The Hardness of Locating MCSP between **P** and **NP**

(second draft)

Marco Carmosino[*] Ngu Dang[†] Fabian Späh[†]

January 09, 2023

Abstract

The Minimum Circuit Size Problem (MCSP) is a fundamental **meta-computational problem** and whether it is **NP**-hard is still beyond the current state-of-the-art. A natural question to ask ourselves is that *would any dramatic ramification follow if* MCSP *is indeed* **NP***-hard?* Kabanets and Cai [KC00] showed that if a positive answer holds, then the class of linear-exponential time problems will have superpolynomial circuit lower-bounds. This work is the landmark for a long and active sequence of work on understanding the power of reductions hard problems to MCSP. In this paper, we provide detailed expositions on the following well-known results on MCSP and **NP**-hardness.

1. The aforementioned result by Kabanets and Cai [KC00] showed if MCSP is **NP**-hard under "natural" many-one reduction, then superpolynomial circuit lower-bounds hold for linear-exponential problems.
2. The same circuit lower-bound also holds under "parametric honest" or "natural" Turing reduction which was shown by Saks and Santhanam [SS20].
3. While **NP**-hardness of many natural problems can be shown under "local" poly-time reductions, Murray and Williams [MW17] showed that such reduction **cannot** be used to show **NP**-hardness of MCSP.

1 Introduction

The Minimum Circuit Size Problem (MCSP) asks: given the truth table of a Boolean function f together with a positive integer s, does there exist a circuit of size at most s that computes f?

Problem:	MCSP
Input:	A tuple $\langle T, s\rangle$ consisting of a truth table $T \in \{0,1\}^{2^n}$ for a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and an integer s
Question:	Is there a circuit C with at most s gates computing T?

MSCP formalizes the *feasibility problem* for integrated circuit design where it is important to minimize space and execution speed of a computational component. The search variant SearchMCSP

[*]IBM
[†]Boston University

— which asks for a witnessing size-s circuit that computes f — formalizes the *circuit design problem* itself, given a truth-table specification. Thus, MCSP is of immediate practical interest to computing technology. As such, the minimization of circuits was already studied in the Soviet Union when investigating whether brute-force search can be generally eliminated [Tra84].

However, motivations to study MCSP are not limited to efficient hardware design. Theoretical results about MCSP have striking implications in many different areas of computer science: complexity, cryptography, and learning theory. Intuitively, MCSP plays such a central role because it is a basic ***meta-computational problem*** — the input itself contains some form of execution instructions. Now, MCSP is exactly such a meta-problem, where the "instructions" are a (maximally explicit) look-up table and the task at hand is to determine if this can be *compressed* into a non-trivial circuit. So, studying MCSP entails determining the complexity of determining complexity itself. To the extent that results about complexity theory are *constructive*, they will somehow involve MCSP.

Moreover, the promise version of MCSP is essential to computational learning theory: If we would want to know what hidden Boolean function produced a certain set of n-dimensional training samples, we can—in accordance with Occam's razor—ask for the minimum circuit which coincides with all training samples. Further, the problems associated with cryptography all hinge on distinguishing between truly random and pseudo-random objects. An efficient MCSP algorithm could defeat *any* pseudo-random object, because such objects must have descriptions in terms of small circuits but there are many more random strings than small circuits. Therefore, resolving the complexity of MCSP is a core open problem across many areas of computer science.

To begin, it is easy to see that MCSP is in **NP**. Namely, we can define a certificate as some proposed circuit C of size at most s, and verify in polynomial time whether C computes each entry of the truth table correctly. Because MCSP for general circuits seems to require brute-force search, it is reasonable to conjecture that it is **NP**-complete. The theory of **NP**-completeness has been enormously successful in explaining why many problems seem intractable. But it has failed so far to explain MCSP — indeed, we have *formal evidence* that a "simple" proof of the statement "MCSP is **NP**-hard" will be *extremely difficult* to obtain.

In a landmark paper titled "Circuit Minimization Problem," Valentine Kabanets and Jin-Yi Cai addressed the difficulty of showing MCSP to be **NP**-hard [KC00]. They are showed that if there is a "natural" many-one reduction R from SAT to MCSP, then we can extract breakthrough complexity lower bounds that seem far out of reach from modern complexity theory. On the other hand, they demonstrate similarly dramatic (but seemingly unlikely) consequences of MCSP $\in$ **P**. Thus, resolving the complexity of MCSP in either direction — easy or hard — seems very difficult.

In this tutorial-level review, we focus on the hardness of proving **NP**-hardness of MCSP. We "spell out" all the details of the proof in their paper, giving an approachable and self-contained presentation. We also give examples that demonstrate how "natural" the reductions studied really are — the vast majority of **NP**-hardness proofs use this sort of reduction, and even parameterized problems are easily shown to be natural in the sense of [KC00]. These notes should make subsequent work on the hardness of **NP**-hardness for MCSP readily accessible by providing a detailed exposition of the initial result. Furthermore, we will also survey a chain of work which examine the **NP**-hardness of MCSP via other "reducibilities." To summarize,

- **On the (Non) NP-hardness of** MCSP **of Computing Circuit Complexity** [MW17]: in this paper, Murray and Williams proved that MCSP is *not* **NP**-hard under local reduction from PARITY. The locality here means that the reduction only acts on "part" of the input, rather than the entire input. Another notable result in this paper is that if MCSP is **NP**-hard under polynomial time reduction, then strong implications would follow, such as **EXP** $\neq$ **ZPP**.

- **Circuit Lower-Bounds from NP-hardness of** MCSP [SS20]: in this paper, Saks and Santhanam showed that a strong circuit lower-bound **E** $\not\subseteq$ SIZE(poly) would follow if MCSP is **NP**-hard under parametric honest and natural Turing reduction. Informally speaking, "parametric honest" here means that the queries made by the reduction on some input have parameters lower-bounded by some polynomial of the input length (i.e. we are promised to have a "large" enough parameter in each query). And "natural" here means that the parameter made by a query on some input depends on the length of the input only.

We will first give some definitions required to understand the main results and key theorems of the paper in Section 2. Section 3 introduces some main consequences of MCSP being **NP**-hard under "natural" reductions. Finally, we conclude the review by giving remarks and directions for further research on this topic in Section 4.

Acknowledgment: We thank Prof. Mark Bun and Ph.D. student Ludmila Glinskih for the constructive feedback and comments on the first draft of this paper. We also thank Prof. Steve Homer and Ph.D. student Tim Jackman for the fruitful discussion and suggestions on the second draft. TO DO: finish the rest.

2 Preliminaries

We begin by introducing some useful background, including the definition of some complexity classes, the concept of natural reductions and pseudorandomness. We assume the reader is familiar with basic complexity theory and circuits.

2.1 Complexity Classes

Here, we provide definitions of and basic relationships between some less-common complexity classes used by [KC00]. For more details and a full discussion, refer to the excellent textbook by Arora and Barak [AB09] on which we based our definitions and notations used throughout the paper.

Definition 1. The class of languages in *deterministic, sub-exponential time* is **SUBEXP** $:= \bigcap_{\epsilon>0} \mathbf{DTIME}(2^{n^{\epsilon}})$.

SUBEXP is not empty, even though it is defined as an infimum over complexity classes. This is because a language might be decided by a different TM for every $\varepsilon > 0$.

Definition 2. The class **QP** of languages decided by a TM in *quasi-polynomial* time defined by

$$\mathbf{QP} := \mathbf{DTIME}(n^{\mathrm{polylog}(n)}) = \bigcup_{c>1} \mathbf{DTIME}(2^{\log^c n}) = \bigcup_{c>1} \mathbf{DTIME}(n^{\log^c n})$$

We obtain the last equality since $2^{(\log n)^{c+1}} = (2^{\log n})^{\log^c n} = n^{\log^c n}$. Note that **QP** contains **P** since $\operatorname{polylog}(n) \in \Omega(1)$. Also, **SUBEXP** contains **QP** since for every $\varepsilon > 0$ and $c > 1$ holds that $\exp(\log^c n) \in o(\exp(n^\varepsilon))$.[1]

Definition 3. The class *exponential time with linear exponent* is defined as $\mathbf{E} := \mathbf{DTIME}(2^{O(n)})$.

Since $\log^c n \in O(n)$ for any $c > 0$, we obtain that $\mathbf{QP} \subseteq \mathbf{E}$. Finally, we introduce *infinitely-often simulations* trough a modifier to a complexity class C.

Definition 4. A language L belongs to the class $\mathsf{i.o.}\,C$ if there is a language $A \in C$ such that A and L agree on infinitely many input lengths, i.e.

$$|\{\, n \in \mathbb{N} \mid \forall x \in \{0,1\}^n \; A(x) = L(x) \,\}| = \infty$$

By definition, C is contained in $\mathsf{i.o.}\,C$.

2.2 Natural Reductions

Let us now introduce the notion of a *natural reduction*. Intuitively, a many-one polynomial-time reduction is natural if its output size depends only on the input size. Formally, we define it as follows.

Definition 5 (Natural Reduction)**.** Assume we are given two languages A and B, where B describes the decision version of a search problem, meaning its instances are of the kind $\langle y, s\rangle \in B$ where $s \in \mathbb{N}$ is a numerical parameter. A many-one reduction $R = \langle R_I, R_P\rangle$ from A to B, whereas R_I maps to instances y of the underlying search problem and R_P to the parameters s, is called *natural* if for all instances x of A holds that

- *PTIME-honesty*: there exists a $c \in \mathbb{R}^+$ such that $|x|^{1/c} \leq |R_I(x)| \leq |x|^c$, meaning R_I does not suddenly grow or shrink, and
- *Parameter-uniform*: R_P depends only on the length of the instance x, i.e. there exists a function $f\colon \mathbb{N} \to \mathbb{N}$ such that $R_P(x) = f(|x|)$.

All **NP**-complete problems we are aware of seem to be complete under natural reductions. This is nothing special for a polynomial-time reduction to an unparameterized problem, as we can usually pad the reduct to conform to the specific length. However, so far we are not aware of any parameterized problem, which is **NP**-complete under general many-one reductions but not under a natural reduction. To illustrate, we provide a reduction from SAT to kOV and show how it applies to the definition of Natural Reduction.

[1]To analyze the relationship of exponentials, the following proves useful: For functions $f, g\colon \mathbb{N}_+ \to \mathbb{N}_+$ holds $2^{f(n)} \in o(2^{g(n)})$ if and only if $g(n) - f(n) \to \infty$ for $n \to \infty$. This is easily seen by using the limit definition of Landau symbols as

$$f(n) \in o(g(n)) \iff 0 = \lim_{n\to\infty} \frac{2^{f(n)}}{2^{g(n)}} = \lim_{n\to\infty} \exp(f(n) - g(n)) \iff \lim_{n\to\infty} f(n) - g(n) = -\infty\,.$$

Example 6. We reduce from SAT to kOV.[2] Given a SAT-formula $\phi(x_1, x_2, \dots, x_k)$ on d clauses, we first define vectors $v(a) \in \{0,1\}^d$ for any literal $a \in \{x_1, \bar{x}_1, \dots, x_k, \bar{x}_k\}$ where $v(a)_\ell = 0$ if the ℓ-th clause contains a and $v(a)_\ell = 1$, otherwise. We can then define $R(\phi) \coloneqq \langle \mathcal{V}, k \rangle$ where $\mathcal{V} = (V^1, \dots, V^k)$ and $V^i = (v(x_i), v(\bar{x}_i))$. Clearly, literals $a^i \in \{x_i, \bar{x}_i\}$ obey

$$\sum_{\ell=1}^{d} v(a^1)_\ell \cdot v(a^2)_\ell \cdots v(a^k)_\ell = 0$$

if and only if for all clauses $\ell \in [d]$ exist a variable $i \in [k]$ such that $v(a^i)_\ell = 0$ or, equivalently, such that setting $x_i = 1$ if $a^i = x_i$ and $x_i = 0$ if $a^i = \bar{x}_i$ satisfies the ℓ-th clause. As such, R indeed reduces from SAT to kOV.

In order to show that the reduction R is also natural, we first need to fix a proper encoding of SAT-formulae. Namely, a formula $\phi(x_1, \dots, x_k)$ is represented as a sequence of clauses, each of which containing a set of indices corresponding to the variables occurring in that clause. The maximum number of used variables is therefore restricted by the formula's length, and we may even assume $k \log k = \Theta(|\phi|)$ without loss of generality.

- PTIME-honesty: Our reduction satisfies $|R_I(\phi)| = |\mathcal{V}| = 2d \cdot k$ where $k \log k = \Theta(|\phi|)$. However, $d = O(|\phi|)$ is the only necessary relationship between the number of clauses and the size of ϕ. To guarantee that the reduction is always PTIME-honest, we thus need to artificially pad $\mathcal{V}$ with $|\phi|$ blocks of all-one vectors (which do not affect the solution). Then, $|R_I(\phi)| = k \cdot \Theta(|\phi|)$ which satisfies $|\phi|^{1/c} \le |R_I(\phi)| \le |\phi|^c$ for every $c > 0$.
- Parameter-uniform: $R_P(\phi) = k$ clearly depends only on the size of ϕ as $k \log k = \Theta(|\phi|)$.

Finally, we conclude this example with an example of "naturalizing" an unnatural reduction.

Example 7. We reduce from Partition[3] to SubsetSum :[4]

$$R(\{a_1, \dots, a_n\}) \coloneqq \begin{cases} \langle \{2,4,8\}, 7 \rangle & \text{if } \sum_{i=1}^n a_i \equiv_2 1 \\ \langle \{a_1, \dots, a_n\}, \frac{1}{2} \sum_{i=1}^n a_i \rangle & \text{otherwise}. \end{cases}$$

While this reduction is correct (i.e. if $\sum_{i=1}^n a_i$ is congruent to 1 modulo 2, it is impossible to divide the set into two partitions of equal sum; the tuple $\langle \{2,4,8\}, 7 \rangle$ is a no-instance to SubsetSum) and can be carried out in polynomial time, it is not natural. First, the output size does not strictly depend on the input size as, in the case of an obvious no-instance to Partition, we map directly to a no-instance of SubsetSum of constant size. Second, the numerical parameter s is not a function of the input size only. For example, the two instances $\{1, \dots, 1\}$ (n-times) and $\{2^n\}$ are of equal size, but have totally different sums. With a little more care, we can, however, make this a natural reduction: Let

$$R'(\{a_1, \dots, a_n\}) \coloneqq \langle \{2a_1, \dots, 2a_n, r, 2^{\sigma+2} \rangle \quad \text{where} \quad \sigma \coloneqq \text{size}(\{a_1, \dots, a_n\}) \\ \text{and} \quad r \coloneqq 2^{\sigma+2} - \textstyle\sum_{i=1}^n a_i \,.$$

[2] A tuple $\langle \mathcal{V}, k \rangle$ where $k \in \mathbb{N}$ and $\mathcal{V}$ is a sequence of k blocks V^i of n vectors $v_1^i, \dots v_n^i \in \mathbb{R}^d$ each, is an instance of k-orthogonal vectors (kOV) if there exists a vector $v^i \in V^i$ from each block such that $\sum_{\ell=1}^d v_\ell^1 \cdot v_\ell^2 \cdots v_\ell^k = 0$.

[3] The partition problem asks, given a multiset of positive integers $\{a_1, \dots, a_n\}$, whether there exists a partition (S, T) of $\{1, \dots, n\}$ such that $\sum_{i \in S} a_i = \sum_{i \in T} a_i$.

[4] Subset sum asks for an instance $\langle A, s \rangle$ of a multiset of positive integers $A = \{a_1, \dots, a_n\}$ and an integer s, whether there exists a subset $S \subseteq \{1, \dots, n\}$ such that $s = \sum_{i \in S} a_i$.

Now, the numerical parameter s is obviously only a function in the input size σ. Regarding correctness, first note that $2^{\sigma+2} > \sum_{i=1}^{n} 2a_i$. This implies that any subset with a sum of $2^{\sigma+2}$ necessarily contains r and thus $2^{\sigma+2} = r + \sum_{i \in S} 2a_i \iff \frac{1}{2}\sum_{i=1}^{n} a_i = \sum_{i \in S} a_i$ where S is the original subset without r.

2.3 Pseudorandomness

Broadly speaking, given a short string of truly random bits, a *pseudorandom generator* (PRG) tries to enlarge this sequence to obtain a string with "artificial" randomness which is defined as computationally indistinguishable from a truly random string by any efficient algorithm called the distinguisher. To be more specific, indistinguishability here means the probability that the distinguisher can detect the difference is *negligible*. For full discussion and formalization on this topic, see Goldreich's Primer on PRGs [Gol10].

In this tutorial, we formulate the construction of PRGs in terms of truth tables and circuit complexity by adopting the notations by Murray and Williams [MW18] as follows. Let $x_1, \ldots, x_{2^\ell}$ be the ℓ-bit strings in lexicographically order and C be a circuit of ℓ input entries, define $tt(C) := C(x_1)\cdots C(x_{2^\ell})$ to be the truth table of C, where $tt(C) \in \{0,1\}^{2^\ell}$. For every string y, let 2^ℓ be the smallest power of 2 such that $2^\ell \geq |y| + 1$. We define the circuit complexity of y, denoted as $CC(y)$, to be the circuit complexity of the ℓ-input function defined by the truth table $y10^{2^\ell - |y| - 1}$. Thus, we can formulate the definition of pseudorandom generators via the following Theorem by Umans [Uma03].

Theorem 8. *There is a universal constant g and a function $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for all s and Y satisfying $CC(Y) \geq s^g$, and for all circuit C of size s it holds that*

$$\left| \Pr_{x \in \{0,1\}^{g \log |y|}} [C(G(Y,x)) = 1] - \Pr_{x \in \{0,1\}^s} [C(x) = 1] \right| < 1/s$$

The existence of certain pseudorandom generators would allow us to derandomize algorithms in **BPP**. The basic idea is to replace the randomness used in a probabilistic algorithm with pseudorandomness generated from a sufficiently small source of true randomness. We then try the algorithm on the pseudorandomness generated from each choice in this small source, while the assumption guarantees that it cannot distinguish it from true randomness. If the stretch function is large enough, this leads to an efficient derandomization.

3 Main Focus

3.1 MCSP and NP-completeness via Natural Many-One Reductions

To begin with, let us make it clear that we do not know whether MCSP is **NP**-hard. Yet, the proof's difficulty can be made explicit by looking at some implications it has for circuit complexity and **BPP**, which are both beyond currently known techniques. In other words, proving the **NP**-hardness of MCSP can be shown to be as challenging as finding proofs for other long-standing problems known to be difficult.

Now, let us look at the first key theorem regarding an implication for circuit complexity if MCSP is **NP**-hard under a natural reduction.

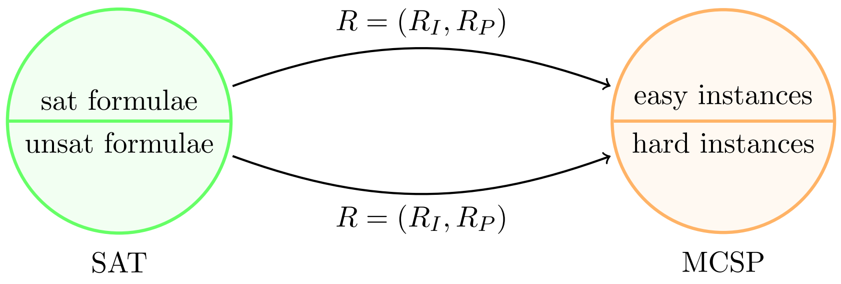

Figure 1: The mapping from SAT to MCSP given by the natural reduction R

Theorem 9. *(Theorem 15.1 in [KC00]) If* MCSP *is* **NP***-hard under a natural reduction from* SAT*, then* **E** *contains a family of Boolean functions f_k not in* i.o. $\mathbf{P}_{/\mathbf{poly}}$*, i.e., of superpolynomial circuit complexity.*

Before we move on to the proof, we provide some classic lemmata related to basic complexity classes and circuit sizes that are useful for establishing this result. The proofs for these are fairly simple and we put them under Appendix A for the readers reference.

Lemma 10. $\mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP}$.

Lemma 11. *If* $\mathbf{NP} \subseteq \mathbf{QP}$ *then* $\mathbf{PH} \subseteq \mathbf{QP}$.

Lemma 12. $\mathbf{QP}^{\mathbf{\Sigma}_k^p}$ *contains a language that does not belong to* i.o. $\mathbf{P}_{/\mathbf{poly}}$ *for some* $k \in \mathbb{N}$.

Lemma 13. *There are $O(s^{3s})$ different circuits of size s. In particular, there are*

1. $n^{\mathrm{polylog}(n)}$ *circuits of size* $\log^c n$ *for any* $c > 0$ *and*
2. $O(2^{n^{2\varepsilon}})$ *circuits of size* n^ε *for any* $\varepsilon > 0$.

We now have enough tools to prove Theorem 9. We remind ourselves of the statement to prove: If there is a natural polynomial-time reduction from SAT to MCSP, then **E** contains a family of Boolean functions f_k of superpolynomial circuit size.

Proof (Theorem 9). We separate the proof into two cases.

- Case 1: $\mathbf{NP} \subseteq \mathbf{QP}$

  Applying Lemma 10 and Lemma 11 to this assumption yields $\mathbf{QP}^{\mathbf{PH}} \subseteq \mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP} \subseteq \mathbf{E}$. Lemma 12 shows that **E** also contains a language of superpolynomial circuit complexity (i.o.). Hence, $\mathbf{E} \nsubseteq$ i.o. $\mathbf{P}_{/\mathbf{poly}}$

- Case 2: $\mathbf{NP} \nsubseteq \mathbf{QP}$

  While it is (computationally) trivial to choose a no-instance for SAT of any given size, it is not clear how to do so for MCSP. The key idea for this part is, therefore, to use the natural reduction R to obtain hard instances for MCSP, i.e., a family of truthtables that cannot be represented by circuits of polynomial size, see Figure 1 for an illustration.

We start out by picking a quite arbitrary infinite set U of unsatisfiable CNF-formulae, say, $U \coloneqq \{ \phi_n \mid n \in \mathbb{N} \}$ where

$$\phi_n(x_1, \ldots, x_n) \coloneqq (x_1 \wedge \bar{x}_1) \wedge (x_3 \wedge x_4 \wedge \cdots\cdots\cdots \wedge x_n) .$$

By construction, $|\phi_n| \in \Theta(n)$. Now, based on this, we want to apply R to define our hard language. For each $k \in \mathbb{N}$, let T_k be the truthtable in k-variables such that $\langle T_k, s_n \rangle = R(\phi_n)$ and n is minimal. Here, s_n represents the size parameter that the R_P part of R (see Definition 5) produces on ϕ_n. If no ϕ_n maps to a truthtable in k variables, simply let $T_k \equiv 0$.

We know that R is natural, which in particular implies $n^{1/c} \leq |T_k| \leq n^c$ (with an appropriate $c \in \mathbb{N}$) for every ϕ_n mapping to a truthtable T_k. Since $|T_k| = 2^k$, this is equivalent to $\log(n^{1/c}) \leq k \leq \log(n^c)$ implying $k = \Theta(\log n)$ or $n = 2^{\Theta(k)}$. Now, we proceed with defining our hard language

$$L \coloneqq \{ x \in \{0,1\}^k \mid k \in \mathbb{N}, T_k(x) = 1 \}$$

to obtain the following.

(i) $L \in \mathbf{E}$: We show how to construct a machine to decide L in time $2^{O(k)}$. Given an input $x \in \{0,1\}^k$, we first need to know which ϕ_n maps to T_k under R.

To each candidate ϕ_n for $n \in \mathbb{N}$, we apply R and obtain $\langle T_{k'}, s_n \rangle = R(\phi_n)$. We then compare whether $k = k'$ and if so, output $T_k(x)$. The reduction R runs in polynomial time, say n^a for an $a \in \mathbb{N}$. By the above construction, $n = 2^{\Theta(k)}$, which means we only need to check $2^{\Theta(k)}$ candidates whereas each check runs in time at most $n^a = (2^{\Theta(k)})^a = 2^{\Theta(k)}$. If no candidate maps to a truthtable on k variables, we know that $T_k \equiv 0$ by definition and reject since $T_k(x) = 0 \neq 1$. Overall, the decision procedure took time $2^{O(k)}$, proving that $L \in \mathbf{E}$. (TO DO: turn this into pseudocode)

(ii) $L \notin \mathbf{P}_{/\mathbf{poly}}$: We start by showing that the parameter s_n produced by R is superpolynomial in $\log n$. Let us assume to the contrary that it is bounded by a polynomial $s_n \leq \log^b n$ for any $b \in \mathbb{N}$. Now, this yields a simple strategy to decide SAT in quasi-polynomial time: Given a CNF-formula ϕ of size n, we apply R to obtain $\langle T_{k'}, s_n \rangle \coloneqq R(\phi)$ with $s_n \leq \log^b n$. We will decide the membership of this instance to MSCP instead of solving the original satisfiability problem. By Lemma 13, there are at most $n^{\mathrm{polylog}(n)}$ circuits of size s_n. We enumerate these and check for each circuit whether it represents $T_{k'}$. Note that testing whether a circuit C of size s_n represents $T_{k'}$ only requires us to do $2^{k'}$ evaluations of C, each of which take time $O(s_n)$. Overall, we can decide the membership to MCSP, and as such the satisfiability of ϕ, in quasi-polynomial time. (TO DO: turn this into pseudocode) which implies that $\mathsf{SAT} \in \mathbf{QP}$. However, SAT is an $\mathbf{NP}$-complete problem, and thus, we have $\mathbf{NP} \subseteq \mathbf{QP}$ contradicting our assumption that $\mathbf{NP} \not\subseteq \mathbf{QP}$.

We, therefore, established that s_n is superpolynomial in $\log n$. At the same time, R is a natural reduction, meaning that s_n is the same for every input of size n. In particular, we obtain that since we set $\langle T_k, s_n \rangle = R(\phi_n)$, the parameter s_n is superpolynomial in $\log n = \Theta(k)$. As ϕ_n is a no-instance to SAT, we conclude that T_k cannot be represented by a polynomial-size circuit family. In other words, $L \notin \mathbf{P}_{/\mathbf{poly}}$. Being more careful, we furthermore obtain that any polynomial-size circuit family can only agree with T_k on finitely many input lengths. The reason for this is that $s_n \in \omega(\mathrm{poly}(k))$ guarantees by definition the existence of a $k_0 \in \mathbb{N}$ such that $\text{poly}(k) < s_n$ for all $k > k_0$. We conclude that $L \notin \text{i.o.}\,\mathbf{P}_{/\mathbf{poly}}$.

$\square$

With the same techniques but a different bound on the circuit size, we can obtain a very similar statement. The only additional ingredient is the exponential-time hypothesis, which is, however, widely assumed to be true [IP99]. The claim is that $\mathbf{NP} \not\subseteq \mathbf{SUBEXP}$, or equivalently, that SAT requires exponential time.

Theorem 14. *(Theorem 15.2 in [KC00]) If* MCSP *is* **NP***-hard under a natural reduction from* SAT *and* $\mathbf{NP} \not\subseteq \mathbf{SUBEXP}$*, then* **E** *contains a family of Boolean functions* f_k *of circuit complexity* $2^{\Omega(k)}$ *(i.o.)*

Proof. This proof follows similarly to the previous statement. In fact, let the language L and pairs $\langle T_k, s_n \rangle$ be defined as in the proof for Theorem 9. The difference is that here, we show that a more restrictive bound on s_n contradicts the stronger assumption $\mathbf{NP} \subseteq \mathbf{SUBEXP}$ as this bound would allow SAT to be solved in subexponential time.

To this end, assume that $s_n \in O(n^\varepsilon)$ for any $\varepsilon > 0$. We pursue the same strategy to decide the satisfiability of a CNF formula ϕ: Using the reduction R, map it to $\langle T_{k'}, s_n \rangle \coloneqq R(\phi)$. Lemma 13 shows that there are $O(2^{n^{2\varepsilon}})$ circuits of size at most s_n. Enumerating all such circuits (1) and checking whether they represent $T_{k'}$ by evaluating (3) all assignments (2) therefore takes time

$$\underbrace{O(2^{n^{2\varepsilon}})}_{(1)} \cdot \underbrace{O(2^{k'})}_{(2)} \cdot \underbrace{O(s_n)}_{(3)} = O(2^{n^{2\varepsilon}}) \cdot \text{poly}(n) \cdot O(n^\varepsilon) = O(2^{n^{3\varepsilon}})\,.$$

So, if we assume to the contrary that $s_n \in O(n^\varepsilon)$ for every $\varepsilon > 0$, we also obtain that we can decide the satisfiability of ϕ in time $O(2^{n^{3\varepsilon}})$ for every ε, i.e. in subexponential time. However, $\text{SAT} \in \mathbf{SUBEXP}$ contradicts the assumption $\mathbf{NP} \subseteq \mathbf{SUBEXP}$.

We can conclude that instead, $s_n \in \Omega(n^\varepsilon)$ for an $\varepsilon > 0$. With $n = 2^{\Theta(k)}$, this shows that $s_n \in \Omega(2^{\varepsilon\Theta(k)}) = 2^{\Omega(k)}$. Now, with the same argument as above, we know that L can only be decided by a $2^{\Omega(k)}$ size circuit family. We have already proven that $L \subseteq \mathbf{E}$, which concludes the proof. $\square$

Now, we will look at the implications for **BPP** when **NP**-hard under a natural reduction from SAT. By appropriate settings of the parameters for Umans' generator (Theorem 8), we obtain the following conditional derandomizations of **BPP**. Note that Umans' generator was not available in 2001 — Kabanets and Cai used different PRGs for different settings of the hardness parameter. This emphasizes that the particular hardness to randomness tradeoff used to obtain derandomization from **NP**-hardness of MCSP was not essential to the ideas of [KC00]. Any black-box construction of PRGs from hardness in **E** would work equally well here.

Theorem 15 ([KC00])**.** *If* MCSP *is* **NP***-hard under a natural reduction from* SAT*, then*

1. $\mathbf{BPP} \subseteq \mathbf{SUBEXP}$ *(i.o.), and*

2. $\mathbf{BPP} = \mathbf{P}$*, unless* $\mathbf{NP} \subseteq \mathbf{SUBEXP}$.

That is, if MCSP is **NP**-hard under a natural reduction from SAT, then every problem in **BPP** can be efficiently solved. Now, we believe that **P** = **BPP**, as most algorithms in **BPP** surrender to derandomization. However, it is known that showing **BPP** = **P** entails proving the existence of certain pseudorandom generators, which in turn would prove **BPP** = **P** all along [Gol11], a result which seems to be challenging in its own right. Therefore, proving the **NP**-hardness of MCSP is as difficult as finding a constructive way for derandomization.

Finally, taking everything together, we obtain a nice corollary as follows.

Corollary 16. *If* MCSP *is* **NP***-hard under a natural reduction from* SAT*, then* **BPP** $\subsetneq$ **E**

Proof. We first show that the inclusion **SUBEXP** $\subseteq$ **E** is strict: By definition, **SUBEXP** is the intersection of $\mathbf{DTIME}(2^{n^{\varepsilon}})$ for all $\varepsilon > 0$. In particular, $\mathbf{SUBEXP} \subseteq \mathbf{DTIME}(2^{\sqrt{n}})$. Now, the deterministic time-hierarchy theorem proves that $\mathbf{DTIME}(2^{\sqrt{n}}) \subsetneq \mathbf{DTIME}(2^{n})$ since $2^{\sqrt{n}} \log\left(2^{\sqrt{n}}\right) = 2^{\sqrt{n}\log\sqrt{n}} \in o(2^n)$ We conclude that $\mathbf{SUBEXP} \subseteq \mathbf{DTIME}(2^{\sqrt{n}}) \subsetneq \mathbf{E}$ Furthermore, from Theorem 15, we know that if MCSP is **NP**-hard under a natural reduction from SAT, then **BPP** $\subseteq$ **SUBEXP**. Combining both gives us **BPP** $\subseteq$ **SUBEXP** $\subsetneq$ **E**. □

3.2 MCSP and NP-Completeness via Turing Reduction

A natural problem rooted from our discussion in the previous section is whether we can derive other strong circuit lower-bounds under the assumption that some seemingly hard reducibility from some **NP**-hard problem to MCSP exists. Indeed, Saks and Santhanam [SS20] have shown that **E** $\not\subseteq$ SIZE(poly) if a parametric Turing Reduction from SAT to MCSP exists.

Broadly speaking, an oracle Turing machine M is a parametric Turing Reduction from SAT to MCSP if M runs in polynomial time and every query made by M has a "large enough" parameter. Specifically, for some input $x \in \{0,1\}^*$, the queries that M makes on x has a parameter of size at least $|x|^{\epsilon}$, where ϵ is a positive constant. Before we move on to the key theorem for this section, we provide a "tool"rooted from the work of Gutfreund, Shaltiel, and Ta-Shma [GSTS05] that plays the critical role in the proof for the theorem.

Lemma 17. *(Lemma 3.1 in [GSTS05]) If* **P** $\neq$ **NP***, then there exists a polynomial time algorithm R such that on input n in unary and a description of a deterministic machine A that tries to decide* SAT *in polynomial time, R outputs three instances whose length is either n or* poly(n^a)*, for some constant a, such that A fails on at least one of these three instances.*

Now, let us look at the key theorem regarding an implication for circuit complexity if MCSP is **NP**-hard under a parametric Turing reduction. We are not going to provide the full proof for the theorem, but rather, a sketch that helps the readers come up with the full detailed solution on their own using the given references and preliminaries.

Theorem 18. *(Theorem 8 in [SS20]) If* MCSP *is* **NP***-hard under a Parametric Honest Turing reduction from* SAT*, then* **E** $\not\subseteq$ SIZE(poly)*.*

Proof. (*sketch*) Assume that MCSP is **NP**-hard under a parametric honest Turing reduction from SAT and let M be the polynomial time oracle Turing machine that implements such reduction. It is useful to remind ourselves that by the definition of Turing reduction, M is a machine that decides SAT by making queries to the embedded oracle machine for MCSP.

Now, we will construct a polynomial time machine A that aims to decide SAT efficiently by simulating M. Namely, on input $x \in \{0,1\}^*$, an instance for SAT, the machine A performs the simulation of M on x as follows

Algorithm 1 $A(x)$:

```
1: for all queries $y$ made by $M$ on the MCSP-oracle do
2:     $A$ only answers yes
3: end for
4: if $M$ accepts then
5:     accept
6: else
7:     reject
8: end if
```

We can assume, w.l.o.g., each query y has length of a power of 2 because it must represent a truth table to be a valid query for the MCSP oracle. Note that, one can argue that the function described by this truth table y must have $O(\log n)$ variables because M is a polynomial-time Turing reduction which means that the size of the query cannot be too big. Lastly, since M is a parametric honest reduction, the query must have a parameter of at least $|x|^\epsilon = n^\epsilon$, for some $\epsilon > 0$.

It is easy to see that A must run in polynomial time since M runs in polynomial time according to our assumption. Now, we consider two cases

- Case 1: $\mathbf{P} = \mathbf{NP}$

  If A solves SAT correctly, then $\mathbf{P} = \mathbf{NP}$. Thus, by a proof by contradiction using Karp-Lipton theorem and the deterministic time hierarchy theorem, one can deduce to $\mathbf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$.

- Case 2: $\mathbf{P} \neq \mathbf{NP}$

  If $\mathbf{P} \neq \mathbf{NP}$, then it must be the case that A fails to decide SAT on infinitely many instances. By Lemma 17, we have a polynomial time machine R such that on input 1^n and the description of A, R produces three instances x_1, x_2, x_3 where each has length of either n or n^a, for some constant a and for infinitely many n, such that A fails on at least one of them. We now construct a machine T that produces a hard truth-table, using M as a subroutine on input x_1, x_2, x_3, as follows

Algorithm 2 $T(x_1, x_2, x_3)$:

```
1: for i = 1, 2, 3 do
2:     $Q_i \leftarrow$ all queries $y$'s of $M(x_i)$
3: end for
4: $Y \leftarrow Q_1 + Q_2 + Q_3$
5: return $Y$
```

It is easy to see that $|Y| = \text{poly}(n)$ and T runs in time $\text{poly}(n) = 2^{O(\log n)}$ which implies $Y \in \mathbf{E}$. Observe that, by Lemma 17, there must be a query y^* wrongly answered by A on the honest parameter n^ϵ which means that the function described by y^*, having $O(\log n)$ variables, requires a circuit of some size strictly greater than n^ϵ which is superpolynomial in terms of $O(\log n)$ which implies $y^* \notin \mathsf{SIZE}(\text{poly})$. Finally, since y^* is a substring of Y by the procedure T, it must be the case that $Y \notin \mathsf{SIZE}(\text{poly})$ which implies $\mathbf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$.

$\square$

With a more sophisticated mathematical analysis which relies on robustness of a complexity class $\mathcal{C}$, Saks and Santhanam managed to prove the same circuit lower-bound with a weaker assumption. Namely, the assumption of a polynomial time Natural Turing reduction M from SAT to MCSP. Specifically, following a similar vein as Natural Reduction in Definition 5 for some input $x \in \{0,1\}^*$, M is Natural if the queries that M makes on x has a parameter depending *only* on $|x|$, and it is the same for all queries made on any input of the same given length.

Theorem 19. *(Theorem 10 in [SS20]) If* MCSP *is* **NP***-hard under a Natural Turing reduction from* SAT*, then* $\mathbf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$.

Even though the proof for this theorem requires much more mathematical work compared to Theorem 18, the general framework of having full control over the reduction to efficiently produce hard instances using Lemma 17 as a tool is similar. Therefore, we refer the readers to the original paper by Saks and Santhanam for full details and discussion on how the whole proof is conducted. Here, we will provide some quick ideas on how it works as a starting point for the readers

- We construct two algorithms for SAT, decision algorithm M_1 and search algorithm M_2 where both algorithms partially do brute-force searching for the circuit. From here, we aim to show that the brute-force search for circuits works well enough to imply the separation or that there are intervals where the queried functions require large circuits which should also imply the separation.
- Specifically, it is helpful to use the fact that SAT is paddable so that we can have control over the reduction parameter. We have two cases.
  - If the reduction parameter is small, then the exact exponential time algorithm M_1 can guess and verify all circuits for all MCSP queries.
  - If the reduction parameter is large, then the algorithm M_2 can try to use the Search-to-Decision reduction to find a satisfying assignment for a SAT instance. If M_2 fails, then we can apply the same technique as in Theorem 18 to find witnessing counter-examples whose circuit sizes are large.

3.3 Non NP-Hardness of MCSP via Local Reductions

We would like to note that our previous discussion in the last two sections does *not* imply that natural reductions from SAT to MCSP does not exist; they are merely just hard to obtain at the current state-of-the-art. Thus, a natural question to ask is which would be a candidate reduction with more potential or which could not be a candidate at all. In one of their breakthroughs, Murray and Williams [MW17] proved that local reductions cannot be used to show **NP**-hardness of MCSP. Namely, a local reduction from PARITY, a simple and tractable language, to MCSP which implies that we won't be able to locally reduce any other **NP**-hard language to MCSP. Here, local reduction means that the reduction R only acts on a certain part of the input string instead of the entire string and thus the running time of the reduction is sub-linear in terms of the input length.

Specifically, let $t : \mathbb{N} \to \mathbb{N}$ and $t(n)$ be some $n^{1-\epsilon}$ for some $\epsilon > 0$, then we define a local reduction as follows

Definition 20. (Definition 1.1 in [MW17]) A local reduction from a language L to L' is an algorithm $R : \Sigma^* \times \Sigma^* \to \{0, 1, *\}$ which runs in time $t(n)$ such that for all $x \in \Sigma^*$ and for some constant $c \geq 1$, the following conditions hold.

- Locality: $\exists \ell_x \leq |x|^c + c$ such that $R(x, i) \in \{0, 1\}$ for all $i \leq \ell_x$, and $R(x, i) = *$ for all $i > \ell_x$. Furthermore, $x \in L \iff R(x, 1) \cdot R(x, 2) \cdots R(x, \ell_x) \in L'$.
- Sub-linear Time Honesty: $R(x, i)$ has random access to x and runs in $O(t(|x|))$ time for all $i \in \{0, 1\}^{\lceil 2c \log_2 |x| \rceil}$.

We will discuss the ideas and intuition of the proof for the following Theorem.

Theorem 21. *(Theorem 1.3 in [MW17]) For any $\delta < 1/2$, there is no* $\mathsf{TIME}(n^\delta)$ *(local) reduction from* PARITY *to* MCSP*.*

Specifically, the proof can be broken down to the following three parts

Naturalizing the Local Reduction via the paddability of PARITY. We first observe that if there exists a reduction from PARITY to MCSP, then it can be made natural with the cost of some extra running time, namely

Claim 22. *(Claim 3.3 in [MW17]) If there exists a* $\mathsf{TIME}(t(n))$ *reduction from* PARITY *to* MCSP*, then there exists a* $\mathsf{TIME}(t(n)\log^2(n))$ *natural reduction (as in Definition 5) from* PARITY *to* MCSP *and the size parameter $s_n \in \tilde{O}(t(n))$.*

The existence of such naturalization relies on the fact that PARITY is a paddable language. Specifically, say we have a length-n instance x of PARITY, then if we create another instance x' by padding n 0's, the parity of x' is still the same as the parity of x and one can show that a reduction R' that runs R on x' maintains the correctness and furthermore, the size parameter s_{2n} of R' becomes a function that only relies on the size of x' which makes R' natural by Definition 5.

Upper-bounding the size parameter of the Naturalized Reduction. Now, we will discuss the argument of the upper bound of the size parameter of R'. We consider a procedure f that first computes the string $z = 0^n$ then runs R' on z. One can observe that since z is a trivial no-instance of PARITY, the computation of R' can speed up in the sense that R' no longer needs to worry about the content of the bits in the input string and can only focus on the indices of the bits to maintain locality. Since R' can be computed in $\mathsf{TIME}(t(n)\log^2(n))$, one can now argue that the size of the circuit computing f is at most $\tilde{O}(t(n))$ owning to the speed-up (TO DO: clarify the speed-up trick more) and therefore the circuit computing the truth table of f has size also at most $\tilde{O}(t(n))$ via the following Lemma which completes the proof of Claim 22.

Lemma 23. *[Wil16] There is a universal constant $c > 1$ such that for any binary string T and any substring S of T, the following inequality holds*

$$CC(f_S) \leq CC(f_T) + c \log |T|$$

Constructing a depth-3 circuit computing PARITY. Recall that $z = 0^n$ is the trivial no-instance of PARITY which means that the tuple $\langle tt(f), s_n \rangle$ is a no-instance of MCSP implying s_n is strictly less than $\tilde{O}(t(n))$, where s_n fixed for all instances of length-n by the naturalness of R'. Upon establishing such upper-bound on the circuit size of a witness for R', we can take advantage of it to build a depth-3 circuit computing PARITY which contradicts Hastad's Theorem which states that PARITY cannot be computed by constant-depth circuits [Has86]. (TO DO: give more details on how each level of the circuit looks like and potentially a picture)

For full details on how all of the introduced steps were done in details, we refer the readers to the actual paper by Murray and Williams.

4 Conclusion and Open Problems

To summarize, the results shown in "Circuit Minimization Problem" by Kabanets and Cai neither give evidence against the **NP**-hardness of MCSP nor refute the conjecture that MCSP $\in$ **P**. Instead, they explain how establishing such proofs is difficult. First, even though we expect that MCSP cannot be efficiently solved, showing its **NP**-hardness under natural reduction entails strong statements about circuit complexity, which in turn imply the existence of certain pseudorandom generators. Both of which are known to be difficult to obtain. On the other hand, it seems likely that MCSP is not contained in **P**. This is mainly because MCSP being efficiently solvable contradicts the existence of one-way functions, which is a widely held belief in cryptography.

Our research is, therefore, taking the direction of showing the hardness of MCSP, which is a crucial step in obtaining the soundness of this cryptographic assumption. Besides [SS20] and [MW17], there has been a long sequence of other similar work ([HP15], [AHK17], [AH19]) which tackles the results given in Section 3.1 to demonstrate further evidence of the difficulty of showing **NP**-hardness of MCSP. However, researchers have managed to prove that some variants of MCSP are **NP**-complete. Namely, DNF-MCSP [Mas79], MCSP for OR – AND – MOD Circuits [HOS18], MCSP for multi-output functions [ILO20], and MCSP for partial functions and constant depth formulae [Ila20] are now known to be **NP**-complete. Furthermore, some theorists have been trying to prove that MCSP is **NP**-intermediate, which, while still implying a separation of **NP** from **P**, avoids some of the hard ramifications we have seen for **NP**-completeness.

To wrap up our tutorial, we would like to provide a list of interesting open problems in this line of work TO DO: write more open problems to the list

- Can we apply a similar proof framework and technique as in [KC00] and [SS20] but for other types of reduction, i.e. truth-table reduction, to obtain some circuit lower-bounds? How about un-restricted many-one or Turing reductions as mentioned at the end of [SS20]?

- Recall in Section 3.2 that one of the key parts of the proof of Theorem 18 is to use the SAT-refuter under the assumption that **P** $\neq$ **NP** [GSTS05] as a subroutine to construct hard instances to conclude the circuit lower-bound. Recently, Lijie Chen *et al.* [CJSW22] have generalized such refuter construction for other assumptions under a variety of other complexity classes such as **P** $\neq$ **PSPACE** or **BPP** $\neq$ **NEXP**. So, can we use these refuters to obtain other (circuit) lower-bounds?

- In [Ila20], the circuits under consideration were under DeMorgan's Basis, so would a similar result holds if we change the basis, e.g. XOR-gate is allowed? Furthermore, can we apply the

Reverse Gate Elimination technique that Rahul introduced to prove hardness of total MCSP, starting with the DeMorgan's Basis? If not, what are the barriers?

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009.

[AH19] Eric Allender and Shuichi Hirahara. New insights on the (non-) hardness of circuit minimization and related problems. *ACM Transactions on Computation Theory (TOCT)*, 11(4):1–27, 2019.

[AHK17] Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. *computational complexity*, 26(2):469–496, 2017.

[CJSW22] Lijie Chen, Ce Jin, Rahul Santhanam, and R. Ryan Williams. Constructive separations and their consequences. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 646–657, 2022.

[Gol10] Oded Goldreich. *A primer on pseudorandom generators*, volume 55. American Mathematical Soc., 2010.

[Gol11] Oded Goldreich. *In a World of P=BPP*, pages 191–232. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[GSTS05] D. Gutfreund, R. Shaltiel, and A. Ta-Shma. If np languages are hard on the worst-case then it is easy to find their hard instances. In *20th Annual IEEE Conference on Computational Complexity (CCC'05)*, pages 243–257, 2005.

[Has86] J Hastad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, page 6–20, New York, NY, USA, 1986. Association for Computing Machinery.

[HOS18] Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. Np-hardness of minimum circuit size problem for or-and-mod circuits. 2018.

[HP15] John M Hitchcock and Aduri Pavan. On the np-completeness of the minimum circuit size problem. In *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[Ila20] Rahul Ilango. Constant depth formula and partial function versions of mcsp are hard. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2020.

[ILO20] Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. Np-hardness of circuit minimization for multi-output functions. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 27, page 21, 2020.

[IP99] R. Impagliazzo and R. Paturi. Complexity of k-sat. In *Proceedings. Fourteenth Annual IEEE Conference on Computational Complexity (Formerly: Structure in Complexity Theory Conference) (Cat.No.99CB36317)*, pages 237–240, 1999.

[KC00] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00, page 73–79, New York, NY, USA, 2000. Association for Computing Machinery.

[Mas79] William J Masek. Some np-complete set covering problems. *Unpublished manuscript*, 1979.

[MW17] Cody D Murray and R Ryan Williams. On the (non) np-hardness of computing circuit complexity. *Theory of Computing*, 13(1):1–22, 2017.

[MW18] Cody Murray and Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for np and nqp. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 890–901, 2018.

[SS20] Michael Saks and Rahul Santhanam. Circuit lower bounds from np-hardness of MCSP under turing reductions. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 26:1–26:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[Tra84] B. A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.

[Wil16] R. Ryan Williams. Natural proofs versus derandomization. *SIAM Journal on Computing*, 45(2):497–529, 2016.

A Appendix

In this section, we provide the proofs of the lemmata that were used to prove Theorem 9 for the readers reference.

A.1 Proof of Lemma 10

Lemma 10 states that giving a quasi-polynomial time oracle access to a machine also running in quasi-polynomial time does not make it more powerful. Namely, $\mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP}$.

Proof. Let M be a TM deciding a language $L \in \mathbf{QP}^{\mathbf{QP}}$ in quasi-polynomial time, say $\exp(\log^c n)$. We show that carrying out the oracle computation instead of calling the oracle does still guarantee a quasi-polynomial running time. To this end, let M' be the TM deciding the $\mathbf{QP}$-oracle, say in time $\exp\left(\log^{c'} m\right)$. Now, any input to M' is at most of length $m = \exp(\log^c n)$. We therefore need time at most $\exp(\log^c(\exp(\log^c n))) = \exp\left(\log^{cc'} n\right)$ for one oracle computation. At the same time, there are at most $\exp(\log^c n)$ calls, resulting in a total running time bound of $\exp\Big(\log^c(\exp\Big(\log^{cc'} n\Big))\Big) = \exp\Big(\log^{c^2c'} n\Big)$, which is still quasi-polynomial. $\square$

A.2 Proof of Lemma 11

Lemma 11 states that if an **NP** problem can be deterministically solved in quasi-polynomial time, then the polynomial hierarchy is contained in **QP**. Namely, if $\mathbf{NP} \subseteq \mathbf{QP}$, then $\mathbf{PH} \subseteq \mathbf{QP}$.

Proof. Recall that we can define **PH** in terms of oracles by

$$\mathbf{PH} = \bigcup_{n>0} \underbrace{\mathbf{NP}^{\mathbf{NP}^{\cdots^{\mathbf{NP}}}}}_{n \text{ times}} .$$

We can use a straightforward induction over n to show that $\mathbf{PH} \subseteq \mathbf{QP}$. Note that the induction basis is just the assumption. Furthermore, by the inductive hypothesis, we have that

$$\underbrace{\mathbf{NP}^{\mathbf{NP}^{\cdots^{\mathbf{NP}}}}}_{n \text{ times}} \subseteq \mathbf{QP} \quad\Longrightarrow\quad \underbrace{\mathbf{NP}^{\overbrace{\mathbf{NP}^{\mathbf{NP}^{\cdots^{\mathbf{NP}}}}}^{n \text{ times}}}}_{n+1 \text{ times}} \subseteq \mathbf{NP}^{\mathbf{QP}} .$$

Now, $\mathbf{NP}^{\mathbf{QP}} \subseteq \mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP}$ by Lemma 10 which concludes the proof. $\square$

A.3 Proof of Lemma 12

Lemma 12 states that $\mathbf{QP}^{\mathbf{\Sigma}_k^p}$ contains a language that does not belong to i.o. $\mathbf{P}_{/\mathbf{poly}}$ for some $k \in \mathbb{N}$.

Proof. The proof follows a nonuniform diagonalization argument. We first define a language that will be hard to compute for any polynomial-size circuit family: Let L' be the language consisting of tuples $\langle x, 1^{\exp(\log^3 n)}\rangle$ with $n := |x|$ such that $C(x) = 1$ where C is the lexicographically first circuit of size $\exp(\log^3 n)$ which is not computed by any circuit of size $\exp(\log^2 n)$. The existence of such a circuit for sufficiently large n follows from a slightly more careful analysis of the nonuniform hierarchy theorem (Theorem 6.22 in [AB09]).

Observe that can decide membership $\langle x, 1^{\exp(\log^3 n)}\rangle \in L'$ by a $\mathbf{\Sigma}_4^p$-type machine. Specifically, the circuit C computing x is the lexicographic first circuit of size $\exp(\log^3 n)$ which is not computed by any circuit of size $\exp(\log^2 n)$ means that: there exists a circuit C whose size $|C| \leq \exp(\log^3 n)$ such that for all $C' <_{\text{lex}} C$, there exists a smaller circuit C'' whose size $|C''| \leq \exp(\log^2 n)$ such that $C' \equiv C''$ on all inputs $z \in \{0,1\}^n$. With a bit of care, one can reformulate this statement to show that $x \in L'$ is equivalent to an expression of the $\exists\forall\exists\forall$-form which deduces to the fact that $L' \in \mathbf{\Sigma}_4^p$ and concludes $\mathbf{\Sigma}_4^p \not\subseteq \mathbf{SIZE}(\text{poly})$ by definition of a member of L' itself.

Finally, we define our language L of superpolynomial circuit complexity as the output of a $\mathbf{QP}^{\mathbf{\Sigma}_4^p}$-machine T as defined above: Given an input $x \in \{0,1\}^n$, query the oracle for L' with $\langle x, 1^{\exp(\log^3 n)}\rangle$ and output its answer.

We constructed this language such that it is hard for a polynomial-size circuit to compute. To see this, assume to the contrary that there is a n^a-size circuit family. However, since $n^a \in o(\exp(\log^2 n))$ and we particularly excluded any circuits of size less than $\exp(\log^2 n)$, this is a contradiction. $\square$

A.4 Proof of Lemma 13

Lemma 13 states that there are $O(s^{3s})$ different circuits of some size s. In particular, if we substitute s with $n^{\mathrm{polylog}(n)}$ and n^{ε}, we obtain the following

1. $n^{\mathrm{polylog}(n)}$ circuits of size $\log^c n$ for any $c > 0$ and
2. $O(2^{n^{2\varepsilon}})$ circuits of size n^{ε} for any $\varepsilon > 0$.

Proof. In a circuit with $s \in \mathbb{N}$ many gates and inputs, each gate is connected to at most two out of s gates and computes one of the functions $\wedge, \vee, \neg$. This means there are at most $3 \cdot s^2$ choices to construct each gate and thus $(3s^2)^s = O(s^{3s})$ choices to construct the whole circuit.

1. Setting $s \coloneqq \log^c n$ gives us

$$O((\log^c n)^{3\log^c n}) = O(2^{(\log^{3c} n)\cdot(\log^c n)}) = O(2^{\log^{4c} n}) = n^{\mathrm{polylog}(n)}$$

   many ways to construct a circuit of size $\log^c n$, while

2. setting $s \coloneqq n^{\varepsilon} = 2^{\varepsilon \log n}$ yields $O((2^{\varepsilon \log n})^{n^{\varepsilon}}) = O(2^{n^{2\varepsilon}})$ different circuits of size n^{ε}.

□