NP-Hardness of Circuit Minimization for Multi-Output Functions

Rahul Ilango
Massachusetts Institute of Technology, Cambridge, MA, US
rilango@mit.edu

Bruno Loff
University of Porto and INESC-Tec, Portugal
bruno.loff@gmail.com

Igor C. Oliveira
University of Warwick, Coventry, UK
igor.oliveira@warwick.ac.uk

Abstract

Can we design efficient algorithms for finding fast algorithms? This question is captured by various circuit minimization problems, and algorithms for the corresponding tasks have significant practical applications. Following the work of Cook and Levin in the early 1970s, a central question is whether minimizing the circuit size of an explicitly given function is NP-complete. While this is known to hold in restricted models such as DNFs, making progress with respect to more expressive classes of circuits has been elusive.

In this work, we establish the first NP-hardness result for circuit minimization of total functions in the setting of general (unrestricted) Boolean circuits. More precisely, we show that computing the minimum circuit size of a given multi-output Boolean function $f\colon \{0,1\}^n \to \{0,1\}^m$ is NP-hard under many-one polynomial-time randomized reductions. Our argument builds on a simpler NP-hardness proof for the circuit minimization problem for (single-output) Boolean functions under an extended set of generators.

Complementing these results, we investigate the computational hardness of minimizing communication. We establish that several variants of this problem are NP-hard under deterministic reductions. In particular, unless $\mathsf{P} = \mathsf{NP}$, no polynomial-time computable function can approximate the deterministic two-party communication complexity of a partial Boolean function up to a polynomial. This has consequences for the class of structural results that one might hope to show about the communication complexity of partial functions.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases MCSP, circuit minimization, communication complexity, Boolean circuit

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.22

Related Version This paper first appeared as an ECCC report at `https://eccc.weizmann.ac.il/report/2020/021/`.

Funding *Rahul Ilango*: This work was supported in part by an Akamai Presidential Fellowship.
Bruno Loff: The author was the recipient of FCT postdoctoral grant number SFRH/BPD/116010/2016. This work is financed by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020.
Igor C. Oliveira: This work was supported in part by a Royal Society University Research Fellowship.

Acknowledgements Igor C. Oliveira would like to thank Ján Pich and Rahul Santhanam for discussions on the complexity of circuit minimization for partial Boolean functions. Bruno Loff would like to thank Eric Allender for posing a question that inspired some of the results in this work, and the Higher School of Economics for inviting him to the conference "Randomness, Information, Complexity", in honor of Alexander Shen and Nikolay Vereshchagin's 60th birthday, where said question

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 22; pp. 22:1–22:36

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

was asked. Rahul Ilango would like to thank Eric Allender, Marco Carmosino, Russell Impagliazzo, Michael Saks, Rahul Santhanam, and Ryan Williams for their encouragement, suggestions, and helpful discussions.

1 Introduction

The Minimum Circuit Size Problem (MCSP) asks for the size (number of gates) of the smallest Boolean circuit that computes a given Boolean function $f\colon \{0,1\}^m \to \{0,1\}$, where f is represented as a string of length $n = 2^m$. Researchers have investigated this problem and its variants from several angles since the early stages of complexity theory (see [75] for some historical perspective). In particular, over the last two decades there has been a significant interest in understanding the computational hardness of circuit minimization. This is motivated in part by the discovery of connections between this problem and a variety of areas, including complexity theory [43], learning theory [17], cryptography and circuit complexity [70], and proof complexity (see e.g. [51, Part VIII]). In addition, Boolean circuit minimization is of high practical relevance, and a number of textbooks and monographs have been written about heuristics and other applied aspects of this problem (cf. [59, 73, 29]).[1]

Despite considerable efforts to understand the computational complexity of circuit minimization, its NP-hardness status has remained wide open. While there is strong evidence that finding optimal circuits is intractable (see Section 1.3), some researchers have suggested that circuit minimization problems such as MCSP might be NP-intermediate (that is, neither in P nor NP-complete). There is a vast literature on MCSP and this question, and we review the references more directly related to our work in Sections 1.2 and 1.3 below.

1.1 Results

We investigate the natural variant of MCSP where the input function $f\colon \{0,1\}^n \to \{0,1\}^m$ is allowed to have multiple output bits. Our main contribution is a proof that the circuit minimization problem for such multi-output functions is NP-hard with respect to randomized reductions. This is the first NP-hardness result for the circuit minimization of total functions that holds with respect to the class of general (unrestricted) Boolean circuits. Previous NP-hardness results for total functions were known when the computational model is considerably restricted, for instance with respect to DNFs [55, 58] (also known as two-level minimization; see e.g. [77]) and DNFs extended with parity gates at the bottom layer [33].

There are well-known connections between computation and communication (see e.g. [53]). In the second part of this work, we explore the complexity of minimizing communication cost with respect to deterministic protocols, a question that dates back to Yao's seminal work on communication complexity [80, Section 4, Problem E]. Among other contributions, we establish the first NP-hardness result for this model, in the setting that the input communication problem is described by a partial Boolean matrix in $\{0,1,*\}^{n\times n}$. In a remarkable paper, Kushilevitz and Weinreb [54] had previously established the intractability of this problem over total Boolean matrices, but their techniques require cryptographic assumptions. Our proof extends to a stronger hardness of approximation result, and this has interesting consequences for communication complexity.

We now describe in more detail each of our NP-hardness results and their implications.

[1] The problem is also referred to as Boolean function or Boolean algebra minimization, logic synthesis, circuit synthesis, logic minimization, circuit optimization, or multi-level minimization in different communities.

1.1.1 NP-hardness of circuit minimization

First, let us fix some notation and terminology. A Boolean circuit consists of fan-in two AND gates, fan-in two OR gates, and NOT gates. The input gates are labelled by variables $x_1, \ldots, x_n$. We measure the size of a Boolean circuit C, denoted $|C|$, using the number of AND, OR, and NOT gates in the circuit. (While this is the convention adopted here, our techniques are robust to modifications of the circuit size measure and of the gate types in the circuit.)

We now introduce the circuit minimization problems considered in our work.

Multi-output Boolean functions. In practice, one is often interested in computing Boolean functions f that have multiple output bits. Indeed, the vast majority of computations, such as addition, multiplication, encryption schemes, error-correcting codes, solutions to search problems, etc., have multiple outputs. In this case, MCSP can give a misleading picture of the circuit complexity of f. For example, if f is the problem of multiplying two $n \times n$ matrices over $\mathbb{F}_2$, then computing any specific choice of one of the n^2 output bits of f requires a circuit with $\Omega(n)$ gates, which seems to suggest a lower bound of $\Omega(n^3)$ on matrix multiplication. Of course, it is widely-known that one can beat the $O(n^3)$ time algorithm for matrix multiplication quite significantly!

This motivates the study of circuit minimization for multi-output Boolean functions. We begin by fixing our notion of multi-output computation. The *components* of a multi-output Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ are the single-output functions that compute the ith output bit of f for $i \in [m]$. We say a Boolean circuit C computes a multi-output Boolean function f if for each component f_i of f, there is a gate or input wire in C that computes f_i.

▶ **Definition 1.** Multi-MCSP *is defined as follows:*

- Input. *Positive integers n, m, and s represented in unary, and a (multi-output) Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ represented by a string of length $m \cdot 2^n$.*
- Output. *The input is accepted if and only if there exists a Boolean circuit C of size at most s that computes f.*

Note that Multi-MCSP is in NP,[2] and that this problem is at least as hard as MCSP.

Partial Boolean functions. Despite the fundamental nature of MCSP, in several natural scenarios arising from practical or theoretical considerations one does not really care about the output of a Boolean function on *every* string of length n.[3] For instance, for a problem on graphs, one might be interested only in graphs that are planar. In such situations, it becomes relevant to understand the complexity of the corresponding function on a subset of inputs. This is more naturally captured by a different formulation of MCSP, where irrelevant or inessential inputs of the Boolean function are omitted. In other words, while MCSP refers to total Boolean functions, it is equally natural to consider circuit minimization over partial Boolean functions.

▶ **Definition 2.** Partial-MCSP *is defined as follows:*

- Input. *A positive integer n represented in unary, a collection $\mathcal{P}$ of pairs (x_i, b_i), where $x_i \in \{0,1\}^n$ and $b_i \in \{0,1\}$, and a positive integer s.*
- Output. *The input is accepted if and only if there exists a Boolean circuit C of size at most s such that $C(x_i) = b_i$ for every pair $(x_i, b_i) \in \mathcal{P}$.*

Note that, like Multi-MCSP, Partial-MCSP is in NP and is at least at hard as MCSP.

[2] If the parameter s is large then the answer is trivial.
[3] Such inputs are associated with "don't care" values in the applied literature.

Boolean functions over an arbitrary set of generators. We also consider circuit minimization for (single-output, total) Boolean functions $f\colon \{0,1\}^m \to \{0,1\}$ under an arbitrary set of *generators*.

To explain, let V be a finite set, called the *ground set*, and $\mathcal{B} = \{B_i\}_{i\in[m]}$ a family of nonempty sets $B_i \subseteq V$ called *generators*, and let $A \subseteq V$. Then we let $D(A \,|\, \mathcal{B})$ denote the minimum number of unions, intersections, and complements that are required to construct A from the sets in $\mathcal{B}$. More precisely, the complement of a set $U \subseteq V$ is defined as $V \setminus U$, and we represent a construction of A from $\mathcal{B}$ as a sequence $B_1, \ldots, B_m, E_1, \ldots, E_\ell$ of sets in V such that $E_\ell = A$ and each set E_j is either the union or intersection of two previously generated sets, or the complement of a previously generated set. We assume for convenience that $D(B \,|\, \mathcal{B}) = 0$ if $B \in \mathcal{B}$. We refer to $D(A \,|\, \mathcal{B})$ as the *discrete complexity*[4] of A with respect to $\mathcal{B}$.

It may then be seen the minimum number of AND, OR and NOT gates needed for a Boolean circuit to compute a Boolean function $f : \{0,1\}^m \to \{0,1\}$ is exactly $D(f^{-1}(1) \mid \mathcal{B})$, where $V = \{0,1\}^m$, and $\mathcal{B} = \{x_1, \ldots, x_m\} \subseteq \{0,1\}^m$ contains the input variables $x_1 \ldots, x_m$, seen as subsets of $\{0,1\}^m$ (i.e.. x_i is the set of strings $w \in \{0,1\}^m$ such that $w_i = 1$). So computing the discrete complexity $D(A \mid \mathcal{B})$, for given A and $\mathcal{B}$, generalizes the task of computing the minimum circuit size, by allowing for the consideration of generator sets $\mathcal{B}$ other than $\{x_1, \ldots, x_m\}$.

Another possibility is to consider circuit complexity over a family $\mathcal{B}$ of generators over a ground set V other than the set of binary strings. For example, the *graph complexity* (see [42] §1.7) of a given bipartite graph $G = (U \times V, E)$, with $E \subseteq U \times V$, is $D(E \mid \mathcal{B})$, where $\mathcal{B}$ contains all product sets $A \times B$ with $A \subseteq U$ and $B \subseteq V$. So computing the discrete complexity $D(A \mid \mathcal{B})$, for given a given A and $\mathcal{B}$, also generalizes the task of computing graph complexity.

In analogy to MCSP, we now introduce the Minimum Discrete Complexity Problem (MDCP).

▶ **Definition 3.** MDCP *is defined as follows:*

- Input. *A positive integer n represented in unary describing the size of the ground set $V = [n]$, a target set $A \subseteq V$, a family $\mathcal{B}$ of nonempty subsets of V, and an integer s.*
- Output. *The input is accepted if and only if $D(A \,|\, \mathcal{B}) \leq s$.*

It is easy to see that MDCP is in NP and that it is more general than MCSP. Our hardness result for MDCP, discussed below, also holds under the assumption that the ground set V is a hypercube $\{0,1\}^m$ and that the collection $\mathcal{B}$ contains the sets generated by $x_1, \ldots, x_m$.

NP-hardness of MDCP, Partial-MCSP, and Multi-MCSP. Note that establishing the hardness of these problems is *necessary* before proving hardness of MCSP. This is because instances of MCSP can be easily converted into instances of each one them.

By adapting techniques from [38], it is not hard to show that MDCP is NP-hard under randomized reductions. Moreover, this result almost immediately implies the NP-hardness of Partial-MCSP, since there is a simple way of converting the circuit minimization of Boolean functions under an arbitrary set of generators into a problem about partial Boolean functions (see Section 3.3). We note that previous works in learning theory [30, 1] implicitly contain a

[4] This general setting was already considered in [68], see also §1.7.2 of Jukna's book [42]. By Stone's representation theorem for Boolean algebras, discrete complexity can be seen as the investigation of circuit complexity with respect to an arbitrary Boolean algebra.

substantially simpler proof that Partial-MCSP is NP-hard, even under *deterministic* reductions. Unfortunately, there is strong evidence that this simpler proof has limitations. For instance, if it could be adapted to show deterministic NP-hardness of MDCP for instances extending the hypercube, then EXP $\neq$ ZPP. This follows from an argument analogous to [62].

Proving the NP-hardness of circuit minimization for total functions seems to require a more sophisticated argument. We are able to combine the technique that we use to show NP-hardness of MDCP and Partial-MCSP with several new ideas to establish the following result.

▶ **Theorem 4.** Multi-MCSP *is* NP*-hard under many-one randomized polynomial time reductions.*

We explain the insights leading to the proof of Theorem 4 in Section 1.2. The final argument is not overly technical, though it took us considerable time to discover the right conceptual ingredients. Could it be the case that MCSP admits a randomized NP-hardness that relies on a clever modification of existing techniques? As far as we know, the existence of a "standard" randomized reduction would not imply a breakthrough such as a complexity class separation.

The hardness results mentioned above come with certain features and consequences that might be of independent interest. We discuss them next.

Search-to-decision reductions for circuit minimization. The formula satisfiability problem (SAT) admits a well-known *search-to-decision* reduction. In other words, if one can easily check if a formula is satisfiable, then it is not much harder to find a satisfiable assignment, whenever one exists. As a consequence of the NP-completeness of SAT, all NP-complete problems must have search-to-decision reductions. On the other hand, designing a search-to-decision reduction for MCSP is open. We refer to [17, 32] for recent developments in this direction which can be interpreted as weak search-to-decision reduction for MCSP.

A corollary of Theorem 4 is that the *search version* of Multi-MCSP and its *decision version* are computationally equivalent under polynomial-time randomized reductions. Inspired by this consequence, we further investigate this phenomenon, and in Section 4.3 we describe a natural *deterministic* search-to-decision reduction for Multi-MCSP. Additionally, we show in Section 3.4 that Partial-MCSP has a simple *deterministic* search-to-decision reduction. These search-to-decision reductions rely on ideas employed in our NP-hardness proofs. This suggests that establishing the NP-hardness of MCSP and obtaining a corresponding search-to-decision reduction might be closely related tasks.

Satisfiability versus Learning. It is known that the appropriate *average-case* formulation[5] of Partial-MCSP captures the complexity of learning general Boolean circuits under the uniform distribution using random examples. This follows by a combination of the ideas in [78] and [15], and for completeness we provide a proof of this equivalence in Appendix A. Consequently, we can base the hardness of learning Boolean circuits on the assumption NP $\not\subseteq$ RP if and only if the existence of an efficient algorithm for average-case Partial-MCSP implies the existence of an efficient (worst-case) algorithm for Partial-MCSP (see Appendix A for details). We refer to [14, 9] for more information about learning algorithms and average-case versus worst-case assumptions.

[5] Here one needs to *distinguish*, for a random choice of polynomially many inputs x_i, whether the labels b_i are randomly generated or are consistent with a fixed circuit of size at most s. We say that an algorithm solves Partial-MCSP on average (for a given choice of the size parameter s) if the distinguishing probability of this experiment is noticeable on every circuit of size at most s.

It seems interesting that the *worst-case* and *average-case* complexities of a natural problem connect to *satisfiability* and *learning*, respectively. Further investigating these relations might be a fruitful research direction.

1.1.2 NP-hardness of communication minimization

Let $f\colon [n] \times [n] \to \{0,1,*\}$ be a partial Boolean function. The two-party communication problem of computing f is defined as follows. Alice is given $x \in [n]$, Bob is given $y \in [n]$, and they are promised that $f(x,y)$ is defined. Their goal is to exchange the minimum number of bits in order to compute $f(x,y)$. We refer to Section 5.1 for definitions, and to [52] for more information about communication complexity in general.

Note that many communication problems of interest are captured by partial Boolean functions, such as Gap-Hamming-Distance (see e.g. [18]) and Unique-Disjointness (cf. [28]).

We are primarily interested in the computational hardness of estimating the communication cost of optimal deterministic protocols for a given function $f\colon [n] \times [n] \to \{0,1,*\}$. This function will be naturally represented by an $n \times n$ matrix $M \in \{0,1,*\}^{n \times n}$, so that $M[x,y] = f(x,y) \in \{0,1\}$ if $f(x,y)$ is defined over the input pair (x,y), and $M[x,y] = *$ otherwise. In order to state our main result in the context of communication complexity, we introduce the Minimum Communication Complexity Problem for partial Boolean functions.

▶ **Definition 5.** Partial-MCCP *is defined as follows:*

- Input. *A positive integer n represented in unary, a matrix $M \in \{0,1,*\}^{n \times n}$ representing a partial Boolean function $f : [n] \times [n] \to \{0,1,*\}$, and a positive integer s.*
- Output. *The input is accepted if and only if there exists a two-party deterministic protocol for computing f whose communication cost is at most s.*

Note that Partial-MCCP is in NP, as the full communication matrix is represented as part of the input, and any non-trivial protocol for f can be described by a string of length polynomial in n.[6]

We prove that computing communication complexity and several related measures is NP-hard under deterministic reductions.

▶ **Theorem 6.** Partial-MCCP *is* NP*-hard under many-one deterministic polynomial time reductions. Furthermore, analogous results hold with respect to leaf complexity, partition number, cover number, and the smallest number of nodes in a* DAG*-like protocol of a partial Boolean function.*

Our hardness results are actually significantly stronger: we show that all these complexity measures are *hard to approximate* in the context of partial Boolean functions. The NP-hardness of approximating communication cost will be discussed in more detail below. The remaining four measures – leaf complexity, partition number, cover number, and the smallest number of nodes in a DAG-like protocol – are NP-hard to approximate up to a factor of $n^{1-\varepsilon}$ (for any fixed $\varepsilon > 0$), which is essentially optimal.

We note that in the setting of NP-hardness results for MCSP with respect to restricted classes of circuits, such as DNF [4] and DNF-XOR [33], a successful strategy has been to first establish hardness of a variant of the problem where the input is the full truth table of a given *partial* function. This is then followed by a reduction to the case of total functions. We leave as an open problem whether Partial-MCCP can be reduced to minimizing communication complexity of total matrices.

[6] In contrast to this definition, note that the NP-hardness result for circuit minimization of partial Boolean functions refers to functions succinctly described by a list of its $\{0,1\}$-valued entries.

Hardness of approximating communication cost and its consequences. Our complexity-theoretic results have implications for the theory of communication complexity. In order to make the discussion more concrete, we first consider an example.

The *log-rank conjecture* of Lovász and Saks [56] states that the deterministic communication complexity of a total Boolean function $f : [n] \times [n] \to \{0,1\}$, denoted $\mathsf{D}(f)$, is characterized up to a polynomial by the logarithm of the rank (over $\mathbb{R}$) of the corresponding communication matrix M_f. It is a basic, well-known fact that $\mathsf{D}(f) \geq \log(\mathrm{rk}\, M_f)$ (cf. [53]). If we do not allow for a super-constant additive error term, the log-rank conjecture says that there is a universal constant $c > 0$ such that $\frac{1}{c}\mathsf{D}(f)^{1/c} - c \leq \log(\mathrm{rk}\, M_f)$ for every total function f.

In the context of our work, the significance of this conjecture is that, if true, it would provide an algorithm (compute the rank and take the logarithm) for approximating the deterministic communication complexity of a given *total* Boolean function. This algorithm runs in time polynomial in the communication matrix, which means that computing such an approximation of $\mathsf{D}(f)$ is not NP-complete, unless $\mathsf{P} = \mathsf{NP}$. While the status of the log-rank conjecture remains unclear,[7] there may be other algebraic or analytic quantities that approximately capture communication cost.

Similar considerations can be made about the communication complexity of *partial* Boolean functions. More generally, we would like it if there were *some* polynomial-time computable function r that would estimate the communication complexity up to a polynomial, in the sense that for some constant $c > 0$ and for every large enough n, any partial function $f\colon [n] \times [n] \to \{0,1,*\}$ satisfies

$$\frac{1}{c} \cdot \mathsf{D}(f)^{1/c} - c \;\leq\; r(M_f) \;\leq\; c \cdot \mathsf{D}(f)^c + c.$$

However, we are able to prove a strong negative result in this direction. We establish that there is no function r as we just described, under the assumption that $\mathsf{P} \neq \mathsf{NP}$. This result is a consequence of the techniques behind the proof of Theorem 6, which also imply certain hardness of approximating results for communication complexity. In more detail, we prove that it is NP-hard to approximate $\mathsf{D}(f)$ up to a sub-exponential function of $\mathsf{D}(f)$. (This result makes sense because $\mathsf{D}(f)$ might be a constant independent of n.) Additionally, we prove that it is NP-hard to estimate $\mathsf{D}(f)$ with an additive error term of $(1 - \Omega(1)) \log n$, or within a fixed but arbitrary constant factor. We refer to Section 5.3 for the precise statements.

1.2 Techniques

1.2.1 Circuit complexity

The proof of Theorem 4 builds on insights from several works on the complexity of circuit minimization, including the references [4], [62] [33], and [38].

In [4], the authors provide a new proof that DNF-MCSP is NP-hard, i.e., the variant of MCSP where the circuit complexity of the input function is measured with respect to DNF size. Their proof employs a deterministic reduction from a set cover problem. More precisely, in the r-Bounded Set Cover Problem (cf. [26]), we are given a collection $\mathcal{S}$ of subsets of $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$, and the goal is to cover $[n]$ with the minimum number of such sets. We are also promised that each set $S \in \mathcal{S}$ has size at most r. The argument of [4] relies on the NP-hardness of solving this problem on certain structured inputs.

[7] In a recent paper, Chattopadhyay et al. [19] (see also [74, 8]) showed that an analogous conjecture for randomized communication complexity is false.

Extending the techniques of [4], a more recent work [33] established that (DNF-XOR)-MCSP is NP-hard under deterministic reductions. Crucial for the argument of [33] to go through is the stronger result proved by [76] showing that r-Bounded Set Cover is NP-hard even to *approximate* (the proof in [76] relies on ideas from [23]). In particular, for any constant-factor approximation parameter α, there exists a parameter r independent of n such that computing an α-approximation of the optimal cover size in r-Bounded Set Cover is NP-hard. Intuitively, this hardness of approximation result provides more flexibility when implementing a reduction from a cover problem to a circuit minimization problem.

Note that the results discussed above rely on the weakness of the circuit classes (DNF and DNF-XOR) to establish the NP-hardness of the corresponding circuit minimization problems. In particular, structural properties of these low-depth circuits are explored in crucial ways. On the other hand, hardly anything is known about the limitations of *unrestricted* Boolean circuits. For instance, while exponential lower bounds are known against DNF-XOR circuits [21], the strongest known explicit lower bounds against general Boolean circuits are of the form cn for a small constant c (cf. [40, 25]). Perhaps this explains in part why many researchers have been pessimistic about the possibility of extending such techniques to show NP-hardness of circuit minimization for more expressive classes of Boolean circuits.

Moreover, some results (see [62], [36], and [38, Appendix B]) strongly suggest that designing *deterministic* reductions for circuit minimization problems that refer to general circuits might be a challenging task. Formally, [62] proved that if MCSP is NP-hard under polynomial-time deterministic many-one reductions, then $\mathsf{EXP} \neq \mathsf{ZPP}$. In other words, it is not possible to design a deterministic reduction showing NP-hardness of MCSP without a breakthrough in complexity theory. While it is not immediately clear to us if this connection applies to problems such as Multi-MCSP, given these results it is more natural to focus on *randomized* reductions.

In sharp contrast to previous works, which have considered the NP-hardness of circuit minimization for *restricted* circuit classes, [38] has recently established the NP-hardness of MCSP for *unrestricted circuits with oracle gates*. In more detail, let MOCSP (Minimum Oracle Circuit Size Problem) be the problem where we are given a parameter s and total functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g_1, \ldots, g_t\colon \{0,1\}^m \to \{0,1\}$, and the goal is to decide if f can be computed by a circuit with at most s AND, OR, NOT, and ORACLE gates, where each ORACLE gate can compute any one of the functions g_i. Perhaps surprisingly, [38] was able to exploit the presence of arbitrary oracle gates to show that MOCSP is NP-hard under randomized reductions.

While computations with oracles might behave very differently than normal computations,[8] this result gave us some optimism, and it was the starting point of our investigation. Our techniques build on the reduction of [38], which relies in part on ideas from [4] and [33].

Similarly to [33] and [38], we will also employ the NP-hardness of approximating r-Bounded Set Cover. Our proofs are technically not very involved, and we focus here on some key conceptual ideas. We refer to Sections 3 and 4 for details.

First, we sketch the proof that MDCP is NP-hard. Building on this argument, we discuss the NP-hardness of Multi-MCSP under many-one polynomial-time randomized reductions (Theorem 4).

[8] For instance, it is well known that there exist oracles A and B such that $\mathsf{P}^A \neq \mathsf{NP}^A$ and $\mathsf{P}^B = \mathsf{NP}^B$, respectively [10].

Hardness of circuit minimization under arbitrary generators (MDCP). We are given a collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ of sets $S_i \subseteq [n]$, where each set S_i has size at most r. The goal is to compute from $\mathcal{S}$ and $[n]$ an input instance of MDCP whose complexity approximates the cover complexity of $[n]$ with respect to $\mathcal{S}$.

One can view the cover complexity of $[n]$ with respect to $\mathcal{S}$ as measuring the complexity of "generating" the set $[n]$ from the sets in $\mathcal{S}$ using only union operations. In more detail, let $\mathsf{cover}([n], \mathcal{S})$ denote the smallest possible number of sets in $\mathcal{S}$ required to cover $[n]$. It is easy to see that, if $\mathsf{cover}([n], \mathcal{S}) \leq \ell$, then $[n]$ can be generated from the sets in $\mathcal{S}$ using at most ℓ fan-in-two union operations. Similarly, it is not hard to see that if $[n]$ can be generated using ℓ fan-in-two union operations when starting from sets in $\mathcal{S}$, then a trivial upper bound is that $\mathsf{cover}([n], \mathcal{S}) \leq 2\ell$. In other words, the minimum number of fan-in-two union operations necessary to generate $[n]$ from sets in $\mathcal{S}$ gives a 2-approximation for $\mathsf{cover}([n], \mathcal{S})$.

The discussion above shows that $\mathsf{cover}([n], \mathcal{S}) = \Theta(D_{\cup}([n] \,|\, \mathcal{S}))$, where $D_{\mathcal{O}}(A \,|\, \mathcal{B})$ denotes the minimum number of $\mathcal{O}$-operations sufficient to generate A from $\mathcal{B}$ when only set operations in $\mathcal{O}$ are allowed. It is not hard to see that intersections are not helpful when generating the entire "ground set" $[n]$. More precisely, one can easily argue that $\mathsf{cover}([n], \mathcal{S}) = \Theta(D_{\{\cup,\cap\}}([n] \,|\, \mathcal{S}))$ by simply replacing intersections by unions. This simple argument and the hardness of approximating set cover can be used to show that, under such a generalization of circuit complexity and when allowing only monotone operations (unions and intersections), it is NP-hard to compute circuit complexity.

We consider next the case of non-monotone operations, which are also present in discrete complexity. Note that when negations (complementations) are allowed, the argument sketched above completely breaks down: by taking the complement of a single set in $\mathcal{S}$, one might be able to cover most of $[n]$. In order to handle this issue, a new ingredient seems necessary. Instead of translating the cover problem given by $([n], \mathcal{S})$ into a direct instance $D([n] \,|\, \mathcal{S})$, we employ a more involved construction based on an idea from [38]. (Intuitively, the construction inoculates the power of negations.) In more detail, we map each element $i \in [n]$ to a block B_i of size n^2 inside the larger ground set $V = [n^3]$. This induces a partition of V into $B_1, \ldots, B_n$. A set $S_j \in \mathcal{S} = \{S_1, \ldots, S_t\}$ is mapped to the union of the blocks B_i with $i \in S_j$. We now consider a certain set $A \subseteq [n^3]$ with nice properties, and use the previously described map to create an instance $D(A \,|\, W_{S_1}, \ldots, W_{S_t})$, where each W_{S_j} is the intersection of A with the union of the sets B_i with $i \in S_j$. (By construction, a cover of $[n]$ by sets in $\mathcal{S}$ provides a way to write any set A as a union of its corresponding sets W_{S_j}.) For a *random set* A (and for this reason we can only get a randomized reduction), we are able to show that with high probability the discrete complexity $D(A \,|\, W_{S_1}, \ldots, W_{S_t})$ *approximates* the cover complexity $\mathsf{cover}([n], \mathcal{S})$. Roughly speaking, this is true because a random set A is so complex that taking negations of the sets W_{S_j} does not really help to considerably reduce its complexity, and the best way to generate A is essentially to use a cover of $[n]$ by sets in $\mathcal{S}$ as a recipe. This allows us to prove that computing discrete complexity is NP-hard under randomized reductions.

Hardness of circuit minimization for multi-output Boolean functions. Next, we try to adapt the NP-hardness result for discrete complexity to Multi-MCSP. Loosely summarizing, the discrete complexity reduction works as follows (continuing with the notation from before):

(1) Randomly convert a set cover problem $([n], \mathcal{S})$ into an essentially equivalent set cover problem $(A, W_{S_1}, \ldots, W_{S_t})$ on a larger ground set in a way that (with high probability) the randomness inoculates against there being way of building A from $W_{S_1}, \ldots, W_{S_t}$ that is significantly better than the naive method of unioning over an optimal set cover.

(2) Compute how hard it is to build A from $W_{S_1}, \ldots, W_{S_t}$ by outputting $D(A \,|\, W_{S_1}, \ldots, W_{S_t})$.

We translate each of these two steps into a Multi-MCSP version separately. For the first step, we translate the sets $(A, W_{S_1}, \ldots, W_{S_t})$ into the truth tables $(T, T_{S_1}, \ldots, T_{S_t})$ corresponding to the sets' characteristic functions, and then replace the notion of building a set from other sets by the notion of computing a function on some input given the values of other functions on that input. In detail, the Multi-MCSP version of Step (1) becomes

(1') Randomly convert a set cover problem $([n], \mathcal{S})$ into an essentially equivalent set cover problem $(A, W_{S_1}, \ldots, W_{S_t})$ on a larger ground set in a way that (w.h.p.) the randomness inoculates against the existence of a circuit C satisfying $T(x) = C(x, T_{S_1}(x), \ldots, T_{S_t}(x))$ that is significantly smaller than the naive method of computing $\bigvee_{S \in \mathcal{S}_0} T_S(x)$ where $\mathcal{S}_0 \subseteq \mathcal{S}$ is an optimal cover.

The main technical challenge for the Multi-MCSP reduction comes from the lack of a simple translation for Step (2). To some degree, this is because there is a "type mismatch" between the problems of computing discrete complexity, where you have a notion of computing "from," and the problem of Multi-MCSP, where there is no notion of computing "from," only a notion of computing "in addition to." Perhaps the closest Multi-MCSP analogue to how hard it is to compute T "from" $T_{S_1}, \ldots, T_{S_t}$ is the quantity

$$\Delta \stackrel{\text{def}}{=} \mathsf{CC}(T \bullet T_{S_1} \bullet \cdots \bullet T_{S_t}) - \mathsf{CC}(T_{S_1} \bullet \cdots \bullet T_{S_t})$$

where the notation $f_1 \bullet \ldots \bullet f_k$ denotes the multi-output function whose components are the functions $f_1, \ldots, f_k$. Informally, the quantity Δ corresponds to how much harder is it to compute T along with $T_{S_1}, \ldots, T_{S_t}$ than it is without T.[9]

However, this quantity does not really compute what we want it to compute. For example, if there is an optimal circuit for computing $T_{S_1} \bullet \cdots \bullet T_{S_t}$ that also computes T at some gate (which might be possible), then $\Delta = 0$. But a solution to a non-trivial set cover problem is never zero! One might hope that we could use randomness again to inoculate against these possibilities, but we have not yet figured out how to do so.

Our key idea for overcoming this barrier is to add additional output functions in order to force $T_{S_1}, \ldots, T_{S_t}$ to be computed in a way such that (with high probability) none of the gates used for computing $T_{S_1}, \ldots, T_{S_t}$ compute T or even help very much in computing T. These new outputs will correspond to the functions we want computed "along the way" to computing $T_{S_1}, \ldots, T_{S_t}$.

The actual implementation of this idea is rather subtle, but here is an, admittedly sketchy, outline. Let D be the circuit that computes $T_{S_1} \bullet \cdots \bullet T_{S_t}$ by just computing each of these functions individually via their naive DNF formula. Our random choice of T can be shown to ensure that none of the functions computed by gates in D "help too much in computing T." Next, define the Evaluation Function induced by D, denoted Eval-D, to be the multi-output function whose components are all those functions which are either computed by a gate in D or an input wire in D. Finally, the Multi-MCSP version of Step (2) will be

(2') Compute how hard it is to compute T at some input from the values of $T_{S_1}, \ldots, T_{S_t}$ at that input by outputting $\Delta' \stackrel{\text{def}}{=} \mathsf{CC}(T \bullet \text{Eval-}D) - \mathsf{CC}(\text{Eval-}D)$.

Since any circuit for computing Eval-D can be converted into a circuit for $T \bullet$ Eval-D using at most t gates (since $T = T_{S_1} \vee \cdots \vee T_{S_t}$), the parameters in our reduction can be set so that the overwhelming number of gates in an optimal circuit for $T \bullet$ Eval-D are functions that are computed in D which we know do not "help too much in computing T." We can

[9] The above definition of Δ brings to mind the chain rule for Kolmogorov complexity $K(x \mid y) = K(xy) - K(y) + O(1)$, so one may think intuitively of Δ as measuring the "complexity", or "entropy", of T given $T_{S_1} \ldots T_{S_t}$.

then show that the quantity Δ' approximates how hard it is to compute T on some input given the values of $T_{S_1}, \ldots, T_{S_t}$ on that input, which in turn, by Step (1'), approximates the size of an optimal cover in $(n, \mathcal{S})$.

1.2.2 Communication complexity

The intractability of computing deterministic communication complexity is known under certain cryptographic assumptions [54]. However, it is unclear how to exploit the techniques in their work to prove a hardness result under a worst-case assumption (see also [54, Remark 4.4]).

While in the context of circuit minimization we have explored reductions from variants of the set cover problem, the proof of Theorem 6 relies on a reduction from graph colorability. The NP-hardness of approximating the chromatic number of a given graph G is now well established (see [57, 31, 24, 82]): it is NP-hard to approximate $\chi(G)$ up to a factor of $n^{1-\varepsilon}$, where n is the number of nodes in G, and $\varepsilon > 0$ is an arbitrary constant.

Our reduction from graph colorability to Partial-MCCP is elementary, and we describe it next. Given a graph $G = ([n], E)$, where $E \subseteq \binom{[n]}{2}$, we construct from it a partial function $f_G : [n] \times [n] \to \{0, 1, *\}$, such that the complexity of f_G under any of the measures considered in Theorem 6 will give us an approximation on $\chi(G)$. The partial function f_G is given by $f_G(i, j) = 1$ if $i = j$, $f_G(i, j) = 0$ if $\{i, j\} \in E$, and $f_G(i, j)$ is undefined otherwise. Note that the matrix $M_G \in \{0, 1, *\}^{n \times n}$ encoding f_G is easily constructed from the input graph G. This completes the description of the communication problem output by the reduction.

For the reader familiar with standard notions from communication complexity, we briefly explain why the communication complexity of f_G (denoted by $\mathsf{D}(f_G)$) provides information about $\chi(G)$. First, it is not hard to show that the 1-cover number of M_G is exactly the chromatic number of G. Using the relation between 1-cover number and deterministic communication complexity, this shows that $\mathsf{D}(f_f) \geq \chi(G)$. On the other hand, it can be shown that there is a deterministic protocol for M_G which has no more than $2 \cdot \chi(G)$ leaves in its protocol tree. Moreover, this protocol is balanced, and this provides a useful upper bound on $\mathsf{D}(f)$. Theorem 6 will then follow from these two claims.

While this reduction and the aforementioned $n^{1-\varepsilon}$-inapproximability results for graph coloring allow us to derive strong hardness of approximation results for communication measures such as leaf complexity and partition number, there is a significant loss with respect to computing $\mathsf{D}(f)$. This happens because in the worst-case $\mathsf{D}(f)$ is only logarithmically related to the other measures. As a consequence, these results are insufficient to establish the consequences described in the second part of Section 1.1.2. To achieve that, we rely on more recent results on the hardness of graph coloring for a different regime of parameters. In more detail, we make crucial use of the works of Huang [37] and Wrochna and Živný [79]. They established that, for any large enough constant k, it is NP-hard to distinguish k-colorable graphs from graphs that are not $g(k)$-colorable, where the function g is *exponential* in k. This translates to new hardness results for approximating $\mathsf{D}(f)$, and we refer to Section 5.3 for more details.

1.3 Further related work

In this section we provide additional pointers to works and research directions related to our results.

Circuit minimization of Boolean functions (MCSP). The circuit minimization problem for single-output Boolean functions with respect to a circuit class $\mathcal{C}$ (denoted by $\mathcal{C}$-MCSP) is known to be NP-complete when $\mathcal{C} \in \{\mathsf{DNF}, \mathsf{DNF\text{-}XOR}\}$. Hardness of DNF-MCSP was first established by Masek [58], with alternate proofs appearing in [22, 4]. This result has also been extended to an almost tight hardness of approximation result for DNF-MCSP (see [48]). The NP-hardness result for (DNF-XOR)-MCSP is more recent [33]. We are not aware of NP-hardness results for $\mathcal{C}$-MCSP for stronger classes. We refer to [69] for more information on circuit minimization for restricted computational models, and for pointers to several related works.

In the case of general Boolean circuits, MCSP is known to be hard for NC^1 (and for slightly stronger classes) under non-uniform AC^0 reductions [64, 27]. Moreover, it has been proved that any function in P can be approximated with noticeable advantage by AC^0 circuits containing a single oracle gate that decides MCSP [64]. Other works have established that every problem in the complexity class SZK (including graph isomorphism) is efficiently reducible to MCSP (see [2, 3]). Interestingly, it has been proved under a cryptographic assumption that a version of MCSP with a large gap between positive and negative instances is NP-intermediate [5].

Several works have shown that establishing the NP-hardness of MCSP with respect to certain classes of reductions would have significant implications to our understanding of algorithms and complexity. For some restricted notions of reduction, hardness results cannot be established even for very simple subclasses of P (see [62]). We refer to [43, 62] and subsequent papers [36, 35, 5, 7] for more information about this line of work. We discussed the influence of these works in our results in Section 1.2.

It is widely known that if solving MCSP is feasible then modern cryptography is insecure [67, 70]. The hardness of MCSP also plays a fundamental role in circuit complexity via the notion of natural proofs [70], and more recently in connection to hardness magnification [63, 60] (see the paragraph on unconditional lower bounds below). As mentioned above, MCSP is closely related to learning algorithms, and we refer to [17] and to Section A for more details.[10] The hardness of MCSP and $\mathcal{C}$-MCSP is also connected to questions in proof complexity (cf. [51, 61]). For several relations between MCSP and complexity theory, we refer to [43].

As mentioned in Section 1.2, it is known that that an extension of MCSP to circuits with oracle gates is NP-hard under randomized reductions [38]. A different formulation of MCSP with oracles has been investigated in [6, 35, 39].

Note that many results discussed above also apply to circuit minimization for partial and multi-output functions, since the corresponding problems generalize MCSP.

Partial Boolean functions (Partial-MCSP) and learning algorithms. The circuit minimization problem for partial Boolean functions with respect to DNF size was shown to be NP-hard by Levin in his seminal work [55].[11] A proof of this NP-hardness result can be found for instance in [77]. Kearns and Valiant [45] showed cryptographic hardness for (Formula)-Partial-MCSP, and a proof that Partial-MCSP is NP-complete under deterministic reductions is implicit in [30] and [1].

[10] Indeed, the opening question in our abstract is also naturally captured by investigations about the power and limitations of learning algorithms. Modern results in complexity theory and learning theory show that circuit minimization and learning are directly related tasks.

[11] This result corresponds to Problem 2 in the English translation of Levin's paper, which can be found in the appendix of [75].

The complexity of Partial-MCSP plays a crucial role in learning theory. More precisely, the search version of $\mathcal{C}$-Partial-MCSP for a concept class $\mathcal{C}$ has long been known to be hard with respect to problems in computational learning theory ([13]; see also [46, Chapter 2]). In other words, an efficient algorithm for the search version of $\mathcal{C}$-Partial-MCSP implies that $\mathcal{C}(\mathsf{poly})$ is PAC learnable in polynomial time (this is often referred to as the "Occam's Razor" principle). This has led to numerous learning algorithms, since the problem is known to be solvable by non-trivial algorithms if $\mathcal{C}$ is simple enough (for example, in the case of decision lists [71]). Other works have established NP-hardness results for slightly more complex classes $\mathcal{C}$, such as decision trees [30][12], or neural networks with a fixed topology [41, 12].

More recently, [78] proved that an efficient algorithm for $\mathcal{C}$-Partial-MCSP also implies PAC learnability. Indeed, he showed that these two tasks are equivalent when one considers a relaxation of worst-case $\mathcal{C}$-Partial-MCSP. (In Section A, we adapt his result to the case of learnability under the uniform distribution.) A certain robust variant of Partial-MCSP is also known to be tightly connected to learnability in the agnostic case (see [50] for more details).

Ko [49] considers the problem MINLT (which roughly corresponds to a variant of Partial-MCSP based on Turing machines rather than circuits) and shows that there exist oracles $\mathcal{O}$ such that $\mathsf{MINLT}^{\mathcal{O}}$ is *not* complete for $\mathsf{NP}^{\mathcal{O}}$ under polynomial-time Turing reductions.

Multi-output Boolean functions (Multi-MCSP) and circuit minimization in practice. There have been quite a few developments on the theoretical aspects of multi-output circuit minimization, and some of these works have had an impact on the practice of circuit minimization. Indeed, chip designers are interested in (and have developed many heuristics for) solving the circuit minimization problem for multi-output (partial) Boolean functions under different input representations. The problem has a long history (see e.g. Karnaugh [44] for two-level minimization and Roth and Karp [72] for multi-level minimization), and we refer to a textbook such as [29] for details.

Regarding theoretical hardness results, Boyar, Matthews, and Peralta [16] show that the multi-output minimization problem for computing linear forms (computing Ax where A is a fixed matrix and x is the input vector) in the restricted model of "linear straight-line programs" (where operations consist of taking linear combinations of inputs) is NP-hard.

Unconditional complexity lower bounds for MDCP, Partial-MCSP, and Multi-MCSP. Results from the emerging area of hardness magnification (see e.g. [63, 60]) show that weak unconditional lower bounds for MCSP and related problems against a variety of computational models can be magnified to complexity separations as strong as $\mathsf{P} \neq \mathsf{NP}$. It is worth noting that Partial-MCSP has played a crucial role in the proof of earlier results in this area, both in circuit complexity [65, Theorem 1] and in proof complexity [61, Proposition 4.14]. Motivated in part by hardness magnification, it is now known that most combinatorial circuit lower bounds established in complexity theory can be shown to hold for MCSP as well (see [27, 20] and references therein). All these results immediately imply state-of-the-art lower bounds for MDCP, Partial-MCSP, and Multi-MCSP, since the instances of MCSP easily embed into these problems.

[12] Note that this NP-hardness result for decision trees is for *partial* functions, represented by a list as in Definition 2. For total functions, the problem is solvable in polynomial time by a simple dynamic programming algorithm.

Hardness of estimating communication complexity. It has long been known that computing the non-deterministic communication complexity of a given total function is NP-hard [66]. We refer to a subsequent work [57] for an inapproximability result.

In the setting of deterministic communication complexity, which is the main focus of our results, the following was known. In [54], Kushilevitz and Weinreb proved under cryptographic assumptions that one cannot efficiently compute the communication complexity of a given total two-player function $f : [n] \times [n] \to \{0, 1\}$. In more detail, if one assumes that there are pseudorandom function generators in NC_1 which fool polynomial size distinguishers, then it is hard to estimate communication complexity with an approximation ratio of ≈ 1.1. On the other hand, if one assumes that there are pseudorandom function generators in NC secure against distinguishers of sub-exponential size, then the hardness result is improved to an approximation ratio of order $n^{1/2}$.

To our knowledge, previously to our work no result was known on the hardness of estimating deterministic communication complexity under *worst-case* assumptions.

2 Preliminaries

We let $[n]$ denote the set $\{1, \ldots, n\}$.

We will use the NP-hardness of approximating Set Cover with respect to sets of bounded size [76]. A weaker version of the result from [76] is sufficient.

We say that sets $S_1, \ldots, S_\ell$ *cover* a set T if $T \subseteq S_1 \cup \cdots \cup S_\ell$. For a collection of sets $\mathcal{S}$ and a set T, we use $\mathsf{cover}(T, \mathcal{S})$ to denote the minimum number of sets in $\mathcal{S}$ necessary to cover T.

▶ **Definition 7** (r-Bounded Set Cover Problem)**.** *For a positive integer r, the r-Bounded Set Cover Problem is defined as follows:*

- Input. *A positive integer n represented in unary, and a collection $\mathcal{S}$ of nonempty subsets of $[n]$. We are promised that $\bigcup_{S \in \mathcal{S}} S = [n]$ and that $|S| \leq r$ for each $S \in \mathcal{S}$.*
- Output. *The value* $\mathsf{cover}([n], \mathcal{S})$.

For convenience, we defined the r-Bounded Set Cover Problem as an optimization problem instead of decision problem.

▶ **Theorem 8** (Hardness of approximating r-Bounded Set Cover [76])**.** *For every constant $\alpha \geq 1$ there exists $r \in \mathbb{N}$ such that approximating the r-Bounded Set Cover Problem within a factor of α is NP-hard. More precisely, for every $L \in \mathsf{NP}$, there exists a polynomial-time algorithm that, on input x, outputs a parameter k and an instance $(1^m, \mathcal{S})$ of the r-Bounded Set Cover Problem such that if $x \in L$ then $\mathsf{cover}([m], \mathcal{S}) \leq k$, and if $x \notin L$ then $\mathsf{cover}([m], \mathcal{S}) > \alpha \cdot k$.*

3 Warm-up: NP-hardness for arbitrary generators and partial functions

We establish in this section the NP-hardness of MDCP, and as an easy corollary, provide a self-contained proof of the NP-hardness of Partial-MCSP. The argument consists of two steps: a randomized (approximate) reduction from r-Bounded Set Cover to MDCP, and a deterministic reduction from MDCP to Partial-MCSP.

3.1 Notation

Recall that we consider a generalization of Boolean circuit complexity originally proposed and investigated in a particular context by [68]. Let V be a finite set (also referred to as the *ground set*). Given a family $\mathcal{B} = \{B_i\}_{i \in [m]}$ of nonempty sets $B_i \subseteq V$ (also referred to

as the family of *generators*) and a set $A \subseteq V$, we let $D(A \mid \mathcal{B})$ denote the minimum number of unions, intersections, and complements that are required to construct A from the sets in $\mathcal{B}$. More precisely, the complement of a set $U \subseteq V$ is defined as $V \setminus U$, and we represent a construction of A from $\mathcal{B}$ as a sequence $B_1, \ldots, B_m, E_1, \ldots, E_\ell$ of sets in V such that $E_\ell = A$ and each set E_j is either the union or intersection of two previously generated sets, or the complement of a previously generated set. We assume for convenience that $D(B \mid \mathcal{B}) = 0$ if $B \in \mathcal{B}$. We refer to $D(A \mid \mathcal{B})$ as the *discrete complexity* of A with respect to $\mathcal{B}$.

3.2 A reduction from r-Bounded Set Cover to MDCP

We will use instances of MDCP with a particular structure. This makes the second reduction from MDCP to Partial-MCSP more transparent.

Let V be a ground set, and $\mathcal{B} = \{B_1, \ldots, B_n\}$ be a collection of nonempty subsets of V. For an element $v \in V$, we use $v^\uparrow \in \{0,1\}^n$ (the "lifted" version of v) to denote the string with the property that $v^\uparrow_i = 1$ if and only if $v \in B_i$. We say that a collection $\mathcal{B}$ is *complete* (with respect to V) if the following holds: if $a, b \in V$ and $a^\uparrow = b^\uparrow$ then $a = b$. In other words, distinct elements of V have different liftings with respect to $\mathcal{B}$. Our reduction from r-Bounded Set Cover to MDCP will always produce a family of generators that is complete.

Let r be a large enough constant, so that say 10-approximating r-Bounded Set Cover is NP-hard. Given an instance $(1^n, \mathcal{S})$ of this problem, the reduction proceed as follows. Fix $V \stackrel{\text{def}}{=} [n^3]$. Partition V into n blocks $V_1, \ldots, V_n$, where $|V_i| = n^2$ for each $i \in [n]$. Given a set $A \subseteq V$, we let $A_i \stackrel{\text{def}}{=} A \cap V_i$. Now view V as the set $\{0,1\}^{3 \log n}$, and for each $j \in [3 \log n]$, let $B_j = \{v \in V \mid v_j = 1\}$. For convenience, we let $\mathcal{F} \stackrel{\text{def}}{=} \{B_1, \ldots, B_{3 \log n}\}$. Note that any family $\mathcal{B}$ of generators that contains $\mathcal{F}$ is complete with respect to V.

Let $\mathcal{A}^{-i} \stackrel{\text{def}}{=} \{A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n\}$. We say that a set A is *critical* if, for every $i \in [n]$,

$$D(A \mid \mathcal{F} \cup \mathcal{A}^{-i}) \;>\; n \cdot \log n.$$

It is not hard to show that a uniformly random set $\boldsymbol{A} \subseteq_{1/2} V$ is typically critical.

▶ **Lemma 9.** *Let $\boldsymbol{A} \subseteq_{1/2} V$ be sampled by letting $v \in \boldsymbol{A}$ independently with probability $1/2$ for each $v \in V$. Then,*

$$\Pr_{\boldsymbol{A}}[\boldsymbol{A} \text{ is critical}] \to 1 \;\text{ as }\; n \to \infty.$$

Proof. For a fixed $i \in [n]$, we argue below that

$$\Pr_{\boldsymbol{A}}[D(\boldsymbol{A} \mid \mathcal{F} \cup \boldsymbol{\mathcal{A}}^{-i}) \leq n \cdot \log n] \;=\; o(1/n).$$

The lemma follows by a union bound.

Note that, conditioning on the choice of $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{i-1}, \boldsymbol{A}_{i+1}, \ldots, \boldsymbol{A}_n$, the set $\boldsymbol{A}_i$ is still a uniformly distributed subset of V_i. Moreover, *after we fix* $\mathcal{F}$ and $\boldsymbol{\mathcal{A}}^{-i}$, any construction of a set $E \subseteq V$ from $\mathcal{F} \cup \boldsymbol{\mathcal{A}}^{-i}$ using s operations can be described by a binary string of length at most $O(s \cdot (\log s + \log n))$. Since $|V_i| = n^2$, the probability that the discrete complexity of $\boldsymbol{A}$ (conditioned on the choice of $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{i-1}, \boldsymbol{A}_{i+1}, \ldots, \boldsymbol{A}_n$) given $\mathcal{F} \cup \boldsymbol{A}^{-i}$ falls below $s = n \cdot \log n$ is at most

$$\frac{2^{O(s \cdot (\log s + \log n))}}{2^{n^2}} \;=\; o(1/n).$$

Since this holds for any choice of $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{i-1}, \boldsymbol{A}_{i+1}, \ldots, \boldsymbol{A}_n$, the desired probability upper bound holds. ◀

We assume from now on that we have efficiently produced a critical set A. (Our randomized reduction will always be correct on a critical set.) Note that so far we have only inspected the input 1^n from $(1^n, \mathcal{S})$. Our approximate reduction outputs a triple $(1^{n^3}, A, \mathcal{B}_\mathcal{S})$, where A is generated as above, and whose family $\mathcal{B}_\mathcal{S}$ is defined as follows. For each set $S \in \mathcal{S}$, let $W_S \stackrel{\text{def}}{=} \bigcup_{i \in S} A_i$. Now set

$$\mathcal{B}_\mathcal{S} \stackrel{\text{def}}{=} \mathcal{F} \cup \{W_S \mid S \in \mathcal{S}\}.$$

Clearly, the triple $(1^{n^3}, A, \mathcal{B}_\mathcal{S})$ can be efficiently computed from $(1^n, \mathcal{S})$.

We argue next that computing $D(A \,|\, \mathcal{B}_\mathcal{S})$ allows us to 2-approximate $\mathsf{cover}([n], \mathcal{S})$.

▶ **Lemma 10.** *If* $\mathsf{cover}([n], \mathcal{S}) = \ell$ *then* $D(A \,|\, \mathcal{B}_\mathcal{S}) \leq \ell$.

Proof. Let $S_1, \ldots, S_\ell$ be a cover of $[n]$ using sets from $\mathcal{S}$. Then, by construction of the sets W_S, we get that $A = W_{S_1} \cup \cdots \cup W_{S_\ell}$. In particular, it is possible to construct A using at most ℓ operations (unions) starting from the sets in $\mathcal{B}_\mathcal{S}$. ◀

▶ **Lemma 11.** *If* $D(A \,|\, \mathcal{B}_\mathcal{S}) = \ell$ *then* $\mathsf{cover}([n], \mathcal{S}) \leq 2\ell$.

Proof. Let $t \stackrel{\text{def}}{=} |\mathcal{S}|$, and suppose that $E_1, \ldots, E_\ell$ represent a construction of $A = E_\ell$ from sets in $\mathcal{B}_S$. For convenience, we write $B_\mathcal{S} = \mathcal{F} \cup \{W_{S_1}, \ldots, W_{S_t}\}$. In order to prove Lemma 11, we need the following simple claim.

▷ Claim 12. $E_1, \ldots, E_\ell$ depend on at most 2ℓ sets from $\{W_{S_1}, \ldots, W_{S_t}\}$.

This claim holds simply because each set E_j depends on at most 2 other sets. One may replace 2ℓ by $\ell + 1$ by arguing more carefully, that any ℓ-node directed acyclic graph with fan-in 2 and a single sink node has at most $\ell + 1$ source nodes. But this tighter bound is unnecessary for our purpose.

Continuing with the proof of Lemma 11, and relabelling some indexes if necessary, we assume for convenience of notation that the construction of A from $\mathcal{B}_\mathcal{S}$ depends only on $\mathcal{F} \cup \{W_{S_1}, \ldots, W_{S_{2\ell}}\}$. In other words,

$$D(A \,|\, \mathcal{F} \cup \{W_{S_1}, \ldots, W_{S_{2\ell}}\}) \leq \ell.$$

Since $W_{S_j} = \bigcup_{i \in S_j} A_i$ and $|S_j| \leq r$, we get that $D(W_{S_j} \,|\, \{A_i\}_{i \in S_j}) \leq r$. In turn, by composing constructions, we obtain that

$$D(A \,|\, \mathcal{F} \cup \{A_i\}_{i \in S_1} \cup \cdots \cup \{A_i\}_{i \in S_{2\ell}}) \;\leq\; \ell \cdot r.$$

We can assume that $r < \log n$ for a large enough n. In addition, notice that if $\ell > n$ then the statement of Lemma 11 is trivial, since by assumption $\mathcal{S}$ always covers $[n]$ and as a consequence $\mathsf{cover}([n], \mathcal{S}) \leq n$. But from $r < \log n$ and $\ell \leq n$ it follows that the expression above can be upper bounded by $n \cdot \log n$. Given that A is critical, it must be the case that the sets $S_1, \ldots, S_{2\ell}$ cover $[n]$. In other words, $\mathsf{cover}([n], \mathcal{S}) \leq 2\ell$, which completes the proof. ◀

Consequently, it immediately follows from Lemmas 9, 10, and 11 and from Theorem 8 that MDCP is NP-hard to compute under polynomial-time randomized reductions.

3.3 A reduction from MDCP to Partial-MCSP

Let $(1^n, A, \mathcal{B}, s)$ be an input to MDCP. We can assume from the properties of the previous reduction that $\mathcal{B}$ is complete. We will also assume without loss of generality that A is nonempty. We create an instance of Partial-MCSP as follows.

For every $v \in V$, where $V = [n]$, we consider the corresponding lifted vector $v^\uparrow \in \{0,1\}^k$ described above, where $\mathcal{B} = \{B_1, \ldots, B_k\}$ and each $B_i \subseteq V$ is nonempty. We let

$$V^\uparrow \stackrel{\text{def}}{=} \{v^\uparrow \mid v \in V\} \subseteq \{0,1\}^k \text{ and } A^\uparrow \stackrel{\text{def}}{=} \{a^\uparrow \mid a \in A\} \subseteq V^\uparrow.$$

Consider the partial Boolean function $f_A \colon V^\uparrow \to \{0,1\}$ defined by $f_A(x) = 1$ if and only if $x \in A^\uparrow$. The reduction outputs the tuple $(1^n, \mathcal{P}, s)$, where

$$\mathcal{P} \stackrel{\text{def}}{=} \{(x, f_A(x)) \mid x \in V^\uparrow\}.$$

It is easy to see that this tuple can be efficiently computed from the input $(1^n, A, \mathcal{B})$. In order to establish the correctness of this reduction, it suffices to prove the following lemma.

▶ **Lemma 13.** *The partial Boolean function $f_A \colon V^\uparrow \to \{0,1\}$ agrees with a Boolean circuit of size at most s if and only if $D(A \mid \mathcal{B}) \leq s$.*

Proof. The lemma is intuitively clear, since the lifting operation $\uparrow$ induces a *bijection* between $V = [n]$ and $V^\uparrow \subseteq \{0,1\}^k$. (This is the case because by assumption $\mathcal{B}$ is complete.) For completeness, we provide more details below.

Let $A \subseteq V$ be an arbitrary nonempty set, and assume that $\mathcal{B}$ is complete. If the circuit size of f_A is 0, then it must be the case that this circuit is simply x_i for some $i \in [k]$. But then $v \in A$ iff $f_A(v^\uparrow) = 1$ iff $v_i^\uparrow = 1$ iff $v \in B_i$. Equivalently, $A = B_i$. By convention, we have $D(B_i \mid \mathcal{B}) = 0$. The other direction is analogous. This establishes the base case corresponding to $s = 0$.

Suppose now that C is a Boolean circuit of size $s > 0$ that agrees with f_A over $V^\uparrow$. Replace each input variable x_i of C by the set $B_i \in \mathcal{B}$, and each Boolean operation AND, OR, and NOT in C by the corresponding set operation $\cap$, $\cup$, and complementation. We claim that this induces a construction of A from $\mathcal{B}$.

In order to see this, fix any element $v \in V$. More generally, we claim that the i-th gate of C outputs 1 on $v^\uparrow$ if and only if the i-th set E_i constructed under this transformation contains the element v. For the input gates, this follows from the discussion above. To prove this for the i-th gate g_i, assume that the result holds for $x_1, \ldots, x_k, g_1, \ldots, g_{i-1}$ (viewed as subsets of $V^\uparrow$), and consider the corresponding construction $B_1, \ldots, B_k, E_1, \ldots, E_k$ (these are subsets of V) induced by the transformation. Then it easily follows from the induction hypothesis that $g_i(v^\uparrow) = 1$ if and only if $v \in E_i$, regardless of the Boolean operation performed at g_i over the preceding gates. For instance, if $g_i = g_{i_1}$ AND g_{i_2} for $i_1, i_2 < i$, then $g_i(v^\uparrow) = 1$ iff $g_{i_1}(v^\uparrow) = 1$ and $g_{i_2}(v^\uparrow) = 1$ iff $v \in E_{i_1}$ and $v \in E_{i_2}$ iff $v \in E_{i_1} \cap E_{i_2} = E_i$.

Obtaining an upper bound on the circuit complexity of f_A from an upper bound on $D(A \mid \mathcal{B})$ can be done using the reverse transformation. This completes the proof of Lemma 13. ◀

Composing the reductions above completes the proof of NP-hardness of Partial-MCSP.

3.4 Search-to-decision reduction for Partial-MCSP

Recall that in Search-Partial-MCSP we are given $(1^n, \mathcal{P})$, where $\mathcal{P} = \{(x_i, b_i)\}_{i \in [t]}$ for some $t \in \mathbb{N}$, $x_i \in \{0,1\}^n$, and $b_i \in \{0,1\}$. The goal is to output a circuit C of minimum size that is consistent with $\mathcal{P}$. In this section, we show that this problem can be solved in deterministic

polynomial-time using an oracle $A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}$ that solves Partial-MCSP. We assume that $A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}$ outputs the optimal circuit size $s \in \mathbb{N}$ (a simple binary search suffices to obtain this value using oracle calls to Partial-MCSP).

We describe a recursive procedure B with access to $A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}$ that solves Search-Partial-MCSP. The input to B is of the form $(1^\ell, \mathcal{Q})$, where $\mathcal{Q}$ is a collection of input pairs in $\{0,1\}^\ell \times \{0,1\}$. The initial call to B that solves Search-Partial-MCSP will be of the form $B(1^n, \mathcal{P})$. For convenience, we rely on a polynomial-time sub-routine $\mathsf{Consistent}(\mathcal{Q}, C)$ that returns true if and only if circuit C over input variables $y_1, \ldots, y_\ell$ is consistent with $\mathcal{Q}$.

Algorithm B.

Input. A pair $(1^\ell, \mathcal{Q})$.

Output. A minimum size circuit C over $y_1, \ldots, y_\ell$ that is consistent with $\mathcal{Q}$.

1. If $\mathsf{Consistent}(\mathcal{Q}, y_i)$ holds for some $i \in [\ell]$, return the circuit represented by input variable y_i.
2. Otherwise, for each operation $\star \in \{\vee, \wedge, \neg\}$, and for each appropriate choice of one or two operands from $\{y_1, \ldots, y_\ell\}$:
   - 2.1 Let $\mathcal{Q}^+$ extend each pair $(y, b) \in \mathcal{Q}$ to a pair $(y^+, b) \in \{0,1\}^{\ell+1} \times \{0,1\}$, where the new coordinate corresponds to the result of the operation. Moreover, let $D(y_1, \ldots, y_\ell)$ be a depth-1 circuit of size 1 corresponding to the same operation.
   - 2.2 If $A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}(1^{\ell+1}, \mathcal{Q}^+) < A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}(1^\ell, \mathcal{Q})$, invoke $B(1^{\ell+1}, \mathcal{Q}^+)$. Let $C^+(y_1, \ldots, y_{\ell+1})$ be the circuit returned by this call. Return the description of $C(y_1, \ldots, y_\ell) \stackrel{\text{def}}{=} C^+(y_1, \ldots, y_\ell, D(y_1, \ldots, y_\ell))$.

We sketch next the proof that $B(1^\ell, \mathcal{Q})$ runs in polynomial time, and that it always returns a consistent circuit of minimum size. Let $s = A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}(1^\ell, \mathcal{Q})$. The proof of correctness is by induction on s, i.e., the induction hypothesis is that the algorithm is correct on every input pair $(1^\ell, \mathcal{Q})$ whose circuit complexity is at most s.

If $s = 0$, then an input variable y_i must be consistent with $\mathcal{Q}$. In this case, algorithm B correctly returns such a circuit in step (1) above. Assume now that $s \geq 1$ and that the induction hypothesis holds for any input whose complexity is at most $s-1$. For the induction step, let $(1^\ell, \mathcal{Q})$ be an input to Search-Partial-MCSP for which $A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}(1^\ell, \mathcal{Q}) = s$. Since $s > 0$, using any bottom layer gate in any optimal circuit for Q, it follows that there is at least one Boolean operation whose corresponding set Q^+ defined in step (2.1) will pass the test performed in step (2.2). Now consider any recursive call in step (2.2), which might not necessarily come from a bottom gate in an optimal circuit for $\mathcal{Q}$. By the induction hypothesis, a circuit C^+ consistent with the corresponding collection Q^+ and of size at most $s-1$ is returned. Clearly, the resulting circuit C obtained from C^+ and from the corresponding circuit D is consistent with Q and has size at most s. This completes the induction step, and the proof of correctness of B.

For the running time, note that on every instance $(1^\ell, \mathcal{Q})$ of B we have $s = A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}(1^\ell, \mathcal{Q}) \leq O(\ell \cdot |\mathcal{Q}|)$. Consequently, at most s nested recursive calls are made. In each call, $\mathsf{Consistent}(\mathcal{Q}, y_i)$ can be computed in time $O(|\mathcal{Q}| \cdot (\ell + s))$. We also consider in the worst case all possible operations over a set of at most $\ell + s$ input coordinates, and there are at most $O(\ell + s)^2$ such operations. Consequently, B runs in time at most $O(s \cdot (|\mathcal{Q}| \cdot (\ell + s) + (\ell + s)^2)) = O(\ell \cdot |\mathcal{Q}|)^3$.

4 Main result: NP-hardness of circuit minimization for multi-output functions

4.1 Definitions

4.1.1 Multi-output Functions, Concatenations, and Truth Tables

For a multi-output Boolean function $f : \{0,1\}^n \to \{0,1\}^m$, the *components* of f are the single-output functions $f_1, \ldots, f_m$ where $f_i : \{0,1\}^n \to \{0,1\}$ is defined so $f_i(x)$ equals the ith output bit of $f(x)$.

For a multi-output Boolean function f, we let the *circuit complexity of* f, denoted $\mathsf{CC}(f)$, be the minimum size of any circuit computing f.

For strings $x, y \in \{0,1\}^\star$, we let $x \bullet y$ denote the concatenated string. We identify a multi-output Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ with the concatenated string $T_1 \bullet \cdots \bullet T_m \in \{0,1\}^{m \cdot 2^n}$ where $T_1, \cdots, T_m$ are the truth tables of the components $f_1, \ldots, f_m$ respectively.

If $f : \{0,1\}^n \to \{0,1\}^{m_1}$ and $g : \{0,1\}^n \to \{0,1\}^{m_2}$ are Boolean functions with the same number of inputs, we define the concatenated Boolean function $f \bullet g : \{0,1\}^n \to \{0,1\}^{m_1+m_2}$ given by $f \bullet g(x) = f(x) \bullet g(x)$.

We also use the $\bullet$ symbol to indicate concatenation in a similar way that $\sum$ acts for addition. For example, if $T_1, \ldots, T_m$ are truth tables of functions with the same number of inputs, then we use the notation $\bullet_{i \in [m]} T_i$ to indicate $T_1 \bullet \cdots \bullet T_m$.

4.1.2 The Evaluation Function and Multi-output Computation

Each circuit C induces a multi-output Boolean function we call the *Evaluation Function of* C, denoted $\mathsf{Eval}\text{-}C$, that computes the outputs of each of the gates in C. In more detail, for a circuit C that takes n inputs and has s gates, the evaluation function induced by C, denoted $\mathsf{Eval}\text{-}C : \{0,1\}^n \to \{0,1\}^{s+n}$, is given by $x_1 \bullet \cdots \bullet x_n \bullet g_1 \bullet \cdots \bullet g_s$ where x_i is the function computed by the ith input wire of C and g_j is the function computed by the jth gate in C (for this to be well-defined, we need to fix an ordering of the gates of C, but, for our purposes, any ordering will do).

Using the Evaluation Function, an equivalent definition of multi-output circuit computation to the one given in the introduction is that a Boolean circuit C computes a (multi-output) Boolean function f if and only if every component of f is a component of $\mathsf{Eval}\text{-}C$.

4.1.3 Windows of Truth Tables

Given a truth table T of length n and a subset $S \subseteq [n]$, we define the *S-window of T*, denoted $T_{\langle S \rangle}$, to be the truth table of length n that (informally) "sees" T on the elements of S and zeroes everywhere else. Rigorously,

$$T_{\langle S \rangle}(x) = \begin{cases} T(x) & \text{if } x \in S, \\ 0 & \text{otherwise.} \end{cases}$$

4.1.4 Canonical DNF Circuits

For each (single output) Boolean function $f : \{0,1\}^n \to \{0,1\}$, it will be useful to fix an algorithm that outputs a single "canonical" Boolean circuit for computing f.

For our purposes, many algorithms are possible, but, for concreteness, we will define the canonical circuit of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ to be the naive DNF formula for f, denoted $\mathbf{DNF}_f$, given by

$$\mathbf{DNF}_f \stackrel{\text{def}}{=} ((x_1 = y_1^1) \wedge \cdots \wedge (x_n = y_n^1)) \vee \cdots \vee ((x_1 = y_1^t) \wedge \cdots \wedge (x_n = y_n^t))$$

where

- $y^1, \ldots, y^t$ are the YES inputs of f in lexicographical order,
- $x_1, \ldots, x_n$ index the bits of the input string x,
- $y_1^j, \ldots, y_n^j$ index the bits of y^j for each $j \in [t]$,
- and for $i \in [n]$ and $j \in [t]$, $(x_i = y_i^j)$ is syntax for $\begin{cases} x_i & \text{if } y_i^j = 1, \\ \neg x_i & \text{if } y_i^j = 0. \end{cases}$

It is easy to see that $\mathbf{DNF}_f$ can be computed in polynomial-time given the truth table of f.

Moreover, reading the above definition of $\mathbf{DNF}_f$ from left to right gives a natural way of defining the kth gate in $\mathbf{DNF}_f$, whereby the kth gate corresponds to the kth gate symbol appearing in the above formula. This fact will later be useful in our analysis.

4.1.5 Lifting Sets

Our reduction will use a way to lift subsets into subsets on larger ground sets. To do this, we will first define a canonical partition of $[m]$ into n parts for $m \geq n$. Let $\mathcal{P}^{m,n} = (P_1^{m,n}, \ldots, P_n^{m,n})$ be the partition of $[m]$ into n parts given by

$$P_i^{m,n} = \{j \in [m] : j \equiv i \mod n\}.$$

From this partition, we can now lift subsets as follows. Let $n \leq m \in \mathbb{N}$. Let $S \subseteq [n]$. The *m-lift* of S, denoted S^m, is the set given by

$$S^m \stackrel{\text{def}}{=} \bigcup_{i \in S} P_i^{m,n}.$$

4.2 A reduction from r-Bounded Set Cover to Multi-MCSP

In order to show that Multi-MCSP is NP-hard, we give a probabalistic polynomial-time many-one reduction with one-sided error from a constant approximation of r-Bounded Set Cover to Multi-MCSP.

In fact, for convenience, our reduction will be from the optimization version of approximating r-Bounded Set Cover to the optimization version of Multi-MCSP (computing CC), but it will be easy to see that our reduction can be converted into the desired reduction for the corresponding decision problems.

Let r be a large enough constant, so that say 10-approximating r-Bounded Set Cover is NP-hard. Given an instance $(1^n, \mathcal{S})$ of this problem, the reduction proceeds as follows. Let $m = O(n^3)$ be the least power of two greater than n^3. Let T be a uniformly random truth table of length m. I.e., T is a binary string in $\{0,1\}^m$, representing a function from $\{0,1\}^{\log m}$ to $\{0,1\}$. Compute the truth table of

$$g \stackrel{\text{def}}{=} \bigodot_{S \in \mathcal{S}} \textsf{Eval-}\mathbf{DNF}_{T_{\langle S^m \rangle}}.$$

Let k be the number of distinct components of g that are not functions computed by an input gate, that is,

$$k \stackrel{\text{def}}{=} |\{g_i : g_i \text{ is a component of } g \text{ and } g_i \neq x_j \text{ for all } j \in [\log m]\}|.$$

The reduction then outputs [13]

$$\Delta \stackrel{\text{def}}{=} \mathsf{CC}(T \bullet g) - k.$$

First, we argue that this procedure runs in polynomial time. The only two steps that may raise concern are whether we can compute the truth table of g efficiently and whether we can compute k efficiently.

To show that the truth table of g can be computed efficiently, it suffices to show that for each $S \in \mathcal{S}$, the truth table of $\mathsf{Eval}\text{-}\mathbf{DNF}_{T_{\langle S^m \rangle}}$ can be computed in time polynomial in n. Computing $T_{\langle S^m \rangle}$ can be done in time $O(m + |T|) = O(m)$ and outputs a truth table of length m. The canonical DNF of the truth table $T_{\langle S^m \rangle}$ can be computed in time polynomial in $|T_{\langle S^m \rangle}| = m$, and the resulting DNF has $\log m$ inputs and size at most $O(m \log m)$. Finally, computing the Evaluation Function of a circuit with $\log m$ inputs and $O(m \log m)$ gates can be done in time $O(m^3)$ by just evaluating the circuit on every input. Hence, putting these all together, computing the truth table of $\mathsf{Eval}\text{-}\mathbf{DNF}_{T_{\langle S^m \rangle}}$ can be done in time polynomial in $m = O(n^3)$.

To see that we can compute k efficiently, realize that we have already computed the full truth table of g and that removing any components computed by one of the $\log m$ input wires along with any duplicate components takes time at most quadratic in the length of the truth table of g.

Now, we will argue for the correctness of the reduction. We will show that

$$\mathsf{cover}([n], \mathcal{S})/4 - 4 \underset{\substack{\text{(with high probability} \\ \text{using Lemma 16)}}}{\leq} \Delta \underset{\substack{\text{(unconditionally} \\ \text{using Lemma 15)}}}{\leq} \mathsf{cover}([n], \mathcal{S}),$$

and thus, with high probability Δ computes a 10-approximation of r-Bounded Set Cover when n is sufficiently large.[14] Moreover, this computation has one-sided error since the upper bound holds unconditionally.

Thus, after proving Lemmas 15 and 16, we will have shown that there is a randomized polynomial-time many-one reduction with one-sided error from 10-approximating r-Bounded Set Cover to Multi-MCSP.

Before proving Lemmas 15 and 16, we make the following observation about computing g.

▶ **Proposition 14.** *If a circuit C computes g, then there are k distinct gates in C that compute components of g. Moreover, $\mathsf{CC}(g) = k$.*

Proof. We begin by proving the first statement, which also implies the lower bound $\mathsf{CC}(g) \geq k$. Suppose C is a circuit that computes g. Then every distinct component of g has a (necessarily distinct) input wire or gate from C that computes that component. Therefore, since g has k distinct components that are not computed by an input wire, C must have at least k distinct gates computing components of g.

Next, we sketch the proof of the upper bound $\mathsf{CC}(g) \leq k$. Let C be the circuit built as follows. For each $S \in \mathcal{S}$, iterate through the gates g in $\mathbf{DNF}_{T_{\langle S^m \rangle}}$ in topological order. Let $\diamond \in \{\wedge, \vee, \neg\}$ be the gate type of g. If g computes a function that is already computed by C,

[13] For the decision problems, we can determine if there is a 10-approximate set cover of size ℓ by outputting $\mathsf{Multi\text{-}MCSP}(T \bullet g, k + \ell)$, which computes whether $\Delta \leq \ell$.

[14] Since $\mathsf{cover}([n], \mathcal{S}) \geq n/r$ (using that the sets in $\mathcal{S}$ have cardinality at most r), we will actually get that, for each $\epsilon > 0$, Δ gives a $4 + \epsilon$ approximation with high probability when n is sufficiently large.

then ignore it. Otherwise, add a $\diamond$ gate to C that takes as input(s) those gate(s) in C that compute the function(s) which are fed as inputs to g in $\mathbf{DNF}_{T_{\langle S^m \rangle}}$ (we are guaranteed to find such gates in C since we are iterating in topological order).

By construction, C computes g. (Recall that $g = \bullet_{S \in \mathcal{S}} \mathsf{Eval}\text{-}\mathbf{DNF}_{T_{\langle S^m \rangle}}$, and our construction ensures C computes every function computed by a gate in $\mathbf{DNF}_{T_{\langle S^m \rangle}}$ for any $S \in \mathcal{S}$.) Moreover, our construction maintains that every gate in C computes a component of g that is not computed by any other gate or input wire. Thus, since g has at most k unique components not computed by input wires, C has at most k gates. ◀

One consequence of Proposition 14 is that $\Delta = \mathsf{CC}(T \bullet g) - \mathsf{CC}(g)$. With this fact, we can prove our two main lemmas.

▶ **Lemma 15.** *If* $\mathsf{cover}([n], \mathcal{S}) = \ell$*, then* $\Delta \leq \ell$*.*

Proof. Let $S_1, \ldots, S_\ell$ be a cover of $[n]$ using sets from $\mathcal{S}$. Then, by construction, we have that $T = T_{\langle S_1^m \rangle} \vee \cdots \vee T_{\langle S_\ell^m \rangle}$. Since $T_{\langle S_1^m \rangle}, \ldots, T_{\langle S_\ell^m \rangle}$ are components of g, this implies that

$$\Delta = \mathsf{CC}(T \bullet g) - \mathsf{CC}(g) \leq \ell$$

as desired. ◀

▶ **Lemma 16.** *Let* ℓ *be the largest integer such that* $\mathsf{cover}([n], \mathcal{S}) \geq 4\ell$*. Then,* $\Delta > \ell$ *with high probability.*

Proof. Our strategy will be as follows. We say that the choice of T is *bad* if, for that choice of T, $\Delta \leq \ell$. We will then upper bound the number of bad T by showing such T have short descriptions.

Fix some bad T. Then $\ell \geq \Delta = \mathsf{CC}(T \bullet g) - k$, so $\mathsf{CC}(T \bullet g) \leq \ell + k$. Since there is a circuit C computing $T \bullet g$ using at most $\ell + k$ gates and k of the gates in C must compute the unique components of g (using Proposition 14), it follows that there is a circuit D that takes $(\log(m) + k)$-inputs and has at most ℓ gates such that

$$D(x, g_1(x), \ldots, g_k(x)) = T(x)$$

for all $x \in \{0,1\}^{\log m}$ where $g_1, \ldots, g_k$ are the unique components of g. Moreover, since D has only ℓ gates of fan-in 2, it uses at most 2ℓ of the components of g in the circuit. Thus, after a possible relabeling of $g_1, \ldots, g_k$, we can assume D takes at most $(\log(m) + 2\ell)$-inputs and that

$$D(x, g_1(x), \ldots, g_{2\ell}(x)) = T(x).$$

Hence, to describe T, we just need to have a description for D as well as a description for $g_1, \ldots, g_{2\ell}$. Indeed, this will be the last step in our eventual description of T. We present the eventual description now so as to guide the reader. Our proof will subsequently proceed working through this description from bottom to top.

1. Given
   - a subset $J \subseteq [n]$ of size at most $\leq n(1 - \frac{1}{2r})$
   - for each $j \in J$ a partial truth table encoding $(j, V_j) \in [n] \times \{0,1\}^{m/n+1}$
   - r-bounded subsets $S_1, \ldots, S_{2\ell} \subseteq [n]$ whose union is J,
   - gate numbers $u_1, \ldots, u_{2\ell} \in [2m \log m]$,
   - and a circuit D of size ℓ with $(n + 2\ell)$ inputs

2. For $j \in J$, let $T_{\langle P_j^{m,n} \rangle}$ be the function whose values on $P_j^{m,n}$ in lexicographic order are given by the binary string V_j and is zero everywhere else
3. For $i \in [2\ell]$, let $T_{\langle S_i^m \rangle} = \bigvee_{j \in S_i \subseteq J} T_{\langle P_j^{m,n} \rangle}$
4. For $i \in [2\ell]$, let g_i be the function computing by the u_ith gate of $\mathbf{DNF}_{T_{\langle S_i^m \rangle}}$
5. Let $T(x) = D(x, g_1(x), \ldots, g_{2\ell}(x))$

Step 4: Describing the g_i. Since $g_1, \ldots, g_{2\ell}$ are components of g and $g = \bullet_{S \in \mathcal{S}} \mathsf{Eval}\text{-}\mathbf{DNF}_{T_{\langle S^m \rangle}}$, there exist $u_1, \ldots, u_{2\ell}$ and $S_1, \ldots, S_{2\ell}$ such that each g_i is the u_ith gate of $\mathbf{DNF}_{T_{\langle S_i^m \rangle}}$ for $i \in [2\ell]$. Moreover, each $u_i \leq 2m \log m$ by the trivial upper bound on the number of gates in a canonical DNF.

Step 3: Describing the $T_{\langle S_i^m \rangle}$. Next, we focus on encoding $T_{\langle S_i^m \rangle}$ for some i. Since $S_i^m = \bigcup_{j \in S_i} P_j^{m,n}$ (by construction of S_i^m), we have (by construction of $T_{\langle S_i^m \rangle}$) that $T_{\langle S_i^m \rangle} = \bigvee_{j \in S_i} T_{\langle P_j^{m,n} \rangle}$. Thus, to compute $T_{\langle S_i^m \rangle}$ for all $i \in [2\ell]$, it suffices to know $S_1, \ldots, S_{2\ell}$ as well as $T_{\langle P_j^{m,n} \rangle}$ for all $j \in J \stackrel{\text{def}}{=} \bigcup_{i \in [2\ell]} S_i$.

Step 2: Describing $T_{\langle P_j^{m,n} \rangle}$ for $j \in J$. The key to our encoding is to realize that $|J|$ cannot be too large, and thus, we do not need to know $T_{\langle P_j^{m,n} \rangle}$ for all $j \in [n]$. Since $\mathsf{cover}([n], \mathcal{S}) \geq 4\ell$, and J is the union of 2ℓ sets from $\mathcal{S}$, it follows that $|J| \leq n - 2\ell$. Moreover, we have that

$$n/r \leq \mathsf{cover}([n], \mathcal{S}) < 4\ell + 4$$

where the first inequality comes from the sets in $\mathcal{S}$ having cardinality at most r and the second inequality comes from the definition of ℓ. Thus,

$$|J| \leq n - 2\ell < n - \frac{n}{2r} + 2 = n(1 - \frac{1}{2r}) + 2.$$

Moreover, we can encode $T_{\langle P_j^{m,n} \rangle}$ for $j \in J$ very efficiently. To describe $T_{\langle P_j^{m,n} \rangle}$, it suffices to describe j and then give the list of values of $T_{\langle P_j^{m,n} \rangle}$ on the set $P_j^{m,n}$ in lexicographic order. Since $|P_j^{m,n}| \leq m/n + 1$ (essentially by construction), we can encode this list of values by a string $V_j \in \{0,1\}^{m/n+1}$ (where we pad this string with extra zeroes if $|P_j^{m,n}| < m/n + 1$).

Step 1: Counting the bits in the description. Now, we count the number of bits in our description of T. Describing J requires n-bits. For $j \in J$, each partial truth table encoding (j, V_j) requires at most $2 \log n + m/n + 1$ bits. Using that $|J| \leq n(1 - \frac{1}{2r})$, we get that all these encodings require at most $n(1 - \frac{1}{2r})(2 \log n + m/n + 1)$ bits. Encoding the r-bounded subsets $S_1, \ldots, S_{2\ell} \subseteq [n]$ requires at most $2n\ell \leq n^2$ bits (here we use the fact that $4\ell \leq n$ since $\mathsf{cover}([n], \mathcal{S}) = 4\ell$). Encoding the gate numbers $u_1, \ldots, u_{2\ell}$ requires at most $4\ell \log(2m \log m) = O(n \log n)$ (using that $4\ell \leq n$ and that $m = n^{O(1)}$). Finally, describing a circuit with ℓ gates and $(n + 2\ell)$ input bits where $4\ell \leq n$ requires $O(n \log n)$ bits. Putting this all together and using that $m = O(n^3)$, we get that T can be described using

$$n + (n(1 - \frac{1}{2r}) + 2)(2 \log n + m/n + 1) + n^2 + O(n \log n) = (1 - \frac{1}{2r})m + O(n^2) = (1 - \Omega(1))m$$

bits. Thus, the number of bad T is upper bounded by $2^{(1-\Omega(1))m}$, so the probability that T is bad is at most $2^{-\Omega(m)}$. ◀

4.3 Search-to-decision reduction for Multi-MCSP

A similar "bottom-up" search-to-decision reduction to the one for Partial-MCSP works for Multi-MCSP. Recall, in the Search-Multi-MCSP problem, the goal is a to output a Boolean circuit C of minimum size computing a (multi-output) Boolean function f.

We will now describe a deterministic polynomial-time procedure to solve Search-Multi-MCSP using an oracle to Multi-MCSP. In our algorithm, we will assume access to an oracle that computes CC, the exact circuit complexity of a (multi-output) Boolean function, but one can compute CC efficiently using an oracle to Multi-MCSP. Finally, our algorithm will work recursively and actually solve a slightly stronger problem.

Our algorithm makes use of the Evaluation Function Eval-C defined in Subsection 4.1, where our precise notion of multi-output computation can also be found. Additionally, we say a circuit C is a *subcircuit* of a circuit D if D can be obtained by adding gates to C.

Algorithm E.

Input. The truth table of a multi-output Boolean function f with n inputs and a circuit C with n input variables $x_1, \ldots, x_n$ and s gates $g_1, \ldots, g_s$.

Output. A minimum-sized circuit D computing f among circuits containing C as a subcircuit.

1. If C computes f, then return C.
2. Otherwise, for each operation $\star \in \{\vee, \wedge, \neg\}$, and for each appropriate choice of one or two operands from $\{x_1, \ldots, x_n, g_1, \ldots, g_s\}$:
   - 2.1 Let C^+ be the circuit obtained by adding a $\star$ gate to C with the chosen operands.
   - 2.2 If $\mathsf{CC}(\text{Eval-}C^+) > \mathsf{CC}(\text{Eval-}C)$ and $\mathsf{CC}(f \bullet \text{Eval-}C^+) = \mathsf{CC}(f \bullet \text{Eval-}C)$, then output $E(f, C^+)$.

If Algorithm E works as claimed, then it is easy to see that invoking E on a multi-output Boolean function f and an empty circuit yields the desired search-to-decision reduction.

We now sketch the proof that E runs in polynomial-time and returns the claimed output on input (f, C). We argue by induction on the quantity $s \stackrel{\text{def}}{=} \mathsf{CC}(f \bullet \text{Eval-}C) - \mathsf{CC}(\text{Eval-}C)$ (intuitively, s is the minimum number of gates that need to be added to C in order to compute f).

If $s = 0$, then it must be that C computes f. (By a similar argument to Proposition 14, $\mathsf{CC}(\text{Eval-}C)$ is exactly the number of distinct components of Eval-C not computed by input wires. It follows that any function computed by a gate in an optimal circuit for Eval-C must be a component of Eval-C and therefore be computed by C.) Thus, Algorithm E correctly returns C in step (1).

Now assume that $s \geq 1$. Let D be a circuit computing f of minimum-size among circuits containing C as a subcircuit.

▷ Claim 17. There is a gate in D whose function that computes a function h such that h is not computed in C and such that h can be computed by applying an operator $\star \in \{\vee, \wedge, \neg\}$ to operand(s) solely from the set $\{x_1, \ldots, x_n, g_1, \ldots, g_s\}$.

Proof. Imagine labeling as depth-0 all the input variables and gates in D whose functions are computed in C. Next, inductively define all other gates to have depth one more than the maximum depth of their input gates. Since D computes f and C does not, there is at least one gate in D with positive depth. Therefore there must be one gate with depth-1. Any gate with depth-1 satisfies the property that it is not computed in C and can be computed by adding a single operator to C. ◁

As a consequence of this, we get that some C^+ will pass the test in Step (2.2).

▷ Claim 18. At least one choice of operator and operand(s) in Step (2.1) will pass the test in Step (2.2).

Proof. Let h be the function guaranteed by Claim 17 that is computed by some choice of apply some operator $\star \in \{\vee, \wedge, \neg\}$ to some operand(s) solely from the set $\{x_1, \ldots, x_n, g_1, \ldots, g_s\}$, and let C^+ be the circuit obtained by adding this gate to C. Since h is not computed by C and C^+ is obtained by adding a gate to C, it follows that $\mathsf{CC}(\mathsf{Eval}\text{-}C^+) > \mathsf{CC}(\mathsf{Eval}\text{-}C)$. Next, since h is computed by D and C is a subcircuit of D, it follows that every function computed by a gate in C^+ is computed by a gate in D. Thus, we have that $\mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^+) \leq |D|$. Combining this with the optimality of D and the fact that C is a subcircuit of C^+, we have that

$$\mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C) \leq \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^+) \leq |D| = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C).$$

Therefore, $\mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^+) = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C)$, and so C^+ passes the test in Step (2.2). ◁

Now, let C^+ be any circuit that passes the test in Step (2.2). Then the quantity

$$s^+ = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^+) - \mathsf{CC}(\mathsf{Eval}\text{-}C^+) < \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C) - \mathsf{CC}(\mathsf{Eval}\text{-}C) = s$$

using the test conditions in Step (2.2). Thus, by the inductive hypothesis, $E(f, C^+)$ returns a circuit D that is a minimum sized circuit for f among circuits containing C^+ as a subcircuit. Since C^+ contains C as a subcircuit, it follows that D also contains C as a subcircuit. Moreover, $|D| = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^+) = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C)$ (by a test condition). Hence, D is a minimum-sized circuit for f among circuits containing C as a subcircuit, as desired.

Finally, we argue that Algorithm E runs in polynomial-time on input (f, C^1). Let T be the truth table of f. Then a lower bound on the input length is $m = |T| + |C^1|$. We will show that E runs in time polynomial in m.

First, we upper bound the number of recursive calls c. Let $(f, C^1), \ldots, (f, C^c)$ denote the successive inputs to E made by the recursive calls where (f, C^1) is the original input. Using induction on the two test conditions in Step (2.2), we have that

$$\mathsf{CC}(\mathsf{Eval}\text{-}C^c) \geq \mathsf{CC}(\mathsf{Eval}\text{-}C^1) + c - 1$$

and that

$$\mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^c) = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^1).$$

On the other hand, we have that

$$\mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^1) \leq \mathsf{CC}(\mathsf{Eval}\text{-}C^1) + O(|T| \log |T|)$$

as witnessed by the circuit that uses trivial DNFs to compute each component of f individually and a minimum-sized circuit for $\mathsf{Eval}\text{-}C^1$ to compute $\mathsf{Eval}\text{-}C^1$. Putting these facts together, we get that

$$\begin{aligned}
\mathsf{CC}(\mathsf{Eval}\text{-}C^1) + c - 1 &\leq \mathsf{CC}(\mathsf{Eval}\text{-}C^c) \\
&\leq \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^c) \\
&= \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^1) \\
&\leq \mathsf{CC}(\mathsf{Eval}\text{-}C^1) + O(|T| \log |T|),
\end{aligned}$$

so the number of recursive calls $c = O(|T| \log |T|) = O(m^2)$.

Next, we analyze the computation required in recursive call $i \in [c]$. Since Algorithm E adds at most one gate to C in each recursive call and $i \leq c$, we have by induction that

$$|C^i| \leq |C^1| + c - 1 = O(|C^1| + m^2) = O(m^2).$$

Therefore, the computation in Step (1) of checking whether C^i computes T can be done in time $O(|T||C^i|) = O(m^3)$ (by just evaluating C^i on all inputs). Next, trying all possible operands on all pairs of input variables and circuit gates from C^i in Step (2) takes at most $O((|C^i| + n)^2) = O(m^4)$ time. Lastly, it is easy to see that Steps (2.1) and Steps (2.2) run in $O(m)$ time. Thus, each recursive step of Algorithm E runs in time $O(m^4)$ and there are $O(m^2)$ recursive calls, so Algorithm E runs in time $O(m^6)$.

5 On the NP-hardness of communication minimization problems

5.1 Background

Hardness of graph coloring. Our NP-hardness reduction is from the chromatic number problem:

▶ **Definition 19** (Chromatic number). *A* coloring *of an undirected graph G, is a partition of the vertices such that no edge has both endpoints in the same part. The* chromatic number *of a graph G, denoted $\chi(G)$, is the smallest number of parts of a coloring of G.*

The NP-hardness of approximating the chromatic number has been established by a series of results [57, 31, 24], culminating in a paper by Zuckerman [82], where the following was proven:

▶ **Theorem 20** ([82]). *For every constant $\varepsilon > 0$ it is NP-hard to approximate $\chi(G)$ for a given n-vertex graph G, with an approximation ratio better than $n^{1-\varepsilon}$. More precisely, for every $L \in$ NP and $\varepsilon > 0$, there exists a polynomial-time algorithm that, on input x, outputs a parameter k and an n-vertex graph G such that if $x \in L$ then $\chi(G) \leq k$, and if $x \notin L$ then $\chi(G) > n^{1-\varepsilon} \cdot k$.*

In the reductions above, the parameter k is $\Theta(n^\varepsilon)$. More recent results on the hardness of approximating chromatic number allow for a different gap, where k is constant as n grows, and we wish to distinguish graphs which are k-colorable from graphs which are not $g(k)$-colorable, for a fast-growing function $g\colon \mathbb{N} \to \mathbb{N}$. An original such result with $g(k) = k^{\Omega(\log k)}$ was shown by Khot [47], which was later improved to $g(k) = 2^{\Omega(k^{1/3})}$ by Huang [37]. The latest result, by Wrochna and Živný [79], achieves $g(k) = \binom{k}{\lfloor k/2 \rfloor}$:

▶ **Theorem 21** ([79]). *Let $k \geq 4$ be a natural number. For every $L \in$ NP, there exists a polynomial-time algorithm that, on input x, outputs an graph G such that if $x \in L$ then $\chi(G) \leq k$, and if $x \notin L$ then $\chi(G) > \binom{k}{\lfloor k/2 \rfloor}$.*

Communication complexity of relations and partial functions. We will need the following definitions.

▶ **Definition 22.** *We will call a* two-player communication relation*, or simply a* relation*, to any subset $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, where $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ are finite sets, such that, for every $(x, y) \in \mathcal{X} \times \mathcal{Y}$, there exists at least one $z \in \mathcal{Z}$ with $(x, y, z) \in F$.*

Given a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, and a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$, we let

$$F(x, y) = \{z \in \mathcal{Z} \mid (x, y, z) \in F\} \neq \varnothing.$$

The matrix with $\mathcal{X}$-indexed rows and $\mathcal{Y}$-indexed columns, whose (x, y) entry is $F(x, y)$, is called the communication matrix *of F.*

Given a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, a rectangle $R = A \times B \subseteq \mathcal{X} \times \mathcal{Y}$, and an element $z \in \mathcal{Z}$, we say R is z-monochromatic for F, *if $(x, y, z) \in F$ for all $(x, y) \in R$. We say R is* monochromatic for F *if there exists a $z \in \mathcal{Z}$ with R being z-monochromatic for F.*

A partial two-player function *is a relation $f \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ such that, for every $(x, y) \in \mathcal{X} \times \mathcal{Y}$, either $|f(x, y)| = 1$, in which case we say f is* defined *at (x, y), or $f(x, y) = \mathcal{Z}$, in which case we say f is* undefined *at (x, y). If f is defined at (x, y), we write $f(x, y) = z$ in place of $f(x, y) = \{z\}$, and if f is undefined at (x, y), we write $f(x, y) = *$, instead of $f(x, y) = \mathcal{Z}$.*

▶ **Definition 23** ($\mathsf{P}(F)$)**.** *Given a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, a* partition *of F is a family Π of monochromatic rectangles for F, such that every $(x, y) \in \mathcal{X} \times \mathcal{Y}$ appears in exactly one rectangle of Π. The* partition number *of a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, denoted $\mathsf{P}(F)$ is the smallest possible size of a partition of F.*

▶ **Definition 24** ($\mathsf{C}(F)$)**.** *Given a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, a* cover *of F is a family Γ of monochromatic rectangles for F, such that every $(x, y) \in \mathcal{X} \times \mathcal{Y}$ appears in at least one rectangle of Γ. The* cover number *of a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, denoted $\mathsf{C}(F)$ is the smallest possible size of a cover of F.*

In the context of Boolean-valued functions, the usual notion of *cover number* (see [42] §4.2, p. 97), and related notion of non-deterministic communication complexity, only require that the 1s of the communication matrix are covered.

▶ **Definition 25** ($\mathsf{C}_1(F)$)**.** *Given a partial two-player function $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, a* 1-cover *of F is a family Γ of 1-monochromatic rectangles for F, such that every $(x, y) \in \mathcal{X} \times \mathcal{Y}$ having $f(x, y) = 1$ appears in at least one rectangle of Γ. The* 1-cover number *of f, denoted $\mathsf{C}_1(F)$ is the smallest possible size of a 1-cover of F. We then define the* non-deterministic communication complexity *of f to be $\mathsf{N}(f) = \lceil \log \mathsf{C}_1(f) \rceil$.*

▶ **Definition 26.** *A* protocol *π over $\mathcal{X} \times \mathcal{Y}$ is a rooted tree:*

- *Each node v is associated with a rectangle $\pi^{-1}(v) = A \times B \subseteq \mathcal{X} \times \mathcal{Y}$.*
- *Each non-leaf node v, with $\pi^{-1}(v) = A \times B$, is labeled by either (a) a partition $A = A_0 \cup A_1$ of A, in which case we say it is* Alice's node *or (b) a partition $B = B_0 \cup B_1$ of B, in which case we say it is* Bob's node.
- *The rectangle associated with the root is $\mathcal{X} \times \mathcal{Y}$.*
- *If a non-leaf node v of Alice has $\pi^{-1}(v) = A \times B$ and is labeled by a partition $A = A_0 \cup A_1$ of A, then for each $c \in \{0, 1\}$ there will be a unique child v_c of v, with $\pi^{-1}(v_c) = A_c \times B$; similarly for Bob's nodes.*

▶ **Definition 27** ($\mathsf{L}(F)$, $\mathsf{D}(F)$)**.** *Let $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ be a relation, and π a protocol over $\mathcal{X} \times \mathcal{Y}$. We say that π is* a protocol for F *if, for any leaf ℓ of π, the rectangle $\pi^{-1}(\ell)$ is monochromatic for F. We then let $\mathsf{L}(F)$ be the smallest possible number of leaves in a protocol for F, and we let $\mathsf{D}(F)$ be the smallest possible depth of a protocol for F.*

Note that the rectangles associated with the leaves of a protocol for F form a partition of F, any partition of F is a cover of F, and any cover of F contains a 1-cover of F, and hence we get the following:

▶ **Lemma 28.** *For any relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, $\mathsf{L}(F) \geq \mathsf{P}(F) \geq \mathsf{C}(F)$, and for a partial two-player function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, $\mathsf{C}(f) \geq \mathsf{C}_1(f)$.*

▶ **Definition 29.** *A* DAG-like protocol *γ over $\mathcal{X} \times \mathcal{Y}$ is a directed acyclic graph:*

- *γ has a single source node, called the* root*, and one or more sink nodes, called* leaves*.*
- *Each node v is associated with a rectangle $\gamma^{-1}(v) = A \times B \subseteq \mathcal{X} \times \mathcal{Y}$.*
- *Each non-leaf node v, with $\gamma^{-1}(v) = A \times B$, is labeled by either (a) a partition $A = A_0 \uplus A_1$ of A, in which case we say it is* Alice's node *or (b) a partition $B = B_0 \uplus B_1$ of B, in which case we say it is* Bob's node*.*
- *The rectangle associated with the root is $\mathcal{X} \times \mathcal{Y}$.*
- *If a non-leaf node v of Alice has $\gamma^{-1}(v) = A \times B$ and is labeled by a partition $A = A_0 \uplus A_1$ of A, then for each $c \in \{0,1\}$ there will be an edge from v to a node v_c of γ, which must be such that $A_c \times B \subseteq \gamma^{-1}(v_c)$; similarly for Bob's nodes.*

▶ **Definition 30** ($\mathsf{S}(F)$)**.** *Let $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ be a relation, and γ a DAG-like protocol over $\mathcal{X} \times \mathcal{Y}$. We say that γ is* a DAG-like protocol for F *if, for any leaf ℓ of γ, the rectangle $\gamma^{-1}(\ell)$ is monochromatic for F. We then let $\mathsf{S}(F)$ be the smallest possible number of nodes in a DAG-like protocol for F.*

Note that: (1) a protocol with k leaves is a DAG-like protocol with $2k-1$ nodes, (2) the rectangles associated with the leaves of a DAG-like protocol form a cover of F, and (3) any DAG with 1 source node, k sink nodes, and maximum out-degree 2 has at least $2k-1$ nodes in total. Hence:

▶ **Lemma 31.** $\mathsf{S}(F) \leq 2\mathsf{L}(F) - 1$ *and* $\mathsf{S}(F) \geq 2\mathsf{C}(F) - 1$.

5.2 A reduction from Graph Coloring to Partial-MCCP

Let us be given a graph $G = ([n], E)$ with $E \subseteq \binom{[n]}{2}$, and consider the partial two-player Boolean function $f_G : [n] \times [n] \to \{0, 1, *\}$, given by

$$f_G(i,j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } \{i,j\} \in E, \\ * & \text{if } \{i,j\} \notin E. \end{cases}$$

By $f_G(i,j) = *$ we mean that $f_G(i,j)$ is undefined, which is to say Alice and Bob may output any value when given (i,j) as input.

▶ **Theorem 32.** *We have $\mathsf{L}(f_G) \leq 2\chi(G)$ and $\mathsf{C}_1(f_G) = \chi(G)$.*

Proof. To show that $\mathsf{L}(f_G) \leq 2\chi(G)$, consider the simple protocol where Alice and Bob have agreed on a coloring of G, so Alice begins by sending the color of her vertex, and Bob replies whether the color of his vertex is the same. If the two colors match, they output 1, and otherwise they output 0.

This is a protocol for f, since a valid coloring of G will only color i and j by the same color if $i = j$, or $\{i,j\} \notin E$, in which case 1 is a valid output. If i and j are colored differently, then $i \neq j$, and so 0 is a valid output. This protocol has $2\chi(G)$ leaves.

To show that $\mathsf{C}_1(f_G) \leq \chi(G)$, we cover the diagonal by 1-monochromatic rectangles; each such rectangle corresponds to a color c, and is the smallest rectangle containing all diagonal entries (i,i) for vertices i of G that are colored c.

Now suppose we have any positive cover Γ for f. The diagonal entries must be covered by 1-monochromatic rectangles of Γ, so consider the coloring of G which colors vertex $i \in [n]$ by the 1-monochromatic rectangle which covers (i, i). By our choice of f, any two diagonal entries belonging to the same 1-monochromatic rectangle are not connected by an edge of G. And so this gives us a valid coloring of G, which implies that $\chi(G) \leq |\Gamma|$. ◀

5.3 Consequences of the reduction

We may now take Theorems 32 and 20, together with Lemmas 28 and 31, to obtain:

▶ **Corollary 33.** *For any constant $\varepsilon > 0$, it is* NP*-hard to approximate* $\mathsf{L}(f)$, $\mathsf{P}(f)$, $\mathsf{C}(f)$, $\mathsf{C}_1(f)$ *and* $\mathsf{S}(f)$, *for a given partial function $f : [n] \times [n] \to \{0, 1, *\}$, with an approximation ratio better than $n^{1-\varepsilon}$.*

Observe that this hardness of approximation result with a ratio of $n^{1-\varepsilon}$ is very close to optimal, since all aforementioned measures for a partial two-player function $f : [n] \times [n] \to \{0, 1, *\}$ are upper-bounded by $2n$. Improving upon this might still be possible, and we leave it as an open problem.

Now notice that the protocol given in the proof of Theorem 20 is balanced. For this reason, its depth is logarithmic in the number of leaves. Thus, since computing $\mathsf{L}(f)$ is hard up to an approximation ratio of $n^{1-\varepsilon}$, it follows that computing the communication complexity $\mathsf{D}(f)$ for a given partial function $f : [n] \times [n] \to \{0, 1, *\}$ is hard, even if we allow for an additive error term of $(1 - \varepsilon) \log n$. Since $\mathsf{D}(f) \leq \log n$, this implies that we cannot approximate $\mathsf{D}(f)$ with an approximation ratio better than $\approx \frac{1}{\varepsilon}$, which we can take as an arbitrarily large constant. The same reasoning applies to non-deterministic communication complexity.

▶ **Corollary 34.** *For any constant $\varepsilon > 0$, it is* NP*-hard to approximate* $\mathsf{D}(f)$ *or* $\mathsf{N}(f)$, *for a given partial function $f : [n] \times [n] \to \{0, 1, *\}$, with an error term smaller than $(1 - \varepsilon) \log n$. For any constant $c > 1$, it is* NP*-hard to approximate* $\mathsf{D}(f)$ *or* $\mathsf{N}(f)$, *for a given partial function $f : [n] \times [n] \to \{0, 1, *\}$, with an approximation ratio better than c.*

Let us now prove that, assuming $\mathsf{P} \neq \mathsf{NP}$, there is no polynomial-time computable approximate characterization of the communication complexity of a partial Boolean function. This is captured by a more general result, which is an immediate consequence of the reduction presented in Section 5.2 and Theorem 21.

▶ **Theorem 35.** *Let $\lambda, \eta : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be functions such that $\eta(1 + \log x) < \lambda(0.99x)$ for all sufficiently large x. Moreover, assume that these functions are non-decreasing for every large enough x. Then, if $\mathsf{P} \neq \mathsf{NP}$, there is no polynomial-time computable function r, which accepts as input the communication matrix $M_f \in \{0, 1, *\}^{n \times n}$ of a partial Boolean function $f : [n] \times [n] \to \{0, 1, *\}$, and for every large enough n outputs a value $r(M_f)$ such that*

$$\lambda(\mathsf{D}(f)) \leq r(M_f) \leq \eta(\mathsf{D}(f)).$$

Proof. Suppose that functions λ, η, and r exist as described in the statement of the theorem. Let L be any language in NP, and assume that k is a sufficiently large constant. For any $x \in \{0, 1\}^n$, let G_x be the corresponding graph given by Theorem 21. Then let $f_x = f_{G_x}$ be as defined in Section 5.2 above, and let M_x be the communication matrix of f_x. If $x \in L$, we then have by Theorem 32 that $k \geq \chi(G_x) \geq \frac{1}{2}\mathsf{L}(f_x)$, and because the protocol for f_x that witnesses this fact is balanced, we get $1 + \log k \geq \mathsf{D}(f_x)$. Hence $\eta(1 + \log k) \geq \eta(\mathsf{D}(f_x)) \geq r(M_x)$. If, on the other hand, $y \notin L$, then we have $2^{0.99k} < \binom{k}{\lfloor k/2 \rfloor} < \chi(G_y) = \mathsf{C}_1(f_y) \leq \mathsf{L}(f_y)$, and

thus $0.99k < \mathsf{D}(f_y)$. Then $\lambda(0.99k) \leq \lambda(\mathsf{D}(f_y)) \leq r(M_y)$. But since $\lambda(0.99k) > \eta(1 + \log k)$ using our assumptions on these functions and on k, it follows that $r(M_y) > r(M_x)$. As a consequence, the polynomial time function r can be used to distinguish positive and negative instances of L. Since L is an arbitrary language in NP, we get that $\mathsf{P} = \mathsf{NP}$. This completes the proof. ◀

References

1 Misha Alekhnovich, Mark Braverman, Vitaly Feldman, Adam R. Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *Journal of Computer and System Sciences*, 74(1):16–34, 2008.

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

3 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.

4 Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing disjunctive normal form formulas and AC^0 circuits given a truth table. *SIAM Journal on Computing*, 38(1):63–84, 2008.

5 Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. In *International Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 54:1–54:14, 2017.

6 Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. *Computational Complexity*, 26(2):469–496, 2017.

7 Eric Allender, Rahul Ilango, and Neekon Vafa. The non-hardness of approximating circuit size. In *International Computer Science Symposium in Russia* (CSR), pages 13–24, 2019.

8 Anurag Anshu, Naresh Goud Boddu, and Dave Touchette. Quantum log-approximate-rank conjecture is also false. *arXiv*, 2018. `arXiv:1811.10525`.

9 Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Symposium on Foundations of Computer Science* (FOCS), pages 211–220, 2008.

10 Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

11 Avrim Blum, Merrick L. Furst, Michael J., and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *International Cryptology Conference* (CRYPTO), pages 278–291, 1993.

12 Avrim L. Blum and Ronald L. Rivest. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1):117–127, 1992.

13 Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, 1987.

14 Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006.

15 Dan Boneh and Richard J. Lipton. Amplification of weak learning under the uniform distribution. In *Conference on Learning Theory* (COLT), pages 347–351, 1993.

16 Joan Boyar, Philip Matthews, and René Peralta. On the shortest linear straight-line program for computing linear forms. In *International Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 168–179, 2008.

17 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Conference on Computational Complexity* (CCC), pages 10:1–10:24, 2016.

18 Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-Hamming-distance. *SIAM Journal on Computing*, 41(5):1299–1317, 2012. doi:10.1137/120861072.

19 Arkadev Chattopadhyay, Nikhil S Mande, and Suhail Sherif. The log-approximate-rank conjecture is false. In *Symposium on Theory of Computing* (STOC), pages 42–53. ACM, 2019.

20 Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit lower bounds for MCSP from local pseudorandom generators. *Electronic Colloquium on Computational Complexity* (ECCC), 26:22, 2019.

21 Gil Cohen and Igor Shinkar. The complexity of DNF of parities. In *Innovations in Theoretical Computer Science* (ITCS), pages 47–58, 2016.

22 Sebastian Czort. The complexity of minimizing disjunctive normal form formulas. Master's Thesis, University of Aarhus, 1999.

23 Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

24 Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57(2):187–199, 1998.

25 Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A better-than-3n lower bound for the circuit complexity of an explicit function. In *Symposium on Foundations of Computer Science* (FOCS), pages 89–98, 2016.

26 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

27 Alexander Golovnev, Rahul Ilango, Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, and Avishay Tal. $\mathrm{AC}^0[p]$ lower bounds against MCSP via the coin problem. *Electronic Colloquium on Computational Complexity* (ECCC), 26:18, 2019.

28 Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM Journal on Computing*, 47(5):1778–1806, 2018. doi:10.1137/16M1082007.

29 Gary D. Hachtel and Fabio Somenzi. *Logic synthesis and verification algorithms*. Springer, 2006.

30 Thomas R. Hancock, Tao Jiang, Ming Li, and John Tromp. Lower bounds on learning decision lists and trees. *Information and Computation*, 126(2):114–122, 1996.

31 Johan Hastad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Symposium on Foundations of Computer Science* (FOCS), pages 627–636, 1996.

32 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Symposium on Foundations of Computer Science* (FOCS), pages 247–258, 2018.

33 Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. NP-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *Computational Complexity Conference* (CCC), pages 5:1–5:31, 2018.

34 Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *Computational Complexity Conference* (CCC), pages 7:1–7:20, 2017.

35 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Conference on Computational Complexity* (CCC), pages 18:1–18:20, 2016.

36 John M. Hitchcock and Aduri Pavan. On the NP-completeness of the minimum circuit size problem. In *Conference on Foundation of Software Technology and Theoretical Computer Science* (FSTTCS), pages 236–245, 2015.

37 Sangxia Huang. Improved hardness of approximating chromatic number. In *International Conference on Approximation Algorithms for Combinatorial Optimization Problems* (APPROX), pages 233–243, 2013.

38 Rahul Ilango. $\mathrm{AC}^0[p]$ lower bounds and NP-hardness for variants of MCSP. *Electronic Colloquium on Computational Complexity* (ECCC), 26:21, 2019.

39 Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The Power of Natural Properties as Oracles. In *Computational Complexity Conference* (CCC), volume 102, pages 7:1–7:20, 2018.

40 Kazuo Iwama and Hiroki Morizumi. An explicit lower bound of $5n - o(n)$ for boolean circuits. In *International Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 353–364, 2002.

41 J. Stephen Judd. Learning in networks is hard. In *International Conference on Neural Networks* (ICNN), volume 2, pages 685–692, 1987.

42 Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer, 2012.

43 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Symposium on Theory of Computing* (STOC), pages 73–79, 2000.

44 Maurice Karnaugh. The map method for synthesis of combinational logic circuits. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, 72(5):593–599, 1953.

45 Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994. doi:10.1145/174644.174647.

46 Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

47 Subhash Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In *Symposium on Foundations of Computer Science* (FOCS), pages 600–609, 2001.

48 Subhash Khot and Rishi Saket. Hardness of minimizing and learning DNF expressions. In *Symposium on Foundations of Computer Science* (FOCS), pages 231–240, 2008.

49 Ker-I Ko. On the complexity of learning minimum time-bounded Turing machines. *SIAM Journal on Computing*, 20(5):962–986, 1990.

50 Pravesh K. Kothari and Roi Livni. Improper learning by refuting. In *Innovations in Theoretical Computer Science* (ITCS), pages 55:1–55:10, 2018.

51 Jan Krajícek. *Forcing with Random Variables and Proof Complexity*. Cambridge University Press, 2011.

52 Eyal Kushilevitz. Communication complexity. In *Advances in Computers*, volume 44, pages 331–360. Elsevier, 1997.

53 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

54 Eyal Kushilevitz and Enav Weinreb. On the complexity of communication complexity. In *Symposium on Theory of Computing* (STOC), pages 465–474, 2009.

55 Leonid Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

56 László Lovász and Michael Saks. Lattices, mobius functions and communications complexity. In *Symposium on Foundations of Computer Science* (FOCS), pages 81–90, 1988.

57 Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

58 William J. Masek. Some NP-complete set covering problems. Unpublished Manuscript, 1979.

59 Edward J. McCluskey. *Introduction to the theory of switching circuits*. McGraw-Hill, 1965.

60 Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Symposium on Theory of Computing* (STOC), 2019.

61 Moritz Müller and Ján Pich. Feasibly constructive proofs of succinct weak circuit lower bounds. *Electronic Colloquium on Computational Complexity* (ECCC), 24:144, 2017.

62 Cody D. Murray and Richard Ryan Williams. On the (non) NP-hardness of computing circuit complexity. In *Conference on Computational Complexity* (CCC), pages 365–380, 2015.

63 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *Conference on Computational Complexity* (CCC), 2019.

64 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017.

65 Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *Symposium on Foundations of Computer Science* (FOCS), pages 65–76, 2018.

66 James Orlin. Contentment in graph theory: covering graphs with cliques. *Indagationes Mathematicae*, 80(5):406–424, 1977.

67 Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

68 Pavel Pudlák, Vojtech Rödl, and Petr Savický. Graph complexity. *Acta Informatica*, 25(5):515–535, 1988.

69 Netanel Raviv. Truth table minimization of computational models. *CoRR/arXiv*, abs/1306.3766, 2013. `arXiv:1306.3766`.

70 Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

71 Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

72 J. Paul Roth and Richard M. Karp. Minimization over boolean graphs. *IBM Journal of Research and Development*, 6(2):227–238, 1962.

73 Christoph Scholl. *Functional decomposition with applications to FPGA synthesis*. Kluwer Academic Publishers, 2001.

74 Makrand Sinha and Ronald de Wolf. Exponential separation between quantum communication and logarithm of approximate rank. *arXiv*, 2018. `arXiv:1811.10090`.

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

76 Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Symposium on Theory of Computing* (STOC), pages 453–461, 2001.

77 Christopher Umans, Tiziano Villa, and Alberto L. Sangiovanni-Vincentelli. Complexity of two-level logic minimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 25(7):1230–1246, 2006.

78 Salil P. Vadhan. On learning vs. refutation. In *Conference on Learning Theory* (COLT), pages 1835–1848, 2017.

79 Marcin Wrochna and Stanislav Živnỳ. Improved hardness for H-colourings of G-colourable graphs. *arXiv*, 2019. `arXiv:1907.00872`.

80 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Symposium on Theory of Computing* (STOC), pages 209–213, 1979. `doi:10.1145/800135.804414`.

81 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *Symposium on Foundations of Computer Science* (FOCS), pages 80–91, 1982.

82 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Symposium on Theory of Computing* (STOC), pages 681–690, 2006.

A The connection between average-case Partial-MCSP and learning

A few years ago, [17] established an equivalence between solving MCSP on average and learning Boolean circuits under the uniform distribution using *membership queries*.[15] In this section, we describe a similar equivalence between the average-case complexity of Partial-MCSP and learning Boolean circuits under the uniform distribution using *random examples*. The result is implicit in the work of [78], and similar ideas have appeared in the literature before (see e.g. [11]). Here we simply translate these ideas to the language of circuit minimization.[16]

[15] See [34] for a discussion on the average-case complexity of MCSP and its connection to natural proofs [70].

[16] Note that [78] considers learnability in the PAC model (i.e. with respect to arbitrary distributions), while here we focus on learnability under the uniform distribution.

First, we formalize the learning model. For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, an *example oracle* $\mathcal{EX}(f)$ for f is a procedure that when invoked returns a pair $(x, f(x))$ consisting of a uniformly distributed string $x \in \{0,1\}^n$ and its label $f(x)$. We say that a circuit C is ε-*close* to a function f if $\Pr_x[C(x) \neq f(x)] \leq \varepsilon$. A randomized algorithm A *learns* a class of Boolean functions $\mathcal{F}$ with *accuracy* ε and *confidence* δ if, for every $f \in \mathcal{F}$ of the form $f\colon \{0,1\}^n \to \{0,1\}$, when A is given access to an example oracle $\mathcal{EX}(f)$,

$$\Pr_{A,\,\mathcal{EX}(f)}[A^{\mathcal{EX}(f)}(1^n) \text{ outputs a circuit } C \text{ that is } \varepsilon(n)\text{-close to } f\,] \;\geq\; 1 - \delta(n).$$

The confidence parameter $\delta(n)$ can be easily boosted without a significant increase of running time. In particular, a learner that succeeds with probability at least $1/\mathsf{poly}(n)$ can be transformed into a learner that fails with probability at most $1/\mathsf{poly}(n)$ with just a polynomial overhead in the running time (cf. [46]). For this reason, the discussion below will concentrate on the accuracy parameter ε, implicitly assuming that $\delta(n) = 1/n$.

We will focus on the class $\mathcal{F} = \mathsf{SIZE}[s]$ that corresponds to Boolean functions computable by (unrestricted) Boolean circuits of size at most s, where $s\colon \mathbb{N} \to \mathbb{N}$. For simplicity, we focus on the regime where $s = \mathsf{poly}(n)$.

Next, we make precise the notion of average-case complexity of Partial-MCSP. We say that a randomized algorithm B *solves* Partial-MCSP (over dimension n) *on average* with *advantage* γ for a *size parameter* s using t *samples* if, for every $f \in \mathsf{SIZE}[s]$ of the form $f\colon \{0,1\}^n \to \{0,1\}$, we have:

$$\left| \Pr_{B,\,\{z^i\}_{i\in[t]}}[B(1^n, z^1, f(z^1), \ldots, z^t, f(z^t)) = 1\,] - \Pr_{B,\,\{z^i\}_{i\in[t]},\,b}[B(1^n, z^1, b_1, \ldots, z^t, b_t)) = 1\,] \right| \;\geq\; \gamma(n),$$

where $b \in \{0,1\}^t$ and $z^1, \ldots, z^t \in \{0,1\}^n$ are independent and uniformly random, and $t = t(n)$.

▶ **Theorem 36.** *The following implications hold.*

(1) *For every $c, d \in \mathbb{N}$ there exists $\ell \in \mathbb{N}$ such that if $\mathsf{SIZE}[n^c]$ can be learned in time $O(n^d)$ with accuracy $\varepsilon = 1/10$, then* Partial-MCSP *over dimension n can be solved on average in polynomial time with advantage $\gamma(n) \to_n 1$ for $s(n) = n^c$ using $t(n) = n^\ell$ samples.*

(2) *If for every $c \in \mathbb{N}$ there exists $\ell \in \mathbb{N}$ such that* Partial-MCSP *over every dimension n can be solved on average in polynomial time with advantage $\gamma = 1/10$ for $s(n) = n^c$ using $t(n) = n^\ell$ samples, then for every $a \in \mathbb{N}$ the class $\mathsf{SIZE}[n^a]$ can be learned in polynomial time with accuracy $\varepsilon(n) = 1/n$.*

In other words, Theorem 36 says that polynomial-size Boolean circuits over $\{0,1\}^n$ can be learned in polynomial time using random examples if and only if Partial-MCSP over $\{0,1\}^{\mathsf{poly}(n)}$ can be solved on average in polynomial time.

Proof Sketch. We start with the proof of (1). Let $\ell \stackrel{\text{def}}{=} n^{d+1} + n$. Assuming the existence of a learning algorithm A for $\mathsf{SIZE}[n^c]$ with accuracy $\varepsilon = 1/10$ and confidence $\delta = 1/n$, the algorithm B for average-case Partial-MCSP employs A as a sub-routine, and computes as follows. Given 1^n and a sequence $(z^1, a_1), \ldots, (z^t, a_t)$ in $\{0,1\}^{t\cdot(n+1)}$, where $t(n) \stackrel{\text{def}}{=} n^\ell$, B uses the first n^{d+1} pairs (z^i, a_i) to simulate the answers to the oracle calls made by A to its example oracle. Let C be the circuit output by $A^{\mathcal{EX}(\cdot)}(1^n)$ after its computation. B uses the last n input pairs (z^i, a_i) to compute the fraction $\alpha \in [0,1]$ of such pairs for which $C(x^i) \neq a_i$. Finally, B outputs 1 if and only if $\alpha \leq 1/3$. Note that B runs in polynomial time under the assumption that A runs in time $O(n^d)$.

Let $f \in \mathsf{SIZE}[n^c]$. In this case, $\Pr_{B,\,\{z^i\}_{i\in[t]}}[B(1^n, z^1, f(z^1), \ldots, z^t, f(z^t)) = 1\,] \geq 1 - 2/n$, since with probability at most $1/n$ algorithm A fails to output a circuit C that is $(1/10)$-close to f, and by a standard concentration bound with probability at most $1/n$ we have $\alpha > 1/3$. On the other hand, it is not hard to see by a standard concentration bound that $\Pr_{B,\,\{z^i\}_{i\in[t]},\,b}[B(1^n, z^1, b_1, \ldots, z^t, b_t)) = 1\,] \leq 1/n$, since in this case no matter the circuit C output by A, the last n input pairs of B contain random bits b_i that are uncorrelated with C. This shows that B has advantage $\gamma(n) \to_n 1$.

In order to prove (2), it is enough to conclude that for every $a \in \mathbb{N}$ there is $\ell \in \mathbb{N}$ such that the class $\mathsf{SIZE}[n^a]$ can be learned to accuracy $\varepsilon(n) = 1/2 - 1/n^{\ell+1}$ in polynomial time. This claim follows from a result of [15, Section 2] showing in particular that, when learning general polynomial-size Boolean circuits under the uniform distribution from random examples, one can boost the accuracy parameter from $\varepsilon(n) = 1/2 - 1/\mathsf{poly}(n)$ to $\varepsilon(n) = 1/\mathsf{poly}(n)$ with only a polynomial overhead in the running time. (From the discussion above, we also know that it is sufficient to achieve confidence $\delta(n) = 1 - 1/\mathsf{poly}(n)$.)

Proceeding with the proof of (2), we describe a learning algorithm A for $\mathsf{SIZE}[n^a]$ with accuracy $\varepsilon(n) = 1/2 - 1/100n^\ell$ using an algorithm B that solves Partial-MCSP over dimension n on average in polynomial time with advantage $\gamma = 1/10$ for $s(n) = n^c$ using $t(n) = n^\ell$ samples. The argument relies on Yao's connection between pseudorandomness and (un)predictability [81], which is established using a hybrid argument. We provide below a self-contained presentation of the argument.

By assumption, for every $f \in \mathsf{SIZE}[n^a]$,

$$\left| \Pr_{B,\,\{z^i\}_{i\in[t]}}[B(1^n, z^1, f(z^1), \ldots, z^t, f(z^t)) = 1\,] - \Pr_{B,\,\{z^i\}_{i\in[t]},\,b}[B(1^n, z^1, b_1, \ldots, z^t, b_t)) = 1\,] \right| \;\geq\; 1/10.$$

For $i \in \{0, 1, \ldots, t\}$, consider the distribution $\mathcal{D}_i$ supported over $\{0,1\}^{t\cdot(n+1)}$ obtained by sampling a string $x^1, b_1, \ldots, x^i, b_i, \ldots, x^t, b_t$, where each x^j is a random n-bit string, and each b_j is set to $f(x^j)$ if $j > i$ and to a random bit otherwise. For convenience, let $p_i \stackrel{\text{def}}{=} \Pr_{B,\,w\sim\mathcal{D}_i}[B(1^n, w) = 1]$. The inequality above implies that

$$\left| \sum_{i=0}^{t-1} (p_i - p_{i+1}) \right| \;\geq\; 1/10.$$

Consequently, for some index $j \in \{0, 1, \ldots, t-1\}$ that might depend on f,

$$\left| \Pr_{B,\,w\sim\mathcal{D}_j}[B(1^n, w) = 1] - \Pr_{B,\,w\sim\mathcal{D}_{j+1}}[B(1^n, w) = 1] \right| \;\geq\; 1/10t.$$

The only distinction between the distributions $\mathcal{D}_j$ and $\mathcal{D}_{j+1}$ is that, for a random $x^{j+1} \in \{0,1\}^n$, one generates the pair $(x^{j+1}, f(x^{j+1}))$, while the other generates the pair (x^{j+1}, b_{j+1}), where b_{j+1} is a random bit. Note that the other coordinates of these two distributions are identically distributed and can be generated using the randomness of the learner and its example oracle $\mathcal{EX}(f)$. For this reason, once one knows the index $j+1$, it is possible to use B (or its negation) to predict the value of f on a random input with advantage $\Omega(1/t)$ over a random guess.

It is not difficult to show that this can be used to design a randomized learning algorithm A that, with probability $\Omega(1/t)$ over its internal randomness and $\mathcal{EX}(f)$, outputs a circuit that is ε-close to f, where $\varepsilon(n) = 1/2 - 1/100t$. Finally, the running time of A is polynomial under the assumption that B runs in polynomial time. ◀

A consequence of this result is that we can base the hardness of learning on the worst-case assumption that $\mathsf{NP} \not\subseteq \mathsf{RP}$ if and only if the existence of an efficient algorithm for average-case Partial-MCSP implies the existence of an efficient (worst-case) algorithm for

Partial-MCSP. In order to see this, note that NP $\subseteq$ BPP if and only if NP $\subseteq$ RP (using a search-to-decision reduction). Now the inclusion NP $\subseteq$ BPP is equivalent to the easiness of (worst-case) Partial-MCSP by Theorem 4, while Theorem 36 establishes an equivalence between learnability and solving Partial-MCSP on average.

Finally, we remark that the connection between learning and Partial-MCSP described here generalizes to any circuit class $\mathcal{C}$ that can efficiently compute the parity function. (This is necessary in order to apply the uniform distribution boosting procedure from [15].) In other words, $\mathcal{C}$-Partial-MCSP is easy on average if and ony if $\mathcal{C}$ can be efficiently learned under the uniform distribution from random examples.