Review on Circuit Minimization Problem by Kabanets-Cai

Date: November 20, 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 $f_n : \{0,1\}^n \rightarrow \{0,1\}$, 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 **P** $\neq$ **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

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

   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 $\exp(\log^c n) \in o(\exp(n^\varepsilon))$.
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 dediciing a language $L \in \mathbf{QP}^{\mathbf{QP}}$ in quasi-polynomial time, say $\exp(\log^c n)$. We show that carrying out the oracle computation instead of calling the oracle does still guarantee a quasi-polynomial running time. To this end, let M' be the TM deciding the $\mathbf{QP}$-oracle, say in time $\exp(\log^{c'} m)$. 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. $\square$

Corollary 2. *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 1. $\square$

We can also establish this resulting by using a padding argument as done in the following.

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.

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

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

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

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 $\mathbf{QP}$. □

Lemma 3. $\mathbf{QP}^{\boldsymbol{\Sigma}_k^p}$ *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 $\boldsymbol{\Sigma}_4^p$-oracle as in Problem (1c) of Homework 7. Finally, we define our language L of superpolynomial circuit complexity as the output of a $\mathbf{QP}^{\boldsymbol{\Sigma}_4^p}$-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 4. *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 $s := \log^c n$ gives us

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

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

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: $\mathbf{NP} \subseteq \mathbf{QP}$

  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, $\mathbf{E} \not\subseteq \mathbf{P}_{/\mathbf{poly}}$

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

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

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

  Now based on this, we want to apply R to define our language. To this end, let first $\langle T_k, s_n \rangle := R(\phi_n)$ 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 $2^k = \Theta(\text{poly}(n))$ and thus $k = \log(\Theta(\text{poly}(n))) = \Theta(\log n)$. Now, define

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

  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 $n := 2^k$. 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 $s_n \le k^b = \log^b n$ for a $b \in \mathbb{N}$.[2] Now, this bound yields a simple strategy to decide $\phi \in \mathsf{SAT}$. As Lemma 4 implies that there are at most $n^{\text{polylog}(n)}$ 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.

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

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

□

2.1 Natural and Unnatural Reductions

Consider the following reduction from Clique[3] to SubIso:[4]

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

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 turn this reduction into a natural reduction. For example, we can define

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

where $G \cup H$ denotes the union of two graphs G and H. Now, the output size becomes $|f'(G,n)| = \Theta(|G|)$ in both cases.

[3]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.

[4]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.