“A first glimpse on why MCSP is interesting” or something like that

...

November 20, 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?

It is easy to see that MCSP is in $\mathbf{NP}$. 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. With that being said, a natural question arises: Is MCSP $\mathbf{NP}$-complete?

In “Circuit Minimization Problem,” Valentine Kabanets and Jin-Yi Cai addressed the difficulty of showing MCSP to be $\mathbf{NP}$-hard [KC00]. They showed some consequences that are unlikely to happen 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 why it is challenging to prove $\mathsf{MCSP} \notin \mathbf{P}$ (again, by providing some consequences whose likelihood is questionable). Clearly, such proof would imply $\mathbf{P} \neq \mathbf{NP}$ which goes beyond the currently known techniques.

In this review, we will provide some definitions required to understand the main results and key theorems of the paper in Section 2. Section 3 introduces some main consequences when $\mathsf{MCSP} \notin \mathbf{P}$ and MCSP is $\mathbf{NP}$-hard under “natural” reductions. Finally, we conclude the review by providing some insightful remarks and directions for further research on this topic in Section 4.

2 Preliminaries

In this section, we provide some definitions that, we believe, are useful for the readers including some complexity classes, Natural (Karp) Reduction, Infinitely-often (**i.o.**) Simulations, and Pseudorandom Generators.

2.1 Some Complexity Classes

We start with providing necessary definitions for some complexity classes. 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]

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 $\mathbf{QP} \subseteq \mathbf{E}$.

2.2 Natural Reductions

In this subsection, we will provide you the definition of *Natural Reduction*. Intuitively speaking, for two problems A and B, and a Karp reduction from A to B, we say the reduction R is natural if, for any instance I of A, the length of the output depends only on the input length, and those lengths are polynomial related. To make sure that the readers can get a better understanding of the proofs that we introduce in the later sections, we provide the formal version which specifies more technical properties of the "mapping" process.

Definition 4. 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 $|x|^{1/c} \le |R(x)| \le |x|^c$, 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 $R_2(x) = f(|x|)$.

[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, namely

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

Example 5. SAT $\leq_p$ 3SAT is "natural," given by $R = R_1(\varphi) = \varphi'$ where φ is some regular CNF and φ' is a 3CNF-formula. The general strategy for the reduction is to split some clause C in φ of size $k > 3$ into a pair of two equivalent clauses C_1 of size $k-1$ and C_2 of size 3 and we repeat the process until we get the desired 3CNF formula, φ'. Thus, it is easy to see that the length of φ' is not stretched by much as we just add more clauses solely based on everything from the original formula φ. In this example, R_2 is not needed as the 3SAT problem does not require any numerical value.

Example 6. 3SAT $\leq_p$ INDSET[2] is "natural," given by $R = \langle R_1(\varphi) = G, R_2(\varphi) = k$ where φ is some 3CNF formula, G is a graph, and k is the target size of the independent set that we want for G. The general strategy for the reduction is to create a graph of $7m$ vertices from a 3CNF formula of m clauses, and the number k can be easily determined using m, the number of clauses. It is easy to see that the length of the output of the reduction is not stretched by much. [3]

In fact, all **NP**-complete parameterized problems we are aware of seem to be complete under natural reductions. To wrap up, we introduce one more example which we found interesting, for the readers' further reference.

Example 7. We reduce from Partition[4] to SubsetSum :[5]

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

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 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

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

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

$$2^{\sigma+2} = r + \textstyle\sum_{i \in S} 2a_i \iff \frac{1}{2}\sum_{i=1}^n = \sum_{i \in S} a_i$$

where S is the rest of the subset without r.

[2] INDSET $= \{\langle G, k \rangle \mid G$ has independent set of size k$\}$

[3] for full details of the reductions of both examples, see [AB09], chapter 2, page 49 and 51 respectively.

[4] 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 $\sum_{i \in S} a_i = \sum_{i \in T} a_i$.

[5] The partition problem 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 $s = \sum_{i \in S} a_i$.

2.3 Infinitely-often Simulations

We introduce a modifier to a complexity class C.

Definition 8. 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 $\mathsf{i.o.}\,C$.

2.4 Pseudoramdom generator

Definition 9. *Pseudorandom generators*: A distribution R over $\{0,1\}^m$ is (S,ϵ)-pseudorandom (for $S \in \mathbb{N}$, $\epsilon > 0$) if for every circuit C of size at most S.

$$|Pr[C(R) = 1] - Pr[C(U_m) = 1]| < \epsilon$$

where U_m denotes the uniform distribution over $\{0,1\}^m$.

Let $S : \mathbb{N} \to \mathbb{N}$ be some function. A 2^n-time computable function $G : \{0,1\}^* \to \{0,1\}^*$ is an $S(l)$-*pseudorandom generator* if $|G(z)| = S(|z|)$ for every $z \in \{0,1\}^*$ and for every $l \in \mathbb{N}$ the distribution $G(U_l)$ is $(S(l)^3, 1/10)$-pseudorandom.

For more details and a full discussion of this topic, prefer to chapter 20 of [AB09]

Definition 10. ([KC00]) The *hardness* $H(G_k)$ of a pseudorandom generator $G_k : \{0,1\}^k \to \{0,1\}^{2k}$ is defined as the minimal s such that there exists a circuit C of size at most s for which

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

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

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 or not. 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 the "natural" Karp reduction.

Before we move on to some key theorems of this section, let us examine some lemmas that are useful for establishing.

Lemma 11. $\mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP}$.

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 $\exp(\log^c(\exp(\log^{cc'} n))) = \exp(\log^{c^2 c'} n)$ which is still quasi-polynomial. $\square$

Lemma 12. *If* $\mathbf{NP} \subseteq \mathbf{QP}$ *then* $\mathbf{PH} \subseteq \mathbf{QP}$.

Proof. Recall that we can define $\mathbf{PH}$ 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 11. $\square$

Lemma 13. $\mathbf{QP}^{\mathbf{\Sigma}_k^p}$ *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. $\square$

Lemma 14. *There are* $O(s^{3s})$ *different circuits of size* s. *In particular, there are*

1. $n^{\operatorname{polylog}(n)}$ *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

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

   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.

$\square$

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 15 ([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.*

Proof. We separate the prove along two cases.

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

  Applying Lemma 11 and Lemma 12 to this assumption yields $\mathbf{QP}^{\mathbf{PH}} \subseteq \mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP} \subseteq \mathbf{E}$. Lemma 13 shows that **E** also contains a language of superpolynomial circuit complexity (i.o.). Hence, $\mathbf{E} \not\subseteq$ 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 14, 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 $L \notin \mathsf{i.o.}\,\mathbf{P}_{/\mathbf{poly}}$.

$\square$

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.) (meaning not in* $\mathsf{i.o.}\,\mathrm{size}(2^{o(k)})$*)*

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 15. 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 14 shows that there are $O(2^{n^{2\varepsilon}})$ circuits of size at most s_n. Enumerating all such circuits and checking whether they represent $T_{k'}$ therefore takes time

$$\underbrace{O(2^{n^{2\varepsilon}})}_{\#\text{circuits}} \cdot \underbrace{O(2^{k'})}_{\#\text{entries}} \cdot \underbrace{O(s_n)}_{\text{evaluate}} = O(2^{n^{2\varepsilon}}) \cdot \mathrm{poly}(n) \cdot O(n^{\varepsilon}) = O(2^{n^{3\varepsilon}})\,.$$

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. $\square$

Now, we will look at the implications for **BPP** when **NP**-hard under a natural reduction from SAT. The following two theorems on hardness-randomness trade-offs are needed to establish the one about **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 15 with the premise of Theorem 17; for (2), combine Theorems 16 and 18 in the same way. □

Intuitively speaking, if MCSP is **NP**-hard under a natural reduction from SAT, then a problem in **BPP** can be efficiently solved. 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

As mentioned previously, it is still unknown to us that whether we would be able to reduce MCSP to SAT using the natural Karp reduction. Namely, if such reduction exists, it yields some astonishing results (as stated in section 3.1) which are unlikely to be found soon. Nonetheless, the assumption that MCSP is in **P** gives some surprising consequences.

For starter, if MCSP can be efficiently solved, then we will be able to factor Blums integers [6] well on the average case. 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* $\mathbf{P}_{/\mathbf{poly}}$*.*

As a reminder, the definition of *strong pseudorandom generators* can be found in section 2.4. This theorem is a direct consequence of the main result of "Natural Proofs" by Razborov and Rudich [RR97], and with that being said, this paper is the best reference for those who are interested in the proof of this theorem. In this sense, we obtain a consequence as follows.

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

[6] a Blum integer is the product of two primes, which congruent to 3 mod 4

An example of a pseudorandom generator which is believed to be a strong one is a generator based on factoring Blum integers. 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 $\mathbf{P}$, then our current cryptography breaks because a fast algorithm for factoring (at least for the average-case) can break any type of cryptography. It is mainly believed that factoring is hard 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 will now explore another surprising consequence that would happen if such proof exists.

If MCSP is in $\mathbf{P}$, then any problem in $\mathbf{BPP}$ can be substituted by an equivalent problem in $\mathbf{ZPP}$, roughly speaking. To see a clearer picture, let us first examine the following theorem.

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

From this theorem, we can easily see that if MCSP is $\mathbf{NP}$-hard under natural Karp reduction, then we obtain the following consequence: $\mathbf{BPP} \subseteq \mathbf{ZPP}^{\mathbf{NP}}$. In fact, this inclusion has been shown previously by other theorists and we recommend the reader to visit the following chain of work [Sip83] [ZH86] [Lau83] [NW94] and [GZ97] for full details and discussion of this statement as well as related topics.

However, as mentioned in section 3.1, establishing the hardness of MCSP is very challenging as such proof will imply a decent amount of breakthroughs in theoretical computer science. Let us now assume that it is indeed true, then if MCSP is in $\mathbf{P}$, then we have $\mathbf{P} = \mathbf{NP}$ which provides us the corollary.

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

Also, it is trivial to see from the definitions that $\mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{BPP}$ (see [AB09], chapter 7). However, it is still unknown to our current knowledge 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 $\mathbf{ZPP} = \mathbf{RP} = \mathbf{BPP}$ (holy cow?!) which would be a huge breakthrough.

4 Conclusion

[Will be added when we are done with section 3, but basically, we plan to briefly introduce some other work that tackled the open problems introduced in this paper and then conclude with some further plans of research.]

References

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

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

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

[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.

[Lau83] Clemens Lautemann. Bpp and the polynomial hierarchy. *Information Processing Letters*, 17(4):215 – 217, 1983.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149 – 167, 1994.

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

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, page 330–335, New York, NY, USA, 1983. Association for Computing Machinery.

[ZH86] S. Zachos and H. Heller. A decisive characterization of bpp. *Inf. Control.*, 69:125–135, 1986.