Review on Circuit Minimization Problem by Kabanets-Cai

Date: November 18, 2020

(First Draft)

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. Formally speaking, given a Boolean function $f_n : \{0,1\}^n \to \{0,1\}$, an instance of MCSP is a tuple $\langle T_n, s_n \rangle$ where T_n is the string of length 2^n representing the truth table of f_n and f_n is computable by a circuit of size at most s_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?

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?

In their paper "Circuit Minimization Problem," Valentine Kabanets and Jin-Yi Cai addressed the difficulty of showing MCSP to be **NP**-hard. Namely, 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 are questionable). Clearly, such proof would imply $\mathbf{P} \neq \mathbf{NP}$ which goes beyond the currently known techniques.

The rest of the review: In Section 2, we will provide some definitions required to understand the main results and theorems of the paper. 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.

Note that for functions $f, g\colon \mathbb{N}_+ \to \mathbb{N}_+$ holds $\exp(f(n)) \in o(\exp(g(n)))$ if and only if $g(n) - f(n) \to \infty$ for $n \to \infty$. This is easily seen by looking at the limit definition of Landau symbols, i.e.

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

2 Some definitions:

1. The class Sub-Exponential: $\mathbf{SUBEXP} = \bigcap_{\epsilon>0} \mathbf{DTIME}(2^{n^\epsilon})$
2. The class **QP** of languages decided by a TM in *quasi-polynomial* time defined by

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

   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 $\text{polylog}(n) \in \Omega(1)$. Also, **SUBEXP** contains **QP** since for every $\varepsilon > 0$ and $c > 1$ holds that $\exp(\log^c n) \in o(\exp(n^\varepsilon))$.
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 $\mathbf{QP} \subseteq \mathbf{E}$.
4. **Natural 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.

   Example: $\mathsf{SAT} \leq_p \mathsf{3SAT}$ is natural.

   Question(s): What is output parameter s_n? Example?

Lemma 1. $\mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP}$.

Proof. Let M be a TM deciding a language $L \in \mathbf{QP}^{\mathbf{QP}}$.

Then, M runs in $O(n^{\log^c n})$ time with oracle access to a $\mathbf{QP}$ machine.

Suppose $f(n) = n^{\log^c n}$ and $g(n) = n^{\log^k n}$ are the run-time upper-bound functions of the "main body" of M and the oracle machine respectively. This implies that the total running time of M with some oracle access during the execution is bounded by $f(g(n))$ since $\mathbf{QP}$ is a deterministic time related class.

Let $t = \log^k n$, then $f(g(n)) = (n^{\log^k n})^{\log^c(n^{\log^k n})} = (n^t)^{(\log(n^t))^c} = (n^t)^{t^c \log^c n} = n^{t^{c+1} \log^c n}$

Substitute $\log^k n$ to t, we have $f(g(n)) = n^{\log^{k(c+1)} n \log^c n} = n^{\log^{k(c+1)+c} n}$ which is quasi-polynomial. In other words, M runs in quasi-polynomial time which implies that $L \in \mathbf{QP}$.

Hence, $\mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP}$

□

Lemma 2. *If* $\mathbf{NP} \subseteq \mathbf{QP}$ *then* $\mathbf{PH} \subseteq \mathbf{QP}$.

Proof. From the assumption $\mathbf{NP} \subseteq \mathbf{QP}$, we have $\mathbf{PH} = \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}^{\cdots\mathbf{NP}}}} \subseteq \mathbf{QP}^{\mathbf{QP}^{\mathbf{QP}^{\cdots\mathbf{QP}}}}$ (how do I put k times in there?)

Also, from Lemma 1, we have $\mathbf{QP}^{\mathbf{QP}^{\mathbf{QP}^{\cdots\mathbf{QP}}}} \subseteq \mathbf{QP}^{\mathbf{QP}^{\mathbf{QP}^{\cdots\mathbf{QP}}}} \subseteq \cdots \subseteq \mathbf{QP}$ (here, it means that we first have a stack of k $\mathbf{QP}$'s then it collapses to the stack of $k-1$ and so on... thus, we finally collapses it to $\mathbf{QP}$)

Thus, $\mathbf{NP} \subseteq \mathbf{QP} \implies \mathbf{PH} \subseteq \mathbf{QP}$ □

Proof. (alternative proof) We first show that $\mathbf{NP} = \exists\mathbf{P} \subseteq \mathbf{QP}$ also implies that $\exists\mathbf{QP} \subseteq \mathbf{QP}$ by a padding argument.[1] To this end, assume that $L \in \exists\mathbf{QP}$ such that it is decided by a $\exp(\log^c n)$-time verifier M. We define our padded language as $L_{\text{pad}} := \{\, x1^{\exp(\log^c n)} \mid x \in L \,\}$. We obtain that $L_{\text{pad}} \in \exists\mathbf{P}$ since applying M to a padded element takes takes only linear time. It follows by our assumption that $L_{\text{pad}} \in \mathbf{QP}$; say L_{pad} is decided by an $\exp(\log^{c'} n)$-time TM M'. Now, the key idea lies in showing that this already implies $L \in \mathbf{QP}$: Let $x \in \{0,1\}^*$ be of length $n := |x|$ and its padded version $y := x1^{\exp(\log^c n)}$ of length $m = \exp(\log^c n)$. Then, the time needed by M' to decide y is

$$\exp(\log^{c'} m) = \exp(\log^{c'}(\exp(\log^c n))) = \exp(\log^{c \cdot c'} n)$$

which is again quasi-polynomial in n. Therefore, we can decide $x \in L$ by applying padding and running M' all in quasi-polynomial time.

The second step is to show that also $\forall\mathbf{QP} \subseteq \mathbf{QP}$. By definition, given a language $L \in \forall\mathbf{QP}$, there exists a polynomial p and a quasi-polynomial time verifier M such that

$$x \in L \iff \forall y \in \{0,1\}^{p(|x|)}\ M(x,y) = 1\,.$$

[1] The class $\exists\mathbf{QP}$ is defined to consist of all languages L such that there exists a quasi-polynomial TM M and a polynomial p with $x \in L \iff \exists y \in \{0,1\}^{p(|x|)}\ M(x,y) = 1$

However, the right hand side is equivalent to the negation of $(\exists y \in \{0,1\}^{p(|x|)}\ M(x,y) = 0)$, a statement which can itself be decided in quasi-polynomial time by the above argument. Since all quasi-polynomial time TMs are also deterministic, we can also negate the output efficiently. It follows that $L \in \mathbf{QP}$.

By an inductive argument, we see that $\mathbf{PH}$ collapses into a subset of $\mathbf{QP}$. $\square$

Lemma 3. *$\mathbf{QP}^{\mathbf{\Sigma}_k^p}$ 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. $\square$

Lemma 4. *There are at most $n^{\text{polylog}(n)}$ 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 $3 \cdot s^2$ choices to construct each gate and thus $(3s^2)^s$ choices to construct the whole circuit. Setting $s := \log^c n$ gives us

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

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

Theorem 5 (Theorem 15). *If* MCSP *is* $\mathbf{NP}$*-hard under a natural reduction from* SAT*, then*

1. $\mathbf{E}$ *contains a family of Boolean functions f_n not in $\mathbf{P}_{/\mathbf{poly}}$ (i.o.), and*

2. $\mathbf{E}$ *contains a family of Boolean functions f_n of circuit complexity $2^{\Omega(n)}$ (i.o.), unless* $\mathbf{NP} \subseteq \mathbf{SUBEXP}$

Proof. - Statement 1: consider two cases

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

  It is obvious to see that $\mathbf{NP} \subseteq \mathbf{QP} \implies \mathbf{PH} \subseteq \mathbf{QP}$ (think of this as a generalisation of $\mathbf{NP} \subseteq \mathbf{P} \implies \mathbf{PH} \subseteq \mathbf{P}$). Also, we make an observation that $\mathbf{QP}^{\Sigma_k^p}$, for some $k \in \mathbb{N}$, contains a language of superpolynomial circuit complexity. The proof of this claim is as follow: (help!!!!)

  Combining these two results, we get the following $\mathbf{NP} \subseteq \mathbf{QP} \implies \mathbf{QP}^{\mathbf{PH}} \subseteq \mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP} \subset \mathbf{E}$ contains a family of function not in $\mathbf{P}_{/\mathbf{poly}}$.

  Applying Lemma 1 and Lemma 2 to this assumption yields that $\mathbf{QP}^{\mathbf{PH}} \subseteq \mathbf{QP}^{\mathbf{QP}} \subseteq \mathbf{QP} \subseteq \mathbf{E}$. Lemma 3 shows that $\mathbf{E}$ also contains a language of superpolynomial circuit complexity.

  Hence, $\mathbf{E} \nsubseteq \mathbf{P}_{/\mathbf{poly}}$

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

  A given natural reduction R from SAT to MCSP maps every family of boolean formulas of size n to truth tables of Boolean functions on $k = \theta(\log n)$ variables (is the symbol θ significant? more elaboration on $\theta(\log n)$, like why is that?) and a parameter s_n.

  Since the reduction is natural, s_n is some function of n only and thus it could be some fixed polynomial $\log^c n$. By Lemma 4, there are at most $n^{\text{polylog(n)}}$ many different circuits on k inputs given $\log^c n$ gates. Thus, all such instances of MCSP where $s_n = \log^c n$, for some constant c, are solvable in quasi-polynomial time. This implies that $\mathsf{SAT} \in \mathbf{QP}$ which is a contradiction to our assumption since SAT is $\mathbf{NP}$-complete. (should we say something about the relationship between k and $\log n$? My understanding is that what we have shown here implies that the circuit size $s_n \neq \log^c n$ which is related to something like k^c, i.e. can't get a circuit of poly(k) size which leads to the thing below)

  Therefore, we can obtain the desired k-variable functions not in $\mathbf{P}_{/\mathbf{poly}}$ by applying the reduction R to any trivial family of unsatisfiable formulas. Clearly, this family of hard functions is computable in time $2^{O(k)}$. (maybe more elaboration on this too...).

- Statement 2: the proof is similar (you want to do it too?)

□