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

(final draft)

Ngu Dang and Fabian Späh

December 11, 2020

1 Introduction

In the Minimum Circuit Size Problem (MCSP), we are given a truth table of some Boolean function together with a positive integer s_n as input, and our task is to answer the question whether there exists a circuit of size at most s_n that computes the function represented by the given truth table.

Problem:	MSCP
Input:	A tuple $\langle T_n, s_n \rangle$ consisting of a truth table T_n for a Boolean function of arity n and an integer s_n
Question:	Is there a circuit C_n of size at most s_n computing T_n?

Finding solutions to MSCP in an important part in the design of integrated circuits where it is important to minimize space and execution speed of a computational component. However, motivations to study this problem are manifold. One lies in meta-complexity which is loosely the study of problems whose instances contain some form of execution instruction themselves. For each individual instance, we want to know how complex the object resulting from this instruction is. So, studying such a meta problem entails determining the complexity of determining complexity itself. Now, MCSP is posed exactly as such a meta problem, whereas the the instruction is stated in terms of a truthtable and the complexity is formalized as the size of a minimal circuit. Given the expressiveness of truthtables and importance of circuits in representing computation, MCSP is therefore central to meta-complexity. In fact, the minimization of circuits was already studied in the Soviet Union as part of the question of whether brute-force search can be eliminated in general [Tra84]. We can also derive MCSP from the more general Kolmogorov complexity: For a universal Turing machine U and $x \in \{0,1\}^*$, define $K_U(x) \coloneqq \min\{\, |d| \mid U(d) = x \,\}$ as the minimum size of a program which generates x. Now, the decision version of K_U is undecidable, so weaker notions restricting at least the execution time of U are studied, and MCSP is one of them. This also relates the study of MCSP to the typcial questions in information theory. In particular, the minimum circuit size puts a number on the information stored in a truth table T_n. 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. The widely applied decision tree algorithms are just a special case of this, but circuits are

of course more succinct. Finally, and we will see this in this paper, MCSP has strong connections to the existence of pseudorandom generators, an important assertion in cryptography.

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_n, and verify whether C computes each entry of the truth table correctly in polynomial time. Despite the huge interest and long history of research in MSCP, we know relatively little beyond that. In "Circuit Minimization Problem," Valentine Kabanets and Jin-Yi Cai addressed the difficulty of showing MCSP to be **NP**-hard [KC00]. They do this by providing some strong implications if there exists a polynomial-time reduction R from SAT to MCSP that is "natural," in the sense that the size of the output depends on the size of the inputs only, and these sizes are polynomially related. Furthermore, the authors pointed out that there has also been no proof that $\mathsf{MCSP} \notin \mathbf{P}$. Clearly, such proof would imply $\mathbf{P} \neq \mathbf{NP}$ which goes beyond the currently known techniques. The authors did, however, provide some clues on why this conjecture seems plausible; again, by providing consequences of questionable likelihood if $\mathsf{MCSP} \in \mathbf{P}$.

In this review, we will first provide 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 and of $\mathsf{MCSP} \in \mathbf{P}$. Finally, we conclude the review by giving remarks and directions for further research on this topic in Section 4.

2 Preliminaries

We begin by introducing some useful background, including the definition of some complexity classes, the concept of natural reductions, and pseudorandomness.

2.1 Complexity Classes

Here, we provide the necessary definitions for complexity classes used throughout this paper. For more details and a full discussion, see [AB09].

Definition 1. The class *sub-exponential* is $\mathbf{SUBEXP} \coloneqq \bigcap_{\epsilon>0} \mathbf{DTIME}(2^{n^\epsilon})$.

Note that **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

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

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 $\mathrm{polylog}(n) \in \Omega(1)$.

Also, **SUBEXP** contains **QP** since for every $\varepsilon > 0$ and $c > 1$ holds that $\exp(\log^c n) \in o(\exp(n^\varepsilon))$.[1]

[1]To analyze the relationship of exponentials, the following proofs 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

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

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

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.

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

Note that C is obviously containend in $\mathsf{i.o.}\, C$. For brevity, we usually write $C \subseteq C$ (i.o.) or $C = C$ (i.o.) in place of $C' \subseteq \mathsf{i.o.}\, C$ and $C' = \mathsf{i.o.}\, C$, respectively.

2.2 Natural Reductions

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

Definition 5. 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 $R = \langle R_1, R_2\rangle$ from A to B, whereas R_1 maps to instances y of the original search problem and R_2 to the parameters s, is called *natural* if for all instances x of A holds that

- there exists a $c \in \mathbb{R}_+$ such that $|x|^{1/c} \leq |R(x)| \leq |x|^c$, meaning R_1 does not suddenly grow or shrink, and
- R_2 depends only on the length of the instance x, i.e. there exists a function $f\colon \mathbb{N} \to \mathbb{N}$ such that $R_2(x) = f(|x|)$.

All **NP**-complete problems we are aware of seem to be complete under natural reductions. This is not 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 give a good understanding, we wrap up with the following example of an unnatural reduction made natural.

Example 6. We reduce from Partition[2] to SubsetSum :[3]

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

Now, while this reduction is correct (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 for two reasons: 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

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

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

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

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

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

Given a short string of truly random bits, a *pseudorandom generator* tries to enlarge this sequence to obtain more "artificial" randomness. We can evaluate the randomness of this resulting sequence is in the fashion of meta-complexity by testing how likely the best adversary is in distinguishing it from true randomness:

Definition 7 (Pseudorandom Generators)**.** A family $\{G_n\}_n$ with $G_n\colon \{0,1\}^n \to \{0,1\}^{S(n)}$ is called an $S(n)$-pseudorandom generator if $\{G_n\}_n \in \mathbf{E}$ and

$$\Pr_{x\in\{0,1\}^n}[C(G(x)) = 1] - \Pr_{y\in\{0,1\}^{S(n)}}[C(y) = 1] < \frac{1}{10}$$

for every circuit C of size at most $S(n)^3$.

The existence of certain pseudorandom generators would allows us to derandomize algorithms in **BPP**. The basic idea is to replace the randomness used in a probabilistic algorithm with pseudorandmoness 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 $S(n)$ is large enough, this leads to an efficient derandomization.

Definition 8 (One-Way Functions)**.** A polynomial-time computable function $f\colon \{0,1\}^* \to \{0,1\}^*$ with $|f(x)| = |x|$ is a *one-way function* if for every constant c and every non-uniform algorithm A running in time $O(n^c)$ holds

$$\Pr_{x\in\{0,1\}^n}\left[A(f(x)) \in f^{-1}(f(x))\right] \leq \frac{1}{n^c}$$

for every $n \in \mathbb{N}$.

In words: no matter how hard an $O(n^c)$ time-constrained adversary tries to find a pre-image of $f(x)$, he can only do so with probability at most $1/n^c$. Note that this statement is made about the average-case hardness of inverting f. As the adversary is time constrained, we can therefore construct one-way functions from problems that we belive to be hard on average. As such, one-way functioins are generally assumed to exist. Moreover, we can show that the existence of certain pseudorandom generators implies the existence of one-way functions [HILL99].

In particular, the existence of one-way functions is vital for cryptography. For example, Bitcoin's blockchain consensus algorithm relies on the hardness of inverting a hash-function which is believed to be one-way. If an adversary would find a way to circumvent the brute-force search efficiently and with good probability, he would be able to take control over the protocol.

We wrap up with defining a special kind of pseudorandom generator:

Definition 9. ([KC00]) The *hardness* $H(G)$ of a pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{2n}$ is defined as the minimal s such that there exists a circuit C of size at most s for which

$$\Pr_{x\in\{0,1\}^n}[C(G_n(x)) = 1] - \Pr_{x\in\{0,1\}^{2n}}[C(y) = 1] \geq 1/s$$

The pseudorandom generator G is called *strong* if it has hardness $H(G) > 2^{n^{\Omega(1)}}$

Definition 10 (Strong Pseudorandom Generators)**.** A pseudorandom generator $G_d \colon G_m$

3 Main Results

3.1 MCSP and NP-completeness

To begin with, we want to emphasize that researchers have not figured out yet whether it is possible to prove the **NP**-hardness of MCSP. The difficulty of such proof was explicitly addressed through some implications for *Circuit Complexity* and **BPP**. In other words, the authors provided some consequences that are still unknown to the current state of the art if MCSP is **NP**-hard under a natural reduction.

Now, we are ready to look at the first key theorem which is about the implication for *Circuit Complexity* if MCSP is **NP**-hard under the *natural* reduction.

Theorem 11 ([KC00])**.** *If* MCSP *is* **NP***-hard under a natural reduction from* SAT*, then* **E** *contains a family of Boolean functions* f_k *not in* $\mathsf{i.o.}\,\mathbf{P}_{/\mathbf{poly}}$*, i.e. of superpolynomial circuit complexity.*

Before we move on to the proof, let us examine some lemmata that are useful for establishing this result.

Lemma 12. $\mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP}$.

Proof. Let M be a TM dedicing 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 **QP**-oracle, say in time $\exp(\log^{c'} m)$. Now, any input to M' is at most of length $m = \exp(\log^c n)$. We therefore need time at most $\exp(\log^c(\exp(\log^c n))) = \exp(\log^{cc'} n)$ 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 $\exp(\log^c(\exp(\log^{cc'} n))) = \exp(\log^{c^2c'} n)$ which is still quasi-polynomial. □

Lemma 13. *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 straithgforward 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 12. □

Lemma 14. $\mathbf{QP}^{\mathbf{\Sigma}_k^p}$ *contains a language which does not belong to* $\mathsf{i.o.}\,\mathbf{P}_{/\mathbf{poly}}$ *for some* $k \in \mathbb{N}$.

Proof. The proof follows a nonuniform diagonalization argument. We first define a language which 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).

We can decide membership $\langle x, 1^{\exp(\log^3 n)} \rangle \in L'$ by a $\mathbf{\Sigma}_4^p$-oracle as in Problem (1c) of Homework 7. Finally, we define our language L of superpolynomial circuit complexity as the output of a $\mathbf{QP}^{\mathbf{\Sigma}_4^p}$-machine: 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. □

Lemma 15. *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* $\varepsilon > 0$.

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

□

We now know have enough tools to prove Theorem 11. We remind ourselves of the statement to prove: If there is a natural polynomial-time reduction from SAT to MCSP, then $\mathbf{E}$ contains a family of Boolean functions f_k of superpolynomial circuit size.

Proof (Theorem 11). We separate the prove along two cases.

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

  Applying Lemma 12 and Lemma 13 to this assumption yields $\mathbf{QP}^{\mathbf{PH}} \subseteq \mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP} \subseteq \mathbf{E}$. Lemma 14 shows that $\mathbf{E}$ also contains a language of superpolynomial circuit complexity (i.o.). Hence, $\mathbf{E} \not\subseteq \mathsf{i.o.}\,\mathbf{P}_{/\mathbf{poly}}$

- Case 2: $\mathbf{NP} \not\subseteq \mathbf{QP}$

  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 which cannot be represented by ciruits of polynomial size.

  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. 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 $L \in \mathbf{E}$.

  (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 thus assume to the contrary that it is bound 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 are going to decide the membership of this instance to MSCP instead of solving the original satisfiability problem. By Lemma 15, 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. However, this implies that $\mathsf{SAT} \in \mathbf{QP}$ 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 $\mathrm{poly}(k) < s_n$ for all $k > k_0$. We conclude that $L \notin \mathsf{i.o.}\,\mathbf{P}_{/\mathbf{poly}}$.

□

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 16 ([KC00])**.** *If* MCSP *is* $\mathbf{NP}$*-hard under a natural reduction from* SAT *and* $\mathbf{NP} \not\subseteq \mathbf{SUBEXP}$*, then* $\mathbf{E}$ *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 11. 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 15 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

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

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, $\mathsf{SAT} \in \mathbf{SUBEXP}$ 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. □

Now, we will look at the implications for $\mathbf{BPP}$ when $\mathbf{NP}$-hard under a natural reduction from SAT. The following two theorems on hardness-randomness trade-offs are needed to establish the one about $\mathbf{BPP}$.

Theorem 17 ([BFNW93])**.** *If the class* **EXP** *contains a family of Boolean functions of superpolynomial circuit complexity (i.o.), then* $\mathbf{BPP} \subseteq \mathbf{SUBEXP}$ *(i.o.).*

Theorem 18 ([IW97])**.** *If the class* **E** *contains a family of Boolean functions* $f_n : \{0,1\}^n \to \{0,1\}$ *of circuit complexity at least* $2^{\epsilon n}$ *for some* $\epsilon > 0$*, (i.o.), then* $\mathbf{BPP} = \mathbf{P}$ *(i.o.).*

Theorem 19 ([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* $\mathbf{NP} \subseteq \mathbf{SUBEXP}$*.*

Proof. For (1), combine the result of Theorem 11 with the premise of Theorem 17; for (2), combine Theorems 16 and 18 in the same way. □

In words, if MCSP is **NP**-hard under a natural reduction from SAT, then every problem in **BPP** can be efficiently solved. Now, we actually believe that $\mathbf{P} = \mathbf{BPP}$ as most algorithms in **BPP** surrender to derandomization. However, it is known that showing $\mathbf{BPP} = \mathbf{P}$ entails proving the existence of certain pseudorandom generators which in turn would prove $\mathbf{BPP} = \mathbf{P}$ 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 20. *If* MCSP *is* **NP***-hard under a natural reduction from* SAT*, then* $\mathbf{BPP} \subsetneq \mathbf{E}$

Proof. We first show that the inclusion $\mathbf{SUBEXP} \subseteq \mathbf{E}$ 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 $2^{\sqrt{n}} \log(2^{\sqrt{n}}) = 2^{\sqrt{n} \log \sqrt{n}} \in o(2^n)$ We conclude that $\mathbf{SUBEXP} \subseteq \mathbf{DTIME}(2^{\sqrt{n}}) \subsetneq \mathbf{E}$ Furthermore, from Theorem 19, we know that if MCSP is **NP**-hard under a natural reduction from SAT, then $\mathbf{BPP} \subseteq \mathbf{SUBEXP}$. Combining both gives us $\mathbf{BPP} \subseteq \mathbf{SUBEXP} \subsetneq \mathbf{E}$. □

3.2 MCSP and P

In the previous section we have discussed the difficulties with proving the **NP**-completeness of MCSP under a natural reduction. Nonetheless, in this section we consider the opposite assumption, namely that MCSP belongs in **P**, which results in some even more surprising and unlikely consequences.

Let us start with providing a strong implication on the field of cryptography, by showing that if MCSP can be efficiently solved, we are be able to factor Blums integers[4] well on the average. Specifically, let us examine the following theorem and its consequence.

Theorem 21. *If* MCSP *is in* $\mathbf{P}_{/\mathbf{poly}}$*, then there is no strong pseudorandom generator in* $\mathbf{P}_{/\mathbf{poly}}$*.*

This theorem is a direct consequence of the "Natural Proofs Barrier" conceived by Razborov and Rudich [RR97]. We highly suggest this read to the interested reader. We generally believe that the pseudorandom generator based on factoring (Blum) integers is strong. However, Theorem 21 rules out such a prospective, so we obtain:

[4] A Blum integer is a product of two distinct primes which are both congruent to 3 mod 4. Factoring Blum integers is believed to be as hard as factoring general integers.

Corollary 22. *If* MCSP *is in* **P**, *then, for any* $\epsilon > 0$, *there is an algorithm running in time* 2^{n^ϵ} *that factors Blum integers well on the average.*

Thus, the corollary above suggests that if MCSP can be efficiently solved, then we can break the strong pseudorandom generator of factoring Blum integers with a good enough average-case algorithm. In other words, if MCSP is in **P**, then current cryptographic schemes relying on the hardness of integer factoring (e.g. RSA or Diffie-Hellman key exchange) are prone to efficient attacks. Even if factoring turns out to be easy, Theorem 21 still puts a questionmark behind the existence of pseudorandom generators and as such on one-way functions. However, it is mainly believed that factoring is hard and one-way functions exist and therefore it is very much unlikely that MCSP can be efficiently solved; however, nothing has been shown to draw an ultimate conclusion about this assumption.

We want to look at a second consequence of $\mathsf{MCSP} \in \mathbf{P}$ for randomized algorithms. To this end, let us first examine the following theorem.

Theorem 23 ([KC00])**.** $\mathbf{BPP} \subseteq \mathbf{ZPP}^{\mathsf{MCSP}}$

This is a stronger version of the previoulsy known inclusion $\mathbf{BPP} \subseteq \mathbf{ZPP}^{\mathbf{NP}}$. In fact, this can be obtained by combining the result $\mathbf{BPP} \subseteq \mathbf{MA}$ achieved through techniques from to the Sipser-Gács Theorem with $\mathbf{MA} \subseteq \mathbf{ZPP}^{\mathsf{MCSP}}$ [GZ97]. We could obtain the Theorem 23 easily from that if MCSP were shown to be **NP**-hard.

Now, using the identity $\mathbf{ZPP}^{\mathbf{P}} = \mathbf{ZPP}$, we obtain the following as a corollary.

Corollary 24. *If* MCSP *is in* **P**, *then* $\mathbf{BPP} \subseteq \mathbf{ZPP}$.

We know that by definition, $\mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{BPP}$ (cf. [AB09], Chapter 7). However, it is yet unknown whether $\mathbf{BPP} \subseteq \mathbf{RP}$ or $\mathbf{BPP} \subseteq \mathbf{NP}$. But if MCSP can be efficiently solved, then by the above corollary, we obtain $\mathbf{ZPP} = \mathbf{RP} = \mathbf{BPP}$ which would be a big breakthrough. In particular, if MCSP were in **P**, then any efficient algorithm with two-sided error could be simulated to run efficiently in expectation with zero-sided error.

4 Conclusion

To summarize, the results shown in "Circuit Minimization Problem" by Kabanets and Cai neither give any evidence against the **NP**-hardness of MCSP nor refute the conjecture that $\mathsf{MCSP} \in \mathbf{P}$. Instead, they suggest that establishing such proofs would be a daunting task since it implies some certain consequences happen for complexity classes - some seem plausible, some not, but all are beyond the scope of our current techniques.

There has been a long sequence of works ([HP15], [AHK17], [MW17], [AH19]) which tackles the results given in Section 3.1 to give 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], and MCSP for multi-output functions [ILO20] are **NP**-complete. Furthermore, some theorists have been trying to prove that MCSP is **NP**-intermediate [5] which is a pretty interesting research direction.

[5] see chapter 3 of [AB09]

As for MCSP and the class **P**, it is very likely that MCSP $\notin$ **P** mainly because it will put an end to cryptography as briefly mentioned in Section 3.2. One thing we want to emphasize is that if the existence of one-way function can be proved one day, we can finally conclude that MCSP is not in **P** which implies the separation of **P** and **NP**.

We expect (but again, do not know) that MCSP cannot be efficiently solved and our research is taking the direction of establishing the hardness of MCSP, which is a crucial step in obtaining the "immortality" of cryptography. This is a wrap up of our review, thank you all the readers for your time, and we wish you a Merry Christmas and a Happy New Year!!!

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.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. Bpp has subexponential time simulations unlessexptime has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

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

[GZ97] Oded Goldreich and David Zuckerman. Another proof that bpp?ph (and more). *Electronic Colloquium on Computational Complexity - ECCC*, 01 1997.

[HILL99] Johan HÅstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, March 1999.

[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.

[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.

[IW97] Russell Impagliazzo and Avi Wigderson. P = bpp if e requires exponential circuits. pages 220–229, 01 1997.

[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.

[RR97] Alexander A Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, August 1997.

[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.