Approaching MCSP from Above and Below: Hardness for a Conditional Variant and $\mathsf{AC}^0[p]$

Rahul Ilango
Massachusetts Institute of Technology, Cambridge, MA, USA
rilango@mit.edu

Abstract

The Minimum Circuit Size Problem (MCSP) asks whether a given Boolean function has a circuit of at most a given size. MCSP has been studied for over a half-century and has deep connections throughout theoretical computer science including to cryptography, computational learning theory, and proof complexity. For example, we know (informally) that if MCSP is easy to compute, then most cryptography can be broken. Despite this cryptographic hardness connection and extensive research, we still know relatively little about the hardness of MCSP unconditionally. Indeed, until very recently it was unknown whether MCSP can be computed in $\mathsf{AC}^0[2]$ (Golovnev et al., ICALP 2019).

Our main contribution in this paper is to formulate a new "oracle" variant of circuit complexity and prove that this problem is NP-complete under randomized reductions. In more detail, we define the Minimum Oracle Circuit Size Problem (MOCSP) that takes as input the truth table of a Boolean function f, a size threshold s, and the truth table of an oracle Boolean function $\mathcal{O}$, and determines whether there is a circuit with $\mathcal{O}$-oracle gates and at most s wires that computes f. We prove that MOCSP is NP-complete under randomized polynomial-time reductions.

We also extend the recent $\mathsf{AC}^0[p]$ lower bound against MCSP by Golovnev et al. to a lower bound against the circuit minimization problem for depth-d formulas, (AC^0_d)-MCSP. We view this result as primarily a technical contribution. In particular, our proof takes a radically different approach from prior MCSP-related hardness results.

2012 ACM Subject Classification Theory of computation → Circuit complexity; Theory of computation → Problems, reductions and completeness

Keywords and phrases Minimum Circuit Size Problem, reductions, NP-completeness, circuit lower bounds

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.34

Related Version A full version of this paper is available at `https://eccc.weizmann.ac.il/report/2019/021/`.

Funding This work was supported (in part) by an Akamai Presidential Fellowship.

Acknowledgements I would like to give a special thanks to Eric Allender for innumerable suggestions and perspectives during all stages of this work. To name just a single example, one of his suggestions led me to improve a PARITY-reduction to the presented MAJORITY-reduction for (AC^0_d)-MCSP. I would also like to thank Abhishek Bhrushundi and Aditi Dudeja for their help in results about constant depth formulas that lead to this paper. I am grateful to Ryan Williams for asking interesting questions and helping to improve the exposition of the paper. Finally, I thank Harry Buhrman, Lance Fortnow, Igor Oliveira, Ján Pich, Aditya Potukuchi, Ninad Rajgopal, Michael Saks, and Rahul Santhanam for answering my questions and engaging in many useful discussions about this work.

1 Introduction

The Minimum Circuit Size Problem (MCSP) takes as input a Boolean function f (represented by its truth table) and a size parameter s and asks if there is a circuit of size at most s computing f. Study of this problem began in the 1950s by complexity theorists in the Soviet Union [31], where MCSP was of such great interest that Levin is said to have delayed

publishing his initial NP-completeness results in hope of showing that MCSP is NP-complete.[1] Interest in MCSP was revitalized when Kabanets and Cai [19] connected the problem with the natural proofs framework of Razborov and Rudich [28]. Since then, MCSP has been the subject of intense research. We begin by (non-exhaustively) reviewing this work.[2]

1.1 Known lower bounds, hardness, and non-hardness for MCSP

It is easy to see that MCSP is in NP (the circuit of size at most s can be used as a witness), but, despite work by numerous researchers, the exact complexity of MCSP remains unknown.

Hardness results for MCSP. We believe MCSP is not easy to compute. Kabanets and Cai [19] show that MCSP $\notin$ P conditioned on a widely-believed cryptographic hypothesis.

Thus, the most natural question about MCSP's complexity is whether it is NP-complete. As of yet, we have not managed to uncover even strong supporting evidence for, or against, MCSP being NP-complete. The strongest known hardness result is by Allender and Das [3] who show MCSP is hard for the cryptographically important class SZK under randomized reductions.

Under less powerful types of reductions, we only know hardness for some subclasses of P. For instance, building on the results of Oliveira and Santhanam [26], Golovnev et al. [12] show that the class of functions computable by polynomial-sized formulas (NC^1) reduces to MCSP under AC^0 reductions.

Lower Bounds for MCSP. Unconditionally, we know lower bounds against MCSP for several restricted computational models. Cheraghchi, Kabanets, Lu, and Myrisiotis [10] prove lower bounds for MCSP against DeMorgan formulas, branching programs, and AC^0 circuits that essentially match the best known explicit lower bounds for these models. Golovnev et al. [12] show that MCSP requires exponential-sized $\mathsf{AC}^0[p]$ circuits by proving MAJORITY $\in (\mathsf{AC}^0)^{\mathsf{MCSP}}$. The MAJORITY hardness result of [12] generalizes to the circuit minimization problem for many circuit classes, however, the techniques fail in the case of constant depth formulas.

Non-hardness results for MCSP. Surprisingly, we know MCSP cannot be NP-hard under certain types of reductions. For example, Hirahara and Watanabe [16] show that "oracle-independent polynomial reductions" cannot show that MCSP is hard for a class larger than P. Moreover, while essentially all NP-complete problems are complete under rather weak reductions such as uniform $\mathsf{TIME}[n^{o(1)}]$ reductions (which we do not define here), Murray and Williams [23] prove that MCSP is not even NP-hard under $\mathsf{TIME}[n^{.49}]$ reductions.

Conditioned on a widely-believed cryptographic hypothesis, Allender and Hirahara [4] show that a very weak approximation of MCSP is actually neither in P nor NP-complete.

Approaching MCSP from below. Given the apparent difficulty of resolving the complexity of MCSP, a natural approach is to progress towards MCSP "from below": considering restricted classes of circuits $\mathcal{C}$ that we understand better and determining whether the $\mathcal{C}$-circuit minimization problem, ($\mathcal{C}$)-MCSP, is NP-hard.

---

[1] [6] cites a personal communication from Levin regarding this.

[2] In our review, we are sometimes slightly informal and imprecise in order to be more accessible. We encourage the reader to look at the corresponding references for precise statements.

As it stands, we know that minimizing DNF formulas and minimizing $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MOD}_m$ circuits are both NP-hard by Masek [20] and by Hirahara, Oliveira, and Santhanam [15] respectively. It is still open whether the circuit minimization problem for, say, depth-3 formulas is NP-hard.

1.2 Implications of lower bounds and hardness for MCSP

Since the main goal of this paper is to shed light on the complexity of MCSP, it is worth motivating why the complexity of MCSP is important.

Indeed, while researchers have not yet managed to establish the complexity of MCSP, a series of works, beginning with Kabanets and Cai [19], connect the computational complexity of MCSP and its variants to longstanding open questions in the field.

Separations of complexity classes. Several works ([17], [23], [3], [19]) show that MCSP being NP-hard, under various notions of reducibility, implies unknown class separations. For example, Hitchcock and Pavan [17] and Murray and Williams [23] show that if MCSP is NP-hard under polynomial-time reductions, then $\mathsf{ZPP} \neq \mathsf{EXP}$, a major open problem.[3]

Worst-case versus average-case complexity for NP. Using tools developed by Nisan and Wigderson [24] and Carmosino, Impagliazzo, Kabanets, and Kolokova [9], Hirahara [14] gives a "worst-case to average-case" reduction for NP conditioned on a certain approximation to MCSP being NP-hard. Thus, if one could show this approximation to MCSP is NP-hard, the worst-case and average-case complexity of NP would be equivalent.

Circuit Lower Bounds. Recent work ([22], [27], [25]) explores a "hardness magnification" phenomenon whereby even weak circuit lower bounds on certain computational problems imply strong lower bounds on other problems. For example, Mckay, Murray, and Williams [22] show that if MCSP (under a small size parameterization) cannot be solved by circuits of size $n \cdot \mathsf{poly}(\log n)$, then NP does not have polynomial-size circuits.

1.3 Our Contributions

In this work, we focus on hardness results for natural variants of MCSP with the aim of shedding light on the complexity of MCSP itself.

NP-Hardness of oracle MCSP (MOCSP)

As mentioned previously, some research has approached the problem of proving NP-hardness for MCSP "from below," that is, showing that the circuit minimization problem is NP-hard for restricted classes of circuits like DNF circuits and $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MOD}_m$ circuits.

In this paper, we attempt to approach MCSP "from above." That is, we study the complexity of a problem that is at least as hard as MCSP. In particular, we define the *Minimum Oracle Circuit Size Problem* (MOCSP) to take as input a truth table T, a size parameter $s \in \mathbb{N}$, and an oracle truth table $\mathcal{O}$. The goal is to determine whether there is an $\mathcal{O}$-oracle circuit with at most s wires that computes T. It is easy to see that $\mathsf{MOCSP} \in \mathsf{NP}$ (the oracle circuit of size s can still act as an NP-witness).

[3] [23] only shows the result under many-one reductions, but their techniques easily generalize to the truth table case. [17] explicitly proves the truth table result using a different approach than [23].

A natural way to think of MOCSP is as a "conditional" version of MCSP where we are now measuring the complexity of computing T when we are given easy access to $\mathcal{O}$.

Although our problem definition is new, this is not the first time someone has considered an "oracle version" of MCSP. Several papers ([2, 5, 16, 18]) study the problem $\mathsf{MCSP}^{\mathsf{A}}$ of minimizing oracle circuits for a *fixed* oracle *language* A. MOCSP differs fundamentally from $\mathsf{MCSP}^{\mathsf{A}}$ in that the oracle circuit gets access to an *arbitrary* Boolean function, not a language, and the function the oracle circuit has access to is an *input* to the problem.

Furthermore, while $\mathsf{MOCSP} \in \mathsf{NP}$, the complexity of $\mathsf{MCSP}^{\mathsf{A}}$ depends on the oracle A. For example, the reference [2] proves $\mathsf{MCSP}^{\mathsf{QBF}}$ is complete for PSPACE under randomized reductions. Moreover, there is a trivial reduction from MCSP to MOCSP: simply provide no oracle truth tables. Therefore, since MOCSP is still in NP and MOCSP and is at least as hard as MCSP, MOCSP is a nice "testing ground" for hardness results we conjecture for MCSP.

Indeed, as is conjectured for MCSP, we *can* prove that MOCSP is NP-hard under randomized reductions.

▶ **Theorem 1.1.** MOCSP *is* NP *hard under:*
- *randomized many-one reductions with one-sided error, and*
- *randomized Turing reductions with zero-sided error.*

These NP-hardness results are both proved by giving a reduction from approximating set cover to MOCSP. It is worth noting that the NP-hardness results of (DNF)-MCSP [20] and $(\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MOD}_m)$-MCSP [15] are also proved via set cover problems, although both of these reductions are deterministic while our reduction is randomized.

Given that we can show MOCSP is NP-hard under randomized reductions, one might even begin to hope that we can prove hardness under, say, polynomial-time reductions. Unfortunately, this seems difficult. Essentially the same proofs Murray and Williams [23] or Hitchcock and Pavan [17] use to show that MCSP being NP-hard under polynomial-time truth table reductions implies $\mathsf{EXP} \neq \mathsf{ZPP}$ also work for MOCSP.

▶ **Theorem 1.2** (Essentially proven in [17] and [23])**.** *If* NP *reduces to* MOCSP *under polynomial time (truth table) reductions, then* $\mathsf{EXP} \neq \mathsf{ZPP}$*.*

Thus, improving our reduction to a polynomial-time truth table reduction requires separating EXP from ZPP, a longstanding open problem. For completeness, we give the MOCSP version of Murray and Williams' proof in Appendix B.

Even so, we expect that the ground truth is that MOCSP is NP-hard under, at least, deterministic polynomial-time Turing reductions.

▶ **Conjecture 1.3.** MOCSP *is* NP*-hard under polynomial-time Turing reductions.*

A perspective on MOCSP and some questions

In light of the fact that hardness for MCSP beyond SZK under even non-uniform reductions is unknown, we found these MOCSP hardness results to be quite surprising. Moreover, the NP-hardness of MOCSP seems to bolster the conjecture that MCSP is in fact NP-hard. To an optimist, NP-hardness results for MOCSP could even be a first step towards proving NP-hardness for MCSP. Indeed, a PSPACE-hardness result was first proved by Buhrman and Torenvliet [8] for an "oracle" version of space-bounded Kolmogorov complexity before Allender et al. [2] showed PSPACE-hardness for the non-oracle version about four years later.[4]

[4] The conference versions of [8] and [2] are four years apart.

Even if stronger hardness results for MCSP remain out of reach, MOCSP could still yield valuable insights about MCSP. For instance, it would be interesting to see which of the barriers and non-hardness results known for MCSP carry over to MOCSP.

▶ **Open Question 1.4.** *Can one show that other barriers or non-hardness results that hold for* MCSP *also hold for* MOCSP*?*

As an example of the insight given by answers to this question, consider Murray and Williams' [23] result that proving MCSP is NP-hard under polynomial-time reductions implies EXP $\neq$ ZPP. A natural question one might ask is whether we can improve this theorem to show that MCSP being NP-hard under randomized reductions implies unknown class separations. As we note in Theorem 1.2, however, Murray and Williams' proof carries over to MOCSP, and Theorem 1.1 shows MOCSP is indeed NP-hard under randomized reductions. Thus, any improvement of Murray and Williams' result to randomized reductions likely requires a fact about MCSP that we do not know for MOCSP.

In another direction, our results seem to imply that proving hardness for MOCSP is more tractable than proving hardness for MCSP. Since we have already noted that MOCSP can be used as a "testing ground" for questions about MCSP (since any hardness result for MCSP must also hold for MOCSP), one can explore whether other conjectured hardness results for MCSP hold for MOCSP. For example, Hirahara's [14] worst-case to average-case reduction for NP can be based on a certain approximation of MCSP being NP-hard, which would imply that a certain approximation of MOCSP is also NP-hard. Given that we can prove the NP-hardness of MOCSP under randomized reductions, we ask if one can prove something similar for the approximation version of MOCSP.

▶ **Open Question 1.5.** *Can one prove that, for some $\epsilon > 0$, approximating* MOCSP *on n-inputs to a factor of n^ϵ is* NP*-hard under, say,* P/poly *reductions? Conversely, can one prove that there is any barrier to showing such a hardness result?*

Finally, while in this paper we concentrate on approaching MCSP from above via MOCSP, it would be interesting to explore other problems that are harder than MCSP but still in NP. Perhaps this could help clarify what aspects of MCSP make it difficult to prove hardness for.

MAJORITY-hardness for (AC^0_d)-MCSP

As mentioned previously, Golovnev et al. [12] proves that MAJORITY $\in (\mathsf{AC}^0)^{\mathsf{MCSP}}$. Using similar techniques, they also show that, for restricted classes of circuits $\mathcal{C}$ such as formulas and constant depth circuits, the $\mathcal{C}$-circuit minimization problem, denoted $(\mathcal{C})$-MCSP, is hard for MAJORITY under AC^0 reductions. For these MAJORITY reductions to work, [12] requires that the size of the minimum $\mathcal{C}$-circuit on truth tables of length n is roughly $(n^{.49})$-Lipschitz, that is, flipping one bit in any truth table can change the $\mathcal{C}$-circuit complexity by at most an $n^{.49}$ additive term. This Lipschitzness hypothesis is unknown (and in our estimation probably false) in the class of depth-d *formulas*, which we denote AC^0_d.[5] As a result, until this work, it was unknown whether, say, (AC^0_d)-MCSP $\in \mathsf{AC}^0[2]$.

We stress that even though AC^0_d is an extremely restrictive class of formulas, (AC^0_d)-MCSP should be a very hard problem. For example, Allender *et al.* [1] prove that if (AC^0_d)-MCSP had polynomial-size circuits for sufficiently large d, then one could factor Blum integers using circuits of size 2^{n^ϵ} for all $\epsilon > 0$.

[5] We will always use the notation AC^0_d to refer to depth-d *formulas* and never depth-d *circuits*.

In this paper, we prove that (AC^0_d)-MCSP is indeed hard for MAJORITY by giving a win-win argument depending on whether the Lipschitzness hypothesis is true. Applying the lower bounds of Razborov [28] and Smolensky [30] then gives an $\mathsf{AC}^0[p]$ lower bound for (AC^0_d)-MCSP.

▶ **Theorem 1.6.** *Let $d \geq 2$. Then* MAJORITY $\leq^{\mathsf{AC}^0}_{tt}$ (AC^0_d)-MCSP. *Consequently, for any prime p,* (AC^0_d)-MCSP $\notin \mathsf{AC}^0[p]$.

One can view this theorem as (small) progress towards the program of approaching MCSP from below. The natural next step in the program is to prove that (AC^0_3)-MCSP is NP-hard, and, until this work, $\mathsf{AC}^0[p]$ lower bounds were open for this problem (despite knowing such $\mathsf{AC}^0[p]$ lower bounds for less restrictive circuit classes!). Perhaps the techniques introduced here could be useful in proving further hardness results for (AC^0_3)-MCSP.

Indeed, our techniques[6] may be the most interesting part of Theorem 1.6. They are entirely different than the ones used by [12] for general MCSP and, to our knowledge, all known MCSP hardness results. For example, a crucial step in the proof is introducing algebraic machinery and examining the size of a function's smallest circuits modulo a certain prime.

Indeed, all previously known MCSP-related reductions (to our knowledge) continue to work even when the reduction is given access to close approximations of a function's minimum circuit size. Because of this, it was not clear if, for the purpose of reductions, there was any difference between knowing whether a function has circuits of size exactly s or $s+1$. However, the algebraic nature of this MAJORITY reduction actually requires knowing such exact quantities, illustrating the existence of useful techniques that can distinguish between MCSP and approximations of MCSP.

1.4 Proof Overviews

In this section, we give fairly detailed overviews of our proofs. In doing this, we will often state results without filling in lower-level details. To make clear to the reader when we are doing this, we mark such sentences with an italicized *we observe*.

NP-hardness of Oracle MCSP (MOCSP)

Recall, the Minimum Oracle Circuit Size Problem, MOCSP, takes as input a truth table T, a threshold $s \in \mathbb{N}$, and an oracle truth table $\mathcal{O}$ and outputs whether there is an $\mathcal{O}$-oracle circuit with at most s wires that computes T. We denote the output of MOCSP on such an input as $\mathsf{MOCSP}(T, s; \mathcal{O})$. We denote the minimum size (measured by wires) of any $\mathcal{O}$-oracle circuit computing T as $\mathsf{CC}^{\mathcal{O}}(T)$.

We prove that MOCSP is NP-hard by giving a reduction from 6-approximating the Set Cover Problem (6-SetCover) to MOCSP. In our notation, 6-SetCover takes as input sets $S_1, \ldots, S_t \subseteq [n]$ whose union is $[n]$ as well as an integer $c \in [n]$ and requires outputting YES when $c \geq \ell$ and NO when $c < \ell/6$ where ℓ is the optimal cover size, i.e.

$$\ell = \min\{|I| : I \subseteq [t] \text{ and } \bigcup_{i \in I} S_i = [n]\}.$$

Approximating SetCover to an $((1 - o(1)) \ln n)$ factor is known to be NP-hard by Dinur and Steurer [11].

[6] Specifically, our techniques in the case when the Lipschitz hypothesis fails (which we expect to be the ground truth).

Informal idea. We begin by giving a high-level overview of the reduction to orient the reader. (It will be very informal, but we are building to a more detailed description.) Say we are given sets $S_1, \dots, S_t \subseteq [n]$ whose union is $[n]$. One can think of each of these sets S_i as "seeing" a small portion of the ground set $[n]$. Then, for a uniformly random truth table T of length $m \geq (nt)^5$, we let each set $S_1, \dots, S_t$ induce truth tables $T_{S_1}, \dots, T_{S_t}$ respectively where each truth table T_{S_i} has the same length at T and "sees" roughly the same part of T as S_i "sees" of $[n]$. We define our oracle $\mathcal{O}$ to be the concatenation of these truth tables. Finally, we ask how hard it is for a circuit to compute T given oracle access to $\mathcal{O}$.

The idea is that if a circuit looks at a collection of induced truth tables corresponding to a set cover, then it can "see" all of T and thus easily compute T. If not, then a large part of T is still random to the circuit and thus hard to compute. Indeed, we show that with high probability the quantity $\mathsf{CC}^{\mathcal{O}}(T)$ can be used to estimate ℓ up to a constant factor.

We now describe the reduction in more detail. Fix sets $S_1, \dots, S_t \subseteq [n]$ whose union is $[n]$ and, let ℓ be the optimal cover size of $[n]$ by $S_1, \dots, S_t$. Let T be a uniformly random truth table of length $m \geq (nt)^5$.

The truth tables induced by $S_1, \dots, S_t$ and T. We now rigorously define the truth tables $T_{S_1}, \dots, T_{S_t}$ of length m induced by $S_1, \dots, S_t$ and T. To do this, we chose a uniformly random partition of $[m]$ into n parts, that is, let $P : [m] \to [n]$ be a uniformly random function, and let $\mathcal{P} = (P_1, \dots, P_n)$ be the partition of $[m]$ into n parts where for $j \in [n]$

$$P_j = \{x \in [m] : P(x) = j\}.$$

We can then use this partition to "lift" any subset of $[n]$ into a subset of $[m]$ as follows. For a subset $S \subseteq [n]$, let S^m denote the subset of $[m]$ given by $S^m = \bigcup_{j \in S} P_j$.

Next, for a subset $S' \subseteq [m]$, we let $T_{\langle S' \rangle}$ be the truth table of length m that "sees" T on the elements of S' and zeroes everywhere else, that is

$$T_{\langle S' \rangle}(x) = T(x) \wedge \mathbb{1}_{x \in S'}$$

Finally, we define the truth table T_{S_i} induced by S_i to be the truth table $T_{\langle S_i^m \rangle}$ (we are dropping the m superscript and the bracketed subscript for concision).

Building the oracle $\mathcal{O}$. We build our oracle $\mathcal{O}$ by concatenating the induced truth tables $T_{S_1}, \dots, T_{S_t}$. In other words, let $\mathcal{O} : \{0,1\}^{\lceil \log t \rceil} \times \{0,1\}^{\log m} \to \{0,1\}$ be given by

$$\mathcal{O}(i, x) = \begin{cases} T_{S_i}(x) & , \text{ if } i \in [t] \\ 0 & , \text{ otherwise} \end{cases}$$

$\mathsf{CC}^{\mathcal{O}}(T)$ is at most $\ell(3 + \log(tm))$. Suppose, without loss of generality, that $S_1 \cup \dots \cup S_\ell$ is an optimal cover of $[n]$. We claim that $\bigvee_{i \in [\ell]} \mathcal{O}(i, x)$ is a circuit computing T. Unraveling our definitions, we have that

$$\bigvee_{i \in [\ell]} \mathcal{O}(i, x) = \bigvee_{i \in [\ell]} T_{S_i}(x) = \bigvee_{i \in [\ell]} (T(x) \wedge \mathbb{1}_{x \in S_i^m}) = T(x) \wedge (\bigvee_{i \in [\ell]} \mathbb{1}_{x \in S_i^m}) = T(x) \wedge \mathbb{1}_{x \in S_1^m \cup \dots \cup S_\ell^m}.$$

Using the fact that $S_1, \dots, S_\ell$ is a cover of $[n]$, we have that

$$T(x) \wedge \mathbb{1}_{x \in S_1^m \cup \dots \cup S_\ell^m} = T(x) \wedge \mathbb{1}_{x \in [m]} = T(x).$$

Thus, $\bigvee_{i \in [\ell]} \mathcal{O}(i, x)$ computes T and has at most

$$2\ell + \ell(1 + \log t + \log m) \leq \ell(3 + \log t + \log m)$$

wires.

With high probability, $\mathsf{CC}^{\mathcal{O}}(T) > \ell \log(tm)/2$. We do this by a union bound argument. Fix any oracle circuit C with at most $\ell \log(tm)/2$ wires. We show that the probability that $C^{\mathcal{O}}$ computes T is very small.

We begin by showing that on any fixed input x that

$$\Pr[C^{\mathcal{O}}(x) = T(x)] \leq 1 - \frac{1}{2n}.$$

Fix some x. We prove the following claim: that with probability at least $\frac{1}{n}$ over the choice of $P(x)$, the output of $C^{\mathcal{O}}(x)$ does not depend whether the value of $T(x)$ is one or zero. If this claim were true, then we would get the desired bound

$$\Pr[C^{\mathcal{O}}(x) = T(x)] \leq 1 - \frac{1}{2n}$$

since, assuming that $C^{\mathcal{O}}(x)$ does not depend on $T(x)$, the probability $C^{\mathcal{O}}(x) = T(x)$ is exactly $\frac{1}{2}$ (since $T(x)$ is chosen uniformly at random).

To prove the claim, we first note that the only way that the output of $C^{\mathcal{O}}(x)$ can depend on the value of $T(x)$ is if during the computation of $C^{\mathcal{O}}(x)$ C makes an oracle query (i', x') to $\mathcal{O}$ such that the value of $\mathcal{O}(i', x')$ depends on the value of $T(x)$. However, from the definition of $\mathcal{O}$, we know that the output of $\mathcal{O}(i', x')$ depends on $T(x)$ if and only if $x' = x$ and $P(x) \in S_{i'}$. Summarizing, $C^{\mathcal{O}}(x)$ depends on $T(x)$ only if $C^{\mathcal{O}}$ makes a query of the form (i', x) such that $P(x) \in S_{i'}$. But $C^{\mathcal{O}}$ has at most $\ell/2$ oracle gates, so it only has $\ell/2$ "tries" to find a set $S_{i'}$ that contains the uniformly random ground set element $P(x)$, which is less than the number of tries needed to cover the ground set! Hence, it is intuitive that with probability at least $\frac{1}{n}$, $P(x)$ takes on a value that is not in any of the $S_{i'}$ that $C^{\mathcal{O}}$ queries. *We observe* that this intuition can be made rigorous (with some more details), and hence, the claim is true.

At this point, we have completed our sketch of why for any fixed x

$$\Pr[C^{\mathcal{O}}(x) = T(x)] \leq 1 - \frac{1}{2n}.$$

Unfixing x, we would like to say that we can multiply these probabilities together for all $x \in [m]$ to bound the probability that $C^{\mathcal{O}}$ computes T. However, we cannot do this, as these events might have some dependencies between each other. Luckily, since, for any x, $C^{\mathcal{O}}(x)$ makes at most $\ell/2 \leq n$ oracle calls, *we observe* that applying the above probability bound can only "spoil" the bound for at most n other values of x. Hence, we can repeatedly apply the bound to about m/n different values of x yielding

$$\Pr[C^{\mathcal{O}} \text{ computes } T] \leq (1 - \frac{1}{2n})^{m/n} \leq e^{-\frac{m}{2n^2}} \leq e^{-n^3 t^5/2}.$$

Finally, union bounding over the at most

$$2^{O(\ell \log(tm)/2)^2} \leq 2^{O(n^2 t)}$$

oracle circuits with at most $\ell \log(tm)/2$ wires yields the desired result.

RP and ZPP reductions. At this point, we have shown that if $S_1, \ldots, S_t$ admits a c-cover, then $\mathsf{MOCSP}(T, c(3 + \log(tm)); \mathcal{O}) = \mathsf{YES}$ (note that this fact did not depend on our random choice of T or $\mathcal{P}$), and we have shown that if $S_1, \ldots, S_t$ does not admit a c-cover, then $\mathsf{MOCSP}(T, c \log(tm)/2; \mathcal{O}) = \mathsf{NO}$ with high probability. It is easy to see that this yields an RP many-one reduction (a randomized many-one reduction with one-sided error) from 6-SetCover to MOCSP.

Furthermore, since this RP reduction only errs by incorrectly classifying NO instances as YES instances, we can use the search-to-decision reduction for SetCover to check the correctness of purported YES instances, transforming our RP reduction into a ZPP-Turing reduction (a randomized zero-error reduction that can make multiple adaptive queries to MOCSP).

Majority Hardness for (AC^0_d)-MCSP

Recall, AC^0_d is the class of depth-d formulas made up of AND and OR gate with unbounded fan-in. We also define $\mathsf{AND} \circ \mathsf{AC}^0_{d-1}$ and $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ be the classes of AC^0_d formulas with a top AND and top OR gate respectively. For $\mathcal{F} \in \{\mathsf{AC}^0_d, \mathsf{AND} \circ \mathsf{AC}^0_{d-1}, \mathsf{OR} \circ \mathsf{AC}^0_{d-1}\}$ and a truth table T, we let $\mathsf{CC}_{\mathcal{F}}(T)$ denote the size of the minimum $\mathcal{F}$-formula computing T where the size of a formula is the number of input leaves.

Our analysis proceeds by considering each $n \in \mathbb{N}$ and splitting into cases depending on whether $\mathsf{CC}_{\mathsf{AC}^0_d}$ is Lipschitz on truth tables of length around n. In more detail, fix some sufficiently large n. Let $q = \Theta(n^2)$ be a power of two. We divide into cases depending on whether there exists an $m \in \{q^{10}, q^{50}\}$ such that $\mathsf{CC}_{\mathsf{AC}^0_d}$ is $(m^{.25})$-Lipschitz on truth tables of length m.

Case 1: Lipschitzness holds for some m

If there does exist an $m \in \{q^{10}, q^{50}\}$ such that $\mathsf{CC}_{\mathsf{AC}^0_d}$ is $(m^{.25})$-Lipschitz on truth tables of length m, then the techniques of Golovnev *et al.* [12] yield an AC^0 truth table reduction from MAJORITY on n-bits to (AC^0_d)-MCSP on m-bits. For completeness, we include a self-contained proof of this case in Appendix A.

Case 2: Lipschitzness fails

Assume that for all $m \in \{q^{10}, q^{50}\}$ $\mathsf{CC}_{\mathsf{AC}^0_d}$ is not $(m^{.25})$-Lipschitz on truth tables of length m. Let $u = q^{10}$ and $v = q^{50}$.

Lipschitzness failing $\implies$ functions easier to compute with a top AND gate. *We observe*, as a straight forward consequence of Lipschitzness failing, that there exists a truth table of length u that has an optimal formula with large top fan-in and a truth table of length v that is easier to compute with a top AND gate:

1. There exists a Boolean function f^u that takes $\log u$ inputs and an AC^0_d formula ϕ^u such that ϕ^u is an optimal AC^0_d formula for f^u and $\phi^u = \phi^u_1 \wedge \cdots \wedge \phi^u_t$ for some $t \geq n$ and some $\phi^u_1, \ldots, \phi^u_t \in \mathsf{AC}^0_{d-1}$.
2. There exists a Boolean function f^v that takes $\log v$ inputs such that $\mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(f^v) > \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^v) + u \log u$.

We will make use of f^u and ϕ^u to reduce MAJORITY to $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$ and we will use f^v to reduce $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$ to $\mathsf{CC}_{\mathsf{AC}^0_d}$.

Using $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$ and optimal subformulas of ϕ^u to compute a dot product. The heart of our MAJORITY reduction is a fairly elementary observation about optimal $(\mathsf{AND} \circ \mathsf{AC}^0_{d-1})$ formulas. Recall, $\phi^u = \phi^u_1 \wedge \cdots \wedge \phi^u_t$ is an optimal (AC^0_d) formula for f^u and, hence, also an optimal $(\mathsf{AND} \circ \mathsf{AC}^0_{d-1})$ formula for f^u. *We observe* that for any $A \subseteq [t]$, the $(\mathsf{AND} \circ \mathsf{AC}^0_{d-1})$-optimality of ϕ^u implies that $\bigwedge_{i \in A} \phi^u_i$ is also an optimal $(\mathsf{AND} \circ \mathsf{AC}^0_{d-1})$ formula for the function it computes.

Introducing some notation, for a string $x \in \{0,1\}^n$, we let f^u_x be the function given by $\bigwedge_{i \in O_x} \phi^u_i$ where $O_x \subseteq [n]$ are the bits in x that are one. Using the above observation about subformulas being optimal, we have that[7] $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^u_x) = \sum_{i \in O_x} |\phi^u_i|$. Thus, one can think of $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^u_x)$ as computing the dot product between x and the vector $\langle |\phi^u_1|, \ldots, |\phi^u_n| \rangle$.

Note that that the definition of f^u_x depends on the labeling of $\phi^u_1, \ldots, \phi^u_t$, in particular the choice of which ϕ^u_i have $i \leq n$. We will later choose a labeling of the ϕ^u_i that is convenient.

Computing MAJORITY (non-uniformly) using $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$. Our goal is to compute MAJORITY on a string $x \in \{0,1\}^n$ using the above "dot product" observation. Before we show how to do this, we give some intuition on how we came up with the idea.

Instead of trying to compute MAJORITY, suppose we relaxed the problem to computing PARITY given access to the integer produced by the dot product $x \cdot \langle |\phi^u_1|, \ldots, |\phi^u_n| \rangle$. Well, if it so happened that all the entries in the vector $\langle |\phi^u_1|, \ldots, |\phi^u_n| \rangle$ were odd, then it is clear that the integer produced by $x \cdot \langle |\phi^u_1|, \ldots, |\phi^u_n| \rangle$ is odd if and only if x has an odd number of ones. Our approach for MAJORITY is a generalization of this.

Let $p = O(n)$ be prime greater than n. *We observe*, via a pigeon-hole principle argument, that there exist integers $k \geq 0$ and $1 \leq r \leq p-1$ and indices $i_1, \ldots, i_n \in [m]$ such that

$$|\phi^u_{i_1}|/p^k \equiv \cdots \equiv |\phi^u_{i_n}|/p^k \equiv r \mod p.$$

Thus, after an appropriate relabeling of ϕ^u_i, we have that

$$|\phi^u_1|/p^k \equiv \cdots \equiv |\phi^u_n|/p^k \equiv r \mod p.$$

Hence, we can determine the weight w of x (and hence compute MAJORITY of x) by computing the value of

$$\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^u_x)/p^k = \sum_{i \in O_x} |\phi^u_i|/p^k \equiv rw \mod p$$

and multiplying by the inverse of r modulo p. [8]

Reducing computing $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$ to computing $\mathsf{CC}_{\mathsf{AC}^0_d}$. Ultimately, we want to compute MAJORITY using $\mathsf{CC}_{\mathsf{AC}^0_d}$ not $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$. By the above procedure, it suffices to show how to compute $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^u_x)$ using $\mathsf{CC}_{\mathsf{AC}^0_d}$.

We can make such a computation as follows. Recall that f^v is a function satisfying

$$\mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(f^v) > \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^v) + u \log u$$

whose existence is guaranteed by the failure of Lipschitzness. Take the direct product of f^u_x with f^v to obtain a function $g_x(y,z) = f^u_x(y) \wedge f^v(z)$. Since the difference between computing f^v with a top AND gate and a top OR gate is larger than $u \log u$ (which is the maximum complexity of f^u_x), *we observe*[9] any optimal AC^0_d formula for g_x must have a top AND gate, so

$$\mathsf{CC}_{\mathsf{AC}^0_d}(g_x) = \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(g_x).$$

---

[7] Recall, our notion of formula size is the number of input leaves.

[8] In case the reader is unsure of whether the last parts of this procedure are implementable in AC^0, realize that the output of $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^u_x)$ is a binary string of length $O(\log n)$ and that any function on a string of length $O(\log n)$ can be computed by a polynomial-sized DNF. See the proof in Section 4 for more details.

[9] both the "*we observe" statements in this paragraph are consequences of standard direct product theorems for formulas.*

Then, *we observe* that

$$\mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(g_x) = \mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^u_x) + \mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^v).$$

Hence, if we are non-uniformly given the value of $\mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^v)$, we can subtract $\mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^v)$ from $\mathsf{CC}_{\mathsf{AC}^0_d}(g_x)$ to find $\mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^u_x)$.

▶ Remark 1.7. We remark that the only time we use the failure of the Lipschitzness hypothesis is to show the existence of functions like f^u with high top fan-in and functions like f^v with a large difference between top AND gate and top OR gate complexity. Using known PARITY lower bounds and depth-hierarchy theorems for AC^0 circuits, we can unconditionally prove the existence of f^u and f^v respectively but with slightly worse parameters that would yield a quasi-polynomial reduction (at least in the $d \geq 3$ case) rather than the polynomial reduction we present.

1.5 Paper Organization

In Section 2 we fix notation and review precise definitions. In Section 3 we prove our MOCSP results, and in Section 4 we prove that MAJORITY reduces to (AC^0_d)-MCSP.

2 Preliminaries

For an integer n, we let $[n]$ denote the set $\{1, \ldots, n\}$. For a binary string $x \in \{0,1\}^*$, the *weight* of x, denoted $\mathsf{wt}(x)$, is the number ones in x. We identify a Boolean function $f : \{0,1\}^n \to \{0,1\}$ with its truth table $T \in \{0,1\}^{2^n}$ and often use them interchangeably.

We let log denote the base-2 logarithm and ln represent the base-e logarithm. For functions f and g, we say $f = \tilde{O}(g)$ if there exists a c such that $f(x) \leq \log^c(g(x))g(x)$ for all sufficiently large x. We say that $f = \tilde{\Omega}(g)$ if $g = \tilde{O}(f)$.

We say a function $f : \{0,1\}^n \to \mathbb{N}$ is *c-Lipschitz* if for all $x, y \in \{0,1\}^n$ that differ in at most one bit, $|f(x) - f(y)| \leq c$. (In this paper, the function f for which we care about c-Lipschitzness is the function that maps a truth table to its circuit complexity.)

Complexity classes and reductions

We assume the reader is familiar with the standard complexity classes such as $\mathsf{AC}^0, \mathsf{P}, \mathsf{ZPP}, \mathsf{RP}, \mathsf{NP}, \mathsf{E}$ and the notion of Turing machines. For background on these, we refer to Arora and Barak's excellent textbook [7]. For us, AC^0 always refers to *non-uniform* AC^0.

We review the types of reductions we use in case the reader is not familiar with randomized reductions, truth table reductions, or our notation.

Many-one reductions. We will make use of the following notions of many-one reduction.

- $L \leq^{\mathsf{P}}_m L'$ if there is a polynomial-time Turing machine M such that $x \in L \iff M(x) \in L'$.
- $L \leq^{\mathsf{RP}}_m L'$ if there is a polynomial-time probabilistic Turing machine M taking in a "random" auxiliary input r such that

$$x \in L \implies \forall r\, M(x,r) \in L', \text{ and}$$

$$x \notin L \implies \Pr_r[M(x,r) \notin L'] \geq 2/3,$$

and $|r|$ is polynomial in the length of x.

Truth table reductions. We will also make use of the following notions of truth table reductions.

- We say an oracle circuit C is a *truth table oracle circuit* if there is no directed path between oracle gates in C.
- $L \leq_{tt}^{\mathsf{AC}^0} L'$ if there is a non-uniform (polynomial-sized) AC^0 truth table oracle circuit C such that C computes L when given oracle access to L'.

Turing reductions. Finally, we say $L \leq_{\mathsf{T}}^{\mathsf{ZPP}} L'$ if L can be computed with zero-error by a polynomial-time probabilistic oracle Turing machine M with oracle access to L'. On any single input, M is allowed to output "don't know" with probability at most $1/2$.

AC^0_d formulas, (AC^0_d)-MCSP, and $\mathsf{CC}_{\mathsf{AC}^0_d}$

For an integer $d \geq 2$, we let AC^0_d denote the class of depth-d formulas that use AND and OR gates with unbounded fan-in and fan-out 1 and that takes as "input leaves" the bits of a binary string and the negation of each of those bits.

For an AC^0_d formula ϕ, we define the size of ϕ, denoted $|\phi|$, to be the total number of input leaves ϕ uses. For a Boolean function f, we let $\mathsf{CC}_{\mathsf{AC}^0_d}(f)$ be the size of the smallest AC^0_d formula computing f.

▶ **Definition 2.1** (Minimum Circuit Size Problem for constant depth formulas). *(AC^0_d)-MCSP, is the language given by*

$$\{(T, s) \in \{0,1\}^\star \times \mathbb{N} : T \text{ is the truth table of a Boolean function, and } \mathsf{CC}_{\mathsf{AC}^0_d}(T) \leq s\}.$$

We will also make use of the classes of formulas $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ and $\mathsf{AND} \circ \mathsf{AC}^0_{d-1}$, defined as the subclasses of AC^0_d formulas with a top OR gate and a top AND gate respectively. For $\mathcal{C} \in \{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}, \mathsf{AND} \circ \mathsf{AC}^0_{d-1}\}$, we define $\mathsf{CC}_{\mathcal{C}}$ and $(\mathcal{C})$-MCSP analogous to $\mathsf{CC}_{\mathsf{AC}^0_d}$ and (AC^0_d)-MCSP.

We also require the following elementary lemmas regarding AC^0_d formulas.

▶ **Lemma 2.2.** *Let f be a Boolean function. Then $\mathsf{CC}_{\mathsf{AC}^0_d}(f) = \mathsf{CC}_{\mathsf{AC}^0_d}(\neg f)$.*

Proof. One can use DeMorgan's laws to turn any AC^0_d formula for f of size s into an AC^0_d formula for $\neg f$ of size s. ◀

We note that our specific notion of AC^0_d formula size is crucial for the next lemma.

▶ **Lemma 2.3** (Direct product theorem for formulas). *Let $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$ be Boolean functions that are both not the constant zero function. Define $h : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ by $h(x,y) = f(x) \wedge g(y)$. Then*

$$\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(h) = \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f) + \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(g), \text{ and}$$

$$\mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(h) \geq \mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(f) + \mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(g).$$

Proof. It is easy to see that

$$\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(h) \leq \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f) + \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(g).$$

On the other hand, since f is not the constant 0 function, it has a 1-valued input x_0. Then, $h(x_0, y)$ computes $g(y)$. Thus, if ϕ is an $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formula for h, then ϕ has at least $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(g)$ y-input leaves. A similar argument shows that ϕ has at least $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f)$ x-input leaves. Hence

$$\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(h) \geq \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f) + \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(g).$$

A similar argument shows that

$$\mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(h) \geq \mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(f) + \mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(g). \quad \blacktriangleleft$$

Oracle Circuits and Oracle MCSP: MOCSP

An oracle circuit C is made up of NOT gates, fan-in two AND and OR gates, and oracle gates $g_1, \ldots, g_t$ for some integer $t \geq 0$. When given an oracle functions $\mathcal{O}$, we let $C^{\mathcal{O}}(x)$ be the value obtained when evaluating C on input x by using the $\mathcal{O}$ function to compute the outputs of the oracle gates $g_1, \ldots, g_t$.

We define the size of an oracle circuit C, denoted $|C|$, to be the number of wires in C. For Boolean functions $f, \mathcal{O}$, we let $\mathsf{CC}^{\mathcal{O}}(f)$ be the size of the smallest $\mathcal{O}$-oracle circuit that computes f. Analogous to MCSP, we define the following.

▶ **Definition 2.4** (The Minimum Oracle Circuit Size Problem). *The Minimum Oracle Circuit Size Problem, denoted* MOCSP*, takes as input a truth table T, a threshold $s \in \mathbb{N}$, and an oracle truth table $\mathcal{O}$ and outputs whether $\mathsf{CC}^{\mathcal{O}}(T) \leq s$. The output of* MOCSP *on such an input is written as* $\mathsf{MOCSP}(T, s; \mathcal{O})$.

Set Cover

We will make use of the following well known NP-complete problem.

▶ **Definition 2.5** (Set Cover). *Set Cover, denoted* SetCover*, is the problem that takes as input a tuple $(n, c, S_1, \ldots, S_t)$, where $n \in \mathbb{N}$ is a universe size, $c \in \mathbb{N}$ is a proposed cover size $1 \leq c \leq n$, and $S_1, \ldots, S_t \subseteq [n]$ are sets whose union is $[n]$, and outputs whether $c \geq \ell$ where ℓ is the optimal cover size given by*

$$\ell = \min\{|I| : I \subseteq [t] \text{ and } \cup_{i \in I} S_i = [n]\}.$$

We will use that SetCover is NP-hard even to approximate to a roughly $\log n$ factor.

▶ **Theorem 2.6** (Dinur and Steurer [11]). *It is* NP*-hard to approximate* SetCover *to within a factor of $(1 - o(1)) \ln n$*

3 On the NP-hardness of MOCSP

First, we introduce some useful notation and definitions. For a truth table T of length m and a set $R \subseteq [m]$, let $T_{\langle R \rangle}$ be the truth table of length m where the jth bit of $T_{\langle R \rangle}$ equals

$$\begin{cases} \text{the } j\text{th bit of } T & \text{, if } j \in R \\ 0 & \text{, otherwise.} \end{cases}.$$

Equivalently,

$$T_{\langle R \rangle}(x) = T(x) \wedge \mathbb{1}_{x \in R}.$$

Next, we define a uniformly random partition $\mathcal{P} = (P_1, \ldots, P_n)$ of $[m]$ into n parts to be such that each element $i \in [m]$ is put into P_j where $j \in [n]$ is chosen uniformly at random. It will be also useful to think of $\mathcal{P}$ as a uniformly random function $P : [m] \to [n]$.

Next, for a partition $\mathcal{P} = (P_1, \ldots, P_n)$ of $[m]$ and any set $S \subseteq [n]$, we define the *$\mathcal{P}$-lift of* S, denoted $S^{\mathcal{P}}$, to be the subset of $[m]$ given by

$$S^{\mathcal{P}} = \bigcup_{i \in S} P_i.$$

Our main theorem shows that MOCSP can approximate set cover.

▶ **Theorem 3.1.** *Let $S_1, \ldots, S_t \subseteq [n]$ be sets that cover $[n]$. Let T be a truth table of length $m \geq (nt)^5$, and let $\mathcal{P} = (P_1, \ldots, P_n)$ be a uniformly random partition of $[m]$ into n parts. Define the oracle function the $\mathcal{O} : \{0,1\}^{\lceil \log t \rceil} \times \{0,1\}^{\log m} \to \{0,1\}$ by*

$$\mathcal{O}(i, y) = \begin{cases} T_{\langle S_i^{\mathcal{P}} \rangle}(y) & \text{, if } i \in [t] \text{ (identifying } i \text{ as an integer in the natural way)} \\ 0 & \text{, otherwise} \end{cases}.$$

Then

- $\mathsf{CC}^{\mathcal{O}}(T) \leq \ell(3 + \log m + \log t)$, *and*
- $\mathsf{CC}^{\mathcal{O}}(T) > \ell(\log m + \log t)/2$ *with high probability*

where ℓ is size of the optimal cover of $[n]$ by $S_1, \ldots, S_t$.

Proof. First, we show the upper bound.

▷ Claim 3.2. $\mathsf{CC}^{\mathcal{O}}(T) \leq \ell(3 + \log m + \log t)$

Proof. Without loss of generality, assume that the optimal cover size ℓ is witnessed by $S_1 \cup \cdots \cup S_\ell = [n]$. Then, by construction, the function computed by oracle circuit C given by

$$C(x) = \mathcal{O}(1, x) \vee \cdots \vee \mathcal{O}(\ell, x)$$

computes T and has at most

$$2\ell + \ell \cdot (\log m + \log t + 1) \leq \ell(3 + \log m + \log t)$$

wires. In more detail,

$$\bigvee_{i \in [\ell]} \mathcal{O}(i, x) = \bigvee_{i \in [\ell]} \bigvee_{j \in S_i} T_{\langle P_j \rangle}(x) = \bigvee_{j \in S_1 \cup \cdots \cup S_\ell} T_{\langle P_j \rangle}(x) = \bigvee_{j \in [n]} T_{P_j}(x) = T(x).$$

Therefore $\mathsf{CC}^{\mathcal{O}}(T) \leq \ell(3 + \log m + \log t)$. ◁

Now, we show the lower bound. We do this by a union bound argument. Fix some $\mathcal{O}$-oracle circuit C such that with at most $\ell(\log m + \log t)/2$ wires. Then C has at most $\ell/2$ oracle gates. We will show that

$$\Pr_{T, \mathcal{P}}[C^{\mathcal{O}} \text{ computes } T]$$

is exponentially small.

We do this by finding a long sequence of inputs $x_1, \ldots, x_d$ on which $C^{\mathcal{O}}$ has a not too large chance of computing T.

We construct this list of inputs recursively as follows. Let $x_1 = 0^{\log m}$, and let

$$Q_1 = \{x : C^{\mathcal{O}}(x_1) \text{ makes a query of the form } (i, x) \text{ to an oracle gate for some } i\}.$$

Now, for $j \geq 1$, if $\{0,1\}^{\log m} \setminus Q_j$ is non-empty, then let x_{j+1} be an element of $\{0,1\}^{\log m} \setminus Q_j$, and let

$$Q_{j+1} = Q_j \cup \{x : C^{\mathcal{O}}(x_{j+1}) \text{ makes a query of the form } (i, x) \text{ to an oracle gate for some } i\}.$$

If $\{0,1\}^{\log m} = Q_j$, then terminate the sequence.

Since C has at most $\ell/2 \leq n$ oracle gates, we know that $|Q_j| \leq j \cdot n$. Thus, since

$$|Q_d| = |\{0,1\}^{\log m}| = m,$$

the length d of this sequence is at least m/n.

It remains to bound

$$\Pr[C^{\mathcal{O}}(x_j) = T(x_j) \forall j] = \prod_{j=1}^{d} \Pr[C^{\mathcal{O}}(x_j) = T(x_j) | \bigwedge_{k \in [j-1]} C^{\mathcal{O}}(x_k) = T(x_k)].$$

Fix some $j \in [d]$. We will bound

$$\Pr[C^{\mathcal{O}}(x_j) = T(x_j) | \bigwedge_{k \in [j-1]} C^{\mathcal{O}}(x_k) = T(x_k)].$$

For convenience, let E denote the event we are conditioning on.

▷ Claim 3.3.

$$\Pr[C^{\mathcal{O}}(x_j) = T(x_j) | E] \leq 1 - \frac{1}{2n}.$$

Proof. By construction of the sequence $x_1, \ldots, x_d$, we know that on all the inputs $x_1, \ldots, x_{j-1}$, $C^{\mathcal{O}}$ does not make an oracle call of the form (i, x_j) for some i. Thus, since the only time the value of $\mathcal{O}$ depends on $T(x_j)$ and $P(x_j)$ is on inputs of the form (i, x_j) for some i, and $T(x_j)$ and $P(x_j)$ are chosen at independently at random, we know that $T(x_j)$ and $P(x_j)$ are still uniform random variables conditioned on E. In other words,

$$\Pr[T(x_j) = 1 | E] = \frac{1}{2}$$

and

$$\Pr[P(x_j) = r | E] = \frac{1}{n}$$

for all $r \in [n]$.

Now, define the alternate oracle $\mathcal{O}'$ that acts the same as $\mathcal{O}$ except that it rejects all queries containing x_j, that is

$$\mathcal{O}'(i, x) = \begin{cases} 0 & , \text{ if } x = x_j \\ \mathcal{O}(i, x) & , \text{otherwise.} \end{cases}$$

Now let $i_1, \ldots, i_v$ with $v \leq \ell/2$ be such that, using the modified oracle $\mathcal{O}'$, the only oracle queries $C^{\mathcal{O}'}(x_j)$ makes that end with x_j are $(i_1, x_j), \ldots, (i_v, x_j)$ (note that queries of this form are the only ones that could reveal information about $T(x_j)$). Since $v < \ell$ (recall ℓ is the size of the optimal cover) there actually exists an element $r^\star \in [n]$ that is not in $S_{i_1} \cup \cdots \cup S_{i_v}$.

Moreover, observe that if $P(x) = r^\star$, then $C^{\mathcal{O}}(x_j)$ will actually make the same oracle queries (and get the same zero responses) as the modified oracle circuit $C^{\mathcal{O}'}(x_j)$. In this case, since $P(x) = r^\star \notin S_{i_1} \cup \cdots \cup S_{i_v}$, it follows from the definition of $\mathcal{O}$ that

$$\mathcal{O}(i_1, x_j) = \cdots = \mathcal{O}(i_v, x_j) = 0$$

regardless of the value of $T(x_j)$. Thus, the output of $C^{\mathcal{O}}$ on input x does not depend at all on the value of $T(x)$. Hence, the probability it correctly guesses $C^{\mathcal{O}}(x) = T(x)$ is at most one half when $P(x_j) = r^\star$.

Since $P(x_j)$ is chosen uniformly at random, we have that $P(x_j) = r^\star$ with probability $1/n$.

Therefore, the probability

$$\Pr[C^{\mathcal{O}}(x_j) = T(x_j) | E] \leq 1 - \frac{1}{2n}$$ ◁

Using this claim, we now have

$$\prod_{j=1}^{d} \Pr[C^{\mathcal{O}}(x_j) = T(x_j) | \bigwedge_{k \in [j-1]} C^{\mathcal{O}}(x_k) = T(x_k)] \leq (1 - \frac{1}{2n})^d \leq e^{-\frac{d}{2n}} \leq e^{-\frac{m}{2n^2}} \leq e^{-\frac{n^3 t^5}{2}}.$$

On the other hand the number of oracle circuits of size $\ell(\log m + \log t)/2 = O(nt \log n)$ is at most $2^{O(n^2 t^2)}$. Thus, by the union bound the probability that there exists an $\mathcal{O}$-oracle circuit of size at most $\ell(\log m + \log t)/2$ computing T is $o(1)$ as desired. ◀

As an immediate consequence of Theorem 3.1, we get that MOCSP is hard for NP under many-one randomized reductions with one-sided error (the reduction is one-sided because the upper bound on the oracle circuit size of T in Theorem 3.1 is independent of the random choices of T and $\mathcal{P}$).

▶ **Corollary 3.4.** $\mathsf{NP} \leq_m^{\mathsf{RP}} \mathsf{MOCSP}$.

Moreover, the RP-reduction in Corollary 3.4 only errs by outputting YES when the answer should be NO. Thus, using the search-to-decision reduction for SetCover, we can check the correctness of purported YES answers, yielding a ZPP Turing reduction.

▶ **Corollary 3.5.** $\mathsf{NP} \leq_{\mathsf{T}}^{\mathsf{ZPP}} \mathsf{MOCSP}$.

4 MAJORITY $\leq_{tt}^{\mathsf{AC}^0}$ (AC^0_d)-MCSP

Let $d \geq 2$. Our goal in this section is to prove the following theorem.

▶ **Theorem 4.1.** $\mathsf{MAJORITY} \leq_{tt}^{\mathsf{AC}^0} (\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}$.

We will do this by showing that for all sufficiently large $n \in \mathbb{N}$, there exists an AC^0 truth table oracle circuit that computes MAJORITY on n-bits when given access to (AC^0_d)-MCSP.

Fix some n, and let q be the least power of two such that $n \leq \sqrt{q}/2$. We will split our analysis into cases depending on whether there exists an $m \in \{q^{10}, q^{50}\}$ such that $\mathsf{CC}^0_{\mathsf{AC}^0_d}$ is $(m^{.25})$-Lipschitz on inputs of length m.

4.1 Case 1: Lipschitzness Holds

If Lipschitzness holds, then the desired (AC^0_d)-MCSP oracle circuit C exists for computing MAJORITY on n-inputs by the work of Golovnev et al. [12]. At a high-level, C works by using the input string to sample a random variable whose circuit complexity spikes (in expectation) depending on the weight of the input and using Lipschitzness to show that this spike happens with such high probability that it can be derandomized using non-uniformity.

For completeness, we give a self-contained proof of this case in Appendix A.

4.2 Case 2: Lipschitzness fails

Assume that for all $m \in \{q^{10}, q^{50}\}$, $\mathsf{CC}_{\mathsf{AC}^0_d}$ is not $(m^{.25})$-Lipschitz on inputs of length m. Thus, for all $m \in \{q^{10}, q^{50}\}$ there exist functions $f^m, h^m : \{0,1\}^{\log m} \to \{0,1\}$ that differ only on a single input $z^m \in \{0,1\}^{\log m}$ such that $\mathsf{CC}_{\mathsf{AC}^0_d}(h^m) - \mathsf{CC}_{\mathsf{AC}^0_d}(f^m) > m^{.25}$.

We assume, without loss of generality, that for all $m \in \{q^{10}, q^{50}\}$, $f^m(z^m) = 0$ and $h^m(z^m) = 1$. (If this is not the case, then replace f^m and h^m by $\neg f^m$ and $\neg h^m$ respectively and apply Lemma 2.2.)

First, we show that the failure of Lipschitzness implies the existence of functions that are much easier to compute by formulas with an AND gate on top. For $m \in \{q^{10}, q^{50}\}$ let $\mathbb{1}_{z^m} : \{0,1\}^{\log m} \to \{0,1\}$ denote the indicator function that accepts just the string z^m.

▶ **Proposition 4.2.** *Let $m \in \{q^{10}, q^{50}\}$. For sufficiently large n, $\mathsf{CC}_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(f^m) \geq \mathsf{CC}_{\mathsf{AC}^0_d}(f^m) + m^{.24}$, and so any optimal AC^0_d formula for f^m has an AND gate on top.*

Proof. Suppose ϕ is an $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formula computing f^m, that is, f^m is computed by $\phi = \phi_1 \vee \cdots \vee \phi_t$ for some AC^0_{d-1} formulas $\phi_1, \ldots, \phi_t$. Then

$$\mathbb{1}_{z^m} \vee \phi_1 \vee \cdots \vee \phi_t$$

computes h^m. Since $\mathbb{1}_{z^m}$ can be computed by a single AND gate of formula size $\log m$, this shows that $\mathsf{CC}_{\mathsf{AC}^0_d}(h^m) \leq |\phi| + \log m$. Combining this with the fact that $\mathsf{CC}_{\mathsf{AC}^0_d}(h^m) - \mathsf{CC}_{\mathsf{AC}^0_d}(f^m) \geq m^{.25}$ gives the desired result. ◀

At this point, we will need to refer to both q^{10} and q^{50} individually, so for convenience let $u = q^{10}$ and $v = q^{50}$.

Let ϕ^u be an optimal AC^0_d formula for f^u. By Proposition 4.2, for sufficiently large n, we know that $\phi^u = \phi^u_1 \wedge \cdots \wedge \phi^u_t$ for some AC^0_{d-1} formulas $\phi^u_1, \ldots, \phi^u_t$. Moreover, we can assume, without loss of generality, that the top gate of ϕ^u_i is OR for all $i \in [t]$. (If some ϕ^u_i has an AND gate on top, then this AND can be carried out by the AND gate on top of ϕ^u without increasing the size of the formula.)

Our next Proposition shows that ϕ^u has high top fan-in.

▶ **Proposition 4.3.** *For sufficiently large n,*

$$t \geq u^{.24}.$$

Proof. We divide into cases depending on d.

Case 1: $d \geq 3$. Realize that

$$(\phi^u_1 \vee \mathbb{1}_{z^u}) \wedge \cdots \wedge (\phi^u_t \vee \mathbb{1}_{z^u})$$

computes h^u. Since $\mathbb{1}_{z^u}$ can be computed by a single AND gate of formula size $\log u$ and the top gate of each ϕ^u_i is an OR gate and $d \geq 3$, this yields a depth-d formula for h^u of size $\mathsf{CC}_{\mathsf{AC}^0_d}(f^u) + t \log u$. Since $\mathsf{CC}_{\mathsf{AC}^0_d}(h^u) - \mathsf{CC}_{\mathsf{AC}^0_d}(f^u) \geq u^{.25}$, the desired bound on t follows.

Case 2: $d = 2$. Let $\mathbb{1}_{z^u,j} : \{0,1\}^{\log u} \to \{0,1\}$ be the function that accepts a string x if and only if the jth bit of x equals the jth bit of z^u. Observe that, since $\bigwedge_{i\in[t]} \phi_i^u$ computes f^u, we have that

$$\bigwedge_{i\in[t]} \bigwedge_{j\in[\log u]} (\phi_i^u \vee \mathbb{1}_{z^u,j})$$

computes h^u. Since $\mathbb{1}_{z^u,j}$ is computed by a single input leaf and ϕ_i^u has an OR gate on top, this yields a depth-2 formula for h^u of size $(|\phi^u|+1)\log u$. Since ϕ^u is an optimal CNF, each clause ϕ_i^u of ϕ^u is the OR of at most $\log u$ input leaves. In other words, $|\phi_i^u| \leq \log u$. Therefore, we have that

$$\mathsf{CC}^0_{\mathsf{AC}^0_d}(h^u) \leq (|\phi^u|+1)\log u \leq (\sum_{i\in[t]} |\phi_i^u|+1)\log u \leq (t\log u+1)\log u = t\log^2 u + \log u.$$

On the other hand, we know by assumption that

$$\mathsf{CC}^0_{\mathsf{AC}^0_d}(h^u) > \mathsf{CC}^0_{\mathsf{AC}^0_d}(f^u) + u^{.25} \geq u^{.25}.$$

Combining these two inequalities gives us the desired bound on t. ◀

Let p be smallest prime greater than n. (Note that $p \leq 2n$ by Betrand's postulate, also known as Chebyshev's theorem. See [13] for a proof.) We say that an integer j is (k,r)-*good* for integers $k \geq 0$ and $1 \leq r \leq p-1$ if p^k divides j and $j/p^k \equiv r \mod p$. In other words, an integer j is (k,r)-good for $k \geq 0$ and $r \in [p-1]$ if the kth largest entry of the base-p representation of the integer j equals r and all previous entries equal zero. From this "base-p" perspective, it is clear that all positive integers j are (k,r)-good for some $k \geq 0$ and $r \in [p-1]$.

We show that, for some k and r, a large subset of the $|\phi_i^u|$ are (k,r)-good.

▶ **Proposition 4.4.** *For all sufficiently large n, there exist integers $k \geq 0$ and $1 \leq r \leq p-1$ and a set $S \subseteq [t]$ of cardinality n such that, for all $i \in S$, the integer $|\phi_i^u|$ is (k,r)-good.*

Proof. We do this by an averaging argument. First, we show that each $|\phi_i^u|$ is (k,r)-good for a k not too large.

▷ Claim 4.5. For all $i \in [t]$, $|\phi_i^u|$ is (k,r)-good for some $0 \leq k \leq \log(u\log u)+1$ and some $r \in [p-1]$.

Proof. Fix some $i \in [t]$. $|\phi_i^u|$ is a positive integer, so $|\phi_i^u|$ is (k,r)-good for some $k \geq 0$ and some $r \in [p-1]$. We still need to upper bound this k. Note that the size of $|\phi_i^u|$ is at most $u\log u$ since ϕ^u is optimal for f^u and f^u can be computed by a DNF of size $u\log u$. Thus, for p^k to divide $|\phi_i^u|$, we must have that $k \leq \log_p(u\log u)+1 \leq \log(u\log u)+1$. ◁

Since for all $i \in [t]$ we have shown that $|\phi_i^u|$ is (k,r) good for some $0 \leq k \leq \log(u\log u)+1$ and some $r \in [p-1]$, a standard averaging argument implies that there exists a set $S \subseteq [t]$ of cardinality at least

$$\frac{t}{(\log(u\log u)+1)(p-1)}$$

such that for all $i \in S$, $|\phi_i^u|$ is (k,r)-good for some fixed $k \geq 0$ and $1 \leq r \leq p-1$. For sufficiently large n, we have that

$$\frac{t}{(\log(u\log u)+1)(p-1)} \geq \frac{u^{.24}}{4n\log u} \geq n$$

using that $u = q^{10} \geq n^{10}$. We then can truncate S so that it has only n elements as desired. ◀

Assume that n is large enough that all the sufficiently large hypotheses in Propositions 4.2, 4.3, and 4.4 apply. For convenience, relabel $\phi_1, \ldots, \phi_t$ so that the set S guaranteed by Proposition 4.4 is just $S = [n]$. Fix $k \geq 0$ and $r \in [p-1]$ to be the values such that for all $i \in S = [n]$, $|\phi_i^u|$ is (k, r)-good.

Introducing notation, for a set $A \subseteq [n]$, let f_A^u be the function computed by $\bigwedge_{i \in A} \phi_i^u$.

▶ **Lemma 4.6.** *Let $A \subseteq [n]$. Then $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f_A^u) = \sum_{i \in A} |\phi_i^u|$.*

Proof. By construction, we have that $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f_A^u) \leq \sum_{i \in A} |\phi_i|$. Suppose for contradiction that $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f_A^u) < \sum_{i \in A} |\phi_i^u|$.

Let $\theta_1 \wedge \cdots \wedge \theta_\ell$ be a minimum-sized $(\mathsf{AND} \circ \mathsf{AC}^0_{d-1})$ formula for f_A^u. By assumption, we have that $\sum_{j=1}^{\ell} |\theta_j| < \sum_{i \in A} |\phi_i^u|$. We can thus replace the $\bigwedge_{i \in A} \phi_i^u$ in the optimal formula for f^u with $\theta_1 \wedge \cdots \wedge \theta_\ell$ and get a smaller formula. In more detail, we have that

$$f^u = f_A^u \wedge \Big(\bigwedge_{i \in [t] \setminus A} \phi_i^u \Big) = (\theta_1 \wedge \cdots \wedge \theta_\ell) \wedge \Big(\bigwedge_{i \in [t] \setminus A} \phi_i^u \Big)$$

which is a formula of size

$$\sum_{j=1}^{\ell} |\theta_j| + \sum_{i \in [t] \setminus A} |\phi_i^u| < \sum_{i \in A} |\phi_i^u| + \sum_{i \in [t] \setminus A} |\phi_i^u| = \sum_{i=1}^{t} |\phi_i^u| = |\phi^u|$$

which contradicts the optimality of ϕ^u for f^u. ◀

For a string $x \in \{0,1\}^n$, let f_x^u be shorthand for $f_{A_x}^u$ where $A_x \subseteq [n]$ is the set of indices where x is one.

▶ **Proposition 4.7.** *Let $x \in \{0,1\}^n$. Then x has weight w if and only if the integer $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f_x^u)$ is (k, rw)-good.*

Proof. By Lemma 4.6 and the fact that $|\phi_i^u|$ is (k, r)-good for all $i \in [n]$, we have that

$$\frac{\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f_x^u)}{p^k} = \frac{\sum_{i \in A_x} |\phi_i^u|}{p^k} \equiv w \cdot r \mod p$$

where $A_x \subseteq [n]$ are bits of x that are ones. The "only if" part of the statement is guaranteed by the fact that $1 \leq r \leq p-1$ has a multiplicative inverse modulo p since p is prime. ◀

▶ **Theorem 4.8.** *Assume n is sufficiently large. Then there is a depth-8 AC^0 truth table oracle circuit C with $O(n^{250})$ wires such that $C^{(\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}}$ computes $\mathsf{MAJORITY}$ on n-bits.*

Proof. It suffices to show that for every $w \in [n]$, there exists a depth-7 AC^0 oracle circuit C_w with $O(n^{249})$ wires such that $C_w^{(\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}}(x) = 1 \iff \mathsf{wt}(x) = w$. Then $\mathsf{MAJORITY}(x) = \bigvee_{w \geq n/2} C_w(x)$.

Fix some $w \in [n]$. The circuit C_w works as follows. On input $x \in \{0,1\}^n$, first check if x is the all zeroes string. If so, then reject. Otherwise, compute the truth table of the direct product function $g_x : \{0,1\}^{\log u} \times \{0,1\}^{\log v} \to \{0,1\}$ given by $g_x(y,z) = f_x^u(y) \wedge f^v(z)$. Compute $s = \mathsf{CC}_{\mathsf{AC}^0_d}(g_x)$ in binary using oracle access to (AC^0_d)-MCSP. Finally accept if the integer s has the property that $s - \mathsf{CC}_{\mathsf{AC}^0_d}(f^v)$ is (k, rw)-good. Reject otherwise.

We now verify this yields a (non-uniform) AC^0 truth table oracle circuit. We can check if x is the all zeroes string with a single OR gate. This requires one level of depth and $O(n)$ wires. Next, realize the jth bit in the truth table of g_x is either zero for all x or equal to

$$f_x^u(j) = \bigvee_{i \in [n] : \phi_i^u(j) = 1} x_i$$

where x_i denotes the ith bit of x. Thus, using non-uniformity, we can compute the truth table of g_x with $O(nuv) = O(nq^{60}) = O(n^{121})$ wires and depth-one. Next, we can compute $s = \mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x)$ in binary with $O(uv\log(uv))$ calls to (AC^0_d)-MCSP using the fact that $\mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x) \leq uv\log(uv)$ by the DNF bound and the fact that

$$\mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x) = s \iff (\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}(g_x, s) = 1 \text{ and } (\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}(g_x, s-1) = 0.$$

This takes at most $\tilde{O}((uv)^2) = O(n^{241})$ wires, an additional three layers of depth, and $2uv\log(uv)$ oracle calls that all do not depend on each other. Finally, the DNF upper bound guarantees that $\mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x) \leq uv\log(uv) \leq n^{61}$, so the length of the integer $s = \mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x)$ in binary is at most $61\log n$. Therefore we can check if s has the property that $s - \mathsf{CC}^0_{\mathsf{AC}^0_d}(f^v)$ is (k, rw)-good using a DNF with at most n^{62} wires and at most an additional two layers of depth. Combining all this yields a AC^0 circuit of depth-7 with at most $O(n^{241})$ wires and no directed path between oracle gates.

Next, we argue for correctness. Clearly, C_w rejects the all zero string, so assume $x \neq 0^n$. By Proposition 4.7, it suffices to show that, for $s = \mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x)$,

$$s - \mathsf{CC}^0_{\mathsf{AC}^0_d}(f^v) = \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^u_x).$$

We confirm that neither f^u_x nor f^v is the constant zero function, so that we can use the direct product theorems in Lemma 2.3.

▷ Claim 4.9. Neither f^u_x nor f^v is the constant zero function.

Proof. If f^v were the constant zero function, then $\mathsf{CC}^0_{\mathsf{AC}^0_d}(h^v) \leq \log v$ by DNF computation which contradicts that

$$\mathsf{CC}^0_{\mathsf{AC}^0_d}(h^v) - \mathsf{CC}^0_{\mathsf{AC}^0_d}(f^v) \geq v^{.25}.$$

Next, let $i \in [n]$ be a bit of x that is not zero. (Recall, we assumed that $x \neq 0^n$.) Then f^u_x has accepts every input that that ϕ^u_i accepts. For contradiction, suppose that ϕ^u_i had no ones. Then we can remove ϕ^u_i from the optimal formula $\phi^u = \phi^u_1 \wedge \ldots \phi^u_t$ for f^u and get a smaller formula for f^u which contradicts the optimality of ϕ^u. ◁

Next we show that the optimal AC^0_d formula for g_x has an AND gate on top.

▷ Claim 4.10. $\mathsf{CC}^0_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(g_x) > \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(g_x)$. Consequently,

$$\mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x) = \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(g_x).$$

Proof. Let $\Delta = \mathsf{CC}^0_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(g_x) - \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(g_x)$. We need to show $\Delta > 0$.

$$\begin{aligned}
\Delta &\geq \mathsf{CC}^0_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(f^v) - \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^v) + \mathsf{CC}^0_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(f^u_x) && \text{(by Lemma 2.3)}\\
&\quad - \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^u_x)\\
&\geq (v)^{.24} + \mathsf{CC}^0_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(f^u_x) - \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^u_x) \geq && \text{(by Proposition 4.2)}\\
&\geq (v)^{.24} - u\log u && \text{(by DNF bound on } f^u_x)\\
&\geq n^{50\cdot.24} - n^{10}\log(n^{10}) && \text{(by definition of } u \text{ and } v)\\
&> 0 && \text{(for sufficiently large } n)
\end{aligned}$$

◁

Using the claim we have that

$$\begin{aligned}
& s - \mathsf{CC}^0_d(f^v) \\
=\ & \mathsf{CC}^0_d(g_x) - \mathsf{CC}^0_d(f^v) && \text{(definition)} \\
=\ & \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(g_x) - \mathsf{CC}^0_d(f^v) && \text{(Claim 4.10)} \\
=\ & \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^u_x) + \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^v) - \mathsf{CC}^0_d(f^v) && \text{(Lemma 2.3)} \\
=\ & \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^u_x) + \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^v) - \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^v) && \text{(Prop 4.2)} \\
=\ & \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^u_x)
\end{aligned}$$

as desired. ◀

References

1. E. Allender, L. Hellerstein, P. McCabe, T. Pitassi, and M. Saks. Minimizing DNF formulas and AC0d circuits given a truth table. In *21st Annual IEEE Conference on Computational Complexity (CCC'06)*, pages 15 pp.–251, July 2006.
2. Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from Random Strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.
3. Eric Allender and Bireswar Das. Zero Knowledge and Circuit Minimization. *Information and Computation*, 256:2–8, 2017.
4. Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. In *Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 54:1–54:14, 2017.
5. Eric Allender, Dhiraj Holden, and Valentine Kabanets. The Minimum Oracle Circuit Size Problem. *Computational Complexity*, 26(2):469–496, 2017.
6. Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *Journal of Computer and System Sciences*, 77(1):14–40, 2011.
7. Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
8. Harry Buhrman and Leen Torenvliet. Randomness is Hard. *SIAM Journal on Computing*, 30:200–1, 2000.
9. Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *Computational Complexity Conference (CCC)*, pages 10:1–10:24, 2016.
10. Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit Lower Bounds for MCSP from Local Pseudorandom Generators. In *46th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 39:1–39:14, 2019.
11. Irit Dinur and David Steurer. Analytical Approach to Parallel Repetition. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 624–633, 2014.
12. Alexander Golovnev, Rahul Ilango, Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, and Avishay Tal. AC0[p] lower bounds against MCSP via the coin problem. *Electronic Colloquium on Computational Complexity*, TR19-018, 2019.
13. G.H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Clarendon Press, Oxford, England, 5th edition, 1979.
14. Shuichi Hirahara. Non-black-box Worst-case to Average-case Reductions within NP. In *Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018.
15. Shuichi Hirahara, Igor C. Oliveira, and Rahul Santhanam. NP-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *Computational Complexity Conference (CCC)*, pages 5:1–5:31, 2018.

16 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Computational Complexity Conference (CCC)*, pages 18:1–18:20, 2016.

17 John Hitchcock and Aduri Pavan. On the NP-Completeness of the Minimum Circuit Size Problem. In *Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, 2015.

18 Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The Power of Natural Properties As Oracles. In *Proceedings of the 33rd Computational Complexity Conference*, CCC '18, pages 7:1–7:20, 2018.

19 Valentine Kabanets and Jin-Yi Cai. Circuit Minimization Problem. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.

20 William J. Masek. Some NP-complete Set Covering Problems. Unpublished Manuscript, 1979.

21 Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*, (Norwich, 1989), London Math. Soc. Lecture Note Ser. 141 . Cambridge Univ. Press, Cambridge, (1989), pp. 48–188.

22 Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 1215–1225, 2019.

23 Cody D. Murray and R. Ryan Williams. On the (Non) NP-hardness of Computing Circuit Complexity. In *Computational Complexity Conference (CCC)*, pages 365–380, 2015.

24 Noam Nisan and Avi Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

25 Igor C. Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. *Electronic Colloquium on Computational Complexity*, TR18-158, 2018.

26 Igor C. Oliveira and Rahul Santhanam. Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *Computational Complexity Conference (CCC)*, pages 18:1–18:49, 2017.

27 Igor C. Oliveira and Rahul Santhanam. Hardness Magnification for Natural Problems. In *Symposium on Foundations of Computer Science (FOCS)*, pages 65–76, 2018.

28 A. A. Razborov. Lower bounds on the size of constant-depth networks over a complete basis with logical addition. *Mathematicheskie Zametki*, 41(4):598–607, 1987.

29 Ronen Shaltiel and Emanuele Viola. Hardness Amplification Proofs Require Majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010.

30 R. Smolensky. On representations by low-degree polynomials. In *FOCS*, pages 130–138, 1993.

31 Boris Trakhtenbrot. A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms. *IEEE Ann. Hist. Comput.*, 6(4):384–400, October 1984.

32 Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, 1983.

A MAJORITY reduces to (AC^0_d)-MCSP when Lipschitzness holds

Our goal in this section is to find a small (AC^0_d)-MCSP-oracle circuit that computes MAJORITY on n-bits for sufficiently large n. We can do this using the techniques of Golovnev et al. [12]. In order to make our proof relatively self-contained, we differ slightly from the presentation in [12]. In particular, our presentation follows a method for computing MAJORITY that is described in Shaltiel and Viola [29].

At a high-level, this procedure works by using the input string to sample a random variable whose circuit complexity spikes depending on the weight of the input and then using Lipschitzness to prove that this spike occurs with high enough probability that we can derandomize using non-uniformity.

Continuing the notation from Section 4, assume that there is an $m \in \{q^{10}, q^{50}\}$ such that $\mathsf{CC}^0_{\mathsf{AC}^0_d}$ is $(m^{.25})$-Lipschitz on inputs of length m.

We define the random variable $T_{p,m} \in \{0,1\}^m$ where each bit in $T_{p,m}$ is independently chosen to be one with probability p and zero with probability $1-p$.

▶ **Lemma A.1.** $\mathbb{E}[\mathsf{CC}^0_{\mathsf{AC}^0_d}(T_{p,m})] = \tilde{O}(pm)$ *if* $p \geq m^{-1/3}$

Proof. By Hoeffding's inequality we have that the probability that $T_{p,m}$ has greater than k ones is at most $\mathsf{exp}(-2\epsilon^2 m)$. Via computation by DNF, if a truth table $T \in \{0,1\}^m$ has at most k ones, $\mathsf{CC}^0_{\mathsf{AC}^0_d}(T) = k \log m = \tilde{O}(k)$. Similarly, we have that $\max\{C(T) : T \in \{0,1\}^m\} = \tilde{O}(m)$. Hence, we get that

$$\mathbb{E}[\mathsf{CC}^0_{\mathsf{AC}^0_d}(T_{p,m})] = \tilde{O}(k) + \tilde{O}(\mathsf{exp}(-2\epsilon^2 m)m) = \tilde{O}(pm + pm\epsilon + \mathsf{exp}(-2\epsilon^2 m)m).$$

If we set $\epsilon = \sqrt{\frac{\ln m}{2m}}$, then we have

$$\mathbb{E}[\mathsf{CC}^0_{\mathsf{AC}^0_d}(T_{p,m})] \leq \tilde{O}(pm + p\sqrt{m}\ln m + 1) \leq \tilde{O}(pm + \sqrt{m}\ln m)$$

Finally, if $p \geq 1/m^{1/3}$, we have $\mathbb{E}[T_{p,m}] = \tilde{O}(pm)$ as desired ◀

We will make use of the following concentration inequality.

▶ **Theorem A.2** (McDiarmid's "bounded differences inequality" [21])**.** *Let* $f : \{0,1\}^n \to \mathbb{R}$ *be* c*-Lipschitz. Let* $X_1, \ldots, X_n$ *be independent random variables with values in* $\{0,1\}$*. Let* $\mu = \mathbb{E}_{X_1,\ldots,X_n}[f(X_1, \ldots, X_n)]$*. Then*

$$\Pr[|f(X_1, \ldots, X_n) - \mu| \geq \epsilon] \leq 2\mathsf{exp}(-\frac{\epsilon^2}{nc^2}).$$

For $t \in \mathbb{N}$ and $w_1 \neq w_2 \in [t]$, we say a Boolean function $f : \{0,1\}^t \to \{0,1\}$ computes $\mathsf{WTDIS}_t[w_1, w_2]$ if $\mathsf{wt}(x) = w_1$ implies $f(x) = 1$ and $\mathsf{wt}(x) = w_2$ implies $f(x) = 0$. (WTDIS is short for weight distinguishing.)

▶ **Theorem A.3.** *If* n *is sufficiently large, then for all* $1 \leq b \leq \sqrt{q}/2$*, there exists a (non-uniform)* NC^0 *oracle circuit* C *with at most* $O(n^{100})$ *wires such that* $C^{(\mathsf{AC}^0_d)\text{-MCSP}}$ *computes* $\mathsf{WTDIS}_q[w_1, w_1 + b]$ *for some* $w_1 \geq \sqrt{q}/2$*. Moreover,* C *has a single gate.*

Proof. For $w \in [q]$, let $p_w = \frac{w}{2q}$. Let w_0 be the largest integer less than $\sqrt{q}$ such that $q - w_0$ is a multiple of b. (Note that $w_0 \geq \sqrt{q} - b \geq \sqrt{q}/2$).

By Lemma A.1, we have that

$$\mathbb{E}[\mathsf{CC}^0_{\mathsf{AC}^0_d}(T_{p_{w_0},m})] = \tilde{O}(\frac{\sqrt{q}}{q}m) = \tilde{O}(m/\sqrt{q}).$$

On the other hand, since $p_q = 1/2$, $T_{p_q,m}$ is just a binary string of length m picked uniformly at random, so the formula size lower bounds of Shannon and Riordan imply

$$\mathbb{E}[\mathsf{CC}^0_{\mathsf{AC}^0_d}(T_{p_q,m})] = \mathop{\mathbb{E}}_{x\in\{0,1\}^m}[C(x)] = \tilde{\Omega}(m)$$

(note that an AC^0_d formula of size s implies an unrestricted formula of size s). Hence, by an averaging argument there exists a $w_1 \geq w_0 \geq \sqrt{q}/2$ such that

$$\mathbb{E}[\mathsf{CC}^0_d(T_{p_{w_1+b},m})] - \mathbb{E}[\mathsf{CC}(T_{p_{w_1},m})] \geq \frac{\tilde{\Omega}(m) - \tilde{O}(m/\sqrt{q})}{q} = \tilde{\Omega}(m/q).$$

Let $t = \frac{\mathbb{E}[T_{p_{w_1+b},m}]+\mathbb{E}[T_{p_{w_1},m}]}{2}$. Then we have that $\mathbb{E}[T_{p_{w_1+b},m}] - t = \tilde{\Omega}(m/q)$ and $t - \mathbb{E}[T_{p_{w_1},m}] = \tilde{\Omega}(m/q)$.

We now outline a probabilistic oracle circuit D that we will later make into a deterministic NC^0 circuit. D takes as input a string $x \in \{0,1\}^n$ and takes as its random "inputs" strings $u_1, \ldots, u_m \in \{0,1\}^{\log q}$ and $v_1, \ldots, v_m \in \{0,1\}$. The reduction then computes the string $y := y_1 \ldots y_m$ where y_i is zero if v_i is zero and y_i is the u_ith bit of x if v_i is one (recall, q is a power of two). D then outputs (AC^0_d)-$\mathsf{MCSP}(y,t)$.

We now argue for correctness with high probability. Realize each y_i is independent with probability $\frac{\mathsf{wt}(x)}{2n}$ of being 1. Hence, y is just the random variable $T_{p_w,m}$ where $w = \mathsf{wt}(x)$. Hence, if $\mathsf{wt}(x) = w_1$, then

$$\Pr[R(x) \neq 1] = \Pr[\mathcal{C}(T_{p_{w_1},m}) > t]$$

Recall that $t - \mathbb{E}[T_{p_{w_1},m}] = \tilde{\Omega}(m/q)$ and, by assumption, $\mathsf{CC}^0_{\mathsf{AC}^0_d}$ on inputs of length m is $(m^{.25})$-Lipschitz, so by Theorem A.2, we have that this probability is bounded by

$$2\mathsf{exp}(-2\frac{\tilde{\Omega}(m^2)}{\tilde{O}(q^2 m^{1.5})}) \leq \mathsf{exp}(-2\frac{\tilde{\Omega}(q^{.5\cdot 10})}{\tilde{O}(q^2)}) = O(\mathsf{exp}(-q^3))$$

using the fact that $m \geq q^{10}$. A similar analysis shows that the probability D errs if $\mathsf{wt}(x) = w_1 + b$ is at most $O(\mathsf{exp}(-q^3))$. This completes the analysis of D.

We now argue that this reduction can be derandomized using non-uniformity. For each input of weight either w_1 or $w_1 + b$, we have shown the fraction of random strings which err on that input is $O(\mathsf{exp}(-q^3))$. Hence, the fraction of random seeds which err on at least one input of weight w_1 or $w_1 + b$ is at most

$$2^q O(\mathsf{exp}(-q^3)) < 1$$

for large enough n. Thus, there exists some fixed $u_1, \ldots, u_m$ and $v_1, \ldots, v_m$ which work on all inputs of length q. Once we are (non-uniformly) given these $u_1, \ldots, u_m$ and $v_1, \ldots, v_m$ which work on all inputs, we can turn D into an NC^0 oracle circuit C which has just a single gate (an oracle gate) whose inputs are the fixed number t and the string y where each bit of y is either a fixed bit of x or zero. This yields a NC^0 oracle circuit with $O(m) = O(q^{50}) = O(n^{100})$ wires. ◀

▶ **Corollary A.4.** *If n is sufficiently large, then for all distinct $w_1, w_2 \in [n]$ there is an NC^0 oracle circuit C with at most two gates and $O(n^{100})$ wires such that $C^{(\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}}$ computes $\mathsf{WTDIS}_n[w_1, w_2]$.*

Proof. Fix some $w_1 \neq w_2$. Without loss of generality assume $w_1 < w_2$ (if this is not the case, then swap the names of w_1 and w_2 in this proof and add a NOT gate to the top of C). Let $b = w_2 - w_1$. Recall q is the least power of two such that $n \leq \sqrt{q}/2$. Note that $q = \Theta(n^2)$ and $b \leq n \leq \sqrt{q}/2$. Theorem A.3 guarantees there exists an NC^0 oracle circuit D of size $O(n^{20})$ such that D^{MCSP} computes $\mathsf{WTDIS}_q[w_3, w_3 + b]$ for some $w_3 \geq \sqrt{q}/2 \geq n$. Finally, let C be the oracle circuit that on input x outputs $D(y)$ where $y = 1^{w_3 - w_1} 0^{q-n-w_3+w_1} x$. The correctness of this output is guaranteed by the fact that $\mathsf{wt}(y) = w_3$ if and only if $\mathsf{wt}(x) = w_1$ and $\mathsf{wt}(y) = w_3 + b$ if and only if $\mathsf{wt}(x) = w_2$. ◀

▶ **Corollary A.5.** *If n is sufficiently large, then there exists a depth-4* AC^0 *truth table oracle circuit C with $O(n^{102})$ wires such that $C^{(\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}}$ computes* MAJORITY *on strings on length n.*

Proof. It suffices to show that, for all $w \in [n]$, one can check if a string $x \in \{0,1\}^n$ has weight w using a depth-3 AC^0 truth-table oracle circuit C_w of size $O(n^{101})$. If one is able to do this, then MAJORITY is computed by $\bigvee_{w \geq n/2} C_w(x)$.

For $w \in [n]$, let $\mathsf{wt}_w : \{0,1\}^n \to \{0,1\}$ be the Boolean function that outputs one if and only if its input is a string of weight w. Now fix some $w \in [n]$. We claim that

$$\mathsf{wt}_w(x) = \bigwedge_{w' \in [n]: w \neq w'} \mathsf{WTDIS}_n[w, w']$$

If x has weight w, then $\mathsf{WTDIS}_n[w,w'](x) = 1$ for all $w' \neq w$, so

$$\mathsf{wt}_{w'}(x) = 1 = \bigwedge_{w' \in [n]: w \neq w'} \mathsf{WTDIS}_n[w, w'].$$

On the other hand, if x has weight $w' \neq w$,then $\mathsf{WTDIS}_n[w,w'](x) = 0$, so

$$\mathsf{wt}_w(x) = 0 = \bigwedge_{w' \in [n]: w \neq w'} \mathsf{WTDIS}_n[w, w'].$$

Finally, by Corollary A.4 we have that $\bigwedge_{w \in [n]: w \neq w'} \mathsf{WTDIS}_n[w', w]$ is computable by a depth-3 AC^0 truth table oracle circuit with $O(n^{101})$ wires. ◀

B $\mathsf{NP} \leq^{\mathsf{P}}_{tt} \mathsf{MOCSP}$ implies $\mathsf{EXP} \neq \mathsf{ZPP}$

The proof of this result follows essentially exactly from Murray and Williams's [23] proof for MCSP. For completeness, we replicate the proof here (even using their words and structure).

▶ **Proposition B.1.** *If* $\mathsf{NP} \leq^{\mathsf{P}}_{tt} \mathsf{MOCSP}$*, then* $\mathsf{EXP} \subseteq \mathsf{P/poly}$ *implies* $\mathsf{EXP} = \mathsf{NEXP}$.

Proof. Assume $\mathsf{NP} \leq^{\mathsf{P}}_{tt} \mathsf{MOCSP}$ and $\mathsf{EXP} \subseteq \mathsf{P/poly}$. Let $L \in \mathsf{NTIME}(2^{n^c})$ for some $c \geq 1$. It suffices to show that $L \in \mathsf{EXP}$.

We pad L into the $L' = \{x01^{2^{|x|^c}} : x \in L\}$. Note that $L' \in \mathsf{NP}$. Hence there is a polynomial-time truth table reduction from L' to MOCSP. Composing the reduction from L to L' with the reduction from L' to MOCSP, we get a $2^{c'n^c}$-time truth table reduction R from n-bit instances of L to $2^{c'n^c}$-bit instances of MOCSP for some constant c'.

Let $Q(x)$ denote the concatenated string of all MOCSP queries produced by R in order on input x. Define the language

$$BITS_Q := \{(x,i) : \text{ the } i\text{th bit of } Q(x) \text{ is } 1\}$$

$BITS_Q$ is clearly in EXP. Since $\mathsf{EXP} \subseteq \mathsf{P/poly}$, for some $d \geq 1$ there is a circuit family C_n of size at most $n^d + d$ computing $BITS_Q$ on n-bit inputs.

Thus, on a given instance x, we have $\mathsf{CC}(Q(x)) \leq s(|x|)$ where $s(|x|) := (|x| + 2c'|x|^c)^d + d$. Therefore, every MOCSP query $(T, s', \mathcal{O})$ produced by the reduction R on input x satisfies

$$\mathsf{CC}^{\mathcal{O}}(T) \leq \mathsf{CC}(T) \leq e \cdot \mathsf{CC}(Q(x)) \leq e \cdot s(|x|)$$

for some constant e since T is a substring of $Q(x)$ (see Lemma 2.2 in [23] for a proof of this substring fact). This leads to the following exponential time algorithm for L:

> On input x, run the exponential-time reduction $R(x)$ by using the following procedure for answering each MOCSP oracle query $(T, s'; \mathcal{O})$. If $s' > e \cdot s(|x|)$, then respond YES to the query. Otherwise, cycle through every oracle circuit E of size at most s'. If $E^{\mathcal{O}}$ computes T, then respond YES. If no such E is found, then respond NO.

It suffices to show the procedure for answering MOCSP oracle queries runs in exponential time. Let $n = |x|$. First, we need to count the number of oracle circuits E on $(\log |T| \leq c'n^c)$-inputs with size at most $s(n)$. The logarithm of the number of oracle circuit of size at most $s(|x|)$ on $(c'n^c)$-inputs with t oracle functions is at most

$$O(s(n)\log(4+t) + s(|x|)\log(c'n^c)\log(s(|x|) + \log(c'n^c))).$$

Since $t \leq 2^{c'n^c}$ and s is polynomial in n, it is easy to see that the number of such circuits E is at most exponential. Second, one can check if an oracle circuit E satisfies $E^{\mathcal{O}}$ computes T in time polynomial in $(|E| + |T| + |\mathcal{O}|)$ and hence exponential in n. As a result, $L \in \mathsf{EXP}$, completing the proof. ◀

▶ **Theorem B.2.** *If* $\mathsf{NP} \leq_{tt}^{\mathsf{P}} \mathsf{MOCSP}$*, then* $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{P/poly}$*. Consequently,* $\mathsf{EXP} \neq \mathsf{ZPP}$*.*

Proof. For contradiction, suppose $\mathsf{NP} \leq_{tt}^{\mathsf{P}} \mathsf{MOCSP}$ and $\mathsf{EXP} = \mathsf{NP} \cap \mathsf{P/poly}$. Then by Proposition B.1 $\mathsf{NEXP} \subseteq \mathsf{EXP} \subseteq \mathsf{NP}$ contradicting the nondeterministic time hierarchy theorem [32]. ◀