $G_0 \equiv G_1$. This yields the desired randomized mapping reduction without false negatives, modulo the rounding issue of s to $\lceil s \rceil$. The latter can be handled by aggregating the permutations τ_i into blocks so as to make the amortized cost of rounding negligible. The details are captured in the Blocking Lemma of Section 4.2.

2.2 General Case

What changes in the case of non-rigid graphs? For ease of exposition, let us again assume that $|\operatorname{Aut}(G_0)| = |\operatorname{Aut}(G_1)|$. There are two complications:

(i) We no longer know how to efficiently compute the threshold $\theta \doteq t(s + \frac{1}{2})$ because $s \doteq \log(N)$ and $N \doteq \log(n!/|\operatorname{Aut}(G_0)|) = \log(n!/|\operatorname{Aut}(G_1)|)$ involves the size of the automorphism group.

(ii) The Lehmer code no longer provides sufficient compression in the isomorphic case as it requires $\log(n!)$ bits per permutation whereas we only have s to spend, which could be considerably less than $\log(n!)$.

In order to resolve (ii) we develop an efficiently decodable indexing of cosets for any subgroup of S_n given by a list of generators (see Lemma 10 in Section 4.3). In fact, our scheme even works for cosets of a subgroup within another subgroup of S_n, a generalization that may be of independent interest (see Lemma 13 in Section 5). Applying our scheme to $\operatorname{Aut}(G)$ and including a minimal list of generators for $\operatorname{Aut}(G)$ in the description of the program p allows us to maintain (2).

Regarding (i), we can deduce a good approximation to the threshold with high probability by taking, for both choices of $r \in \{0, 1\}$, a polynomial number of samples of R_{G_r} and using the oracle for MKTP to compute the exact KT-complexity of their concatenation. This leads to a randomized reduction from GI to MKTP with bounded error (from which one without false positives follows as mentioned before), reproving the earlier result of [2] using our new approach (see Remark 11 in Section 4.3 for more details).

In order to avoid false negatives, we need to improve the above approximation algorithm such that it never produces a value that is too small, while maintaining efficiency and the property that it outputs a good approximation with high probability. In order to do so, it suffices to develop a *probably-correct overestimator* for the quantity $n!/|\operatorname{Aut}(G)|$, i.e., a randomized algorithm that takes as input an n-vertex graph G, produces the correct quantity with high probability, and never produces a value that is too small; the algorithm should run in polynomial time with access to an oracle for MKTP. Equivalently, it suffices to develop a probably-correct *under*estimator of similar complexity for $|\operatorname{Aut}(G)|$.

The latter can be obtained from the known search-to-decision procedures for GI, and answering the oracle calls to GI using the above two-sided error reduction from GI to MKTP. There are a number of ways to implement this strategy; here is one that generalizes to a number of other isomorphism problems including Linear Code Equivalence.

1. Find a list of permutations that generates a subgroup of $\operatorname{Aut}(G)$ such that the subgroup equals $\operatorname{Aut}(G)$ with high probability.
Finding a list of generators for $\operatorname{Aut}(G)$ deterministically reduces to GI. Substituting the oracle for GI by a two-sided error algorithm yields a list of permutations that generates $\operatorname{Aut}(G)$ with high probability, and always produces a subgroup of $\operatorname{Aut}(G)$. The latter property follows from the inner workings of the reduction, or can be imposed explicitly by checking every permutation produced and dropping it if it does not map G to itself. We use the above randomized reduction from GI to MKTP as the two-sided error algorithm for GI.

2. Compute the order of the subgroup generated by those permutations.
   There is a known deterministic polynomial-time algorithm to do this [29].

The resulting probably-correct underestimator for $|\operatorname{Aut}(G)|$ runs in polynomial time with access to an oracle for MKTP. Plugging it into our approach, we obtain a randomized reduction from GI to MKTP without false negatives. A reduction with zero-sided error follows as discussed earlier. Thus, we have established the following result.

▶ **Theorem 1.** $\mathrm{GI} \in \mathsf{ZPP}^{\mathrm{MKTP}}$.

Before applying our approach to other isomorphism problems, let us point out the important role that the Orbit–Stabilizer Theorem plays. A randomized algorithm for finding generators for a graph's automorphism group yields a probably-correct underestimator for the size of the automorphism group, as well as a randomized algorithm for GI without false positives. The Orbit–Stabilizer Theorem allows us to turn a probably-correct underestimator for $|\operatorname{Aut}(G)|$ into a probably-correct overestimator for the size of the support of R_G, thereby switching the error from one side to the other, and allowing us to avoid false negatives instead of false positives.

3 Generalization

Our approach extends to several other isomorphism problems. We first present a definition of a generic isomorphism problem, and then informally develop the generalization of Theorem 1. We refer to the full version [3] for the formal proofs.

3.1 Framework

We consider the following framework, parameterized by an underlying family of group actions (Ω, H) where H is a group that acts on the universe Ω. We typically think of the elements of Ω as abstract objects, which need to be described in string format in order to be input to a computer; we let $\omega(z)$ denote the abstract object represented by the string z.

▶ **Definition 2** (Isomorphism Problem)**.** An instance of an Isomorphism Problem consists of a pair $x = (x_0, x_1)$ that determines a universe Ω_x and a group H_x that acts on Ω_x such that $\omega_0(x) \doteq \omega(x_0)$ and $\omega_1(x) \doteq \omega(x_1)$ belong to Ω_x. Each $h \in H_x$ is identified with the permutation $h : \Omega_x \to \Omega_x$ induced by the action. The goal is to determine whether there exists $h \in H_x$ such that $h(\omega_0(x)) = \omega_1(x)$.

When it causes no confusion, we drop the argument x and simply write H, Ω, ω_0, and ω_1. We often blur the—sometimes pedantic—distinction between z and $\omega(z)$. For example, in GI, each z is an $n \times n$ binary matrix (a string of length n^2), and represents the abstract object $\omega(z)$ of a graph with n labeled vertices; thus, in this case the correspondence between z and $\omega(z)$ is a bijection. The group H is the symmetric group S_n, and the action is by permuting the labels.

For completeness, we include below the definitions of the instantiations we mentioned in Section 1.1. Table 1 summarizes how they can be cast in the framework.

Linear code equivalence. A *linear code* over the finite field $\mathbb{F}_q$ is a d-dimensional linear subspace of $\mathbb{F}_q^n$ for some n. Two such codes are (permutationally) *equivalent* if there is a permutation of the n coordinates that makes them equal as subspaces.

Linear Code Equivalence is the problem of deciding whether two linear codes are equivalent, where the codes are specified as the row-span of a $d \times n$ matrix (of rank d), called a *generator*

Table 1 Instantiations of the Isomorphism Problem

Problem	H	Ω
Graph Isomorphism	S_n	graphs with n labeled vertices
Linear Code Equivalence	S_n	subspaces of dimension d in $\mathbb{F}_q^n$
Permutation Group Conjugacy	S_n	subgroups of S_n
Matrix Subspace Conjugacy	$\mathrm{GL}_n(\mathbb{F}_q)$	subspaces of dimension d in $\mathbb{F}_q^{n\times n}$

matrix. There exists a mapping reduction from GI to Linear Code Equivalence over any field [26, 15]; Linear Code Equivalence is generally thought to be harder than GI.

In order to cast Code Equivalence in our framework, we consider the family of actions $(S_n, \Omega_{n,d,q})$ where $\Omega_{n,d,q}$ denotes the linear codes of length n and dimension d over $\mathbb{F}_q$, and S_n acts by permuting the coordinates.

Permutation Group Conjugacy. Two permutation groups $\Gamma_0, \Gamma_1 \leq S_n$ are *conjugate* (or permutationally isomorphic) if there exists a permutation $\pi \in S_n$ such that $\Gamma_1 = \pi\Gamma_0\pi^{-1}$; such a π is called a conjugacy.

The *Permutation Group Conjugacy* problem is to decide whether two subgroups of S_n are conjugate, where the subgroups are specified by a list of generators. The problem is known to be in $\mathsf{NP} \cap \mathsf{coAM}$, and is at least as hard as Linear Code Equivalence. Currently the best known algorithm runs in time $2^{O(n)} \operatorname{poly}(|\Gamma_1|)$ [7]—that is, the runtime depends not only on the input size (which is polynomially related to n), but also on the size of the groups generated by the input permutations, which can be exponentially larger.

Matrix Subspace Conjugacy. A *linear matrix space* over $\mathbb{F}_q$ is a d-dimensional linear subspace of $n \times n$ matrices. Two such spaces V_0 and V_1 are *conjugate* if there is an invertible $n \times n$ matrix X such that $V_1 = XV_0X^{-1} \doteq \{X \cdot M \cdot X^{-1} : M \in V_0\}$, where "$\cdot$" represents matrix multiplication.

Matrix Subspace Conjugacy is the problem of deciding whether two linear matrix spaces are conjugate, where the spaces are specified as the linear span of d linearly independent $n \times n$ matrices. There exist mapping reductions from GI and Linear Code Equivalence to Matrix Subspace Conjugacy [15]; Matrix Subspace Conjugacy is generally thought to be harder than Linear Code Equivalence.

3.2 Generic Result

We generalize our construction for GI to any Isomorphism Problem by replacing $R_G(\pi) \doteq \pi(G)$ where $\pi \in S_n$ is chosen uniformly at random, by $R_\omega(h) \doteq h(\omega)$ where $h \in H$ is chosen uniformly at random. The analysis that the construction yields a randomized reduction without false negatives from the Isomorphism Problem to MKTP carries over, provided that the Isomorphism Problem satisfies the following properties.

1. The underlying group H is *efficiently samplable*, and the action $(\omega, h) \mapsto h(\omega)$ is efficiently computable. We need this property in order to make sure the reduction is efficient.
2. There is an efficiently computable *normal form* for representing elements of Ω as strings. This property trivially holds in the setting of GI as there is a unique adjacency matrix that represents any given graph on the vertex set $[n]$. However, uniqueness of representation need not hold in general. Consider, for example, Permutation Group Conjugacy. An instance of this problem abstractly consists of two permutation groups (Γ_0, Γ_1), represented

(as usual) by a sequence of elements of S_n generating each group. In that case there are many strings representing the same abstract object, i.e., a subgroup has many different sets of generators.

For the correctness analysis in the isomorphic case it is important that H acts on the abstract objects, and *not* on the binary strings that represent them. In particular, the output of the reduction should only depend on the abstract object $h(\omega)$, and not on the way ω was provided as input. This is because the latter may leak information about the value of the bit r that was picked. The desired independence can be guaranteed by applying a normal form to the representation before outputting the result. In the case of Permutation Group Conjugacy, this means transforming a set of permutations that generate a subgroup Γ into a canonical set of generators for Γ.

In fact, it suffices to have an efficiently computable *complete invariant* for Ω, i.e., a mapping from representations of objects from Ω to strings such that the image only depends on the abstract object, and is different for different abstract objects.

3. There exists a probably-correct overestimator for $N \doteq |H|/|\operatorname{Aut}(\omega)|$ that is computable efficiently with access to an oracle for MKTP. We need this property to set the threshold $\theta \doteq t(s+\frac{1}{2})$ with $s \doteq \log(N)$ correctly.
4. There exists an encoding for cosets of $\operatorname{Aut}(\omega)$ in H that achieves KT-complexity close to the information-theoretic optimum. This property ensures that in the isomorphic case the KT-complexity is never much larger than the entropy.

Properties 1 and 2 are fairly basic. Property 4 may seem to require an instantiation-dependent approach. However, we develop a *generic* hashing-based encoding scheme that meets the requirements. In fact, we give a nearly-optimal encoding scheme for any samplable distribution that is almost flat, without reference to isomorphism. Unlike the indexings from Lemma 10 for the special case where H is the symmetric group, the generic construction does not achieve the information-theoretic optimum, but it comes sufficiently close for our purposes.

The notion of a probably-correct overestimator in Property 3 can be further relaxed to that of a *probably-approximately-correct overestimator*, or *pac overestimator* for short. This is a randomized algorithm that with high probability outputs a value within an absolute deviation bound of Δ from the correct value, and never produces a value that is more than Δ below the correct value. More precisely, it suffices to efficiently compute with access to an oracle for MKTP a pac overestimator for $s \doteq \log(|H|/|\operatorname{Aut}(\omega)|)$ with deviation $\Delta = 1/4$. The relaxation suffices because of the difference of about 1/2 between the threshold θ and the actual KT-values in both the isomorphic and the non-isomorphic case.

Moreover, Properties 1 and 2 are sufficient to generalize the construction of Allender and Das [2], which yields randomized reductions of the isomorphism problem to MKTP without false positives (irrespective of whether a search-to-decision reduction is known). This leads to the following generalization of Theorem 1.

▶ **Theorem 3.** *Let* Iso *denote an Isomorphism Problem as in Definition 2. Consider the following conditions:*

1. [action sampler] *The uniform distribution on H_x is uniformly samplable in polynomial time, and the mapping $(\omega, h) \mapsto h(\omega)$ underlying the action (Ω_x, H_x) is computable in* ZPP*.*
2. [complete universe invariant] *There exists a complete invariant ν for the representation ω that is computable in* ZPP*.*
3. [entropy estimator] *There exists a probably-approximately-correct overestimator for $(x, \omega) \mapsto \log(|H_x|/|\operatorname{Aut}(\omega)|)$ with deviation $\Delta = 1/4$ that is computable in randomized time* $\operatorname{poly}(|x|)$ *with access to an oracle for* MKTP*.*

With these definitions:

(a) *If conditions 1 and 2 hold, then* Iso $\in \mathsf{RP}^{\mathrm{MKTP}}$.

(b) *If conditions 1, 2, and 3 hold, then* Iso $\in \mathsf{coRP}^{\mathrm{MKTP}}$.

As $s = \log|H| - \log|\operatorname{Aut}(\omega)|$ in Property 3, it suffices to have a pac overestimator for $\log|H|$ and a pac *under*estimator for $\log|\operatorname{Aut}(\omega)|$, both to within deviation $\Delta/2 = 1/8$ and of the required efficiency. Generalizing our approach for GI, one way to obtain the desired underestimator for $\log|\operatorname{Aut}(\omega)|$ is by showing how to efficiently compute with access to an oracle for MKTP:

(a) a list L of elements of H that generates a subgroup $\langle L\rangle$ of $\operatorname{Aut}(\omega)$ such that $\langle L\rangle = \operatorname{Aut}(\omega)$ with high probability, and

(b) a pac underestimator for $\log|\langle L\rangle|$, the logarithm of the order of the subgroup generated by a given list L of elements of H.

Further mimicking our approach for GI, we know how to achieve (a) when the Isomorphism Problem allows a search-to-decision reduction. Such a reduction is known for Linear Code Equivalence, but remains open for problems like Matrix Subspace Conjugacy and Permutation Group Conjugacy. However, we show that (a) holds for a *generic* isomorphism problem provided that products and inverses in H can be computed efficiently. The proof hinges on the ability of MKTP to break the pseudo-random generator construction of [16] based on a purported one-way function (see Theorem 45 from [1]).

As for (b), we know how to efficiently compute the order of the subgroup *exactly* in the case of permutation groups ($H = S_n$), even without an oracle for MKTP, and in the case of many matrix groups over finite fields ($H = \mathrm{GL}_n(\mathbb{F}_q)$) with oracle access to MKTP, but some cases remain open. Instead, we show how to *generically* construct a *pac underestimator* with small deviation given access to MKTP as long as products and inverses in H can be computed efficiently, and H allows an efficient complete invariant. The first two conditions are sufficient to efficiently generate a distribution $\widetilde{p}$ on $\langle L\rangle$ that is uniform to within a small relative deviation [8]. The entropy $\widetilde{s}$ of that distribution equals $\log|\langle L\rangle|$ to within a small additive deviation. As $\widetilde{p}$ is (essentially) flat, our generic encoding scheme shows that $\widetilde{p}$ has an encoding whose length does not exceed $\widetilde{s}$ by much, and that can be decoded by small circuits. Given an efficient complete invariant for H, we can use an approach similar to the one we used to approximate the threshold θ to construct a pac underestimator for $\widetilde{s}$ with small additive deviation, namely the amortized KT-complexity of the concatenation of a polynomial number of samples from $\widetilde{p}$. With access to an oracle for MKTP we can efficiently evaluate KT. As a result, we obtain a pac underestimator for $\log|\langle L\rangle|$ with a small additive deviation that is efficiently computable with oracle access to MKTP.

This gives the following specialization of Theorem 3:

▶ **Theorem 4.** *Let* Iso *denote an Isomorphism Problem as in Definition 2. Suppose that the ensemble* $\{H_x\}$ *has a representation* η *such that conditions 1 and 2 of Theorem 3 hold as well as the following additional conditions:*

4. [group operations] *Products and inverses in* H_x *are computable in* ZPP.

5. [sample space estimator] *The map* $x \mapsto |H_x|$ *has a pac overestimator with deviation* $\Delta = 1/8$ *computable in* $\mathsf{ZPP}^{\mathrm{MKTP}}$.

6. [complete group invariant] *There exists a complete invariant* ζ *for the representation* η *that is computable in* ZPP.

Then Iso $\in \mathsf{ZPP}^{\mathrm{MKTP}}$.

Theorem 4 allows us to show that all of the isomorphism problems in Table 1 reduce to MKTP under randomized reductions with zero-sided error. We refer to the full version [3] for a complete treatment.

▶ **Corollary 5.** *Linear Code Equivalence, Permutation Group Conjugacy, and Matrix Subspace Conjugacy are in* $\mathsf{ZPP}^{\mathrm{MKTP}}$.

4 Technical Development for Graph Isomorphism

This section is dedicated to a rigorous proof of Theorem 1. We start with the formal definition of MKTP, and then follow the outline of Section 2, taking the same four steps, and filling in the missing details.

4.1 KT Complexity

Theorem 1 states a reduction from GI to the Minimum KT Problem. The measure KT that we informally described in Section 1, was introduced and formally defined as follows in [1]. We refer to that paper for more background and motivation for the particular definition.

▶ **Definition 6** (KT)**.** Let U be a universal Turing machine. For each string x, define $\mathrm{KT}_U(x)$ to be

$$\min\{\, |d| + T : \quad (\forall \sigma \in \{0,1,*\})\ (\forall i \le |x|+1)\ U^d(i,\sigma) \text{ accepts in } T \text{ steps iff } x_i = \sigma \,\}.$$

We define $x_i = *$ if $i > |x|$; thus, for $i = |x|+1$ the machine accepts iff $\sigma = *$. The notation U^d indicates that the machine U has random access to the description d.

$\mathrm{KT}(x)$ is defined to be equal to $\mathrm{KT}_U(x)$ for a fixed choice of universal machine U with logarithmic simulation time overhead [1, Proposition 5]. In particular, if d consists of the description of a Turing machine M that runs in time $t_M(n)$ and some auxiliary information a such that $M^a(i) = x_i$ for $i \in [n]$, then $\mathrm{KT}(x) \le |a| + c_M T_M(\log n)\log(T_M(\log n))$, where $n \doteq |x|$ and c_M is a constant depending on M. It follows that $(\mu/\log n)^{\Omega(1)} \le \mathrm{KT}(x) \le (\mu\cdot\log n)^{O(1)}$ where μ represents the circuit complexity of the mapping $i \mapsto x_i$ [1, Theorem 11]. The Minimum KT Problem is defined as $\mathrm{MKTP} \doteq \{(x,\theta) \mid \mathrm{KT}(x) \le \theta\}$.

4.2 Rigid Graphs

The crux of Theorem 1 is the randomized mapping reduction from deciding whether a given pair of n-vertex graphs (G_0, G_1) is in GI to deciding whether $(y,\theta) \in \mathrm{MKTP}$, as prescribed by (1). Recall that (1) involves picking a string $r \doteq r_1 \dots r_t \in \{0,1\}^t$ and permutations π_i at random, and constructing the string $y = y_1 \dots y_t$, where $y_i = \pi_i(G_{r_i})$. We show how to determine θ such that a sufficiently large polynomial t guarantees that the reduction has no false negatives.

We first consider the simplest setting, in which both G_0 and G_1 are rigid. We argue that $\theta \doteq t(s+\frac{1}{2})$ works, where $s = \log(n!)$.

Nonisomorphic Case If $G_0 \not\equiv G_1$, then (by rigidity), each choice of r and each distinct sequence of t permutations results in a different string y, and thus the distribution on the strings y has entropy $t(s+1)$ where $s \doteq \log(n!)$. By a straightforward counting argument, a typical string sampled from such a distribution will have high KT-complexity. Formally, we have the following:

▶ **Proposition 7.** *Let y be sampled from a distribution with min-entropy s. For all k, we have $\mathrm{KT}(y) \ge s-k$ except with probability at most 2^{-k}.*

Our y is sampled from a uniform distribution, hence the entropy and min-entropy coincide. Thus $\mathrm{KT}(y) > \theta = t(s+1) - \frac{t}{2}$ with all but exponentially small probability in t, and so with high probability the algorithm declares G_0 and G_1 nonisomorphic.

Isomorphic Case. If $G_0 \equiv G_1$, we need to show that $\mathrm{KT}(y) \leq \theta$ always holds. The key insight is that the information in y is precisely captured by the t permutations $\tau_1, \tau_2, \ldots, \tau_t$ such that $\tau_i(G_0) = y_i$. These permutations exist because $G_0 \equiv G_1$; they are unique by the rigidity assumption. Thus, y contains at most ts bits of information. We show that its KT-complexity is not much larger than this.

We do this using an *efficiently decodable indexing* of the symmetric groups S_n. This is, for each n, a bijective map $[n!] \to S_n$ so that, on input i, the image of i can be computed in time $\mathrm{poly}(n)$. We rely on the following indexing, due to Lehmer (see, e.g., [21, pp. 12–33]):

▶ **Proposition 8** (Lehmer code). *The symmetric groups S_n have indexings that are uniformly decodable in time* $\mathrm{poly}(n)$.

To bound $\mathrm{KT}(y)$, we consider a program d that has the following information hard-wired into it: n, the adjacency matrix of G_0, and the t integers $k_1, \ldots, k_t \in [n!]$ encoding $\tau_1, \ldots, \tau_t$. We use the decoder from Proposition 8 to compute the i-th bit of y on input i. This can be done in time $\mathrm{poly}(n, \log(t))$ given the hard-wired information.

As mentioned in Section 1, a naïve method for encoding the indices $k_1, \ldots, k_t$ only gives the bound $t\lceil s \rceil + \mathrm{poly}(n, \log(t))$ on $\mathrm{KT}(y)$, which may exceed $t(s+1)$ and—*a fortiori*—the threshold θ, no matter how large a polynomial t is. We remedy this by aggregating multiple indices into blocks, and amortizing the encoding overhead across multiple samples. The following technical lemma captures the technique.

▶ **Lemma 9** (Blocking Lemma for Indexings). *Let $\{T_x\}$ be an ensemble of sets of strings such that all strings in T_x have the same length* $\mathrm{poly}(|x|)$. *Suppose that each T_x has an indexing decodable by circuits of size* $\mathrm{poly}(|x|)$. *Then there are constants $\alpha_0 > 0$ and c so that, for all $t \in \mathbb{N}$ and all sufficiently large $x \in \{0,1\}^*$, and every y that is the concatenation of t elements of T_x*

$$\mathrm{KT}(y) \leq t \log |T_x| + t^{1-\alpha_0} |x|^c$$

We first show how to apply the Blocking Lemma and then prove it. For a given rigid graph G, we let T_G be the set of adjacency matrices of permutations of G. To index T_G, we associate to each permutation $\tau(G)$ the index k of τ from the Lehmer code. Then there is a circuit of size $\mathrm{poly}(|G|)$ which takes as input k, computes τ, and then outputs $\tau(G)$. By the Blocking Lemma, we have that

$$\mathrm{KT}(y) \leq ts + t^{1-\alpha_0} n^c \tag{3}$$

for some constants $\alpha_0 > 0$ and c, and all sufficiently large n. Taking $t = n^{1+c/\alpha_0}$, we see that for all sufficiently large n, $\mathrm{KT}(y) \leq t(s + \frac{1}{2}) \doteq \theta$.

Proof of Lemma 9. Let T_x and D_x be the hypothesized ensemble of sets of strings and corresponding decoders. Fix x and t, let $m = \mathrm{poly}(|x|)$ denote the length of the strings in T_x, and let $b \in \mathbb{N}$ be a parameter to be set later.

To bound $\mathrm{KT}(y)$, we first write $y = y_1 \cdots y_t$ where each $y_j \in T_x$, and let $k_j \in [|T_x|]$ be the index of y_j via D_x. (i.e., $D_x(k_j) = y_j$.) We group the y_j's into $\lceil t/b \rceil$ size-b blocks $\widetilde{y}_1, \widetilde{y}_2, \ldots, \widetilde{y}_{\lceil t/b \rceil}$. For each block $\widetilde{y}_j$, let $\widetilde{k}_j$ the number whose base-$|T_x|$ representation is written $k_{b(j-1)+1}k_{b(j-1)+2}\cdots k_{bj}$. This is a number between 0 and $|T_x|^b - 1$, and hence can be expressed in binary with $\lceil b \log |T_x| \rceil$ bits. Given a circuit computing D_x, $\widetilde{y}_j$ can be computed from $\widetilde{k}_j$ in time polynomial in $|D_x|$, $\log |T_x|$, and b.

Consider a program d that has x, t, m, b, the circuit for computing D_x, and the indices $\widetilde{k}_1, \widetilde{k}_2, \ldots, \widetilde{k}_{\lceil t/b \rceil}$ hardwired, takes an input $i \in \mathbb{N}$, and determines the i-th bit of y as follows. It first computes $j_0, j_1 \in \mathbb{N}$ so that i points to the j_1-th bit position in $\widetilde{y}_{j_0}$. Then, using D_x, $\widetilde{k}_{j_0}$, and j_1, it computes $\widetilde{y}_{j_0}$ and outputs its j_1-th bit, which is the i-th bit of y.

The bit-length of d is at most $\lceil t/b \rceil \cdot \lceil b \log |T_x| \rceil$ for the indices, plus $\text{poly}(|x|, b, \log t)$ for the rest. The time needed by d is bounded by $\text{poly}(|x|, b, \log t)$. Thus $\text{KT}(y) \leq \lceil t/b \rceil \lceil b \log |T_x| \rceil + \text{poly}(|x|, b, \log t) \leq t \log |T_x| + t/b + \text{poly}(|x|, b, \log t)$. The lemma follows by choosing $b = \lceil t^{\alpha_0} \rceil$ for a sufficiently small constant α_0. ◀

4.3 Known Number of Automorphisms

We generalize the case of rigid graphs to graphs for which we know the size of their automorphism groups. Specifically, in addition to the two input graphs G_0 and G_1, we are also given numbers N_0, N_1 where $N_i \doteq n!/|\operatorname{Aut}(G_i)|$. Note that if $N_0 \neq N_1$, we can right away conclude that $G_0 \not\equiv G_1$. Nevertheless, we do not assume that $N_0 = N_1$ as the analysis of the case $N_0 \neq N_1$ will be useful in Section 4.4.

The reduction is the same as in Section 4.2 with the correct interpretation of s. The main difference lies in the analysis, where we need to accommodate for the loss in entropy that comes from having multiple automorphisms.

Let $s_i \doteq \log(N_i)$ be the entropy in a random permutation of G_i. Set $s \doteq \min(s_0, s_1)$, and $\theta \doteq t(s + \frac{1}{2})$. In the nonisomorphic case the min-entropy of y is at least $t(s+1)$, so $\text{KT}(y) > \theta$ with high probability. In the isomorphic case we upper bound $\text{KT}(y)$ by about ts. Unlike the rigid case, we can no longer afford to encode an entire permutation for each permuted copy of G_0.

Instead, we need an indexing for the cosets of $\text{Aut}(G_0)$ within S_n. Formally, this means a map $[n!/|\operatorname{Aut}(G_0)|] \to S_n$ so that for every coset of $\text{Aut}(G_0)$ in S_n, there is an index whose image through the map is in the coset. Given such an indexing, we can get an indexing of the set T_{G_0} of strings consisting of the adjacency matrices of permutations of G_0. The following indexing, applied to $\Gamma = \text{Aut}(G_0)$, suffices for this purpose.

▶ **Lemma 10.** *For every subgroup Γ of S_n there exists an indexing of the cosets of Γ that is decodable by circuits of size* $\text{poly}(n)$*.*

We prove Lemma 10 in Section 5 as a corollary to a more general lemma that gives, for each $\Gamma \leq H \leq S_n$, an efficiently computable indexing for the cosets of Γ in H.

▶ **Remark 11.** Before we continue towards Theorem 1, we point out that the above ideas yield an alternate proof that $\text{GI} \in \mathsf{BPP}^{\text{MKTP}}$ (and hence that $\text{GI} \in \mathsf{RP}^{\text{MKTP}}$). This weaker result was already obtained in [2] along the well-trodden path discussed in Section 1; this remark shows how to obtain it using our new approach.

The key observation is that in both the isomorphic and the nonisomorphic case, with high probability $\text{KT}(y)$ stays away from the threshold θ by a growing margin. Moreover, the above analysis allows us to efficiently obtain high-confidence approximations of θ to within any constant using sampling and queries to the MKTP oracle.

More specifically, for $i \in \{0, 1\}$, let $\widetilde{y}_i$ denote the concatenation of $\widetilde{t}$ independent samples from R_{G_i}. Our analysis shows that $\text{KT}(\widetilde{y}_i) \leq \widetilde{t} s_i + \widetilde{t}^{1-\alpha_0} n^c$ always holds, and that $\text{KT}(\widetilde{y}_i) \geq \widetilde{t} s_i - \widetilde{t}^{1-\alpha_0} n^c$ holds with high probability. Thus, $\widetilde{s}_i \doteq \text{KT}(\widetilde{y}_i)/\widetilde{t}$ approximates s_i with high confidence to within an additive deviation of $n^c/\widetilde{t}^{\alpha_0}$. Similarly, $\widetilde{s} \doteq \min(\widetilde{s}_0, \widetilde{s}_1)$ approximates s to within the same deviation margin, and $\widetilde{\theta} \doteq t(\widetilde{s} + \frac{1}{2})$ approximates θ to within an additive deviation of $tn^c/\widetilde{t}^{\alpha_0}$. The latter bound can be made less than 1 by setting $\widetilde{t}$ to a sufficiently large polynomial in n and t. Moreover, all these estimates can be computed in time $\mathrm{poly}(\widetilde{t}, n)$ with access to MKTP as MKTP enables us to evaluate KT efficiently.

4.4 Probably-Correct Underestimators for the Number of Automorphisms

The reason the $\mathsf{BPP}^{\mathrm{MKTP}}$-algorithm in Remark 11 can have false negatives is that the approximation $\widetilde{\theta}$ to θ may be too small. Knowing the quantities $N_i \doteq n!/|\operatorname{Aut}(G_i)|$ exactly allows us to compute θ exactly and thereby obviates the possibility of false negatives. In fact, it suffices to compute overestimates for the quantities N_i which are correct with non-negligible probability. We capture this notion formally as follows:

▶ **Definition 12** (probably-correct overestimator)**.** Let $g : \Omega \to \mathbb{R}$ be a function, and M a randomized algorithm that, on input $\omega \in \Omega$, outputs a value $M(\omega) \in \mathbb{R}$. We say that M is a *probably-correct overestimator* for g if, for every $\omega \in \Omega$, $M(\omega) = g(\omega)$ holds with probability at least $1/\mathrm{poly}(|\omega|)$, and $M(\omega) > g(\omega)$ otherwise. A *probably-correct underestimator* for g is defined similarly by reversing the inequality.

We point out that, for any probably-correct over-/underestimator, taking the min/max among $\mathrm{poly}(|\omega|)$ independent runs yields the correct value with probability $1 - 2^{-\mathrm{poly}(|\omega|)}$.

We are interested in the case where $g(G) = n!/|\operatorname{Aut}(G)|$. Assuming this g on a given class of graphs Ω has a probably-correct overestimator M computable in randomized polynomial time with an MKTP oracle, we argue that GI on Ω reduces to MKTP in randomized polynomial time without false negatives.

To see this, consider the algorithm that, on input a pair (G_0, G_1) of n-vertex graphs, computes $\widetilde{N}_i = M(G_i)$ as estimates of the true values $N_i = \log(n!/|\operatorname{Aut}(G_i)|)$, and then runs the algorithm from Section 4.3 using the estimates $\widetilde{N}_i$.

- In the case where G_0 and G_1 are not isomorphic, if both estimates $\widetilde{N}_i$ are correct, then the algorithm detects $G_0 \not\equiv G_1$ with high probability.
- In the case where $G_0 \equiv G_1$, if $\widetilde{N}_i = N_i$ we showed in Section 4.3 that the algorithm always declares G_0 and G_1 to be isomorphic. Moreover, increasing θ can only decrease the probability of a false negative. As the computed threshold θ increases as a function of $\widetilde{N}_i$, and the estimate $\widetilde{N}_i$ is always at least as large as N_i, it follows that G_0 and G_1 are always declared isomorphic.

4.5 Arbitrary Graphs

A probably-correct overestimator for the function $G \mapsto n!/|\operatorname{Aut}(G)|$ on *any* graph G can be computed in randomized polynomial time with access to MKTP. The process is described in full detail in Section 1, based on a $\mathsf{BPP}^{\mathrm{MKTP}}$ algorithm for GI (taken from Remark 11 or from [2]). This means that the setting of Section 4.4 is actually the general one. The only difference is that we no longer obtain a mapping reduction from GI to MKTP, but an oracle reduction: We still make use of (1), but we need more queries to MKTP in order to set the threshold θ.

This shows that $\mathrm{GI} \in \mathsf{coRP}^{\mathrm{MKTP}}$. As $\mathrm{GI} \in \mathsf{RP}^{\mathrm{MKTP}}$ follows from the known search-to-decision reduction for GI, this concludes the proof of Theorem 1 that $\mathrm{GI} \in \mathsf{ZPP}^{\mathrm{MKTP}}$.

5 Coset Indexings for Permutation Groups

In this section we develop the efficiently decodable indexings for cosets of permutation subgroups claimed in Lemma 10. In fact, we present a further generalization that may be of independent interest, namely an efficiently decodable indexing for cosets of permutation subgroups within another permutation subgroup.

▶ **Lemma 13.** *For all $\Gamma \leq H \leq S_n$, there exists an indexing of the cosets*[1] *of Γ within H that is uniformly decodable in polynomial time when Γ and H are given by a list of generators.*

Lemma 10 is just the instantiation of Lemma 13 with $H = S_n$ followed by hardwiring generators for Γ and S_n into a circuit simulating the decoder from Lemma 13. The proof of Lemma 13 requires some elements of the theory of permutation groups. Given a list of permutations $\pi_1, \ldots, \pi_k \in S_n$, we write $\Gamma = \langle \pi_1, \ldots, \pi_k \rangle \leq S_n$ for the subgroup they generate. Given a permutation group $\Gamma \leq S_n$ and a point $i \in [n]$, the Γ-orbit of i is the set $\{g(i) : g \in \Gamma\}$, and the Γ-stabilizer of i is the subgroup $\{g \in \Gamma : g(i) = i\} \leq \Gamma$.

We make use of the fact that (a) the number of cosets of a subgroup Γ of a group H equals $|H|/|\Gamma|$, and (b) the orbits of a subgroup Γ of H form a refinement of the orbits of H. We also need the following basic routines from computational group theory (see, for example, [19, 29]).

▶ **Proposition 14.** *Given a set of permutations that generate a subgroup $\Gamma \leq S_n$, the following can be computed in time polynomial in n:*

(1) *the cardinality $|\Gamma|$,*

(2) *a permutation in Γ that maps u to v for given $u, v \in [n]$, or report that no such permutation exists in Γ, and*

(3) *a list of generators for the subgroup Γ_v of Γ that stabilizes a given element $v \in [n]$.*

The proof of Lemma 13 makes implicit use of an efficient process for finding a *canonical representative* of $\pi\Gamma$ for a given permutation $\pi \in H$, where "canonical" means that the representative depends on the coset $\pi\Gamma$ only. The particular canonical representative the process produces can be specified as follows.

▶ **Definition 15.** For a permutation $\pi \in S_n$ and a subgroup $\Gamma \leq S_n$, the *canonical representative* of π modulo Γ, denoted $\pi \bmod \Gamma$, is the lexicographically least $\pi' \in \pi\Gamma$, where the lexicographic ordering is taken by viewing a permutation π' as the sequence $(\pi'(1), \pi'(2), \ldots, \pi'(n))$.

We describe the process as it provides intuition for the proof of Lemma 13.

▶ **Lemma 16.** *There exists a polynomial-time algorithm that takes as input a generating set for a subgroup $\Gamma \leq S_n$ and a permutation $\pi \in S_n$, and outputs the canonical representative* $\pi \bmod \Gamma$.

Proof of Lemma 16. Consider the element 1 of $[n]$. Permutations in $\pi\Gamma$ map 1 to an element v in the same Γ-orbit as $\pi(1)$, and for every element v in the Γ-orbit of $\pi(1)$ there exists a permutation in $\pi\Gamma$ that maps 1 to v. We can canonize the behavior of π on the element 1 by replacing π with a permutation $\pi_1 \in \pi\Gamma$ that maps 1 to the minimum element m in the

[1] The choice of left ($\pi\Gamma$) vs right ($\Gamma\pi$) cosets is irrelevant for us; all our results hold for both, and one can usually switch from one statement to the other by taking inverses. Related to this, there is an ambiguity regarding the order of application in the composition gh of two permutations: first apply g and then h, or vice versa. Both interpretations are fine. For concreteness, we assume the former.

Γ-orbit of $\pi(1)$. This can be achieved by multiplying π to the right with a permutation in Γ that maps $\pi(1)$ to m.

Next we apply the same process to π_1 but consider the behavior on the element 2 of $[n]$. Since we are no longer allowed to change the value of $\pi_1(1)$, which equals m, the canonization of the behavior on 2 can only use multiplication on the right with permutations in Γ_m, i.e., permutations in Γ that stabilize the element m. Doing so results in a permutation $\pi_2 \in \pi_1\Gamma$.

We repeat this process for all elements $k \in [n]$ in order. In the k-th step, we canonize the behavior on the element k by multiplying on the right with permutations in $\Gamma_{\pi_{k-1}([k-1])}$, i.e., permutations in Γ that pointwise stabilize all of the elements $\pi_{k-1}(\ell)$ for $\ell \in [k-1]$. ◀

Proof of Lemma 13. The number of canonical representatives modulo Γ in H equals the number of distinct (left) cosets of Γ in H, which is $|H|/|\Gamma|$. We construct an algorithm that takes as input a list of generators for Γ and H, and an index $i \in [|H|/|\Gamma|]$, and outputs the permutation σ that is the lexicographically i-th canonical representative modulo Γ in H.

The algorithm uses a prefix search to construct σ. In the k-th step, it knows the prefix $(\sigma(1), \sigma(2), \ldots, \sigma(k-1))$ of length $k-1$, and needs to figure out the correct value $v \in [n]$ to extend the prefix with. In order to do so, the algorithm needs to compute for each $v \in [n]$ the count c_v of canonical representatives modulo Γ in H that agree with σ on $[k-1]$ and take the value v at k. The following claims allow us to do that efficiently when given a permutation $\sigma_{k-1} \in H$ that agrees with σ on $[k-1]$. The claims use the notation $T_{k-1} \doteq \sigma_{k-1}([k-1])$, which also equals $\sigma([k-1])$.

▶ **Claim 17.** *The canonical representatives modulo Γ in H that agree with $\sigma \in H$ on $[k-1]$ are exactly the canonical representatives modulo $\Gamma_{T_{k-1}}$ in $\sigma_{k-1}H_{T_{k-1}}$.*

Proof. The following two observations imply Claim 17.

(i) A permutation $\pi \in H$ agrees with $\sigma \in H$ on $[k-1]$
$\Leftrightarrow$ π agrees with σ_{k-1} on $[k-1]$
$\Leftrightarrow$ $\sigma_{k-1}^{-1}\pi \in H_{T_{k-1}}$
$\Leftrightarrow$ $\pi \in \sigma_{k-1}H_{T_{k-1}}$.

(ii) Two permutations in $\sigma_{k-1}H_{T_{k-1}}$, say $\pi \doteq \sigma_{k-1}g$ and $\pi' \doteq \sigma_{k-1}g'$ for $g, g' \in H_{T_{k-1}}$, belong to the same left coset of Γ iff they belong to the same left coset of $\Gamma_{T_{k-1}}$. This follows because if $\sigma_{k-1}g' = \sigma_{k-1}gh$ for some $h \in \Gamma$, then h equals $g^{-1}g' \in H_{T_{k-1}}$, so $h \in \Gamma \cap H_{T_{k-1}} = \Gamma_{T_{k-1}}$. ◀

▶ **Claim 18.** *The count c_v for $v \in [n]$ is nonzero iff v is the minimum of some $\Gamma_{T_{k-1}}$-orbit contained in the $H_{T_{k-1}}$-orbit of $\sigma_{k-1}(k)$.*

Proof. The set of values of $\pi(k)$ when π ranges over $\sigma_{k-1}H_{T_{k-1}}$ is the $H_{T_{k-1}}$-orbit of $\sigma_{k-1}(k)$. Since $\Gamma_{T_{k-1}}$ is a subgroup of $H_{T_{k-1}}$, this orbit is the union of some $\Gamma_{T_{k-1}}$-orbits. Combined with Claim 17 and the construction of the canonical representatives modulo $\Gamma_{T_{k-1}}$, this implies Claim 18. ◀

▶ **Claim 19.** *If a count c_v is nonzero then it equals $|H_{T_{k-1}\cup\{v\}}|/|\Gamma_{T_{k-1}\cup\{v\}}|$.*

Proof. Since the count is nonzero, there exists a permutation $\sigma' \in H$ that is a canonical representative modulo Γ that agrees with σ_{k-1} on $[k-1]$ and satisfies $\sigma'(k) = v$. Applying Claim 17 with σ replaced by σ', k by $k' \doteq k+1$, T_{k-1} by $T'_k \doteq T_{k-1} \cup \{v\}$, and σ_{k-1} by any permutation $\sigma'_k \in H$ that agrees with σ' on $[k]$, yields Claim 19. This is because the number of canonical representatives modulo $\Gamma_{T'_k}$ in $\sigma'_k H_{T'_k}$ equals the number of (left) cosets of $\Gamma_{T'_k}$ in $H_{T'_k}$, which is the quantity stated in Claim 19. ◀

Algorithm 1

```
Input: positive integer $n$, $\Gamma \leq H \leq S_n$, $i \in [|H|/|\Gamma|]$
Output: lexicographically $i$-th canonical representative modulo $\Gamma$ in $H$
 1: $\sigma_0 \leftarrow id$
 2: for $k = 1$ to $n$ do
 3:     $O_1, O_2, \ldots \leftarrow$ $\Gamma$-orbits contained in the $H$-orbit of $\sigma_{k-1}(k)$, in increasing order of $\min(O_i)$
 4:     find integer $\ell$ such that $\sum_{j=1}^{\ell-1} c_{\min(O_j)} < i \leq \sum_{j=1}^{\ell} c_{\min(O_j)}$, where $c_v \doteq |H_v|/|\Gamma_v|$
 5:     $i \leftarrow i - \sum_{i=1}^{\ell-1} c_{\min(O_j)}$
 6:     $m \leftarrow \min(O_\ell)$
 7:     find $\tau \in H$ such that $\tau(\sigma_{k-1}(k)) = m$
 8:     $\sigma_k \leftarrow \sigma_{k-1}\tau$
 9:     $H \leftarrow H_m$; $\Gamma \leftarrow \Gamma_m$
10: return $\sigma_n$
```

The algorithm builds a sequence of permutations $\sigma_0, \sigma_1, \ldots, \sigma_n \in H$ such that σ_k agrees with σ on $[k]$. It starts with the identity permutation $\sigma_0 = id$, builds σ_k out of σ_{k-1} for increasing values of $k \in [n]$, and outputs the permutation $\sigma_n = \sigma$.

Pseudocode for the algorithm is presented in Algorithm 1. Note that the pseudocode modifies the arguments Γ, H, and i along the way. Whenever a group is referenced in the pseudocode, the actual reference is to a list of generators for that group.

The correctness of the algorithm follows from Claims 18 and 19. The fact that the algorithm runs in polynomial time follows from Proposition 14. ◀

6 Future Directions

We end with a few directions for further research.

6.1 What about Minimum Circuit Size?

We suspect that our techniques also apply to MCSP in place of MKTP, but we have been unsuccessful in extending them to MCSP so far. To show our result for the complexity measure $\mu = \mathrm{KT}$, we showed the following property for polynomial-time samplable flat distributions R: There exists an efficiently computable bound $\theta(s,t)$ and a polynomial t such that if y is the concatenation of t independent samples from R, then

$$\mu(y) > \theta(s,t) \text{ holds with high probability if } R \text{ has entropy } s+1 \text{, and} \quad (4)$$

$$\mu(y) \leq \theta(s,t) \text{ always holds if } R \text{ has entropy } s. \quad (5)$$

We set $\theta(s,t)$ slightly below $\kappa(s+1,t)$ where $\kappa(s,t) \doteq st$. (4) followed from a counting argument, and (5) by showing that

$$\mu(y) \leq \kappa(s,t) \cdot \left(1 + \frac{n^c}{t^\alpha}\right) \quad (6)$$

always holds for some positive constants c and α. We concluded by observing that for a sufficiently large polynomial t the right-hand side of (6) is significantly below $\kappa(s+1,t)$.

Mimicking the approach with μ denoting circuit complexity, we set

$$\kappa(s,t) = \frac{st}{\log(st)} \cdot \left(1 + (2 - o(1)) \cdot \frac{\log\log(st)}{\log(st)}\right).$$

Then (4) follows from [31]. As for (5), the best counterpart to (6) we know of (see, e.g., [10]) is

$$\mu(y) \leq \frac{st}{\log(st)} \cdot \left(1 + (3 + o(1)) \cdot \frac{\log\log(st)}{\log(st)}\right).$$

However, in order to make the right-hand side of (6) smaller than $\kappa(s+1,t)$, t needs to be exponential in s.

One possible way around the issue is to boost the entropy gap between the two cases. This would not only show that all our results for MKTP apply to MCSP as well, but could also form the basis for reductions between different versions of MCSP (defined in terms of different circuit models, or in terms of different size parameters), and to clarify the relationship between MKTP and MCSP. Until now, all of these problems have been viewed as morally equivalent to each other, although no efficient reduction is known between *any* two of these, in either direction. Given the central role that MCSP occupies, it would be desirable to have a theorem that indicates that MCSP is fairly robust to minor changes to its definition. Currently, this is lacking.

On a related point, it would be good to know how the complexity of MKTP compares with the complexity of the KT-random strings: $R_{\text{KT}} = \{x : \text{KT}(x) \geq |x|\}$. Until now, all prior reductions from natural problems to MCSP or MKTP carried over to R_{KT}—but this would seem to require even stronger gap amplification theorems. The relationship between MKTP and R_{KT} is analogous to the relationship between MCSP and the special case of MCSP that is denoted MCSP′ in [23]: MCSP′ consists of truth tables f of m-ary Boolean functions that have circuits of size at most $2^{m/2}$.

6.2 Statistical Zero Knowledge

Allender and Das [2] generalized their result that GI $\in \mathsf{RP}^{\text{MKTP}}$ to $\mathsf{SZK} \subseteq \mathsf{BPP}^{\text{MKTP}}$ by applying their approach to a known SZK-complete problem. Our proof that GI $\in \mathsf{coRP}^{\text{MKTP}}$ similarly generalizes to $\mathsf{SZK} \subseteq \mathsf{BPP}^{\text{MKTP}}$. We use the problem Entropy Approximation, which is complete for SZK under oracle reductions [12, Lemma 5.1]:[2] Given a circuit C and a threshold θ with the promise that the distribution induced by C has entropy either at most $\theta - 1$ or else at least $\theta + 1$, decide whether the former is the case. By combining the Flattening Lemma [13] with our hashing-based generic encoding mentioned in the introduction, one can show that for any distribution of entropy s sampled by a circuit C, the concatenation of t random samples from C has, with high probability, KT complexity between $ts - t^{1-\alpha_0} \cdot \text{poly}(|C|)$ and $ts + t^{1-\alpha_0} \cdot \text{poly}(|C|)$ for some positive constant α_0. Along the lines of Remark 11, this allows us to show that Entropy Approximation, and hence all of SZK, is in $\mathsf{BPP}^{\text{MKTP}}$.

We do not know how to eliminate the errors from those reductions: Is $\mathsf{SZK} \subseteq \mathsf{ZPP}^{\text{MKTP}}$, or equivalently, is Entropy Approximation in $\mathsf{ZPP}^{\text{MKTP}}$? Our approach yields that Entropy Approximation is in $\mathsf{coRP}^{\text{MKTP}}$ (no false negatives) *when the input distributions are almost flat*, i.e., when the difference between the max- and min-entropy is small. However, it is not known whether that restriction of Entropy Approximation is complete for SZK[3] (Goldreich

[2] Entropy Approximation is complete for NISZK (Non-Interactive SZK) under *mapping* reductions [12]. Problems that are complete for SZK under mapping reductions include Statistical Difference [28] and Entropy Difference [13]. The mapping reduction from Statistical Difference to Entropy Difference in [14, Theorem 3] is similar to our reduction from Isomorphism Problems to MKTP.

[3] The Flattening Lemma [13] allows us to restrict to distributions that are almost flat in an *average-case*

and Vadhan, personal communication). Moreover, we do not see how to eliminate the false positives.

Trying to go beyond SZK, we mention that except for the possible use of the MKTP oracle in the construction of the probably-correct overestimator from condition 3 in Theorem 3, the reduction in Theorem 3 makes only one query to the oracle. It was observed in [18] that the reduction also works for any relativized KT problem MKTP^A (where the universal machine for KT complexity has access to oracle A). More significantly, [18] shows that any problem that is accepted with negligible error probability by a probabilistic reduction that makes only one query, relative to *every* set MKTP^A, must lie in $\mathsf{AM} \cap \mathsf{coAM}$. Thus, without significant modification, our techniques cannot be used in order to reduce any class larger than $\mathsf{AM} \cap \mathsf{coAM}$ to MKTP.

The property that only one query is made to the oracle was subsequently used in order to show that MKTP is hard for the complexity class DET under mapping reductions computable in nonuniform NC^0 [4]. Similar hardness results (but for a more powerful class of reducibilities) hold also for MCSP [25]. This has led to unconditional lower bounds on the circuit complexity of MKTP [4, 17], showing that MKTP does not lie in the complexity class $\mathsf{AC}^0[p]$ for any prime p; it is still open whether similar circuit lower bounds hold for MCSP.

Acknowledgments. We thank V. Arvind for helpful comments about the graph automorphism problem and rigid graphs, Alex Russell and Yoav Kallus for helpful ideas on encoding and decoding graphs, Laci Babai and Peter Brooksbank for answering questions about computational group theory, and Oded Goldreich and Salil Vadhan for answering questions about SZK. We also thank the anonymous reviewers for their helpful suggestions.

References

1 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35:1467–1493, 2006.

2 Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In *Mathematical Foundations of Computer Science 2014*, pages 25–32, 2014.

3 Eric Allender, Joshua Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum circuit size, graph isomorphism, and related problems. Technical Report TR17-158, Electronic Colloquium on Computational Complexity, 2017.

4 Eric Allender and Shuichi Hirahara. New insights on the (non)-hardness of circuit minimization and related problems. In *Proceedings of the 42nd International Symposium on Mathematical Foundations of Computer Science*, 2017. To appear.

5 Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *Journal of Computer and System Sciences*, 77:14–40, 2010.

6 László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the 48th annual ACM Symposium on Theory of Computing*, pages 684–697, 2016.

7 László Babai, Paolo Codenotti, and Youming Qiao. Polynomial-time isomorphism test for groups with no abelian normal subgroups. In *Automata, Languages, and Programming*, pages 51–62, 2012.

sense, but we need almost flatness in the above *worst-case* sense. For example, consider a circuit C that induces a distribution of entropy less than $\theta - 1$ whose support contains all strings of length n where $n \gg \theta$. In that case, there is no nontrivial worst-case bound on the KT complexity of samples from C; with positive probability, t samples from C may have KT-complexity close to $t \cdot n \gg t \cdot (\theta + 1)$.

8 László Babai. Local Expansion of Vertex-transitive Graphs and Random Generation in Finite Groups. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 164–174, 1991.

9 Marco Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Algorithms from natural lower bounds. In *Proceedings of the 31st Computational Complexity Conference*, pages 10:1–10:24, 2016.

10 Gudmund Skovbjerg Frandsen and Peter Bro Miltersen. Reviewing bounds on the circuit size of the hardest functions. *Information Processing Letters*, 95(2):354–357, 2005.

11 Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

12 Oded Goldreich, Amit Sahai, and Salil Vadhan. Can statistical zero knowledge be made non-interactive? or on the relationship of SZK and NISZK. In *Advances in Cryptology — CRYPTO '99*, pages 467–484, 1999.

13 Oded Goldreich and Salil Vadhan. Comparing entropies in statistical zero knowledge with applications to the structure of SZK. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity*, pages 54–73, 1999.

14 Oded Goldreich and Salil Vadhan. On the complexity of computational problems regarding distributions. In Oded Goldreich, editor, *Studies in Complexity and Cryptography – Miscellanea on the Interplay between Randomness and Computation*, pages 13–29. Springer, 2011.

15 Joshua A. Grochow. Matrix Lie algebra isomorphism. In *Proceedings of the 2012 IEEE Conference on Computational Complexity*, pages 203–213, 2012.

16 Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:1364–1396, 1999.

17 Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *Proceedings of the 32nd Computational Complexity Conference*, pages 7:1–7:20, 2017.

18 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Proceedings of the 31st Conference on Computational Complexity*, pages 18:1–18:20, 2016.

19 Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien. *Handbook of Computational Group Theory*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, 2005.

20 Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 73–79, 2000.

21 Donald E. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, 1973.

22 Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhauser Verlag, Basel, Switzerland, Switzerland, 1993.

23 Cody D. Murray and R. Ryan Williams. On the (Non) NP-Hardness of Computing Circuit Complexity. *Theory of Computing*, 13:1–22, 2017.

24 Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

25 Igor C. Carboni Oliveira and Rahul Santhanam. Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *Proceedings of the 32nd Computational Complexity Conference*, pages 18:1–18:49, 2017.

26 Erez Petrank and Ron M. Roth. Is code equivalence easy to decide? *IEEE Transactions on Information Theory*, 43:1602–1604, 1997.

27 Michael Rudow. Discrete logarithm and minimum circuit size. *Information Processing Letters*, 128:1–4, 2017.

28 Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM*, 50(2):196–249, 2003.

29 Ákos Seress. *Permutation group algorithms*, volume 152 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 2003.

30 Boris A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984.

31 Masaki Yamamoto. A tighter lower bound on the circuit size of the hardest Boolean functions. Technical Report TR11-086, Electronic Colloquium on Computational Complexity, 2011.