A Note on Hardness under Projections for Graph Isomorphism and Time-Bounded Kolmogorov Complexity

Eric Allender
Department of Computer Science, Rutgers University
allender@cs.rutgers.edu

Azucena Garvía Bosshard
School of Informatics, University of Edinburgh
agarvia@student.ethz.ch

Amulya Musipatla
School of Computer Science, Carnegie Mellon University
amusipat@andrew.cmu.edu

October 26, 2020

Abstract

This paper focuses on a variant of the circuit minimization problem (MCSP), denoted MKTP, which studies resource-bounded Kolmogorov complexity in place of circuit size. MCSP is not known to be hard for any complexity class under any kind of m-reducibility, but recently MKTP was shown to be hard for DET under m-reductions computable in NC^0, by presenting an AC^0 reduction from the rigid graph isomorphism problem to MKTP, and combining that with a theorem of Torán, showing that DET AC^0-reduces to the rigid graph isomorphism problem, and then appealing to the "Gap Theorem" of [1]. Here, we show that these reductions can be accomplished by means of *projections*. Thus MKTP is hard for DET under projections, and the rigid graph isomorphism problem is hard for DET under *uniform* projections.

1 Introduction

The Minimum Kolmogorov Time-Bounded Complexity Problem (MKTP) has attracted attention due to its connection to the Minimum Circuit Size Problem (MCSP). MKTP is the problem of determining whether a string s has complexity less than a certain threshold, where this complexity is defined as the minimum of $|d|+t$ over all descriptions, d, of s given that a Universal Turing Machine can

output the i^{th} bit of s in time t on input (d, i). This problem and its variants are introduced in [3].

In this paper, we improve the known hardness result for MKTP by Allender and Hirahara, that MKTP is hard for DET under non-uniform NC^0 reductions [7], and show that MKTP is in fact hard for DET under non-uniform projections. In other words, the existing reduction had each output bit depending on a constant number of input bits while in our reduction each output bit depends on at most one input bit.

Clearly this type of reduction is much more restrictive, which helps motivate interest in this result. For instance, if someone were to show that there is a problem in DET that requires superpolynomial size on depth-three threshold circuits, then this lower bound would now carry over immediately to MKTP; whereas this would not follow immediately from the NC^0-reduction of [7].

Time-Bounded Kolmogorov complexity and circuit complexity are polynomially related, although there is no known $\leq_T^{\mathsf{P}}$ reduction between MKTP and MCSP. Due to this connection, results regarding MKTP tend to follow through immediately with MCSP and it has historically been easier to find hardness results for MKTP. Recently, Allender and Hirahara [7] proved an implication between the hardness of MKTP^A and MCSP^A, for many oracles A. These problems are widely studied due to the belief that they might be NP-intermediate, at least under uniform AC^0 reductions (which one might be able to prove without settling the P vs NP question).

There is ample reason to believe that $\mathsf{MCSP} \notin \mathsf{P}$; if MCSP were computable in P then there would be no cryptographically-secure one-way functions [14]. Additionally, many supposedly-intractable problems, whose conjectured difficulty underlies various cryptographic systems, such as the Discrete Log assumption, are solvable in $\mathsf{BPP}^{\mathsf{MCSP}}$ and $\mathsf{BPP}^{\mathsf{MKTP}}$ [3].

Some evidence that MCSP might not be NP-hard comes from the fact that many canonical NP-complete problems, such as SAT, are actually complete under very restrictive reductions ($\mathsf{TIME}(poly(\log n))$), whereas Murray and Williams [16] show that MCSP is not complete under this type of reduction; they prove that for every $\delta < \frac{1}{2}$, there is no $\mathrm{TIME}(n^\delta)$ reduction from PARITY to MCSP. [16] also proves that there would be interesting consequences of NP-hardness results for MCSP, for instance, if MCSP is NP-hard under polynomial-time reductions, then $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{P/poly}$ and consequently, $\mathsf{EXP} \neq \mathsf{ZPP}$. Further improvements in this direction were recently presented by Fu [10].

Torán [17] proved that Graph Isomorphism (GI) is hard for DET under AC^0 reductions, and showed that this holds even for "rigid" graphs (graphs with no nontrivial automorphisms). Graph isomorphism is in $\mathsf{BPP}^{\mathsf{MCSP}}$ and $\mathsf{BPP}^{\mathsf{MKTP}}$ [4] and even lies in $\mathsf{ZPP}^{\mathsf{MKTP}}$ [6]. The Rigid Graph Isomorphism Problem (RGI) reduces to MKTP under NC^0 reductions [7]. Since RGI is hard for DET under AC^0 reductions [17] and even under NC^0 reductions [7], this yielded the first proof that MKTP is hard for a well-studied complexity class under some notion of m-reduction.

Here, we show that the reduction from DET to RGI can even be a uniform projection, and we observe that the reduction from RGI to MKTP in [7] is

actually a non-uniform projection, thus yielding the first proof that MKTP is hard for a well-studied complexity class under projections. Subsequently, a larger complexity class (which is believed to contain intractable problems and which does contain RGI) was also shown to reduce to MKTP under (nonuniform) projections [5].

For that reason, this article does not focus on the reduction from RGI to MKTP (which has been superseded by [5]). It may still be of use to other researchers, to know that RGI is hard for DET under uniform projections, and this does not follow from the later work. Thus this reduction may now be viewed as the main contribution of the current paper.

2 Preliminaries

We assume familiarity with basic complexity theory, such as the complexity classes L and NL (deterministic and nondeterministic logspace, respectively) and the circuit complexity classes AC^0 and NC^1. The required background can be found in a textbook such as [18].

The Graph Isomorphism problem (GI) is the problem of determining whether two graphs are isomorphic. We sometimes assume that various vertices are assigned a "color" (and that isomorphisms must respect the colors of vertices); this so-called Colored GI is well-known to be equivalent to GI [15]. We will also refer to RGI: the restriction of GI to rigid graphs. A rigid graph is one that has no automorphisms, and RGI asks to determine whether two rigid graphs are isomorphic.

#L is the class of counting functions $f(x)$ that correspond to problems in NL [8]. More formally, it is the class of functions $f : \{0,1\}^* \to \mathbb{N}$ where $f(x)$ computes the number of accepting paths on input x for some Turing Machine deciding a problem in NL. For example, the problem of determining whether a graph has a path from node s to t is in NL. Its corresponding counting problem of counting the number of paths between s and t is in #L. In fact, computing any $f(x)$ for $f \in \#\mathsf{L}$ is reducible to computing the number of paths in a directed graph G_x. That is, there is an AC^0-computable function G such that $f(x)$ is the number of $s-t$ paths in the graph $G(x)$.

It is useful to identify a class of *languages* whose complexity coincides with that of #L. If $f \in \#\mathsf{L}$, define L_f to be $\{(x,i,b) :$ the i-th bit of $f(x)$ is $b\}$. Given a function $f \in \#\mathsf{L}$ and a problem A, we say that f reduces to A if L_f reduces to A.

The determinant class (DET) is the class of problems that are NC^1-Turing reducible to the problem of computing the determinant of a matrix. This class can equivalently be described as $\mathsf{NC}^1(\#\mathsf{L})$ [8], i.e. the class of problems that are NC^1-Turing reducible to #L.

A *projection* is a very restrictive type of reduction, which is computed by a family of circuits that have no gates (other than NOT gates). Thus every output bit (other than bits that are set to constants in $\{0,1\}$) is connected by a wire to some input bit or its negation. We consider only projections computed

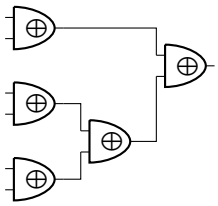

Figure 1: Funnel structure on 6 inputs.

by circuits of polynomial size. We write $A \leq_m^{\mathsf{proj}} B$ if A is reducible to B via a projection (possibly computed by a nonuniform family of circuits). If $A \leq_m^{\mathsf{proj}} B$ via a *uniform* family of circuits (consult [18] for a discussion of uniformity), then we write $A \leq_m^{\mathsf{fop}} B$. Here, "fop" stands for *first-order projection*, because of their use in the study of descriptive complexity [13, 2] (where uniform AC^0 can be viewed as characterizing the expressive power of first-order logic).

2.1 Chinese remainder representation

Let $B_n = \{p_1, ..., p_n\}$, with $p_1 < ... < p_n$, denote the set of the first m primes and let $M = \prod_{p \in B_n} p$. The Chinese remainder representation of an integer x in base B_n is defined as $(x_1, ..., x_n)$, where $x_i = x \mod p_i$ for all $i \in [m]$. This representation is unique whenever $0 \leq x < M$.

Theorem 2.1. *The problem of obtaining the binary representation of an integer given its Chinese remainder representation is in* NC^1 *(and even in uniform* TC^0*) [9, 12].*

3 Main Theorem

In this section, we show that $\mathsf{DET} \leq_m^{\mathsf{fop}} \mathsf{RGI}$. Our proof closely follows the development in [17]. The complexity of DET can be characterized by the complexity of counting paths in graphs. The first step toward building our reduction is to show that counting paths mod$k$ reduces to RGI via projections.

3.1 Counting paths mod k with RGI

It is convenient to consider arithmetic circuits consisting of mod gates of fan-in two. For that reason, we introduce the following definition:

Definition 3.0.1. $C_n^*[k]$ denotes a circuit on n inputs with a funnel-like (or tree-like) structure of fan-in 2 mod k gates. This structure has depth $\log(n)$ such that the i^{th} layer connects the outputs of the $i - 1^{th}$ layer in pairs.

See Figure 1 for an example of such a structure.

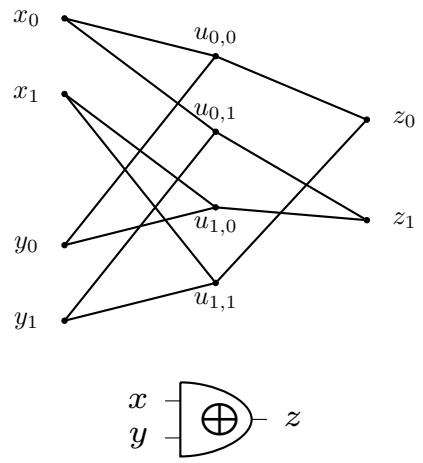

Figure 2: The graph G^2 representing a mod2 gate.

The motivation behind this definition is to turn a mod k gate of arbitrary fan-in into a structure that computes the same function but using only gates of fan-in 2.

For completeness, we include the following definition and lemma from [17].

Definition 3.0.2. Let $k \geq 2$ and denote by $\oplus$ the addition in $\mathbb{Z}_k$. We define the undirected graph $G^k = (V, E)$, given by the set of $k^2 + 3k$ nodes

$$V = \{x_a, y_a, z_a \mid a \in \{0, ..., k-1\}\} \cup \{u_{a,b} \mid a, b \in \{0, ..., k-1\}\}$$

and edges

$$E = \{(x_a, u_{a,b}) \mid a, b \in \{0, ..., k-1\}\} \cup \{(y_b, u_{a,b}) \mid a, b \in \{0, ..., k-1\}\} \cup \{(u_{a,b}, z_{a\oplus b}) \mid a, b \in \{0, ..., k-1\}\}$$

Figure 2 shows the graph that corresponds to the mod2 gate.

Lemma 3.1. *Fix $k \geq 2$, for any $a, b \in \{0, ..., k-1\}$,*

1. *there is an automorphism φ in G^k mapping x_0 to x_a and y_0 to y_b, and*
2. *every automorphism φ in G^k mapping x_0 to x_a and y_0 to y_b, maps z_0 to $z_{a\oplus b}$*

Armed with this gadget, we can now make further progress toward reducing counting paths to RGI.

Lemma 3.2. *For any* NL *machine M and for any $k \in \mathbb{N}$, there is a first-order projection that maps string y to a string x and a circuit C containing only fan-in two mod k gates such that C on input x outputs $P \bmod k$, where P is the number of accepting paths of M on input x. The size of the projection is polynomial in $|y|$ and k.*

Proof. We may assume without loss of generality that the NL machine M has exactly one accepting configuration, which is also a halting configuration. We may also assume that M keeps track of the number of steps that it has executed, so that its configuration graph is acyclic. Consider the graph G consisting of all configurations of M on input y, and note that the number of nodes in G is polynomial in $|y|$. Let s denote the node of G corresponding to the initial configuration of M on input y, and let t denote the node corresponding to the accepting configuration. Clearly P is equal to the number of paths in G from s to t, and also G can be constructed from y via a uniform projection.

First, given s and t, consider assigning the following weight function w to all the edges of G. Let $N(u)$ denote the set of all neighbors of u, and let V be the set of vertices of G.

$$w(u,v) = \begin{cases} 1 & \text{if } u = s \\ 0 & \text{if } u = t (\text{there are no edges}(t,v)) \\ \sum_{x \in N(u)} w(x,u) & \text{otherwise} \end{cases}$$

Under this weight function, P equals the sum of the edge weights going into t. Expanding on this, for any number k, computing $P \bmod k$ is as simple as modifying each weight such that $w^*(u,v) = w(u,v) \bmod k$.

This can be envisioned as a circuit C such that each node except s is a $\bmod k$ gate, and the output of a gate is an input of another if there is a directed edge between the two in G. The only other input to any gate is a 1 if there is an edge directed from s to its corresponding vertex. Each gate has at most $|V|$ inputs.

Now, we replace each gate in C by $C^*_{|V|}(k)$. This structure has $|V|-1$ gates at the level closest to the inputs and depth $\lceil \log(|V|) \rceil$, where $|V|$ is the number of vertices in G (Fig. 1). As explained above, this structure has the property that every gate in the circuit has a maximum of two inputs, but the output of the structure still computes $\bmod k$ of all inputs to the structure. The structures are connected in the same way as in C, with the exception that the i^{th} input to the funnel for gate v is connected to the output of the structure for gate i if there is an edge from i to v in G, and otherwise it is connected to a constant 0.

Clearly, there is a projection that produces an encoding of this circuit (as an adjacency matrix), given G, and hence there is a projection that produces the circuit given y, since the composition of two projections is a projection. The input to the circuit consists of a 1 feeding into the node s, and 0's feeding into any other input gate.

There are three classes of bits in the adjacency matrices of the circuit.

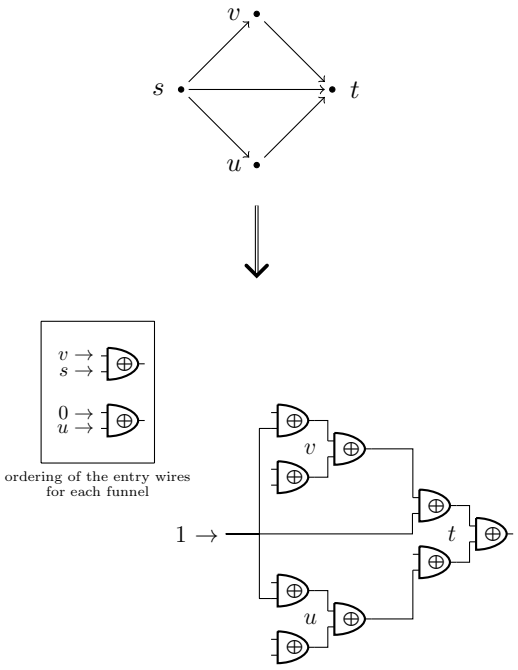

Figure 3: An example of a graph being transformed into a circuit with fan-in 2 addition mod $\mathbb{Z}_k$ as described in Lemma 3.2 The output of the circuit corresponds to $P \bmod \mathbb{Z}_k$ where P is the number of $s-t$ paths in the graph, assuming the unconnected wires have input 0.

- Those that represent the internal structure of a funnel and are therefore fixed.
- Those that are fixed to be 0 because they represent the connection between two gates that cannot be connected due to the ordering of the wires.
- Those that correspond exactly to an edge in G_x and are therefore dependent only on the bit that represents that edge in the adjacency matrix of G_x because of the ordering imposed on the input wires of any structure.

It is routine to verify that the connections are sufficiently uniform, so that this is a first-order projection. (In order to make the uniformity argument more transparent, one can adopt a naming convention for the gates in the circuit, where the gates have names of the form (D, w) where D is a configuration of M and w is the name of the w^{th} gate in the copy of $C^*_{|V|}(k)$ for configuration D.) ■

See Figure 3 for an example of the transformation in Lemma 3.2.

Lemma 3.3. *For each fixed $p \in \{0, 1, ..., k-1\}$, there is a projection that takes as input a fan-in 2 circuit C consisting of* mod$k$ *gates (as in Lemma 3.2) and some input x, and produces as output two graphs G_1 and G_2 such that there is an isomorphism between the two if and only if $C(x) = p$.*

Proof. We first proceed by mapping any mod$k$ gate to the graph structure described in Lemma 3.1. This can be done via a projection since each structure is fixed. If one gate's output feeds into another gate's input, we connect the k outputs to the k inputs correspondingly. A naive approach would lead to a reduction that is not a projection, because checking if, say, gate h is the "first" or the "second" input to gate g would involve checking several bits of the adjacency matrix.

Under the construction in Lemma 3.2, however, for any funnel structure, we know exactly which input wire corresponds to any gate's output. This means that for any input vertex in the mod k graph, we know exactly which connection to look for in the circuit, meaning that each bit of the adjacency matrix will depend on at most one bit of the circuit's matrix. As a consequence, we know that all bits but one in each column are fixed to be 0, making this a very sparse matrix.

In order to create an instance of RGI, we create two copies of the graph described above; let's call them G_0 and G_1. Let the nodes corresponding to the input gates (in each graph) be $\{(g_i, j) : 1 \leq i \leq n, 0 \leq j \leq k-1\}$, and let the nodes corresponding to the output gates be $\{h, j) : 0 \leq j \leq k-1\}$. In G_0 we give color i to $(g_i, 0)$ and we give color $n+1$ to $(h, 0)$; in G_1 we give color i to (g_i, x_i) and we give color $n+1$ to (h, p). Note that the coloring given to G_0 yields a simulation of the circuit C on the all-zero input (in which case it is guaranteed to output 0), whereas the coloring given to G_1 yields a simulation of the circuit on input x. (Nodes that are in the gadgets corresponding to a given gate g can be given color g, in order to maintain rigidity.) As in [17], we can

see that the circuit outputs $p \in \mathbb{Z}_k$ if and only if G_0 and G_1 are isomorphic. It is also observed in [17] that both graphs are rigid. ■

Theorem 3.4. $\mathsf{DET} \leq_{\mathrm{m}}^{\mathsf{fop}} \mathsf{RGI}$

Proof. With the preceding two lemmas in hand, the rest of the proof proceeds precisely as in the proof of [17, Theorem 5.3] (which is credited there to van Melkebeek). A crucial ingredient of this proof is to make use of the set PGI, which is defined to be

$$\{((G,H),(I,J)) : (G,H) \in \mathsf{GI} \Leftrightarrow (I,J) \notin \mathsf{GI}\}.$$

(In our proof, the graphs G, H, I and J will all be rigid.) A tuple $((G,H),(I,J))$ in PGI is considered to encode the Boolean value 1 if $(G,H) \in \mathsf{GI}$, and will encode 0 if $(I,J) \in \mathsf{GI}$.

Lemma 3.3 shows how, for any function $f \in \#\mathsf{L}$, any number p (where we will be interested only in the case where p is prime), and any $k < p$, one can obtain a first-order projection q such that, for any string x, $q(x)$ is a pair (G,H) of rigid graphs with the property that G and H are isomorphic iff $f(x) \equiv k \bmod p$. We now want to create an instance of PGI, which means that we also want to have a pair of graphs (I,J) that are isomorphic iff $f(x) \not\equiv k \bmod p$. Note that, if p is prime, then $f(x) \not\equiv k \bmod p$ if and only if $(f(x)+(p-k))^{p-1} \equiv 1 \bmod p$. Note also that the function that maps $(x, 1^p)$ to $f(x)^{p-1}$ is also in $\#\mathsf{L}$. Thus the construction from Lemma 3.3 can be used to create the complete instance of PGI encoding whether or not $f(x) \equiv k \bmod p$.

As in [17, Theorem 5.3] for any problem $A \in \mathsf{DET}$, we observe that since A is NC^1-Turing reducible to the oracle L_f for some $f \in \#\mathsf{L}$, it is also NC^1-Turing reducible to the oracle

$$L'_f = \{(x,p,k) | p < a \log |x| \text{ and} f(x) \equiv k \bmod p\}$$

by Theorem 2.1. (Here, the constant a is chosen so that the product of the first n^a primes is larger than the numeric value of $f(x)$ for any x of length n.) The reduction presented in [17, Theorem 5.3] that produces an instance of RGI can actually be implemented as a first-order projection with no additional changes, where each query about $f(x)$ in the original NC^1-Turing reduction is replaced by a list of queries asking if $(x,p,k) \in L'_f$ for all small primes p and all $k < p$. Thus the only modification to the original construction that is required is provided by Lemmas 3.2 and 3.3. ■

Theorem 3.5. *[AH]* RGI *is many-one reducible to* MKTP *under projections.*

Proof. We consider the reduction from RGI to MKTP given in [7]. On input (G_0, G_1), the reduction uses t permutations $\{\pi_i\}$ and a complex binary string x of length t. The input is then mapped to the string $\pi_0(G_{x_0})\pi_1(G_{x_1})...\pi_i(G_{x_i})$, which is complex if and only if G_0 and G_1 are not isomorphic. We note that to obtain any bit of $\pi_i(G_{x_i})$, the reduction only relies on the bit of G_{x_i} that maps to it through the permutation π_i. That is, if $G_{x_i}(a,b)$ refers to the bit denoting if vertex a is adjacent to b, then $\pi_i(G_{x_i})(\pi_i(a), \pi_i(b)) = G_{x_i}(a,b)$. This is clearly a projection. ■

Our main result now follows directly from Theorems 3.4 and 3.5 .

Corollary 3.5.1. MKTP *is hard for* DET *under many-one projections.*

4 Conclusions and Open Questions

It is interesting to note that the simplicity of this reduction relies on enforcing an inefficient structure of the output, meaning that a large majority of the bits are fixed. In Lemmas 3.3 and 3.4, we turn a graph into a circuit and then into a graph isomorphism problem. In both proofs, in order to maintain that the reduction is projection, we require establishing an ordering of vertices. Using this ordering means that most bits are fixed as zeros and are mainly used just for formatting purposes.

Given that MCSP and MKTP are related problems, we believe our result should follow through for MCSP however we currently have no hardness results for MCSP with relation to DET.

Acknowledgments

The work of the second and third authors was performed while they were participants in the 2019 DIMACS REU program [11], and was supported by NSF grant CCF-1852215. They also thank Lazaros Gallos and Parker Hund for organizing the REU program's activities. The first author is supported in part by NSF grant CCF-1909216.

References

[1] Manindra Agrawal, Eric Allender, and Steven Rudich. Reductions in circuit complexity: An isomorphism theorem and a gap theorem. *Journal of Computer and System Sciences*, 57(2):127–143, 1998.

[2] Eric Allender, José L. Balcázar, and Neil Immerman. A first-order isomorphism theorem. *SIAM Journal on Computing*, 26(2):557–567, 1997.

[3] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35(6):1467–1493, June 2006.

[4] Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. *Information and Computation*, 256:2–8, 2017. Special issue for MFCS '14.

[5] Eric Allender, John Gouwar, Shuichi Hirahara, and Caleb Robelle. Cryptographic hardness under projections for time-bounded Kolmogorov complexity. Manuscript, 2020.

[6] Eric Allender, Joshua A Grochow, Dieter Van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum circuit size, graph isomorphism, and related problems. *SIAM Journal on Computing*, 47(4):1339–1372, 2018.

[7] Eric Allender and Shuichi Hirahara. New insights on the (non-) hardness of circuit minimization and related problems. *ACM Transactions on Computation Theory (TOCT)*, 11(4):1–27, 2019.

[8] Eric Allender and Mitsunori Ogihara. Relationships among PL, #L, and the determinant. *RAIRO - Theoretical Informatics and Applications*, 30:1–21, 01 1996.

[9] A. Chiu, G.I. Davida, and B. Litow. Division in logspace-uniform NC^1. *RAIRO Theoretical Informatics and Applications*, 35:259–276, 2001.

[10] Bin Fu. Hardness of sparse sets and minimal circuit size problem. In *Proc. Computing and Combinatorics - 26th International Conference (COCOON)*, volume 12273 of *Lecture Notes in Computer Science*, pages 484–495. Springer, 2020.

[11] Azucena Garvia-Bosshard and Amulya Musipatla. A note on the relationship between the determinant and time-bounded Kolmogorov Complexity. Manuscript, see reu.dimacs.rutgers.edu/2019/projects.php, 2019.

[12] William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65:695–716, 2002.

[13] Neil Immerman. *Descriptive complexity*. Springer Science & Business Media, 2012.

[14] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 73–79, New York, NY, USA, 2000. ACM.

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

[16] Cody D. Murray and R. Ryan Williams. On the (non) NP-hardness of computing circuit complexity. *Theory of Computing*, 13(4):1–22, 2017.

[17] Jacobo Torán. On the hardness of graph isomorphism. *SIAM Journal on Computing*, 33(5):1093–1108, 2004.

[18] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag New York Inc., 1999.