Electronic Colloquium on Computational Complexity, Report No. 45 (1999)

ECCC

Circuit Minimization Problem

Valentine Kabanets* Jin-Yi Cai†

November 15, 1999

Abstract

We study the complexity of the circuit minimization problem: given the truth table of a Boolean function f and a parameter s, decide whether f can be realized by a Boolean circuit of size at most s. We argue why this problem is unlikely to be in P (or even in P/poly) by giving a number of surprising consequences of such an assumption. We also argue that proving this problem to be NP-complete (if it is indeed true) would imply proving strong circuit lower bounds for the class E, which appears beyond the currently known techniques.

Keywords: hard Boolean functions, derandomization, natural properties, NP-completeness.

1 Introduction

An n-variable Boolean function $f_n : \{0,1\}^n \to \{0,1\}$ can be given by either its truth table of size 2^n, or a Boolean circuit whose size may be significantly smaller than 2^n. It is well known that most Boolean functions on n variables have circuit complexity at least $2^n/n$ [Sha49], but so far no family of sufficiently hard functions has been proven to exist in any relatively small uniform complexity class. As far as we know, every language in $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ may be decided by a family of linear-size circuits.

So the state of affairs is this: extremely hard Boolean functions abound, but we cannot exhibit any particular example of a hard function that is computable within reasonable time bounds. Can we at least recognize a hard function when we see one? In other words, is there an efficient algorithm that solves the following problem?

Minimum Circuit Size Problem (MCSP)
Instance: A Boolean function $f_n : \{0,1\}^n \to \{0,1\}$ given by its truth table (of length 2^n) and a number $s_n \in \mathbb{N}$ (in binary).
Question: Is f_n computable by a Boolean circuit of size at most s_n?

We would like to point out that the above problem was considered in the past; in fact, it was studied in the USSR already in the 50's (see, e.g., [Yab59b, Yab59a]). Actually, Yablonski [Yab59b, Yab59a] believed that he had shown the impossibility of eliminating the "brute-force search" when solving a related problem: "Compute a family $\{f_n\}_{n\geqslant 0}$ of n-variable Boolean functions where each f_n has the maximum circuit complexity among all n-variable Boolean functions". However, his proof had to do with a restricted class of algorithms, and cannot be interpreted to mean that such a family of Boolean functions is impossible to construct in time polynomial in the sizes of

*Department of Computer Science, University of Toronto, Toronto, Canada. Email: kabanets@cs.toronto.edu.
†Department of Computer Science, State University of New York at Buffalo, Buffalo, NY 14260. Email: cai@cs.buffalo.edu.

ISSN 1433-8092

their truth tables (see [Tra84] for a more detailed discussion). It is not hard to see that if such a family of n-variable Boolean functions *cannot* be constructed in time $\mathrm{poly}(2^n)$, then P $\neq$ NP. So, if Yablonski succeeded in proving his intended claim, he would have found a negative solution to the P vs. NP problem even before that problem was formally stated in [Coo71].

Returning to our problem, we observe that MCSP is obviously in NP (just note that the input size is $O(2^n)$, and so we have enough time to check that a guessed circuit of appropriate size computes a given function of n variables). We would like to argue that MCSP is intractable. The most convincing argument would be a proof that MCSP is not in P. But this would prove a separation of NP from P, which appears to be well beyond the currently known techniques.

The next best argument would be a proof that MCSP is NP-complete. However, as we argue below, any natural proof of this would imply non-trivial circuit lower bounds for languages in E, and hence is unlikely to be found soon. Here, by "natural", we mean a proof that gives a Karp reduction from, say, SAT to MCSP such that the size of the output depends on the size of the input only, and these sizes are polynomially related. We note that all the NP-completeness proofs that we are aware of are natural in this sense.

Unable to reduce SAT to MCSP, we nonetheless show that the assumption that MCSP is in P does have a number of surprising consequences. In particular, it would imply the existence of an average-case algorithm for factoring integers which is faster than any known algorithm, the existence in E^{NP} of a family of Boolean functions of maximum circuit complexity, the inclusion BPP $\subseteq$ ZPP, the equivalence between E containing a language of circuit complexity at least $2^{\epsilon n}$, for some $\epsilon > 0$, and E containing a language of circuit complexity at least $\frac{2^n}{n}(1 + (1-\gamma)\frac{\log n}{n})$, for any $\gamma < 1$, and the equivalence of certain local and global complexity assumptions sufficient for derandomization.

The rest of the paper. In Section 2, we give some consequences of the assumption that MCSP is easy. Section 3 contains an argument why it seems unlikely that one can prove MCSP to be NP-complete without proving strong circuit lower bounds. We give concluding remarks and present some open problems in Section 4.

2 MCSP and P

2.1 Natural Properties

Recall that 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

$$|\mathbf{Pr}_{x\in\{0,1\}^k}[C(G_k(x)) = 1] - \mathbf{Pr}_{y\in\{0,1\}^{2k}}[C(y) = 1]| \geqslant 1/s.$$

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

Let Γ be a complexity class. Following Razborov and Rudich [RR97], we call a combinatorial property $\{C_n\}_{n\geqslant 0}$ of n-variable Boolean functions f_n Γ-*natural* with density δ_n if each C_n contains a subset C_n^* such that

1. the predicate $f_n \stackrel{?}{\in} C_n^*$ is computable in Γ, where f_n is given by its truth table, and
2. C_n^* contains at least δ_n fraction of all n-variable Boolean functions.

Informally, a natural property is easy to check, and it holds for a significant fraction of all Boolean functions. One standard setting of the parameters in the above definition is $\Gamma = \mathrm{P}$ and $\delta_n = 2^{-O(n)}$.

For a complexity class Λ, a combinatorial property $\{C_n\}_{n\geqslant 0}$ is *useful against* Λ if every family of Boolean functions $\{f_n\}_{n\geqslant 0}$ such that $f_n \in C_n$ i.o. is not in Λ.

The main result in [RR97] can be stated as follows.

Theorem 1 (Razborov-Rudich) *If a* P/poly*-natural property useful against* P/poly *exists, then there is no strong pseudorandom generator in* P/poly*.*

As an immediate corollary of Theorem 1, we get the following.

Theorem 2 *If MCSP is in* P/poly*, then there is no strong pseudorandom generator in* P/poly*.*

Proof: It is easy to see that if MCSP is in P/poly, then we get a P/poly-natural property useful against P/poly, and hence the claim follows by Theorem 1. ■

An example of a generator which is believed to be strong pseudorandom is the generator based on factoring Blum integers (recall that a Blum integer is a product of two primes, each congruent to 3 mod 4). Breaking this generator implies being able to factor Blum integers well on the average. Theorem 2 shows that the existence of an efficient algorithm for MCSP yields an average-case algorithm for factoring that beats any known factoring algorithm; the best known (worst-case) deterministic factoring algorithm has the running time approximately $2^{n/4}$ on n-bit integers [Pol74, Str76], while the best probabilistic algorithm runs in time approximately $2^{\sqrt{n}}$ [LP92].

Corollary 3 *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.*

The widely believed hardness of factoring may be taken as the most compelling piece of evidence that MCSP is hard. However, we give more examples below of some unlikely consequences to the assumption that MCSP is easy.

2.2 Hardness Amplification

Suppose that one has an n-variable Boolean function of high circuit complexity, say, $2^{\epsilon n}$ for some $\epsilon > 0$. Given the truth table of such a function, can one efficiently (i.e., in time polynomial in 2^n) produce the truth table of a *harder* Boolean function in $m \in \Omega(n)$ variables, e.g., of circuit complexity greater than $2^m/m$?

The affirmative answer to this question would be surprising. However, we can show that such an algorithm exists, under the assumption that MCSP is in P.

Theorem 4 *Assume MCSP is in* P*. Then there exists a polynomial-time algorithm that, given the truth table of an n-variable Boolean function of circuit complexity at least $2^{\epsilon n}$, for some $\epsilon > 0$, outputs $2^{\Omega(n)}$ Boolean functions on $m \in \Omega(n)$ variables each, such that all of the output functions have circuit complexity greater than $\frac{2^m}{m}(1 + (1-\gamma)\frac{\log m}{m})$, for any $\gamma > 0$.*

Our proof will use the following result that can be readily extracted from [IW97].

Theorem 5 (Impagliazzo-Wigderson) *For every $\epsilon > 0$, there exist $c, d \in \mathbb{N}$ such that the truth table of a Boolean function $f_{cn} : \{0,1\}^{cn} \to \{0,1\}$ of circuit complexity $2^{\epsilon cn}$ can be transformed, in time $2^{O(n)}$, into a pseudorandom generator $G_{dn} : \{0,1\}^{dn} \to \{0,1\}^{2^n}$ running in time $2^{O(n)}$ that has hardness $H(G_{dn}) > 2^n$.*

We also need a lower bound on the circuit complexity of most Boolean functions from [Lup59, Lup63].

Theorem 6 (Lupanov) *For any $\epsilon > 0$ and sufficiently large n, almost all n-variable Boolean functions require Boolean circuits of size greater than $\frac{2^n}{n}\left(1 + (1-\epsilon)\frac{\log n}{n}\right)$.*

Proof of Theorem 4. Let $\gamma > 0$ be arbitrary, and let $s(n) = \frac{2^n}{n}(1 + (1-\gamma)\frac{\log n}{n})$. Assuming that MCSP is in P, we get a polynomial-size circuit family that accepts only the truth tables of n-variable Boolean functions of circuit complexity greater than $s(n)$, by fixing the parameter $s_n = s(n)$. Clearly, the acceptance probability of our circuits will be very close to one, by Theorem 6.

Since the size of these circuits is bounded by some fixed polynomial in the input size, the Impagliazzo-Wigderson generator G from Theorem 5 will fool them. That is, almost all 2^n-bit strings output by G will be the truth tables of n-variable Boolean functions of circuit complexity greater than $s(n)$. We can tell which functions are hard by running an algorithm for MCSP, which is assumed to be in P, and hence we can output hard functions only. ■

As a consequence of the theorem above, we get, under the assumption that MCSP is easy, that E contains a relatively hard Boolean function iff it contains a very hard function. More precisely, we have the following.

Corollary 7 *Assume MCSP is in* P. *Then* 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 $\epsilon > 0$, iff* E *contains a family of Boolean functions $g_n : \{0,1\}^n \to \{0,1\}$ of circuit complexity greater than $\frac{2^n}{n}(1 + (1-\gamma)\frac{\log n}{n})$, for any $\gamma > 0$.*

Proof: $\Leftarrow$. This direction is obvious.

$\Rightarrow$. As in the proof of Theorem 4, let $\gamma > 0$ be arbitrary and let $s(n) = \frac{2^n}{n}(1 + (1-\gamma)\frac{\log n}{n})$. Assuming that MCSP is in P, we obtain that most of the outputs of the Impagliazzo-Wigderson generator G from Theorem 5 are the truth tables of n-variable Boolean functions of circuit complexity greater that $s(n)$. Choose the lexicographically first string of length dn which is mapped by G_{dn} into the truth table of such a hard function. Call this hard function $g_n : \{0,1\}^n \to \{0,1\}$.

If 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 $\epsilon > 0$, then we can compute the truth table of f_{cn} on cn inputs in time $2^{O(n)}$. Thereupon, we can compute the truth table of the hard function $g_n : \{0,1\}^n \to \{0,1\}$ as described in the previous paragraph, using time $2^{O(n)}$. Thus, given an n-bit string x as input, we can compute $g_n(x)$ in time $2^{O(n)}$, which means that E contains a family of n-variable Boolean functions of circuit complexity greater than $s(n)$. ■

2.3 Natural Properties Revisited

Here we observe that the results of the previous subsection are just particular cases of a more general phenomenon. Recall that a property of n-variable Boolean functions is called *natural* if it holds for sufficiently many functions, and if it can be decided efficiently in the size of an input truth table. Let $N = 2^n$. Below, by a natural property, we will mean a P-natural property $\{C_n\}_{n\geqslant 0}$ with density $1/N$.

An obvious question one may ask about a given natural property $\{C_n\}_{n\geqslant 0}$ is this: What is the uniform complexity of computing a particular family of n-variable Boolean functions satisfying property $\{C_n\}_{n\geqslant 0}$? At present, the best answer to this question is the trivial one: we need time 2^{2^n}.

On the other hand, suppose that we are given access to an arbitrary fixed family of sufficiently hard n-variable Boolean functions. Then, for every natural property $\{C_n\}_{n\geqslant 0}$, we can find, in time $2^{O(n)}$, the truth table of a particular n-variable Boolean function satisfying C_n. In other words, any *single* hard family of Boolean functions contains enough information for an efficient search of witnesses for *every* given natural property. Formally, we have the following.

Theorem 8 *Let $f = \{f_n\}_{n\geqslant 0}$ be an arbitrary fixed family of Boolean functions of circuit complexity at least $2^{\epsilon n}$, for some $\epsilon > 0$. Then, for every natural property $C = \{C_n\}_{n\geqslant 0}$, the class E^f contains a family of Boolean functions satisfying C.*

Proof: As in the proofs of Theorem 4 and Corollary 7, we use Theorem 5 to transform the truth table of a hard Boolean function f_{cn} into a pseudorandom generator G_{dn} that fools the circuit deciding C_N, for appropriate $c, d \in \mathbb{N}$. The hardness of G_{dn} implies that its range will contain an n-variable Boolean function satisfying C_N. We can fix one such function g_n by choosing the lexicographically first input α such that $G_{dn}(\alpha)$ satisfies C_N. ■

It follows from the proof of Theorem 8 that, on inputs of size n, the size of oracle queries is $O(n)$. Hence, we get the following.

Corollary 9 *Suppose* 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 $\epsilon > 0$. Then, for every natural property $C = \{C_n\}_{n\geqslant 0}$, the class* E *contains a family of Boolean functions satisfying C.*

2.4 Hard Functions in Uniform Complexity Classes

It is well-known that $\mathrm{E}^{\Sigma_2^p}$ contains a family of Boolean functions of maximum circuit complexity. If MCSP were easy, we would get the following improvement to this result.

Theorem 10 *If MCSP is in* P, *then* E^{NP} *contains a family of Boolean functions of maximum circuit complexity.*

Proof: We essentially follow the proof of a similar result from [MVW99, Lemma 2]. First, for a given n, we find the maximum circuit complexity over all n-variable Boolean functions by asking a series of questions of the form: "Is there a string $t_1 \dots t_{2^n}$ representing the truth table of a Boolean function that requires circuit size at least s?", for $s = 2^n, 2^n - 1, \dots$; the first value of $s = s^*$ that gets the positive answer will be the required maximum circuit size. Note that, under our assumption that MCSP is in P, these questions will be NP-questions.

Now we find the lexicographically first truth table $T = t_1 \dots t_{2^n}$ of a Boolean function with circuit complexity s^*, by starting with the empty truth table $T = \epsilon$, and appending 0 to T if the answer to the following NP-question is positive: "Can the string $T0$ be extended to a truth table of a Boolean function with circuit complexity at least s^*?", and appending 1 otherwise. Continuing in this way for 2^n steps, we completely specify the truth table of a Boolean function with maximum circuit complexity. Clearly, the overall running time of the described algorithm is $2^{O(n)}$, given access to an NP-oracle. ■

It was shown in [Kan82] that, for every $k \in \mathbb{N}$, $\Sigma_2^p \cap \Pi_2^p$ contains a family of Boolean functions f_n of circuit complexity greater than n^k; in [KW98], $\Sigma_2^p \cap \Pi_2^p$ was replaced by the class $\mathrm{ZPP}^{\mathrm{NP}}$. By a padding argument, we easily get from Theorem 10 the following.

Corollary 11 *If MCSP is in* P, *then, for every $k \in \mathbb{N}$, there exists a language L_k in* P^{NP} *that requires circuit size at least n^k.*

As we noted above, the best unconditional result along the lines of Corollary 11 states that languages of circuit complexity at least n^k exist in $\mathrm{ZPP}^{\mathrm{NP}}$ [KW98]. This is about the best possible one can get using relativizable techniques since there are oracles with respect to which all of P^{NP} can be computed by linear-size circuits [Wil85]. In particular, it follows that MCSP is not in P, with respect to the same oracle.

2.5 Two-Sided Error vs. Zero Error

It is well-known that $\mathrm{BPP} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$ [ZH86] (see also [Sip83, Lau83, NW94, GZ97]). It is also obvious from the definitions that $\mathrm{ZPP} \subseteq \mathrm{RP} \subseteq \mathrm{BPP}$. On the other hand, it is not known whether $\mathrm{BPP} \subseteq \mathrm{RP}$ or $\mathrm{BPP} \subseteq \mathrm{NP}$.

We observe that if MCSP is easy, then any probabilistic algorithm with a two-sided error can be replaced by an equivalent probabilistic algorithm with no error. We prove the following theorem first.

Theorem 12 $\mathrm{BPP} \subseteq \mathrm{ZPP}^{\mathrm{MCSP}}$.

Proof: Impagliazzo and Wigderson [IW97] show how to use a hard Boolean function on $O(\log n)$ variables to derandomize BPP. We use their result to get the following algorithm in $\mathrm{ZPP}^{\mathrm{MCSP}}$ for every given BPP algorithm. (A similar argument was given in [NW94] to obtain another proof that $\mathrm{BPP} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$.)

First, our algorithm guesses a truth table of a Boolean function on $O(\log n)$ variables of circuit complexity $n^{\Omega(1)}$. This step is in $\mathrm{ZPP}^{\mathrm{MCSP}}$ since most Boolean functions are sufficiently hard and we reject any easy function with the help of the MCSP oracle. Having found a hard Boolean function, we use Theorem 5 to obtain an efficient deterministic simulation of the given BPP algorithm on any n-bit input. Since the second step of our algorithm is in P, the claim follows. ■

We should point out that the result of Theorem 12 would follow trivially from the well-known inclusion $\mathrm{BPP} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$ if one could show that MCSP is NP-hard. However, as we argue below, the proof that MCSP is NP-hard (if it is indeed true) is beyond the current state of the art of theoretical computer science.

Now we can state an easy corollary to Theorem 12.

Corollary 13 *If MCSP is in* P*, then* $\mathrm{BPP} \subseteq \mathrm{ZPP}$.

2.6 Global vs. Local Conditions Sufficient for Derandomization

In the study of the P vs. BPP question, several conditions were formulated that are sufficient for derandomizing BPP. They can be split into two categories: *global* conditions and *local* conditions. Roughly speaking, a global condition assumes the existence of an efficient algorithm for generating a certain combinatorial object (usually, a set of binary strings) which contains some information "useful" with respect to *all* small circuits. On the other hand, a local condition assumes the existence of an efficient algorithm that, given a small circuit as input, produces some information "useful" with respect to this *particular* circuit.

Intutively, local conditions seem much weaker than global ones. Below, we give the standard examples of both global and local conditions, and show that the assumption that MCSP is easy leads to a surprising conclusion: the two kinds of conditions are equivalent.

The global conditions usually have to do with the existence of pseudorandom generators that can "fool" *every* sufficently small Boolean circuit. One standard meaning of the term *fooling* is

that every small circuit C on n inputs must accept the fraction of outputs of the generator that is sufficiently close to $\mathbf{Pr}_{x\in\{0,1\}^n}[C(x)=1]$, the actual acceptance probability of C. The other one is that C must accept at least one of the outputs of the generator, provided that the acceptance probability of C is sufficiently high (say, at least $1/2$).

Generators of the first kind are usually called *discrepancy set generators*, and those of the second kind *hitting set generators*; a discrepancy set generator is also a hitting set generator, but the converse need not be true. Let us call a generator *efficient* if it outputs n bits on an input of $O(\log n)$ bits, runs in time $\mathrm{poly}(n)$, and fools every circuit of size n on n inputs.

It should be obvious that the existence of efficient discrepancy set generators implies BPP = P. Remarkably, Andreev et al. [ACR98] proved that the same conclusion can be achieved under the seemingly weaker assumption that efficient hitting set generators exist (see also [ACRT97, BF99, GW99] for simpler proofs). It turns out that these two assumptions are equivalent to the assumption that E contains a Boolean function of high circuit complexity. Namely, given an efficient hitting set generator, one can construct a Boolean function computable in E that has very high circuit complexity; the idea of such a construction was implicit already in [NW94], and is stated explicitly in [ISW99, Theorem 9]. Conversely, an efficient discrepancy set generator can be obtained from a hard Boolean function, using the results in [IW97] (recall Theorem 5 above).

An example of a local condition is the existence of an efficient *circuit approximator*, the algorithm that sufficiently closely approximates the acceptance probability of a given circuit. This condition is obviously sufficient for derandomizing BPP, and it is trivially implied by the global conditions stated above. In fact, it can be viewed as a local version of the condition that efficient discrepancy set generators exist.

A local version of the condition that efficient hitting set generators exist is the existence of an efficient algorithm for solving the following promise problem.

Promise SAT
Given: A Boolean circuit C on n inputs.
Output: 0 if $\mathbf{Pr}_{x\in\{0,1\}^n}[C(x)=1]=0$, and 1 if $\mathbf{Pr}_{x\in\{0,1\}^n}[C(x)=1]\geqslant 1/2$.

As in the case of their global counterparts, the two local conditions stated above are also equivalent; the proof can be extracted from [ACRT97] (see also [BF99]).

Now we show that, under the assumption that MSCP is easy, all of the global and local conditions stated above are equivalent. That is, if MCSP is in P, the following conditions are equivalent:

1. 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 $\epsilon>0$,
2. there is an efficient discrepancy set generator,
3. there is an efficient hitting set generator,
4. there is a polynomial-time algorithm solving Promise SAT, and
5. there is a polynomial-time circuit approximator.

As we mentioned above, it is known that, without any assumptions, (1) $\Leftrightarrow$ (2) $\Leftrightarrow$ (3) and (4) $\Leftrightarrow$ (5). Hence, it suffices to prove the following theorem.

Theorem 14 *If MCSP is in* P*, then the following conditions are equivalent:*

1. 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* $\epsilon>0$*, and*

2. there is a polynomial-time circuit approximator.

Proof [Sketch]: (1) $\Rightarrow$ (2). Given a hard function in E, we get an efficient discrepancy set generator as in [IW97]. Obviously, such a generator can be used as a circuit approximator.

(2) $\Rightarrow$ (1). Given an efficient circuit approximator, we can construct an efficient algorithm that, when given a circuit with acceptance probability of at least 1/2, finds an input accepted by this circuit. The idea is to look for an accepted input by fixing one bit at a time, using the circuit approximator to guide the search: fix the bit value so as to get a greater estimate for the acceptance probability of the resulting circuit. Note that since this witness-finding algorithm is deterministic, its output is uniquely determined by the input circuit.

Under our assumption, there is a polynomial-time uniform family $\{C_m\}_{m\geqslant 0}$ of Boolean circuits deciding MCSP. By fixing the parameter s_n in MCSP to be $2^{\epsilon n}$, we obtain the family of circuits C'_{2^n} accepting only the truth tables of n-variable Boolean functions of circuit complexity greater than $2^{\epsilon n}$. Clearly, each circuit C'_m accepts more than a half of its inputs (see Theorem 6 above).

Now we can apply our witness-finding algorithm to the family of circuits C'_{2^n} and find a unique family of n-variable Boolean functions of high circuit complexity. The total running time for constructing a particular n-variable function from this family will be $\mathrm{poly}(2^n)$. ■

3 MCSP and NP-completeness

3.1 Implications for Circuit Complexity

Even though we have given some evidence that MCSP is probably not in P, we cannot show that it is NP-hard. The difficulty is that any "natural" proof of the NP-hardness of a problem A yields a way to construct hard instances of A. In the case of MCSP, such a proof would give rise to an explicit Boolean function in E with superpolynomial circuit complexity.

More formally, for two problems A and B and a Karp reduction R from A to B, we call the reduction R *natural* if, for any instance I of problem A, the size of $R(I)$ (as well as the possible numerical parameters of $R(I)$) depends only on the size of I, and the sizes of I and $R(I)$ are polynomially related. For example, the text-book reductions from SAT to 3-SAT, and from 3-SAT to Vertex Cover [GJ79] are natural in the above sense. In fact, all "natural" NP-complete problems that we are aware of are complete under natural reductions; this includes the Minimum Size DNF Problem, for which a natural reduction from SAT is given in [Mas79].

In the next theorem, we use the notation $\mathrm{SUBEXP} = \cap_{\epsilon>0}\mathrm{DTIME}(2^{n^\epsilon})$.

Theorem 15 *If MCSP is* NP*-hard under a natural reduction from SAT, then*

1. E *contains a family of Boolean functions* f_n *not in* P/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.

Proof: Statement 1. First, we observe that if NP $\subseteq$ QP, where $\mathrm{QP} = \mathrm{DTIME}(n^{\mathrm{polylog}(n)})$, then PH $\subseteq$ QP. Also, it can be easily shown that $\mathrm{QP}^{\Sigma_k^p}$, for some $k \in \mathbb{N}$, contains a language of superpolynomial circuit complexity. Combining these two results, we get that NP $\subseteq$ QP implies that $\mathrm{QP}^{\mathrm{PH}} \subseteq \mathrm{QP}^{\mathrm{QP}} \subseteq \mathrm{QP} \subset$ E contains a family of functions not in P/poly.

Now suppose that NP $\not\subseteq$ QP. A given natural reduction R from SAT to MCSP maps every family of formulas of size n to the truth tables of Boolean functions on $k = \theta(\log n)$ variables and

a parameter s_n. Since the reduction is natural, s_n is a function of n only. If s_n could be upper-bounded by some fixed polynomial $(\log n)^c$, then all such instances of MCSP would be solvable in deterministic time $n^{\text{polylog}(n)}$ (since there are at most that many different circuits on k inputs with $(\log n)^c$ gates). This would imply that SAT is in QP.

Thus, under the assumption that NP $\not\subseteq$ QP, we can obtain the desired family of k-variable functions not in P/poly by applying reduction R to any trivial family of unsatisfiable formulas. Clearly, this family of hard functions would be computable in time $2^{O(k)}$.

Statement 2 is similar. If s_n could be upper-bounded by $2^{\epsilon \log n}$ for every $\epsilon > 0$, then SAT would be solvable in deterministic time 2^{n^δ} for every $\delta > 0$. Assuming that NP $\not\subseteq$ SUBEXP, we get that any trivial family of unsatisfiable formulas will be transformed by R to a family of Boolean functions on $k = \theta(\log n)$ variables of circuit complexity $2^{\Omega(k)}$ for infinitely many k. ■

3.2 Implications for BPP

We need the following two theorems on hardness-randomness trade-offs from [IW97] and [BFNW93], respectively.

Theorem 16 (Impagliazzo-Wigderson) *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* BPP = P *(i.o.).*

Theorem 17 (Babai-Fortnow-Nisan-Wigderson) *If the class* EXP *contains a family of Boolean functions of superpolynomial circuit complexity (i.o.), then* BPP $\subseteq$ SUBEXP *(i.o.).*

Using Theorems 16 and 17 above, we now easily obtain the following corollary from Theorem 15.

Theorem 18 *If MCSP is* NP*-hard under a natural reduction from SAT, then*

1. BPP $\subseteq$ SUBEXP *i.o., and*
2. BPP = P *i.o., unless* NP $\subseteq$ SUBEXP.

In other words, assuming that MCSP is NP-complete under a natural reduction from SAT, we get the following: if NP is hard i.o. (a.e.), then BPP is easy i.o. (a.e.). We should contrast this with the fact implied by the inclusion BPP $\subseteq \Sigma_2^p$ [Sip83, Lau83]: if NP is easy a.e., then BPP is also easy a.e.

Corollary 19 *If MCSP is* NP*-hard under a natural reduction from SAT, then* BPP $\subsetneq$ E.

Proof: If MCSP is NP-hard under a natural reduction from SAT, then BPP is in SUBEXP for infinitely many input lengths, by Theorem 18. Since we can diagonalize against SUBEXP with a Turing machine in E so that this diagonalizing machine differs from every machine in SUBEXP on at least one input for all sufficiently large input lengths, the claim follows. ■

4 Concluding Remarks and Open Problems

In Section 3, we have argued that proving the NP-hardness of MCSP would be difficult because of the lack of any superlinear lower bounds for a language in E. However, we have very strong lower bounds for some restricted models of computation, e.g., constant-depth circuits and monotone circuits. Is the Minimum Depth-d (Unbounded Fan-in) Circuit Size Problem NP-complete for every $d \geqslant 2$? At present, only the case of $d = 2$ is known [Mas79].

Unfortunately, one obstacle to proving the NP-completeness result for minimum circuit size of depth-d AC^0-circuits is the lack of strongly exponential lower bounds; the known lower bounds for AC^0 (e.g., for parity) are exponential in some root of n only. We do not have a proof that E contains a Boolean function with a strongly exponential lower bound for AC^0. On the other hand, the output of a natural reduction, when given an unsatisfiable formula, will need to produce a function in E with strongly exponential lower bound, unless NP $\subseteq$ SUBEXP.

There also appear to be no strongly exponential lower bounds for the case of monotone Boolean circuits; the lower bounds for CLIQUE and BMS (Broken Mosquito Screen) are exponential in some root of the input size only. So we have the same obstacle in proving the NP-completeness result for monotone circuits as we do for AC^0.

We point out two more open problems. Can Corollary 7 be improved to say that, under the assumption that MCSP is in P, the class E contains a language of circuit complexity at least $2^{\epsilon n}$, for some $\epsilon > 0$, iff E contains a language of *maximum* circuit complexity? Our proof used the fact that a significant fraction of n-variable Boolean functions have high circuit complexity, whereas there may be very few functions of maximum circuit complexity.

Another question is whether MCSP is self-reducible. Namely, is it possible to find a minimum-size circuit for a given Boolean function f in time polynomial in the size of the truth table of f, when given oracle access to the language of MCSP? If MCSP is NP-complete, then, obviously, the answer should be positive.

Acknowledgments. The first author wishes to thank Stephen Cook for many remarks and insightful comments on the results of this paper, Richard Lipton for a fruitful discussion at an early stage of the research described here, and Charles Rackoff for his helpful remarks. Also, many thanks to Stephen Cook and Dieter van Melkebeek for reading and commenting on an earlier version of this paper.

References

[ACR98] A.E. Andreev, A.E.F. Clementi, and J.D.P. Rolim. A new general derandomization method. *Journal of the Association for Computing Machinery*, 45(1):179–213, 1998. (preliminary version in ICALP'96).

[ACRT97] A.E. Andreev, A.E.F. Clementi, J.D.P. Rolim, and L. Trevisan. Weak random sources, hitting sets, and BPP simulations. In *Proceedings of the Thirty-Eighth Annual IEEE Symposium on Foundations of Computer Science*, pages 264–272, 1997.

[BF99] H. Buhrman and L. Fortnow. One-sided versus two-sided error in probabilistic computation. In C. Meinel and S. Tison, editors, *Proceedings of the Sixteenth Annual Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 100–109. Springer Verlag, 1999.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Complexity*, 3:307–318, 1993.

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[GW99] O. Goldreich and A. Wigderson. Improved derandomization of BPP using a hitting set generator. In D. Hochbaum, K. Jansen, J.D.P. Rolim, and A. Sinclair, editors, *Randomization, Approximation, and Combinatorial Optimization*, volume 1671 of *Lecture Notes in Computer Science*, pages 131–137. Springer Verlag, 1999. (RANDOM-APPROX'99).

[GZ97] O. Goldreich and D. Zuckerman. Another proof that BPP⊆PH (and more). *Electronic Colloquium on Computational Complexity*, TR97-045, 1997.

[ISW99] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In *Proceedings of the Fortieth Annual IEEE Symposium on Foundations of Computer Science*, pages 181–190, 1999.

[IW97] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[Kan82] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55:40–56, 1982.

[KW98] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. *SIAM Journal on Computing*, 28(1):311–324, 1998.

[Lau83] C. Lautemann. BPP and the polynomial time hierarchy. *Information Processing Letters*, 17:215–218, 1983.

[LP92] H.W. Lenstra Jr. and C. Pomerance. A rigorous time bound for factoring integers. *Journal of the American Mathematical Society*, 5(3):483–516, 1992.

[Lup59] O.B. Lupanov. A method of circuit synthesis. *Izvestiya VUZ, Radiofizika*, 1(1):120–140, 1959. (in Russian).

[Lup63] O.B. Lupanov. On the synthesis of certain classes of control systems. In *Problemy Kibernetiki 10*, pages 63–97. Fizmatgiz, Moscow, 1963. (in Russian).

[Mas79] W.J. Masek. Some NP-complete set covering problems. Manuscript, 1979.

[MVW99] P. B. Miltersen, N.V. Vinodchadran, and O. Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In T. Asano, H. Imai, D.T. Lee, S. Nakano, and T. Tokuyama, editors, *Proceedings of the Fifth Annual International Conference on Computing and Combinatorics*, volume 1627 of *Lecture Notes in Computer Science*, pages 210–220. Springer Verlag, 1999. (COCOON'99).

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[Pol74] J.M. Pollard. Theorems on factorization and primality testing. *Proceedings of the Cambridge Philosophical Society*, 76:521–528, 1974.

[RR97] A.A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55:24–35, 1997.

[Sha49] C.E. Shannon. The synthesis of two-terminal switching circuits. *Bell Systems Technical Journal*, 28(1):59–98, 1949.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 330–335, 1983.

[Str76] V. Strassen. Einige Resultate über Berechnungskomplexität. *Jahresberichte der DMV*, 78:1–8, 1976.

[Tra84] B.A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force search) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[Wil85] C.B. Wilson. Relativized circuit complexity. *Journal of Computer and System Sciences*, 31:169–181, 1985.

[Yab59a] S.V. Yablonski. The algorithmic difficulties of synthesizing minimal switching circuits. In *Problemy Kibernetiki 2*, pages 75–121. Fizmatgiz, Moscow, 1959. English translation in *Problems of Cybernetics II.*

[Yab59b] S.V. Yablonski. On the impossibility of eliminating perebor in solving some problems of circuit theory. *Doklady Akademii Nauk SSSR*, 124(1):44–47, 1959. English translation in *Soviet Mathematics Doklady.*

[ZH86] S. Zachos and H. Heller. A decisive characterization of BPP. *Information and Control*, 69(1-3):125–135, 1986.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'