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

Your name, should you choose to include it

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 **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 **NP**-complete?

In "Circuit Minimization Problem," Valentine Kabanets and Jin-Yi Cai addressed the difficulty of showing MCSP to be **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 MCSP $\notin$ **P** (again, by providing some consequences whose likelihood is questionable). Clearly, such proof would imply **P** $\neq$ **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 MCSP $\notin$ **P** and MCSP is **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 the list of some definitions that, we believe, are useful for the readers.

Definition 1. The class Sub-Exponential: $\mathbf{SUBEXP} = \bigcap_{\epsilon>0} \mathbf{DTIME}(2^{n^\epsilon})$

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

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

where we obtain the last equality since

$$2^{(\log n)^{c+1}} = \exp(\log^{c+1} n) = \exp(\log n \log^c n) = n^{\log^c n}.$$

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 $\exp(\log^c n) \in o(\exp(n^\varepsilon))$.

Definition 3. The class exponential time with linear exponential 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 $\mathbf{QP} \subseteq \mathbf{E}$.

2.1 Natural Reductions

Definition 4. *Natural (Karp) Reduction*: 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 $|R(I)|$, as well as all possible output parameters s_n, depend only on the input length $|I|$. Furthermore, $|I|$ and $|R(I)|$ are polynomial related.

For example, $\mathsf{SAT} \leq_p \mathsf{3SAT}$ is "natural." Namely, given φ - an instance in SAT, the general idea 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 $3-CNF$ formula, φ'. Thus, the length of φ' is only dependent on φ as we just add more clauses solely based on everything from the original formula, intuitively speaking.

Other textbook reductions that we know and love (such as $\mathsf{3SAT} \leq_p \mathsf{VERTEXCOVER}$, etc.) are "natural" in this sense. Unnatural reductions, on the other hand, are more contrived. Consider, for example, the following reduction from Clique[1] to SubIso:[2]

$$f(G,n) = \begin{cases} \langle K_1, K_2 \rangle & \text{if } |V(G)| < n \\ \langle G, K_n \rangle & \text{otherwise} \end{cases}$$

where K_n is the complete graph on n vertices. Note that the reduction is correct since G has a clique of size n if and only if it has a connected subgraph K_n. Also, G cannot have a clique of size n if it does not even have n vertices. Also, the reduction can be carried out in polynomial time: In fact, we can commpute the output size as

$$|f(G,n)| = \begin{cases} O(1) & \text{if } |V(G)| < n \\ \Theta(|G|) & \text{otherwise} \end{cases}$$

However, $|f(G,n)|$ is certainly not a function in the input size $(|G| + n)$, so this reduction is not natural. With a little more care, we can still turn this reduction into a natural reduction. We can define

$$f'(G,n) = \begin{cases} \langle G \cup \bar{K}_{\log n}, G \rangle & \text{if } |V(G)| < n \\ \langle G, K_n \rangle & \text{otherwise} \end{cases}$$

[1]The clique problem (Clique) asks for an instance $\langle G, n \rangle$ of a graph G and integer n, whether G has a completely connected subgraph of size n.

[2]The subgraph isomorphism problem (SubIso) asks, given two graphs G and H, whether there is a subgraph G' of G which is isomorphic to H.

where $G \cup H$ denotes the union of two graphs G and H where $\bar{K}_m$ is the empty graph on m vertices. Now, the output size is $|f'(G,n)| = \Theta(|G|)$ in both cases.

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 5. $\mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP}$.

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. □

Lemma 6. *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 5. □

Lemma 7. $\mathbf{QP}^{\mathbf{\Sigma}_k^p}$ *contains a language which does not belong to* $\mathbf{P}_{/\mathbf{poly}}$ *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 8. *There are at most $n^{\mathrm{polylog}(n)}$ different circuits of size $\log^c n$ for a constant c.*

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 $(3s^2)^s$ choices to construct the whole circuit. Setting $s \coloneqq \log^c n$ gives us

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

many ways to construct a circuit of size $\log^c n$. □

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 9. *If* MCSP *is* **NP***-hard under a natural reduction from* SAT*, then*

1. **E** *contains a family of Boolean functions f_n not in $\mathbf{P}_{/\mathbf{poly}}$ (i.o.), and*
2. **E** *contains a family of Boolean functions f_n of circuit complexity $2^{\Omega(n)}$ (i.o.), unless* **NP** $\subseteq$ **SUBEXP** *[the proof for this is yet missing]*

Proof. We separate the prove along two cases.

- Case 1: **NP** $\subseteq$ **QP**

  Applying Lemma 5 and Lemma 6 to this assumption yields that $\mathbf{QP}^{\mathbf{PH}} \subseteq \mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP} \subseteq \mathbf{E}$. Lemma 7 shows that **E** also contains a language of superpolynomial circuit complexity. Hence, $\mathbf{E} \not\subseteq \mathbf{P}_{/\mathbf{poly}}$

- Case 2: **NP** $\not\subseteq$ **QP**

  We pick any infinite set U of unsatisfiable Boolean formulae, say, $U \coloneqq \{\, \phi_n \mid n \in \mathbb{N} \,\}$ where

  $$\phi_n(x_1, \ldots, x_n) \coloneqq (x_1 \wedge \bar{x}_1) \wedge \underbrace{(x_3 \wedge x_4 \wedge \cdots\cdots\cdots \wedge x_n)}_{\text{arbitrary term of size } \Theta(i)} .$$

  Now based on this, we want to apply R to define our language. To this end, let first $\langle T_k, s_n\rangle \coloneqq R(\phi_n)$ for each $\phi_n \in U$. Note that a truth table T_k for a Boolean function in k variables is of size 2^k. Since R is a polynomial-time natural reduction, we have $2^k = \Theta(\mathrm{poly}(n))$ and thus $k = \log(\Theta(\mathrm{poly}(n))) = \Theta(\log n)$. Now, define

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

  to obtain the following.

(i) $L \in E$: We can easily construct a machine running to decide L in time $2^{O(n)}$: Given an input $x \in \{0,1\}^k$, first construct the Boolean formula $\phi_n \in U$ for $n \coloneqq 2^k$. In particular, we can write down ϕ_n in time $2^{O(k)}$. Next, we apply our natural reduction R to ϕ_n to obtain $\langle T_k, s_n \rangle = R(\phi_n)$. As R is polynomial-time, say n^a, the time needed for this application is also bound by $(2^{O(k)})^a = 2^{O(k)}$. In a final step, we output $T_k(x)$. Overall, we can carry out the decision in time $2^{O(k)}$ proving that $L \in E$.

(ii) $L \notin \mathbf{P}_{/\mathbf{poly}}$: We assume to the contrary that $L \in \mathbf{P}_{/\mathbf{poly}}$. Remember that the reduction R is natural. In particular, this means for a mapping $R(\phi) = \langle T_k, s_n \rangle$ where ϕ is any Boolean formula of size n, that the parameter s_n depends only on a polynomial in $k = \Theta(\log n)$; say it is upper bounded by $s_n \leq k^b = \log^b n$ for a $b \in \mathbb{N}$.[3] Now, this bound yields a simple strategy to decide $\phi \in \mathsf{SAT}$. As Lemma 8 implies that there are at most $n^{\mathrm{polylog}(n)}$ circuits of size s_n, we can simply enumerate these and check for each circuit whether it represents T_k to decide $\langle T_k, s_n \rangle \in \mathsf{MCSP}$. 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 membership in quasi-polynomial time. This, in turn, allows us to decide $\phi \in \mathsf{SAT}$ by applying R and carrying out the above enumeration, implying that $\mathsf{SAT} \in \mathbf{QP}$. We have, however, assumed that $\mathbf{NP} \not\subseteq \mathbf{QP}$, which is a contradiction.

□

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 10 ([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 11 ([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 12. *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* $\mathbf{NP} \subseteq \mathbf{SUBEXP}$*.*

Taking everything together, we obtain a nice corollary as follows.

Corollary 13. *If* MCSP *is* **NP***-hard under a natural reduction from* SAT*, then* $\mathbf{BPP} \subsetneq \mathbf{E}$

Proof. Idea: diagonalize against **SUBEXP** with a Turing Machine M in E and M should mess up when the input length is large enough. □

3.2 MCSP and P

[We plan to discuss some implication for hard functions in uniform complexity class (2.4) and Zero-sided and One-sided error (2.5)]

[3] [We are also not exactly clear about why this is the case.]

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

[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.

[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.