Electronic Colloquium on Computational Complexity, Report No. 118 (2019)

Hardness Magnification for all Sparse NP Languages[*]

Lijie Chen
MIT
lijieche@mit.edu

Ce Jin
Tsinghua University
jinc16@mails.tsinghua.edu.cn

R. Ryan Williams
MIT
rrw@mit.edu

September 5, 2019

Abstract

In the Minimum Circuit Size Problem ($\mathsf{MCSP}[s(m)]$), we ask if there is a circuit of size $s(m)$ computing a given truth-table of length $n = 2^m$. Recently, a surprising phenomenon termed as *hardness magnification* by [Oliveira and Santhanam, FOCS 2018] was discovered for $\mathsf{MCSP}[s(m)]$ and the related problem MKtP of computing time-bounded Kolmogorov complexity. In [Oliveira and Santhanam, FOCS 2018], [Oliveira, Pich, and Santhanam, CCC 2019], and [McKay, Murray, and Williams, STOC 2019], it was shown that minor ($n^{1+\varepsilon}$-style) lower bounds for $\mathsf{MCSP}[2^{o(m)}]$ or $\mathsf{MKtP}[2^{o(m)}]$ would imply breakthrough circuit lower bounds such as $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$, $\mathsf{NP} \not\subset \mathsf{NC}^1$, or $\mathsf{EXP} \not\subset \mathsf{P}_{/\mathsf{poly}}$.

We consider the question: What is so special about MCSP and MKtP? Why do they admit this striking phenomenon? One simple property is that all variants of MCSP (and MKtP) considered in prior work are *sparse* languages. For example, $\mathsf{MCSP}[s(m)]$ has $2^{\widetilde{O}(s(m))}$ yes-instances of length $n = 2^m$, so $\mathsf{MCSP}[2^{o(m)}]$ is $2^{n^{o(1)}}$-sparse.

We show that there is a hardness magnification phenomenon for *all* equally-sparse NP languages. Formally, suppose there is an $\varepsilon > 0$ and a language $L \in \mathsf{NP}$ which is $2^{n^{o(1)}}$-sparse, and $L \notin \mathsf{Circuit}[n^{1+\varepsilon}]$. Then NP does not have n^k-size circuits for all k. We prove analogous theorems for De Morgan formulas, B_2-formulas, branching programs, $\mathsf{AC}^0[6]$ and TC^0 circuits, and more: improving the state of the art in NP lower bounds against any of these models by an ε factor in the exponent would already imply NP lower bounds for *all* fixed polynomials. In fact, in our proofs it is not necessary to prove a (say) $n^{1+\varepsilon}$ circuit size lower bound for L: one only has to prove a lower bound against $n^{1+\varepsilon}$-time n^{ε}-space deterministic algorithms with n^{ε} advice bits. Such lower bounds are well-known for *non-sparse* problems.

Building on our techniques, we also show interesting new hardness magnifications for search-MCSP and search-MKtP (where one must output small circuits or short representations of strings), showing consequences such as $\oplus\mathsf{P}$ (or PP, PSPACE, and EXP) is not contained in $\mathsf{P}_{/\mathsf{poly}}$ (or NC^1, $\mathsf{AC}^0[6]$, or branching programs of polynomial size). For instance, if there is an $\varepsilon > 0$ such that search-$\mathsf{MCSP}[2^{\beta m}]$ does not have De Morgan formulas of size $n^{3+\varepsilon}$ for all constants $\beta > 0$, then $\oplus\mathsf{P} \not\subset \mathsf{NC}^1$.

[*]Supported by NSF CCF-1741615 and a Google Faculty Research Award. Portions of this work were completed while L.C. and R.W. were visiting the Simons Institute at UC Berkeley.

ISSN 1433-8092

1 Introduction

Recently there has been a surge of interest in *hardness magnification* [OS18, OPS19, MMW19], a set of results showing how very *weak* lower bounds for certain problems would imply breakthrough separations in computational complexity, such as $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$ or $\mathsf{EXP} \not\subset \mathsf{NC}^1$.

We illustrate the phenomena with three representative results. It has been shown that if there is an $\varepsilon > 0$ such that for all small enough constants $\beta > 0$,

1. If $\mathsf{MCSP}[2^{\beta m}]$ doesn't have $n^{1+\varepsilon}$ size circuits on input of length $n = 2^m$, then $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$ [MMW19].

2. If $\mathsf{Gap\text{-}MKtP}[2^{\beta m}, 2^{\beta m} + O(m)]$ doesn't have $n^{3+\varepsilon}$ size De Morgan formulas on input of length $n = 2^m$, then $\mathsf{EXP} \not\subset \mathsf{NC}^1$ [OPS19].

3. If $\mathsf{Gap\text{-}MKtP}[2^{\beta m}, 2^{\beta m} + O(m)]$ doesn't have $n^{1+\varepsilon}$ size De Morgan formulas with unbounded XORs at the bottom layer, then $\mathsf{EXP} \not\subset \mathsf{NC}^1$ [OPS19].

(For now, it is not important to know *what* $\mathsf{MCSP}[2^{\beta m}]$ and $\mathsf{Gap\text{-}MKtP}[2^{\beta m}, 2^{\beta m} + O(m)]$ are, except that they are problems in NP and EXP that are widely believed to be very hard to solve.) The above three results show that "minor" lower bounds on parameterized versions of certain NP (or EXP) problems would imply breakthrough lower bounds. Item (1) says that $n^{1+\varepsilon}$ size lower bounds for a certain NP problem implies NP does not have polynomial size circuits; items (2) and (3) say that super-cubic De Morgan formula lower bounds or super-linear De Morgan formula lower bounds (with unbounded XORs at the bottom) for a certain EXP problem implies EXP does not have polynomial-size formulas.

Other examples of similar phenomenon are known, for $n^{1-\varepsilon}$ approximation to CLIQUE [Sri03], low-depth circuit lower bounds for NC^1 [AK10, CT19], sublinear-depth circuit lower bounds for P [LW13], proof complexity [MP17], lower bounds for non-commutative arithmetic circuits [CILM18], and $n^{1+\varepsilon}$-size circuit lower bounds for MrKtP (The Minimum rKt Problem) [Oli19].

An Optimistic Perspective. Such results may suggest intriguing approaches to attacking central separation problems in complexity theory. For instance, [OPS19] (adapting [HS17]) showed $\mathsf{Gap\text{-}MKtP}[2^{\beta m}, 2^{\beta m} + O(m)]$ does not have De Morgan formulas of size $n^{2-\varepsilon}$. If this lower bound could be improved from $n^{2-\varepsilon}$ to $n^{3+\varepsilon}$, then by Item (2) above, major lower bounds in complexity theory would follow.

A concern about Item (2) is that the $n^{3-o(1)}$ formula-size lower bounds have resisted improvement for over 20 years ([Hås98, Tal14, DM18, Bog18]), so new techniques are probably required for a $n^{3+\varepsilon}$-size lower bound. However, it is known that MOD2-Inner-Product satisfies the required lower bound in Item (3), even for $n^{2-\varepsilon}$ size [Tal16]. The challenge then is to extend known those lower bound techniques to problems in EXP, such as Gap-MKtP.

Over the years, complexity theory has developed ways to reason about the limits of lower bound techniques. In particular, the natural proof barrier [RR97, NR04] shows, assuming widely accepted conjectures in cryptography, if a proof technique is strong enough to efficiently prove hardness for a *random function*, then it is unlikely to succeed on circuit classes (such as TC^0) which can compute strong pseudorandom functions. It has been argued that hardness magnification results offer a way to bypass the natural proofs barrier, as the results *only apply to special meta-problems such as* MCSP *and* MKtP. The heuristic argument is that, if a proof is based on hardness magnification, it is unlikely to work for a random function. (More precisely, the proof method would violate the "largeness condition" of natural proofs.)

A Pessimistic Perspective. An alternative perspective is that hardness magnification results indicate "weak" circuit lower bounds are even harder to prove than previously thought. Previously, it was understood that super-polynomial lower bounds are very hard to prove, but there did not appear to be serious obstacles for proving "weak" ($n^{1+\varepsilon}$ or $n^{2+\varepsilon}$ size) lower bounds for various computational models. If we believe that circuit lower bounds such as $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$ are very hard to prove, then hardness magnification suggests there should be other deep reasons why we cannot prove even small lower bounds for MCSP and other variants.

1.1 Our Results

Given the above discussions, a natural question arises: *What is so special about* MCSP *and* MKtP*? Why do they admit such a surprising phenomenon?* This question is well-motivated from both the optimistic and pessimistic perspectives. For the optimist, given the possibility of proving breakthrough complexity separations via hardness magnification, it is of central interest to understand for what classes of functions such a phenomenon is possible, and to explore more possibilities of hardness magnification. For the pessimist, in the heuristic argument that hardness magnification avoids the natural proof barrier [OS18], it is suggested that hardness magnification crucially uses properties of MCSP or MKtP that do not hold for random functions. Therefore, it is interesting to understand what properties of MCSP or MKtP suffice for hardness magnification, to gain a better understanding on how natural proofs may be avoided. This understanding may in turn inspire a new barrier to circuit lower bounds (recall we are assuming one is pessimistic).

Notation. We first introduce some notation to succinctly describe our results. The class $\mathsf{Circuit}[s]$ (a.k.a. $\mathsf{SIZE}[s]$) contains the problems solvable by a family of (fan-in two) circuits of size at most $s(n)$. We also consider constant-depth circuit classes of unbounded fan-in, such as $\mathsf{AC}_d[m][s]$ (circuits of size $s(n)$ and depth d over AND, OR, NOT, and MOD_m gates) and $\mathsf{TC}_d[s]$ (threshold circuits of size $s(n)$ and depth d). We measure the size of $\mathsf{AC}_d[m]$ circuits by number of *gates*, and the size of TC_d circuits by number of *wires*. The class $\mathsf{BP}[s]$ contains problems solvable by a family of (deterministic) branching programs of size at most $s(n)$ (for a definition of branching programs, see [Juk12, Chapter 1.3]).

We consider formulas over the De Morgan basis U_2 (NOT, $\mathsf{AND}(x,y)$, $\mathsf{OR}(x,y)$), the basis of all two-input Boolean functions B_2, and extended U_2-formulas where the leaves may be constants or parities over input bits of arbitrary arity. We denote the corresponding classes for formulas of at most s leaves by $\mathsf{U_2\text{-}Formula}[s]$ (or simply $\mathsf{Formula}[s]$), $\mathsf{B_2\text{-}Formula}[s]$, and $\mathsf{U_2\text{-}Formula\text{-}\oplus}[s]$, respectively.

Usually n refers to the input length to problems such as MCSP or MKtP, and we often identify m with $\log n$. $\mathsf{MCSP}[s(m)]$ asks if a given truth table of length $n = 2^m$ has a circuit of size at most $s(m)$. $\mathsf{MKtP}[p(n)]$ asks if a given string of length n can be printed by a Turing Machine of description length c in at most t steps, such that $c + \log(t) \leq p(n)$.

Hardness Magnification for All Sparse NP Languages. Observe $\mathsf{MCSP}[2^{\beta m}]$ and $\mathsf{MKtP}[2^{\beta m}]$ are *sparse* languages in NP (or EXP) with $n = 2^m$ inputs. For exampe, $\mathsf{MCSP}[s(m)]$ has at most $2^{\widetilde{O}(s(m))}$ yes-instances of length $n = 2^m$, since there are $O(s(m) \log s(m))$ many circuits of size $s(m)$. Our main result shows, surprisingly, the subexponential sparsity of a language is already enough for a strong hardness magnification result! That is, analogous weak circuit lower bounds as in [OS18, OPS19, MMW19] for **any** equally-sparse NP language also imply major separations in circuit complexity.

Theorem 1.1. *Let $\mathcal{C}$ be any complexity class such that $\exists \cdot \mathcal{C} = \mathcal{C}$ (e.g., $\mathcal{C} =$ NP, MA, or AM).*
If there is an $\varepsilon > 0$ and a family of languages $\{L_\beta\}$ (indexed over $\beta \in (0,1)$) such that L_β is a 2^{n^β}-sparse language in $\mathcal{C}$ and for all β:

1. *$L_\beta \notin \mathsf{Circuit}[n^{1+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{Circuit}[n^k]$ for all k.*
2. *$L_\beta \notin \mathsf{U_2\text{-}Formula\text{-}\oplus}[n^{1+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{Formula}[n^k]$ for all k.*
3. *$L_\beta \notin \mathsf{B_2\text{-}Formula}[n^{2+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{Formula}[n^k]$ for all k.*

4. *$L_\beta \notin \mathsf{U}_2\text{-}\mathsf{Formula}[n^{3+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{Formula}[n^k]$ for all k.*

5. *$L_\beta \notin \mathsf{BP}[n^{2+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{BP}[n^k]$ for all k.*

6. *$L_\beta \notin \mathsf{AC}_{d+2}[m][n^{1+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{AC}_d[m][n^k]$ for all k, for all constants d and even integers $m \geq 2$.*

7. *$L_\beta \notin \mathsf{TC}_{d+O(\log 1/\varepsilon)}[n^{1+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{TC}_d[n^k]$ for all k, for all constants d.*

Moreover, the converse of each item above also holds, except for the last two.

Theorem 1.1 says that, if the state-of-the-art NP lower bounds can be improved for these models [Hås98, Neč66, IPS97] by an ε factor in the exponent, for any subexponentially sparse NP language, we would have arbitrary fixed-polynomial NP bounds for these models. (As previously discussed, for U_2-Formula-$\oplus$ we would only have to adapt existing $n^{2-\varepsilon}$ lower bounds [Tal16] to a sufficiently sparse NP language.)

Setting $L_\beta = \mathsf{MCSP}[2^{0.99\beta n}]$ and $\mathcal{C} = \mathsf{NP}$ in Theorem 1.1, we recover a main result of [MMW19], although with a weaker (yet still very strong) consequence $\mathsf{NP} \not\subset \mathsf{Circuit}[n^k]$ for all k. In earlier recent work, [CMMW19] proved a result similar to Item (1) in the above theorem, but with the much weaker consequence $\mathsf{NEXP} \not\subset \mathsf{P}_{/\,\mathrm{poly}}$.

Hardness Magnification against Uniform Algorithms with Small Advice. Given that the best known explicit circuit size lower bound for NP (even E^{NP}) functions is only $(3 + 1/86 - o(1))n$ [FGHK16], it is natural to wonder if we can further weaken the hypothesis "L_β does not have $n^{1+\varepsilon}$-size circuits". We observe that it is in fact enough to prove lower bounds against $n^{1+\varepsilon}$-time n^ε-space deterministic algorithms using n^ε bits of advice (a special case of $n^{1+\varepsilon}$ size circuits), or against $O(n)$-time randomized algorithms (with constant failure probability) using n^ε bits of advice and only $O(\log n)$ random bits.

Theorem 1.2. *Let $\mathcal{C}$ be any complexity class such that $\exists \cdot \mathcal{C} = \mathcal{C}$ (e.g., $\mathcal{C} = \mathsf{NP}$, MA, or AM). If there is an $\varepsilon > 0$ and a family of languages $\{L_\beta\}$ (indexed over $\beta \in (0,1)$) such that L_β is a 2^{n^β}-sparse language in $\mathcal{C}$ and for all β,*

- *L_β is not computable by an $n^{1+\varepsilon}$-time n^ε-space deterministic algorithm with n^ε bits of advice, then $\mathcal{C} \not\subset \mathsf{Circuit}[n^k]$ for all k.*
- *L_β is not computable by an $O(n)$-time randomized algorithm with n^ε bits of advice and $O(\log n)$ random bits, then $\mathcal{C} \not\subset \mathsf{Circuit}[n^k]$ for all k.*

Moreover, the converse of each item above also holds.

An appealing aspect of Theorem 1.2 is that, without the sparsity requirement, such lower bounds *can* be proved easily, by modifying the deterministic time hierarchy theorem (see Appendix A for details).

Theorem 1.3 (Adaptation of [HS65])**.** *For all $\varepsilon \in (0,1)$, there is a $(2^{n^\varepsilon} \cdot n)$-sparse language L in time $n^{1+\varepsilon} \cdot \mathrm{poly}\log(n)$ which is not computable by any $n^{1+\varepsilon}$-time deterministic algorithm with n^ε bits of advice.*

Therefore, if we could only make the language L of Theorem 1.3 sparser (note that we are allowed to use $L \in \mathsf{NP}$, instead of $L \in \mathsf{TIME}[n^{1+\varepsilon} \cdot \mathrm{poly}\log(n)]$ in Theorem 1.3), we would prove significant lower bounds by Theorem 1.2. The sparsity requirement can even be made weaker, if we only want to show a super-linear lower bound for NP (which is still notoriously open [FGHK16]).

Theorem 1.4. *If there is an $\varepsilon \in (0,1)$ and a 2^{n^ε}-sparse language $L \in \mathsf{NP}$ which is not computable by any $\tilde{O}(n)$-time deterministic algorithm with $\tilde{O}(n^\varepsilon)$ bits of advice, then $\mathsf{NP} \not\subset \mathsf{SIZE}(n \cdot \log^c n)$ for all $c \geq 1$.*

As another example of lower bounds for sparse languages, we also observe that time-space trade-off lower bounds for SAT [FLvMV05, Wil07] can be extended to hold for sparse languages. Let $\mathsf{TS}[T(n), S(n)]$ denote the class of languages decidable by $O(T(n))$-time $O(S(n))$-space algorithms.

Theorem 1.5. *For every $c < \phi$, there is a 1-sparse language $L \in \mathsf{coNTIME}[n^{1+o(1)}]$, such that $L \notin \mathsf{TS}[n^c, n^{o(1)}]$. Here $\phi = \frac{1+\sqrt{5}}{2} = 1.618\cdots$ is the golden ratio.*

Theorem 1.5 is surprising as it goes against the intuition that such time-space trade-offs relied crucially on the NP-completeness of SAT: polynomially-sparse languages cannot be NP-complete (or coNP-complete) unless $\mathsf{P} = \mathsf{NP}$ [For79, Mah80]. The proof of Theorem 1.5 is deferred to Appendix B.

1.1.1 Stronger Hardness Magnification for search-MCSP and MKtP

Using similar ideas as the proof of Theorem 1.1, we can show that mild *worst-case* lower bounds for search-MCSP or search-MKtP can imply super-polynomial lower bounds for a variety of well-studied non-uniform models of computation:

Theorem 1.6. *Let $\mathcal{C} \in \{\oplus\mathsf{P}, \mathsf{PP}, \mathsf{PSPACE}\}$, and $m \le s(m) \le 2^{(1-\Omega(1))m}$. Let the input length $n = 2^m$.*

1. *If $\text{search-MCSP}[s(m)] \notin \mathsf{Circuit}[n \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{Circuit}[\mathrm{poly}(n)]$.*
2. *If $\text{search-MCSP}[s(m)] \notin \mathsf{U}_2\text{-}\mathsf{Formula}\text{-}\oplus[n \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{Formula}[\mathrm{poly}(n)]$.*
3. *If $\text{search-MCSP}[s(m)] \notin \mathsf{B}_2\text{-}\mathsf{Formula}[n^2 \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{Formula}[\mathrm{poly}(n)]$.*
4. *If $\text{search-MCSP}[s(m)] \notin \mathsf{U}_2\text{-}\mathsf{Formula}[n^3 \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{Formula}[\mathrm{poly}(n)]$.*
5. *If $\text{search-MCSP}[s(m)] \notin \mathsf{BP}[n^2 \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{BP}[\mathrm{poly}(n)]$.*
6. *If $\text{search-MCSP}[s(m)] \notin \mathsf{AC}_{d+2}[m_*][n \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{AC}_d[m_*][\mathrm{poly}(n)]$, for all constants d and even integers $m_* \ge 2$.*
7. *If there is an $\varepsilon > 0$ such that, for all small enough $\beta > 0$, $\text{search-MCSP}[2^{\beta m}] \notin \mathsf{TC}_{d+O(\log 1/\varepsilon)}[n^{1+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{TC}_d[\mathrm{poly}(n)]$, for all constants d.*

Moreover, all above implications also hold for $\mathcal{C} = \mathsf{EXP}$, with search-MCSP *replaced by* search-MKtP.

Remark 1.7. *Indeed, one can even enforce that the* search-MCSP (search-MKtP) *outputs the lexicographically first circuit (program) in the above theorem. This makes the problem harder, and the corresponding lower bound easier to prove.*

Comparison with Previous Works on MKtP. Oliveira and Santhanam [OS18] show that $n^{1+\varepsilon}$-size lower bounds for *approximating* $\mathsf{MKtP}[n^\beta]$ with additive error $O(\log n)$ (for all small $\beta > 0$) would imply $\mathsf{EXP} \notin \mathsf{P}_{/\mathsf{poly}}$. Oliveira, Pich and Santhanam [OPS19] generalize this connection to other computational models listed in Theorem 1.6.

McKay, Murray and Williams [MMW19] improve some of these results, by showing that the same lower bounds on *exact* search-MKtP problem already yield similar consequences.[1] In particular, Item (1) and (a weaker version of) Items (6) and (7) in our Theorem 1.6 were already proved by [MMW19]. However, their

[1] They also considered the oracle version of search-MKtP, which we do not discuss.

techniques are not fine-grained enough to apply to more restricted models such as BP or $\mathsf{Formula}$, or fixed depth circuits such as $\mathsf{AC}_d[m]$ and TC_d.[2]

Our results on MKtP improve both of them, as they not only apply to all reasonable computational models, including BP, $\mathsf{Formula}$, or various fixed depth circuits, but also establish a connection with *exact* search-MKtP.

Comparison with Previous Works on MCSP. [OPS19] show that $n^{1+\varepsilon}$-size lower bounds for *approximating* $\mathsf{MCSP}[2^{\beta m}]$ with multiplicative error $O(m)$ (for all small $\beta > 0$) would imply $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$. [MMW19] improve that by showing the same lower bounds on *exact* search-$\mathsf{MCSP}[2^{\beta m}]$ suffice (that is, they already proved Item (1) of Theorem 1.6). But these two sets of results on MCSP do not generalize to more restricted computational models such as BP, $\mathsf{Formula}$ or various fixed depth circuits.

Our techniques are fine-grained enough to apply to all reasonable computational models, but with a weaker conclusion such as $\oplus\mathsf{P} \not\subset \mathsf{Formula}[\mathrm{poly}(n)]$, instead of $\mathsf{NP} \not\subset \mathsf{Formula}[\mathrm{poly}(n)]$. Still, such a circuit lower bound would already imply major consequences in complexity theory, given that even $\mathsf{NEXP} \subset \mathsf{Formula}[\mathrm{poly}(n)]$ is still open.

Against Uniform Algorithms with Small Advice. Similar to the case of Theorem 1.2, we also show that it suffices to prove lower bounds against $n^{1+\varepsilon}$-time n^{ε}-space deterministic algorithms with n^{ε} bits of advice, instead of $\mathsf{Circuit}[n^{1+\varepsilon}]$.

Theorem 1.8.

- *Let* $\mathcal{C} \in \{\oplus\mathsf{P}, \mathsf{PP}, \mathsf{PSPACE}\}$, *and* $m \le s(m) \le 2^{(1-\Omega(1))m}$. *If* search-$\mathsf{MCSP}[s(m)]$ *on input length* $n = 2^m$ *is not computable by an* $n \cdot \mathrm{poly}(s(m))$*-time* $\mathrm{poly}(s(m))$*-space deterministic algorithm with* $\mathrm{poly}(s(m))$ *bits of advice, then* $\mathcal{C} \not\subset \mathsf{Circuit}[\mathrm{poly}(n)]$.
- *Let* $\log n \le p(n) \le n^{1-\Omega(1)}$. *If* search-$\mathsf{MKtP}[p(n)]$ *is not computable by an* $n \cdot \mathrm{poly}(p(n))$*-time* $\mathrm{poly}(p(n))$*-space deterministic algorithm with* $\mathrm{poly}(p(n))$ *bits of advice, then* $\mathsf{EXP} \not\subset \mathsf{Circuit}[\mathrm{poly}(n)]$.

1.1.2 Hardness Magnification for Zero-Error Heuristics

We remark that our techniques can also be applied to some other settings considered in prior work [OS18]. In the following we discuss a hardness magnification phenomenon for lower bounds against zero-error heuristics. A (zero-error) average-case algorithm A for a function $f : \{0,1\}^n \to \{0,1\}$ is a deterministic algorithm that always outputs a value in $\{0, 1, ?\}$, such that A is never incorrect and A outputs ? with probability at most $1/n$ over uniform random n-bit inputs. (To implement this output behavior in Boolean circuits, we let the circuit output two bits encoding 0, 1, or ?.)

Theorem 1.9. *Let* $\mathscr{C}$ *be any circuit class (e.g.,* $\mathscr{C}$ *could be* $\mathsf{Circuit}$, $\mathsf{Formula}$, $\mathsf{AC}^0[6]$, *etc.). If there is an* $\varepsilon > 0$ *and a family of languages* $\{L_\beta\}$ *(indexed over* $\beta \in (0,1)$*) such that* L_β *is a* 2^{n^β}*-sparse* NP *language not solvable on average with zero error by* $\mathscr{C}$*-circuits of size* n^ε *for all* β*, then* $\mathsf{NP} \not\subset \mathscr{C}[n^k]$ *for all* k.

Theorem 1.10. *Let* $\mathscr{C}$ *be any circuit class (e.g.,* $\mathscr{C} = \mathsf{Circuit}, \mathsf{Formula}$, *or* $\mathsf{AC}^0[6]$*).*

- *Let* $s(m) \ge m$. *If* $\mathsf{MCSP}[s(m)]$ *on input length* $n = 2^m$ *cannot be solved on average with zero error by* $\mathscr{C}[\mathrm{poly}(s(m))]$*, then* $\mathsf{NP} \not\subset \mathscr{C}[\mathrm{poly}(n)]$.
- *Let* $p(n) \ge \log n$. *If* $\mathsf{MKtP}[p(n)]$ *cannot be solved on average with zero error by* $\mathscr{C}[\mathrm{poly}(p(n))]$*, then* $\mathsf{EXP} \not\subset \mathscr{C}[\mathrm{poly}(n)]$.

[2] Roughly speaking, their techniques seem to inherently induce at least a multiplicative constant factor increase in the depth, so these techniques seem incapable of proving Items (6) or (7) in Theorem 1.6.

Comparison with [OS18]. Oliveira and Santhanam [OS18] show that if for some $k \geq 1, c > 0$, $\mathsf{MCSP}[m^k]$ on input length $n = 2^m$ is not solvable on average by $\mathsf{AC}^0[2^{cm}]$ circuits, then $\mathsf{NP} \not\subset \mathsf{NC}^1$. Our Theorem 1.10 implies the same consequence, while only assuming the average-case lower bound against $\mathsf{Formula}[\mathrm{poly}(m)]$, which is much weaker than $\mathsf{AC}^0[2^{cm}]$. Oliveira and Santhanam also show that if for some $k \geq 1, c > 0$ $\mathsf{MCSP}[m^k]$ is not solvable on average by $\mathsf{Circuit}[2^{cm}]$, then $\mathsf{NP} \not\subset \mathsf{Circuit}[2^{n^{o(1)}}]$. We did not state a theorem comparable to this result. However, our proof of Theorem 1.10 can easily show (just plug in different parameters) that the same consequence $\mathsf{NP} \not\subset \mathsf{Circuit}[2^{n^{o(1)}}]$ still follows even if we only assume average-case lower bound against $\mathsf{Circuit}[2^{m^{o(1)}}]$.

1.2 Intuition

In this section we give a high-level description of our proof techniques.

Kernelization. The key component of our technique is a *kernelization* method for sparse NP languages L (Theorem 3.4). In parameterized complexity, kernelization involves taking a problem instance with a "low parameter" and producing a smaller instance (depending on the parameter), where these smaller instances can potentially come from a different problem. Given L we design an auxiliary NP function H, such that L can be efficiently decided on n-bit instances by making n queries to H with short query length (depending logarithmically on the sparsity of L). This is a "Turing-kernelization" from L to H with respect to the log-sparsity parameter.

Let $L \in \mathsf{NP}$ be a 2^t-sparse language where $t = t(n) \leq n^{\beta}$ for a small constant $\beta \in (0, 1)$. Using good error correcting codes [Spi96] and expander-walk sampling, we can design a linear-time computable hash function $M_v : \{0, 1\}^n \to \{0, 1\}^{\Theta(t)}$ with short seed length $|v| = O(t)$, such that a random seed v is likely to make all yes-instances $x' \in L_n$ hash to a distinct value $M_v(x')$.

Fix such a good seed v. Given a hash value h, an NP machine can then unambiguously find the $x' \in L_n$ such that $M_v(x') = h$ (if such an x' exists): nondeterministically guess (x', y), and reject if and only if $M_v(x') \neq h$ or y is not a witness for $x' \in L_n$. Hence we can define an auxiliary NP function $H(h, v, i, w)$, which accepts if and only if there is an $x' \in L_n$ such that $M_v(x') = h$ and $x'_i = w$. Notice that H only takes $O(t)$ bits of input. With a good seed v as advice and oracle access to H, a deterministic algorithm O^H_{det} can compute $L_n(x)$: we define $O^H_{\mathsf{det}}(x)$ to accept if and only if for all $i \in [n]$, $H(M_v(x), v, i, x_i)$ accepts. That is, O_{det} compares its input x bit-by-bit with the unique yes-instance x' (if it exists) that has the same hash value as x, by making n queries to H.[3]

Hardness magnification for all sparse NP languages. Now we are ready to describe the first item of Theorem 1.1. Suppose for contradiction that $\mathsf{NP} \subset \mathsf{Circuit}[n^k]$ for some k. For all small $\varepsilon > 0$, we choose $\beta = \beta(\varepsilon) = \varepsilon/(2k)$. Then given a $2^{n^{\beta}}$-sparse NP language L_n, we define the auxiliary NP function H taking $O(t) \leq O(n^{\beta})$ bits of input, which can be implemented by a circuit of size $O(t^k) \leq O(n^{k\beta}) \leq O(n^{\varepsilon/2})$. To compute $L_n(x)$, the algorithm O_{det} first computes the linear function $M_v(x)$, then returns the AND of n oracle queries to H of length $O(n^{\beta})$. Implementing this with circuitry, we find that L is computable with circuits of size $\tilde{O}(n + n \cdot t^k) < n^{1+\varepsilon}$.

Generalization to other computational models. The above result generalizes easily to the other computational models mentioned (the remaining items in Theorem 1.1), such as U_2-$\mathsf{Formula}$-$\oplus$, TC^0, and $\mathsf{AC}^0[m]$. We only need to observe two properties of the algorithm O_{det}: (1) the hash function $M_v(x)$ that we used is linear over $\mathbb{F}_2$ so it can be implemented with a small number of PARITY gates and wires, and (2) the output

[3] In fact, in the proof of Theorem 3.4 where we construct H, we use only n/t queries, which is more efficient.

of O_{det} is an AND of *non-adaptive* queries to the H-oracle. Thus our circuits for L are ANDs of n copies of H, which take $O(t)$ bits of input from $O(t)$ PARITYs over the n inputs.

Generalization to $n^{1+\varepsilon}$ time deterministic algorithms with n^{ε} advice. The above ideas extend to prove Theorem 1.2, which shows it is enough to prove lower bounds against $n^{1+\varepsilon}$-time deterministic algorithms with n^{ε} bits of advice, which is a special case of $n^{1+\varepsilon}$ size circuits. The idea is simple: hardwire the good seed v and the description of the $O(t^k)$-size circuit for H into the advice string for O_{det}; the advice length is then $\tilde{O}(t+t^k) < n^{\varepsilon}$.

Hardness Magnification for MKtP. Our kernelization technique can also be applied to prove hardness magnification results for MCSP and MKtP (and their search versions, as in Theorem 1.6). We first describe the proof for MKtP case, assuming $\mathsf{EXP} \subset \mathsf{Circuit}[\mathrm{poly}(n)]$. For $\beta \in (0,1)$, consider the language $L = \mathsf{MKtP}[n^{\beta}]$, which is 2^t sparse for $t = \Theta(n^{\beta})$. We modify the implementation of the "kernelization function" $H(h,v,i,w)$ as follows. Instead of guessing a string $x' \in \mathsf{MKtP}[n^{\beta}]$, we (deterministically) enumerate all Turing machines M of description length at most n^{β}, and compute its output x' after simulating M for in $2^{n^{\beta}-|\langle M\rangle|}$ steps. Then $x' \in \mathsf{MKtP}[n^{\beta}]$ (we skip it if M does not stop in time, or $|x'| \neq n$). Then we proceed to verify whether $M_v(x') = h$ and $x'_i = w$.

As before, $O_{\mathsf{det}}(x)$ can decide $\mathsf{MKtP}[n^{\beta}]$ by querying $H(M_v(x), v, i, x_i)$. Moreover, the above implementation of H runs in deterministic $2^{O(t)}$ time, where the constant in the big-O is *independent* of β, so $H \in \mathsf{EXP}$. By assumption, H has $O(t^k)$-size circuits on input of $\Theta(t)$ bits. Then we can similarly obtain $n^{1+\varepsilon}$ size circuits for $\mathsf{MKtP}[n^{\beta}]$.

Hardness Magnification for MCSP. To adapt the above proof to MCSP under the assumption that $\oplus\mathsf{P}$ has polynomial-size circuits, we can similarly modify the implementation of the function H. Instead of guessing a string $x' \in \mathsf{MCSP}[2^{\beta m}]$, we guess a circuit C of size at most $2^{\beta m}$ with m inputs. The truth table of C (call it $tt(C)$) is a yes-instance of $\mathsf{MCSP}[2^{\beta m}]$; we need to check that $M_v(tt(C)) = h$ and $tt(C)_i = w$.

Note that $tt(C)_i = w$ if and only if $C(i) = w$. To check $M_v(tt(C)) = h$ efficiently, we use fully *explicit* ε-biased sets $S_{n,\varepsilon} \subseteq \{0,1\}^n$ (Theorem 2.4) to design a hash family so that each bit of the hash value can be efficiently computed by a $\oplus\mathsf{P}$ oracle with only $\mathrm{poly}(t)$ input bits. In particular, our hash function will compute $\langle tt(C), w_i\rangle$ modulo 2, where $tt(C)$ is the truth table of the guessed circuit C, and $w_i \in S_{n,\varepsilon}$. We can see that $\langle tt(C), w_i\rangle$ can be computed in $\mathrm{poly}(t)$ given inputs C, n, i and a $\oplus\mathsf{P}$ oracle. Therefore, the function H can now be implemented in $\mathsf{NP}^{\oplus\mathsf{P}}$. Toda's theorem [Tod91] implies that $\mathsf{NP}^{\oplus\mathsf{P}} \subset \oplus\mathsf{P}_{/\mathsf{poly}}$; together with $\oplus\mathsf{P} \subset \mathsf{P}_{/\mathsf{poly}}$, we conclude $\mathsf{NP}^{\oplus\mathsf{P}} \subset \mathsf{P}_{/\mathsf{poly}}$. Thus the function H has polynomial-size circuits under the hypothesis, and we can complete the argument as before.

Hardness Magnification Against TC^0 Circuits. To get our results for TC^0 circuits (the last item of Theorem 1.1 and Theorem 1.6), we make use of the recent construction of error correcting codes computable by uniform and extremely sparse TC^0 circuits [CT19].

Oliveira, Pich, and Santhanam [OPS19] also apply sparse TC^0-computable error correcting codes to prove hardness magnification results for TC^0. Their argument requires the error correcting codes to have an efficient decoder. As it is unclear how to decode the codes constructed in [CT19], one can only use a construction in [Tel18] with a worse dependence on depth in their argument (they achieve $O(1/\varepsilon)$ instead of $O(\log \varepsilon^{-1})$ in Theorem 1.6). Our argument does not need an efficient decoder at all, so we can apply the better construction from [CT19] to achieve a better magnification.

Moreover, to prove hardness magnification for MCSP against TC^0, we further require that the codes computable in TC^0 are *fully explicit*. We observe that the codes are fully explicit using a $\oplus\mathsf{P}$ oracle (see Theorem 2.7), which is enough for our application.

2 Preliminaries

We use $\tilde{O}(f)$ to denote $O(f \operatorname{poly}\log f)$ throughout the paper. All logarithms are base-2. We say a language L is $f(n)$-sparse if $|L_n| \le f(n)$, where $L_n = L \cap \{0,1\}^n$. We assume knowledge of basic complexity theory (see [AB09, Gol08] for excellent references). We assume a RAM model when we describe algorithms.

For a complexity class $\mathcal{C}$, we define $\exists \cdot \mathcal{C}$ to be the class of languages L such that there is a relation $R \in \mathcal{C}$ so that for all strings x, $x \in L$ if and only if there is a $\operatorname{poly}(|x|)$-length y such that $R(x, y)$ accepts. So for example, $\exists \cdot \mathsf{NP} = \mathsf{NP}$, $\exists \cdot \mathsf{MA} = \mathsf{MA}$, and $\exists \cdot \mathsf{coNP} = \Sigma_2\mathsf{P}$.

2.1 MCSP and MKtP

The Minimum Circuit Size Problem (MCSP) and its variant Minimum Kt Complexity Problem (MKtP, [Lev84]) are studied in this paper. We first recall their definitions.

Definition 2.1 (MCSP)**.** Let $s : \mathbb{N} \to \mathbb{N}$ satisfy $s(m) \ge m$ for all m.
Problem: $\mathsf{MCSP}[s(m)]$.
Input: A function $f : \{0,1\}^m \to \{0,1\}$, presented as a truth table of $n = 2^m$ bits.
Decide: Does f have a (fan-in two) Boolean circuit C of size at most $s(m)$?

Definition 2.2 (MKtP)**.** Let $p : \mathbb{N} \to \mathbb{N}$ satisfy $p(n) \ge \log n$ for all n.
Problem: $\mathsf{MKtP}[p(n)]$.
Input: A string $x \in \{0,1\}^n$.
Decide: Is there a Turing machine M of description length c that prints x in at most t steps, where $c + \log(t) \le p(n)$?

We can also define the search versions of these two problems, search-MCSP and search-MKtP, where a witness circuit C (or Turing Machine M) should be output when the answer is YES.

2.2 ε-Biased Sets and Error Correcting Codes

Constructions of ε-biased sets are used several times in our hardness magnification constructions. We first introduce ε-biased sets, and then state a fully explicit construction from [AGHP90].

Definition 2.3. Let $\varepsilon \in (0, 1/2)$. A set $S \subseteq \{0,1\}^n$ is ε-biased if for all non-zero $v \in \{0,1\}^n$,

$$\Pr_{w \in S}[\langle v, w\rangle = 0 \text{ over } \mathbb{F}_2] \in (1/2 - \varepsilon, 1/2 + \varepsilon).$$

Theorem 2.4 ([AGHP90])**.** *For every positive n and $\varepsilon \in (0, 1/2)$, there is an ε-biased set $S_{n,\varepsilon} \subseteq \{0,1\}^n$ of cardinality $\widetilde{O}(n^2/\varepsilon^2)$, such that, given inputs $n, \varepsilon, i \in [|S_{n,\varepsilon}|]$ and $j \in [n]$, one can compute the j-th bit of the i-th vector from $S_{n,\varepsilon}$ in $\operatorname{poly}(\log n, \log 1/\varepsilon)$ time.*

We also need linear-time computable error correcting codes [Spi96, GI02].

Theorem 2.5 ([Spi96, GI02])**.** *There is a linear error correcting code E (i.e., E is a linear function over $\mathbb{F}_2$) with constant rate and constant minimum relative distance, such that E can be computed in $O(n)$ time and by an $O(n)$-size circuit family.*[4]

We also need error correcting codes computable by uniform and extremely sparse TC^0 circuits from [CT19] (which builds on [GHK$^+$13] and [CRVW02]).

[4] The code in [Spi96] requires a polynomial-time preprocessing stage, which is removed in the later work [GI02].

Theorem 2.6 (Uniform Sparse TC^0 Circuits For "Almost-Good" Codes [CT19])**.** *For some universal constants $\rho > 0$ and $c_0 > 1$, and for all constant $d \geq 2$, there is a deterministic polynomial-time algorithm A_d that on the input 1^n (where n is a sufficiently large power of two), outputs a TC^0 circuit C_n such that:*

1. *C_n computes the encoding function of a linear code $\{0,1\}^n \to \{0,1\}^{\hat{n}}$ with constant relative distance $\rho > 0$, where $\hat{n} = n \cdot \exp(\operatorname{poly}\log\log(n))$.*
2. *C_n has depth $d + 2$ and at most $n^{1+c_0\cdot\phi^{-d}+o(1)}$ wires, where $\phi = \frac{1+\sqrt{5}}{2}$.*

We also observe that the above construction is *fully explicit* with a $\oplus\mathsf{P}$ oracle.

Theorem 2.7. *The linear code E of Theorem 2.6 is* $\oplus\mathsf{P}$-fully explicit*, in that for all i and vectors $v_i \in \mathbb{F}_2^n$ such that $E(x)_i := \langle x, v_i \rangle$, given input $n, i \in [\hat{n}]$ and $j \in [n]$, the j-th bit of v_i can be computed in* $\operatorname{poly}(\log n)$ *time with an oracle in* $\oplus\mathsf{P}$.

Remark 2.8. *A crucial component of the proof of Theorem 2.6 in [CT19] is the lossless (bipartite) expander construction of [CRVW02], which is* fully explicit*. That is, its neighbor function $E : [N] \times [D] \to [M]$ (map (u, i) to the i-th neighbor of the vertex u on the left) is computable in* $\operatorname{poly}\log(N)$ *time. Given this observation, it is easy to verify the construction in Section 5.1 of [CT19] is $\oplus\mathsf{P}$-fully explicit.*

2.3 Expander Graphs

Expander graphs and their strongly explicitly constructions are also served as important tools in our paper. We first recall their definition, and state a strongly explicit construction from [RVW02].

Definition 2.9 (Expander Graphs)**.** An n-vertex undirected graph G is an (n, d, λ)-expander graph if G is d-regular and $\lambda(G) \leq \lambda$, where $\lambda(G)$ denotes the second largest eigenvalue (in absolute value) of the normalized adjacency matrix of G (i.e., the adjacency matrix of G divided by d).

For constants $d \in \mathbb{N}$ and $\lambda < 1$, a family of graphs $\{G_n\}_{n\in\mathbb{N}}$ is a (λ, d)*-expander graph family* if for every n, G_n is an (n, d, λ)-expander graph.

Theorem 2.10 (Strongly Explicit Expanders, e.g., [GG81])**.** *There exists a (λ, d)-expander graph family $\{G_n\}$ for some constants $d \in \mathbb{N}$ and $\lambda < 1$, such that there is an algorithm that on inputs $n, v \in [n], i \in [d]$ outputs the i-th neighbor of v in graph G_n in* $\operatorname{poly}(\log n)$ *time.*

We also need the following expander Chernoff bound, which shows that a random walk on an expander graph behaves similarly to a sequence of i.i.d. random vertices.

Theorem 2.11 (Expander Chernoff Bound, [Gil98])**.** *Let G be an (n, d, λ)-expander graph. Let $f : [n] \to \{0,1\}$ be a function on the vertices of G, and $\mu = \mathbb{E}_{v\in[n]} f(v)$. Let $v_1, v_2, \ldots, v_t$ be a random walk on G (where v_1 is uniformly chosen). Then for $\delta > 0$,*

$$\Pr_{v_1,\ldots,v_t}\left[\frac{1}{t}\sum_{i=1}^{t} f(v_i) < \mu - \delta\right] \leq e^{-(1-\lambda)\delta^2 t/4}.$$

3 Kernelization of Sparse Languages

In this section we present kernelization constructions for generic sparse languages.

3.1 The Hash Family

We begin by describing the hash families we use for kernelizing sparse languages. In short, we construct a family of k-perfect linear hash functions that is very efficiently computable for a wide range of values of k. The problem of constructing k-perfect hash families has a long history (see for example [SS90, FKS84, AYZ95, NSS95]). For example, in their famous FPT paper on color-coding, Alon, Yuster, and Zwick [AYZ95] derandomize the color-coding method by showing there are k-perfect hash families from $\{0,1\}^n$ to $\{0,1\}^{\log(k)}$, where each function in the family takes $O(k)+\log n$ bits to specify, and each function can be efficiently evaluated given its specification. For our purposes, we want our functions to also be *linear* over $\mathbb{F}_2$, so that their formula complexity is guaranteed to be low. One can easily obtain a family of k-perfect linear hash functions by choosing random linear transformations from $\{0,1\}^n$ to $\{0,1\}^{2\log(k)+O(1)}$. However, uniform random matrices take too many bits to describe; we achieve a short seed length using standard tools from pseudorandomness: error-correcting codes, ε-biased sets, and expander walks. These constructions are likely folklore; we did not find an explicit reference.

Definition 3.1 (k-Perfect Linear Hash Family)**.** Let $n \in \mathbb{N}$, and let k be an integer in $[n, 2^n]$. We say that a family of functions $\{M_v : \{0,1\}^n \to \{0,1\}^t\}$ is a *k-perfect linear hash family* if:

1. (Succinct) Each M_v is specified by an $O(\log k)$-bit seed v, and $t \le O(\log k)$.
2. (Spreading) For all $S \subseteq \{0,1\}^n$ with $|S| \le k$, there is a seed v such that $|\{M_v(x) \mid x \in S\}| = |S|$. We say such a v is a *good seed for S*.
3. (Linear) For all seeds v and $i \in [t]$, the i-th bit of $M_v(x)$ can be expressed as $(\langle x, w_{v,i}\rangle \bmod 2)$ for some $w_{v,i} \in \{0,1\}^n$.

We first show that such hash functions can be constructed in linear time and with linear-size circuits.

Lemma 3.2 (Linear-Time Construction of Perfect Linear Hash Functions)**.** *There is a $d > 0$ such that for all $k(n) \in [n, 2^{n/\log^d n}]$, there is a k-perfect linear hash family $\{M_v\}$ (for every n), an $O(n)$-time algorithm A, and an $O(n)$-size circuit family $\{C_n\}$, such that for all $(v,x) \in \{0,1\}^{O(\log k(n))+n}$, $A(v,x) = M_v(x) = C_n(v,x)$.*

Proof of Lemma 3.2. Let $E : \{0,1\}^n \to \{0,1\}^{n/c_r}$ be the linear error correcting code from Theorem 2.5, where the constant $c_r \in (0,1)$ is the rate of the code. For all distinct $x, y \in S$ (i.e., where $x \ne y$), we have

$$\Pr_{v_0 \in [n/c_r]}[E(x)_{v_0} \ne E(y)_{v_0}] \ge c_d,$$

where constant $c_d \in (0,1)$ is the relative distance of E.

Let G be the strongly explicit expander graph on the vertex set $[n/c_r]$ (with constant parameters $\lambda < 1, d \in \mathbb{N}$) from Theorem 2.10. Let $t = c \cdot \log(k)$ for a constant $c > 0$ to be specified later. Given the vector of t elements $v = (v_1, v_2, \ldots, v_t) \in [n/c_r]^t$, we define

$$M_v(x) := (E(x)_{v_1}, E(x)_{v_2}, \ldots, E(x)_{v_t}) \in \{0,1\}^t.$$

For all distinct $x, y \in S$, by the Expander Chernoff Bound (Theorem 2.11), if $v = (v_1, v_2, \ldots, v_t)$ is a random walk on G, then

$$\begin{aligned}\Pr_v[M_v(x) = M_v(y)] &\le \Pr_v\left[\frac{1}{t}\sum_{i=1}^{t}[E(x)_{v_i} \ne E(y)_{v_i}] < c_d/2\right] \\ &\le e^{-(1-\lambda)(c_d/2)^2 t/4}.\end{aligned}$$

Now let $S \subseteq \{0,1\}^n$ satisfy $|S| \le k$. Choosing $t = c \cdot \log(k)$ for a sufficiently large constant $c > 0$, and union-bounding over all $\binom{k}{2}$ distinct pairs (x, y), we have

$$\Pr_v[\text{for every pair of distinct } x, y \in S,\, M_v(x) \neq M_v(y)] \ge 0.99.$$

Note that a walk on G of length t can be specified by $O(\log n + t) = O(t)$ bits. Hence we have a $O(t)$-bit good seed for S. We can recover the vertices $v_1, \ldots, v_t$ from these $O(t)$ bits, by computing all relevant edges of the expander graph in time (and circuit size)

$$t \cdot \operatorname{poly}\log(n) \le O(\log k) \operatorname{poly}\log(n) \le O(n/\log^d n) \operatorname{poly}\log(n).$$

Setting d to be sufficiently large, this time bound is $O(n)$.

Finally, the computation of M_v involves computing E on the input x, and extracting a subset of t bits of $E(x)$. We already argued that the subset of t bits can be computed in $O(n)$ time and size; the computation of E can be also done in $O(n)$ time and size, by Theorem 2.5. This completes the proof. □

Next, we show that the same family of hash functions can also be implemented in a fully explicit way, where each bit of the vectors defining the hash functions can be computed in polylog time.

Lemma 3.3 (Fully Explicit Perfect Linear Hash Functions)**.** *There is a $d > 0$ such that for all $k(n)$, there is a k-perfect linear hash family $\{M_v\}$ (for every n) such that for all $j \in [n]$, the j-th bit of the vector $w_{v,i}$ defining M_v can be computed in $\log(k) \cdot \operatorname{poly}\log(n)$ time and space. It follows that $M_v(x)$ can be computed in $n \cdot \log(k) \cdot \operatorname{poly}\log(n)$ time and $\log(k) \cdot \operatorname{poly}\log(n)$ space on any input (v, x).*

Proof of Lemma 3.3. Let $W = \{w_1, \ldots, w_\ell\} \subseteq \{0,1\}^n$ be the fully explicit 0.1-biased set constructed in Theorem 2.4, where $\ell = \operatorname{poly}(n)$. For all distinct $x, y \in \{0,1\}^n$, since W is 0.1-biased we have

$$\Pr_{v_0 \in [\ell]}[\langle x, w_{v_0}\rangle \neq \langle y, w_{v_0}\rangle] \ge 0.4.$$

Given a vector of $t = O(\log k)$ elements $v = (v_1, v_2, \ldots, v_t) \in [\ell]^t$, we define

$$M_v(x) := (\langle x, w_{v_1}\rangle, \langle x, w_{v_2}\rangle, \ldots, \langle x, w_{v_t}\rangle) \in \{0,1\}^t.$$

Now let $S \subseteq \{0,1\}^n$ satisfy $|S| \le k$. By an analogous expander-walk argument as in the proof of Lemma 3.2, we conclude there is an $O(t)$-bit good seed v for S.

To compute the j-th bit of w_{v_i}, we first compute v_i from the $O(t)$-bit seed in $t \cdot \operatorname{poly}(\log n) \le \log(k) \cdot \operatorname{poly}(\log n)$ time. Since W is fully explicit, we can then compute the j-th bit of w_{v_k} in $\operatorname{poly}(\log n)$ time. □

3.2 Kernelization

Analogous to the definition of sparsity for languages, define the sparsity of a function $f : \{0,1\}^n \to \{0,1\}$ to be $|f^{-1}(1)|$. The main theorem of this section is that every sparse f has a corresponding "kernel" function H taking only $O(\log |f^{-1}(1)|)$ inputs which can be used to efficiently compute f.

Theorem 3.4. *Let $f : \{0,1\}^n \to \{0,1\}$ be any function of sparsity S_{sparse}, where $\log n \le \log(S_{\mathsf{sparse}}) \le n^{1-\Omega(1)}$. There is a function $H : \{0,1\}^{\Theta(\log S_{\mathsf{sparse}})} \to \{0,1\}$ computable in nondeterministic $O(n)$ time with one oracle query to f, and a deterministic $O(n)$-time algorithm O_{det} computing f with $O(\log S_{\mathsf{sparse}})$ bits of advice, making $O(n/\log S_{\mathsf{sparse}})$ non-adaptive queries to H. O_{det} can also be computed by a $O(n)$-size circuit with $O(n/\log S_{\mathsf{sparse}})$ H-oracle gates.*

Proof. Let $L_n = f^{-1}(1) \subseteq \{0,1\}^n$ denote the language determined by f. Applying Lemma 3.2, let $M_v(x) : \{0,1\}^n \to \{0,1\}^t$ be a hash function with length-t seed v, for some $t = \Theta(\log S_{\mathsf{sparse}})$, where there exists a good seed v such that M_v is injective on L_n (all values $M_v(x)$ are distinct, over all $x \in L_n$).

The Function H. Now we define the function H by giving an algorithm for it.

Algorithm 1: The algorithm H.

```
Given an input $h \in \{0,1\}^t$, a seed $v \in \{0,1\}^t$, an integer $i \in [n]$, and a bitstring $w \in \{0,1\}^t$:
Guess an $x \in \{0,1\}^n$. Set $x_j = 0$ for $j = n+1, \ldots, n+t$.
If $f(x) = 1$, $M_v(x) = h$, and $x_{[i\ldots(i+t-1)]} = w$ then accept else reject.
```

Note that guessing x and computing $M_v(x)$ can be done in $O(n)$ time. Therefore Algorithm 1 runs in nondeterministic $O(n)$ time with oracle f. By definition, it queries f at most once on any computation path. Intuitively, H on (h, v, i, w) accepts if and only if there is an x such that $f(x) = 1$, the hash function indexed by v hashes x to h, and the t-bit block $x_{[i\ldots(i+t-1)]}$ equals the t-bit string w.

The Deterministic Oracle Algorithm O_{det} with Advice. Now we describe the deterministic oracle algorithm O_{det} which computes f given oracle H.

Algorithm 2: The algorithm O_{det}.

```
Given an input $x \in \{0,1\}^n$, and given a good $v \in \{0,1\}^t$ for $L_n$ as advice:
Compute $h = M_v(x)$.
Accept if for all $i \in \{1, 1+t, 1+2t, \ldots\} \cap [n]$, $H(h, v, i, x_{[i\ldots(i+t-1)]}) = 1$. Reject otherwise.
```

Note O_{det} makes $O(n/t)$ queries to H and each query has length $O(t)$, so its total running time is $O(n)$. We can also hardwire the advice and compute O_{det} by an $O(n)$-size circuit with H-oracle gates.

If $f(x) = 1$, then every call to H accepts (by guessing x). If $f(x) = 0$, then by assumption on the advice v, there is at most one x' such that $f(x') = 1$ and $M_v(x) = M_v(x')$, but we must have $x_{[i\ldots(i+t-1)]} \neq x'_{[i\ldots(i+t-1)]}$ for some i, so at least one of the H-oracle queries rejects. □

If we simply replace the hash function $M_v(x)$ in the above proof by the fully explicit version from Lemma 3.3, then the resulting implementation of O_{det} runs in $n \cdot \mathrm{poly}(t)$ time and $\mathrm{poly}(t)$ space.

Corollary 3.5. *The oracle algorithm O_{det} in Theorem 3.4 can be implemented to run in $O(n \cdot \mathrm{poly}\log S_{\mathsf{sparse}})$ time and $O(\mathrm{poly}\log S_{\mathsf{sparse}})$ space with an oracle for H.*

By a simple modification we obtain a randomized version of Theorem 3.4. It will be used later for proving Theorem 1.2.

Theorem 3.6. *Let $f : \{0,1\}^n \to \{0,1\}$ be a function of sparsity S_{sparse}, where $\log n \leq \log(S_{\mathsf{sparse}}) \leq n^{1-\Omega(1)}$. There is a function $H : \{0,1\}^{\Theta(\log S_{\mathsf{sparse}})} \to \{0,1\}$, computable in nondeterministic $O(n)$ time with one oracle query to f, and a randomized $O(n)$-time algorithm O_{rand} computing f (with one-sided error probability 0.01) using $O(\log n)$ random bits, $O(\log S_{\mathsf{sparse}})$ bits of advice, and only $O(1)$ non-adaptive oracle queries to H.*

Proof sketch. We slightly modify the algorithms O_{det} and H from the proof of Theorem 3.4. Let $E : \{0,1\}^n \to \{0,1\}^{n/c_r}$ be the error correcting code from Theorem 2.5. Modify the function H so that it takes another two input parameters $i' \in [n/c_r]$, $w' \in \{0,1\}$, and have H reject if $E(x)_{i'} \neq w'$, i.e., the i'-th bit of the code of x is not equal to w'.

This modified function H is still computable in nondeterministic $O(n)$ time (with oracle f), since $E(x)$ can be computed in linear time.

Our new algorithm O_{rand} works as follows on an input x. Instead of querying H for $O(n/t)$ times, we only make $O(1)$ (non-adaptive) queries. For each query, we pick a random $i' \in [n/c_r]$, and send i', $E_v(x)_{i'}$ as the two new parameters to the H oracle.

If $f(x) = 1$, then $O_{\mathsf{rand}}(x)$ accepts with probability 1.

If $f(x) = 0$, since E has constant relative distance, every query to H is rejected with constant probability. Repeating $O(1)$ times (non-adaptively), the error probability is reduced to 0.01. □

4 Hardness Magnification for All Sparse NP Languages

In this section, we prove our hardness magnification results for all sparse NP languages against various computational models.

4.1 Hardness Magnification for Non-Uniform Models

We first prove Theorem 1.1 below, which is based on our kernelization results in Section 3.

Reminder of Theorem 1.1. *Let $\mathcal{C}$ be any complexity class such that $\exists \cdot \mathcal{C} = \mathcal{C}$ (e.g., $\mathcal{C} = \mathsf{NP}$, MA, or AM). If there is an $\varepsilon > 0$ and a family of languages $\{L_\beta\}$ (indexed over $\beta \in (0,1)$) such that L_β is a 2^{n^β}-sparse language in $\mathcal{C}$ and for all β:*

1. *$L_\beta \notin \mathsf{Circuit}[n^{1+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{Circuit}[n^k]$ for all k.*
2. *$L_\beta \notin \mathsf{U}_2\text{-}\mathsf{Formula}\text{-}\oplus[n^{1+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{Formula}[n^k]$ for all k.*
3. *$L_\beta \notin \mathsf{B}_2\text{-}\mathsf{Formula}[n^{2+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{Formula}[n^k]$ for all k.*
4. *$L_\beta \notin \mathsf{U}_2\text{-}\mathsf{Formula}[n^{3+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{Formula}[n^k]$ for all k.*
5. *$L_\beta \notin \mathsf{BP}[n^{2+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{BP}[n^k]$ for all k.*
6. *$L_\beta \notin \mathsf{AC}_{d+2}[m][n^{1+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{AC}_d[m][n^k]$ for all k, for all constants d and even integers $m \geq 2$.*
7. *$L_\beta \notin \mathsf{TC}_{d+O(\log 1/\varepsilon)}[n^{1+\varepsilon}]$, then $\mathcal{C} \not\subset \mathsf{TC}_d[n^k]$ for all k, for all constants d.*

Moreover, the converse of each item above also holds, except for the last two.

Proof. We will only prove the theorem for the case when $\mathcal{C} = \mathsf{NP}$. But it is easy to see that we only use the property that $\exists \cdot \mathsf{NP} = \mathsf{NP}$.

The $\mathsf{Circuit}$ case. We first prove the case for Circuit, and then argue it for other computational models.

The $\Leftarrow$ direction can be proved by a simple padding argument. Set $\varepsilon = 1$. For every $\beta \in (0,1)$, by assumption, there is a language $L_\beta \in \mathsf{NP}$ without $n^{2/\beta}$ size circuits. Then we can define another language $L'_\beta \in \mathsf{NP}$ as $\{x10^{(|x|^{1/\beta}-|x|-1)} \mid x \in L_\beta\}$. Clearly, L'_β does not have $n^{1+\varepsilon} = n^2$ size circuits, and it is a 2^{n^β}-sparse language. This padding argument also works for other computational models (except the last two items in the theorem statement).

The $\Rightarrow$ direction is the hardness magnification part. Suppose $\mathsf{NP} \subset \mathsf{Circuit}[n^k]$ for a constant k. We wish to show for any $\varepsilon > 0$, there is a $\beta > 0$ such that every 2^{n^β}-sparse NP language has $n^{1+\varepsilon}$-size circuits.

Let $\varepsilon > 0$, and $\beta = \beta(\varepsilon)$ to be specified later. Let $L \in \mathsf{NP}$ be a 2^{n^β}-sparse language and $t = t(n) = n^\beta$. Applying Theorem 3.4, we define the "kernel" function H_t on $\Theta(t)$ input bits; by Algorithm 1 it is decided in nondeterministic $O(n) = \mathrm{poly}(t)$ time using one L_n-oracle query. Note that Algorithm 1 actually returns the answer of the L_n-oracle, provided that the other two conditions in Line 3 (Algorithm 1) hold. Hence, H_t is in $\exists \cdot \mathsf{NP} = \mathsf{NP}$, and has an $O(t^k)$-size circuit. Recall that $L_n = O_{\mathsf{det}}^{H_t}$ (in an input to H_t, we also use $O(\log n)$ bits to specify the input length n), where O_{det} can be implemented by an $O(n)$-size circuit with $O(n/t)$ H_t-oracle gates. We replace the oracle gates by a small circuit for H_t, and obtain a circuit for L_n of size $O((n/t) \cdot t^k + n)$. Setting $\beta = \varepsilon/k$ completes the proof.

The $\mathsf{AC}^0[m]$ case. The proof is very similar to the $\mathsf{Circuit}$ case. Suppose $\mathsf{NP} \subset \mathsf{AC}_d[m][n^k]$ for a constant k and an even integer m. To implement O_{det}, recall that the hash function M_v used by O_{det} is linear over $\mathbb{F}_2$, so $M_v(x)$ can be computed by $O(t)$ XOR gates (which can be simulated by MOD_m gates when m is even). The outputs of these gates are fed into $O(n/t)$ H_t oracles in parallel, and then an AND is taken over the outputs of the H_t oracles. Using $O(n/t)$ copies of $O(t^k)$-size $\mathsf{AC}_d[m]$ circuits for $H_t \in \mathsf{NP}$, we obtain an $\mathsf{AC}_{d+2}[m]$ circuit (with XOR gates at the bottom layer and an AND gate at the top) for L_n of size $O((n/t) \cdot t^k)$. Again, setting $\beta = \varepsilon/k$ completes the proof.

The U_2-$\mathsf{Formula}$, B_2-$\mathsf{Formula}$, and U_2-$\mathsf{Formula}$-$\oplus$ cases. The above proof for $\mathsf{AC}^0[m]$ also works for formulas. The differences in the sizes ($n^{1+\varepsilon}$, $n^{2+\varepsilon}$, or $n^{3+\varepsilon}$) lie solely in the complexity of computing the bottom XOR layer with these various models. An XOR gate over at most n input bits can be computed by a U_2-formula of size $O(n^2)$, by a B_2-formula of size $O(n)$, and XOR gates are "free" on the leaves of U_2-$\oplus$ formulas.

The branching program (BP) case. Again the proof is similar. Suppose $\mathsf{NP} \subset \mathsf{BP}[n^k]$. Then H_t has branching programs of size $O(t^k)$. Each bit fed to oracle H_t is either fixed, or an XOR over at most n input bits, each of which can be computed by a branching program of size $O(n)$. Therefore the output of every H_t call can be computed by a branching program of size $O(n \cdot t^k)$ on inputs of length n. Finally, the AND of all $O(n/t)$ H_t calls can be computed by a branching program of size $O(n \cdot t^k \cdot (n/t))$. Again, setting $\beta = \varepsilon/k$ completes the proof.

The threshold circuit (TC^0) case. Our argument is similar to the AC^0 case. Suppose $\mathsf{NP} \subset \mathsf{TC}_d[n^k]$ for a constant k. We replace the error correcting code used in Lemma 3.2 by the code $E : \{0,1\}^n \to \{0,1\}^{\hat{n}}$ from Theorem 2.6, and adjust O_{det} and H in Theorem 3.4 accordingly. Then O_{det} needs to compute $E(x)$ in order to get the hash value $M_v(x)$. By Theorem 2.6, this could be done by a $(d'+2)$-depth TC^0 circuit with $n^{1+c_0\phi^{-d'}+o(1)}$ wires for any constant $d' \ge 2$. Then we feed them into $O(n/t)$ copies of TC_d circuits with t^k wires each, and obtain a $(d+d'+3)$-depth TC^0 circuit for O_{det} with $O((n/t)t^k + n^{1+c_0\phi^{-d'}+o(1)})$ wires. Setting $\beta = \varepsilon/k$ and $d' = O(\log 1/\varepsilon)$ completes the proof. □

4.2 Hardness Magnification for Uniform Algorithms with a Small Amount of Advice

Next we observe that it in fact suffices to consider $n^{1+\varepsilon}$ time deterministic algorithm with n^{ε} bits of advice, instead of $n^{1+\varepsilon}$ circuits in the Item (1) of Theorem 1.1. Formally, we have

Reminder of Theorem 1.2 *Let $\mathcal{C}$ be any complexity class such that $\exists \cdot \mathcal{C} = \mathcal{C}$ (e.g., $\mathcal{C} = \mathsf{NP}$, MA, or AM). If there is an $\varepsilon > 0$ and a family of languages $\{L_\beta\}$ (indexed over $\beta \in (0,1)$) such that L_β is a 2^{n^β}-sparse language in $\mathcal{C}$ and for all β,*

- *L_β is not computable by an $n^{1+\varepsilon}$-time n^{ε}-space deterministic algorithm with n^{ε} bits of advice, then $\mathcal{C} \not\subset \mathsf{Circuit}[n^k]$ for all k.*
- *L_β is not computable by an $O(n)$-time randomized algorithm with n^{ε} bits of advice and $O(\log n)$ random bits, then $\mathcal{C} \not\subset \mathsf{Circuit}[n^k]$ for all k.*

Moreover, the converse of each item above also holds.

Proof. We will only prove our theorem for the case when $\mathcal{C} = \mathsf{NP}$. But it is easy to see that we only use the property that $\exists \cdot \mathsf{NP} = \mathsf{NP}$.

To show the $\Leftarrow$ direction, we apply the same padding argument from the proof of Theorem 1.1, showing that for all $\beta > 0$ there is a 2^{n^β}-sparse NP language L'_β without n^{10}-size circuits. Then L'_β is not computable by an $O(n^2)$-time deterministic algorithm with $O(n)$ bits of advice, otherwise (by a standard translation of algorithms into circuits) L'_β would have $o(n^{10})$-size circuits. Similarly, L'_β is not computable by an $O(n)$-time randomized algorithm with $O(n)$ advice and $O(\log n)$ random bits, since derandomizing it would also lead to $o(n^{10})$-size circuits.

To show the $\Rightarrow$ direction, we prove the contrapositive. Suppose $\mathsf{NP} \subset \mathsf{Circuit}[n^k]$ for some constant k. We need to show that for all $\varepsilon > 0$, there is a $\beta > 0$ such that all 2^{n^β}-sparse NP languages admit an $n^{1+\varepsilon}$-time deterministic algorithm with n^ε bits of advice, as well as an $O(n)$ time randomized algorithm (with constant error probability) with n^ε bits of advice and $O(\log n)$ random bits.

For any $\varepsilon > 0$, let $\beta = \beta(\varepsilon)$ be a constant to be determined later. Let L be a 2^{n^β}-sparse NP language. Let $t = t(n) = \Theta(n^\beta)$. Applying Theorem 3.4 and Corollary 3.5, we can define a "kernel" function $H_t \in \mathsf{NP}$ on $\Theta(t)$ input bits, such that $L_n = O_{\mathsf{det}}^{H_t}$, where O_{det} is an $O(n \cdot t^c)$-time $O(t^c)$-space deterministic algorithm (for some constant $c > 0$) with $O(t)$ bits of advice, making $O(n/t)$ queries to H_t. We hardwire the t^k-size circuit for H_t into advice, so the answer of each query can be computed in $\tilde{O}(t^k)$ time and space. We set $\beta = \varepsilon/(k + c)$, so O_{det} runs in $n^{1+\varepsilon}$ time and n^ε space with n^ε bits of advice. Similarly, applying Theorem 3.6, we can obtain a randomized algorithm O_{rand} solving L in $O(n)$ time with n^ε advice bits and $O(\log n)$ random bits. □

The sparsity requirement can be further weakened, if we only ask for a quasi-linear circuit size bound.

Reminder of Theorem 1.4 *If there is an $\varepsilon \in (0, 1)$ and a 2^{n^ε}-sparse $L \in \mathsf{NP}$ which is not computable by any $\tilde{O}(n)$-time deterministic algorithm with $\tilde{O}(n^\varepsilon)$ advice, then $\mathsf{NP} \not\subset \mathsf{SIZE}(n \cdot \log^c n)$ for all $c \geq 1$.*

Proof. We prove the constrapositive. Suppose $\mathsf{NP} \subset \mathsf{SIZE}(n \cdot \log^c n)$ for some $c \geq 1$. We show that for all $\varepsilon \in (0, 1)$, every 2^{n^ε}-sparse language $L \in \mathsf{NP}$ can then be decided by an $n \cdot \operatorname{poly}\log(n)$-time deterministic algorithm with $\tilde{O}(n^\varepsilon)$ bits of advice.

Let $t = t(n) = \Theta(n^\varepsilon)$. Applying Theorem 3.4, we can define another language $H \in \mathsf{NP}$ such that $L_n = O_{\mathsf{det}}^{H_t}$, where O_{det} is an $O(n)$-time deterministic algorithm with $O(t)$ bits of advice, which makes $O(n/t)$ queries to H_t. We hardwire the $\tilde{O}(t)$-size circuit for H_t into our advice, so the answer of each query can be computed in $\tilde{O}(t)$ time. So O_{det} runs in $O(n \cdot \operatorname{poly}\log(t))$ time with $\tilde{O}(n^\varepsilon)$ bits of advice. □

4.3 Hardness Magnification for Zero-Error Heuristics

A (zero-error) average-case algorithm A of a function $f : \{0, 1\}^n \to \{0, 1\}$ is a deterministic algorithm that always outputs a value in $\{0, 1, \text{“?”}\}$, such that: (1) A is never incorrect; (2) A outputs "?" with probability at most $1/n$ over uniform random inputs.

Reminder of Theorem 1.9 *Let $\mathscr{C}$ be any circuit class (e.g., $\mathscr{C}$ could be $\mathsf{Circuit}$, $\mathsf{Formula}$, $\mathsf{AC}^0[6]$, etc.). If there is an $\varepsilon > 0$ and a family of languages $\{L_\beta\}$ (indexed over $\beta \in (0, 1)$) such that for all β, L_β is a 2^{n^β}-sparse NP language not solvable on average with zero error by $\mathscr{C}$-circuits of size n^ε, then NP does not have n^k-size circuits in $\mathscr{C}$, for all k.*

Proof. Suppose $\mathsf{NP} \subset \mathscr{C}[n^k]$ for some k (i.e., NP has n^k-size circuits from $\mathscr{C}$). It suffices to show that for every $\varepsilon > 0$, there is a $\beta > 0$ such that every 2^{n^β}-sparse $L \in \mathsf{NP}$ can be solved on average with zero error by $\mathscr{C}[n^\varepsilon]$.

For any $\varepsilon > 0$, let $\beta = \beta(\varepsilon) = \varepsilon/(2k)$, and let $t = t(n) = n^\beta + \log n$. Define function $H_t(h)$ as following: if there exists $x \in L_n$ such that $x_{[1\ldots t]} = h$, then $H_t(h) = 1$; otherwise $H_t(h) = 0$

(in an input to H_t, we also use $O(\log n)$ bits to specify the input length n). Note that $H \in \mathsf{NP}$, so H_t can be computed by a $\mathscr{C}$-circuit of size $t^k < n^{\varepsilon}$. To solve $L_n(x)$ on average, we simply return 0 if $H_t(x_{[1\dots t]})$ rejects, and return "?" if $H_t(x_{[1\dots t]})$ accepts. Obviously this algorithm is never incorrect. Moreover, $\Pr_{x \in_R \{0,1\}^n}[H_t(x_{[1\dots t]})$ accepts$] \le 2^{n^{\beta}}/2^t \le 1/n$. Hence L can be solved on average with zero error by $\mathscr{C}$-circuits of size n^{ε}. □

Reminder of Theorem 1.10 *Let $\mathscr{C}$ be any circuit class (e.g., $\mathscr{C} = \mathsf{Circuit}, \mathsf{Formula}$, or $\mathsf{AC}^0[6]$).*

- *Let $s(m) \ge m$. If $\mathsf{MCSP}[s(m)]$ on input length $n = 2^m$ cannot be solved on average with zero error by $\mathscr{C}[\mathrm{poly}(s(m))]$, then $\mathsf{NP} \not\subset \mathscr{C}[\mathrm{poly}(n)]$.*
- *Let $p(n) \ge \log n$. If $\mathsf{MKtP}[p(n)]$ cannot be solved on average with zero error by $\mathscr{C}[\mathrm{poly}(p(n))]$, then $\mathsf{EXP} \not\subset \mathscr{C}[\mathrm{poly}(n)]$.*

Proof. Our proof is similar to Theorem 1.9. We first prove the MCSP case. Note that $\mathsf{MCSP}[s(m)]$ has sparsity $S_{\mathsf{sparse}} = 2^{100 s(m) \log s(m)}$. Let $t = \log S_{\mathsf{sparse}} + \log n$. Define function $H(h)$ as following: if there exists $x \in \mathsf{MCSP}[s(m)]$ such that $x_{[1\dots t]} = h$, then $H(h) = 1$; otherwise $H(h) = 0$ (in an input to H_t, we also use $O(\log n)$ bits to specify the input length n). Observe that $H \in \mathsf{NP}$, as we can guess a witness circuit C of size $s(m)$, and check whether $C(i) = h_i$ for all $1 \le i \le t$. So H_t can be computed by a $\mathscr{C}$-circuit of size $\mathrm{poly}(t)$. Thus by the same argument in the proof of Theorem 1.9, we can decide $\mathsf{MCSP}[s(m)]$ on average with zero error by $\mathscr{C}$-circuits of size $\mathrm{poly}(s(m))$.

For the MKtP case, the proof is similar as above. We define H in the same way, and simply note that H is solved in deterministic exponential time by straightforwardly enumerating and simulating every possible Turing machine. □

5 Better Hardness Magnification for search-MCSP and MKtP

In this section we discuss hardness magnification for search-$\mathsf{MCSP}[s(m)]$ and search-$\mathsf{MKtP}[p(n)]$. To work with circuits and formulas, we assume the output is a fixed-length string for convenience. We assume the output for search-$\mathsf{MCSP}[s(m)]$ is always an $(L+1)$-length string where $L = 100 s(m) \log s(m)$, being either 0^{L+1} (for NO instance) or $1\langle C_{\mathsf{padded}}\rangle$ where $\langle C_{\mathsf{padded}}\rangle = \langle C\rangle 10 \cdots 0$ is the (padded) description of a witness circuit. Similarly we assume the output for search-$\mathsf{MKtP}[p(n)]$ is either 0^{L+1} or $1\langle M_{\mathsf{padded}}\rangle$ where $|\langle M_{\mathsf{padded}}\rangle| = L = p(n) + 1$.

Reminder of Theorem 1.6 *Let $\mathcal{C} \in \{\oplus\mathsf{P}, \mathsf{PP}, \mathsf{PSPACE}\}$, and $m \le s(m) \le 2^{(1-\Omega(1))m}$. Let the input length $n = 2^m$.*

1. *If search-$\mathsf{MCSP}[s(m)] \notin \mathsf{Circuit}[n \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{Circuit}[\mathrm{poly}(n)]$.*
2. *If search-$\mathsf{MCSP}[s(m)] \notin \mathsf{U}_2\text{-}\mathsf{Formula}\text{-}\oplus[n \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{Formula}[\mathrm{poly}(n)]$.*
3. *If search-$\mathsf{MCSP}[s(m)] \notin \mathsf{B}_2\text{-}\mathsf{Formula}[n^2 \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{Formula}[\mathrm{poly}(n)]$.*
4. *If search-$\mathsf{MCSP}[s(m)] \notin \mathsf{U}_2\text{-}\mathsf{Formula}[n^3 \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{Formula}[\mathrm{poly}(n)]$.*
5. *If search-$\mathsf{MCSP}[s(m)] \notin \mathsf{BP}[n^2 \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{BP}[\mathrm{poly}(n)]$.*
6. *If search-$\mathsf{MCSP}[s(m)] \notin \mathsf{AC}_{d+2}[m_*][n \cdot \mathrm{poly}(s(m))]$, then $\mathcal{C} \not\subset \mathsf{AC}_d[m_*][\mathrm{poly}(n)]$, for all constants d and even integers $m_* \ge 2$.*

7. *If there is an $\varepsilon > 0$ such that, for all small enough $\beta > 0$,* search-MCSP$[2^{\beta m}] \notin \mathsf{TC}_{d+O(\log 1/\varepsilon)}[n^{1+\varepsilon}]$, *then $\mathcal{C} \not\subset \mathsf{TC}_d[\mathrm{poly}(n)]$, for all constants d.*

Moreover, all above implications also hold for $\mathcal{C} =$ EXP*, with* search-MCSP *replaced by* search-MKtP.

5.1 Results for search-MKtP

Using ideas similar to Theorem 3.4, we first prove a lemma showing that search-MKtP can be solved with short *non-adaptive* queries to an oracle in EXP. Prior work [MMW19] proved a similar result requiring (highly) adaptive query access; as far as we can tell, their approach cannot be used to prove tight hardness magnification results for more restricted classes such as branching programs and Boolean formulas, as in Theorem 1.6.

Lemma 5.1. *There exists a language $H \in$* EXP *(specified by Algorithm 3) such that, for any $p(n) \geq \log n$, there is a deterministic $O(n \cdot p(n))$-time algorithm O_{det} computing* search-MKtP$[p(n)]$ *(specified by Algorithm 4) with oracle access to H (only making* **non-adaptive** *queries to H_t where $t = \Theta(p(n))$) with $O(p(n))$ bits of advice.*

Proof. Note that MKtP$[p(n)]$ is a S_{sparse}-sparse language for some $S_{\mathsf{sparse}} = 2^{\Theta(p(n))}$. Our proof is similar to that of Theorem 3.4.

We apply Lemma 3.2 and define the hash function $M_v : \{0,1\}^n \to \{0,1\}^t$ specified by t-bit seed v, where $t = \Theta(\log S_{\mathsf{sparse}}) = \Theta(p(n))$. There is a good seed v such that for any two different $x, y \in$ MKtP$[p(n)]$, $M_v(x) \neq M_v(y)$.

The function H. Let $L = p(n) + 1$. Now we define the function H analogous to Algorithm 1.

Algorithm 3: The algorithm H specialized for search-MKtP$[p(n)]$.

```
Given an input $h \in \{0,1\}^t$, a seed $v \in \{0,1\}^t$, an integer $i \in [n]$, a bitstring $w \in \{0,1\}^t$, and an
  integer $j \in \{0, 1, \dots, L\}$.
By enumerating every Turing Machine $M$ with $k := p(n) - |\langle M \rangle| \geq 0$, find the lexicographically
  smallest $\langle M \rangle$ such that within $2^k$ steps $M$ outputs $x \in \{0,1\}^n$ satisfying
  $M_v(x) = h, x_{[i \dots (i+t-1)]} = w$. Accept if $\langle M_{\mathsf{padded}} \rangle_j = 1$. Reject if $M$ is not found or
  $\langle M_{\mathsf{padded}} \rangle_j = 0$.
```

The parameter j refers to the index for the output string of search-MCSP. We assume $\langle M_{\mathsf{padded}} \rangle_0 = 1$.

Note that n and $p(n)$ should also be considered as part of the input to H. It's easy to see that H runs in deterministic $2^{O(p(n))}$ time on input length $\Theta(p(n))$, so $H \in$ EXP.

The Deterministic Oracle Algorithm O_{det} with Advice. Next we describe the deterministic oracle algorithm O_{det} which computes search-MKtP. O_{det}^H takes $O(p(n))$ bits of advice, runs in $O(Ln) = O(n \cdot p(n))$ time, and makes $O(Ln/t) = O(n)$ non-adaptive queries to H. For NO instance, the output is 0^{L+1}. For YES instance it outputs $1\langle M_{\mathsf{padded}} \rangle$ corresponding to the lexicographically smallest witness Turing Machine $\langle M \rangle$. □

Proof of Theorem 1.6 **(search-MKtP).** We only prove the case for $\mathcal{C} =$ EXP and Circuit. For other computational models, our argument is similar to the proof of Theorem 1.1 (note that for the TC^0 result we need to use the code from Theorem 2.6).

We prove the contrapositive. Suppose EXP $\subset \mathsf{P}_{/\mathsf{poly}}$. We want to show search-MKtP$[p(n)]$ admits $n \cdot \mathrm{poly}(p(n))$-size circuits. Recall that the algorithm H defined in Lemma 5.1 is in EXP, so H has t^k-size circuit on input length t, for some constant k. For $t = t(n) = \Theta(p(n))$, by Lemma 5.1, $O_{\mathsf{det}}^{H_t}$ solves

Algorithm 4: The algorithm O_{det} specialized for search-$\mathsf{MKtP}[p(n)]$.

```
Given an input $x \in \{0,1\}^n$, and oracle access to $H$.
Given a good seed $v \in \{0,1\}^t$ for $\mathsf{MKtP}[p(n)]$ as advice.
for $j \in \{0,1,\dots,L\}$ do
    Compute $h = M_v(x)$.
    if for all $i \in \{1, 1+t, 1+2t, \dots\} \cap [n]$, $H(h, v, i, x_{[i\dots(i+t-1)]}, j) = 1$ then
        Output 1.
    else
        Output 0.
```

search-$\mathsf{MKtP}[p(n)]$, where O_{det} is an $O(n \cdot p(n))$-time deterministic algorithm with $O(t)$ bits of advice, which makes $O(n)$ non-adaptive queries to oracle H_t.

We can hard-wire the $O(t)$ bits of advice into the circuit. This yields an $O(p(n)^k \cdot n + n \cdot p(n))$-size circuit for search-$\mathsf{MKtP}[p(n)]$. □

5.2 Results for search-MCSP

Recall that $n = 2^m$. First we prove a lemma analogous to Lemma 5.1.

Lemma 5.2. *There exists a language $H \in (\Sigma_2\mathsf{P})^{\oplus\mathsf{P}}$ (specified by Algorithm 5) such that, for any $s(m) \geq m$, there is a deterministic $n \cdot \mathrm{poly}(s(m))$-time algorithm O_{det} computing* search-$\mathsf{MCSP}[s(m)]$ *with oracle access to H (only making* **non-adaptive** *queries to H_t where $t = \Theta(s(m)\log s(m))$) with $O(s(m)\log s(m))$ bits of advice.*

Proof. Note that $\mathsf{MCSP}[s(m)]$ is an S_{sparse}-sparse language where $\log(S_{\mathsf{sparse}}) = \Theta(s(m)\log s(m))$. We apply Lemma 3.3 (fully explicit version) and define hash function $M_v : \{0,1\}^n \to \{0,1\}^t$ specified by length-t seed v, where $t = \Theta(s(m)\log s(m))$. There is a good seed v such that for any two different $x, y \in \mathsf{MCSP}[s(m)]$, $M_v(x) \neq M_v(y)$.

The Function H. Now we define the function H. Note that m and $s(m)$ should also be considered as part of the input to H. Let $L = 100s(m)\log s(m)$. For a circuit C with m input bits, let $tt(C) = C(1)C(2)\dots C(2^m)$ denote the truth table of C.

Algorithm 5: The algorithm H specialized for search-$\mathsf{MCSP}[s(m)]$.

```
Given an input $h \in \{0,1\}^t$, a seed $v \in \{0,1\}^t$, an integer $i \in [n]$, a bitstring $w \in \{0,1\}^t$, and an
  integer $j \in \{0,1,\dots,L\}$.
Guess a circuit $C$ of size $s(m)$ with $m$ input bits. Accept if $M_v(tt(C)) = h$,
  $(tt(C))_{[i\dots(i+t-1)]} = w$, $\langle C_{\mathsf{padded}}\rangle_j = 1$, and for every $C'$ of size $s(m)$ that is lexicographically
  smaller than $C$, $M_v(tt(C')) \neq h$. Reject otherwise.
```

By Lemma 3.3, for any $k \in [t]$, $M_v(tt(C))_k = \langle tt(C), w_{v,k}\rangle$, where the j-th bit of $w_{v,k}$ can be computed in $\mathrm{poly}(t)$ time. Then we can compute this dot product using a $\oplus\mathsf{P}$ oracle taking $\mathrm{poly}(t)$ bits of input. It simply counts (modulo 2) the number of $j \in [n]$ such that $C(j) = (w_{v,k})_j = 1$. Hence $H \in (\Sigma_2\mathsf{P})^{\oplus\mathsf{P}}$.

The Deterministic Oracle Algorithm O_{det} with Advice. The algorithm O_{det} is essentially the same as in the proof of Lemma 5.1 for search-MKtP (see Algorithm 4). It takes $O(t)$ bits of advice, runs in $O(L \cdot n\,\mathrm{poly}(t)) \leq n \cdot \mathrm{poly}(t)$ time, and makes $O(Ln/t) = O(n)$ non-adaptive queries to H_t. For NO instance, the output is 0^{L+1}. For YES instance it outputs $1\langle C_{\mathsf{padded}}\rangle$ corresponding to the lexicographically smallest witness circuit C. □

Proof of Theorem 1.6 (search-MCSP)**.**

$\mathcal{C} = \oplus\mathsf{P}$**.** We prove the case for Circuit. For other computational models, our argument is similar to the proof of Theorem 1.1. (Note that for the TC^0 result we need to use the code E from Theorem 2.6. To make Lemma 5.2 work, we use the fact that E is $\oplus\mathsf{P}$-fully explicit, and that $\oplus\mathsf{P}^{\oplus\mathsf{P}} = \oplus\mathsf{P}$.)

Note that $(\Sigma_2\mathsf{P})^{\oplus\mathsf{P}} \subseteq \mathsf{BPP}^{\oplus\mathsf{P}} \subset \mathsf{P}^{\oplus\mathsf{P}}_{/\mathsf{poly}} = \oplus\mathsf{P}_{/\mathsf{poly}}$, where the first inclusion follows from Fortnow's proof [For09] of Toda's first lemma [Tod91].

Suppose $\oplus\mathsf{P} \subset \mathsf{Circuit}[\mathrm{poly}(n)]$. Then $(\Sigma_2\mathsf{P})^{\oplus\mathsf{P}} \subset \oplus\mathsf{P}_{/\mathsf{poly}} \subseteq \mathsf{Circuit}[\mathrm{poly}(n)]$, so the algorithm H defined in Lemma 5.2 has t^k-size circuit on input length t, for some constant k. Let $t = t(n) = \Theta(s(m)\log s(m))$. Then using the same argument as in the proof for search-MKtP in the previous section, we can show that search-MCSP$[s(m)]$ admits $n \cdot \mathrm{poly}(s(m))$-size circuits.

$\mathcal{C} = \mathsf{PP}$**.** We prove the case for Formula. For other computational models the proof follows similarly.

Suppose $\mathsf{P} \subseteq \mathsf{PP} \subset \mathsf{Formula}[\mathrm{poly}(n)]$. Then $\mathsf{P}^{\mathsf{PP}} \subset \mathsf{P}_{/\mathsf{poly}}$, by hardwiring polynomial-sized formulas simulating the PP oracle, as advice. Hence $\oplus\mathsf{P} \subseteq \mathsf{P}^{\mathsf{PP}} \subset \mathsf{P}_{/\mathsf{poly}}$. Since $\mathsf{P} \subset \mathsf{Formula}[\mathrm{poly}(n)]$, we also have $\mathsf{P}_{/\mathsf{poly}} \subset \mathsf{Formula}[\mathrm{poly}(n)]$. Therefore $\oplus\mathsf{P} \subset \mathsf{Formula}[\mathrm{poly}(n)]$. The rest of the proof follows from the $\mathcal{C} = \oplus\mathsf{P}$ case.

$\mathcal{C} = \mathsf{PSPACE}$**.** The proof directly follows from $\oplus\mathsf{P} \subseteq \mathsf{PSPACE}$. □

We can also prove a result for uniform deterministic algorithms with a small amount of advice similar to Theorem 1.2.

Reminder of Theorem 1.8

- *Let* $\mathcal{C} \in \{\oplus\mathsf{P}, \mathsf{PP}, \mathsf{PSPACE}\}$, *and* $m \le s(m) \le 2^{(1-\Omega(1))m}$. *If* search-MCSP$[s(m)]$ *on input length* $n = 2^m$ *is not computable by an* $n \cdot \mathrm{poly}(s(m))$ *time* $\mathrm{poly}(s(m))$ *space deterministic algorithm with* $\mathrm{poly}(s(m))$ *bits of advice, then* $\mathcal{C} \not\subset \mathsf{Circuit}[\mathrm{poly}(n)]$.
- *Let* $\log n \le p(n) \le n^{1-\Omega(1)}$. *If* search-MKtP$[p(n)]$ *is not computable by an* $n \cdot \mathrm{poly}(p(n))$ *time* $\mathrm{poly}(p(n))$ *space deterministic algorithm with* $\mathrm{poly}(p(n))$ *bits of advice, then* $\mathsf{EXP} \not\subset \mathsf{Circuit}[\mathrm{poly}(n)]$.

Proof sketch. We slightly modify the proof for Theorem 1.6 in the previous two sections: use the hash function M_v from Lemma 3.3 (fully explicit version), so that O_{det} can compute $M_v(x)$ using small space and small advice (as in the proof of Theorem 1.2). □

6 Open Problems

Our results suggest several further directions in hardness magnification to study. Here we highlight two.

- First, the most interesting open question is whether Theorem 1.1 or Theorem 1.2 can be applied to prove actual circuit lower bounds. Along the same lines, are there any other natural sparse NP languages for which one can prove some concrete lower bounds?

- Second, is it possible to show hardness magnification results for "denser" variants of $\mathsf{MCSP}[s(m)]$ or $\mathsf{MKtP}[s(m)]$, such as $\mathsf{MCSP}[2^m/m^3]$?[5] The hardness magnification in this work and [OS18, OPS19, MMW19] crucially depend on the condition that $s(m)$ is $2^{o(m)}$ or smaller.

Acknowledgments

We would like to thank Chi-Ning Chou, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Rahul Santhanam, and Roei Tell for helpful discussions. We thank the FOCS reviewers for helpful comments.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[AGHP90] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 544–553, 1990.

[AK10] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010.

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, July 1995.

[Bog18] Andrej Bogdanov. Small bias requires large formulas. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 22:1–22:12, 2018.

[CILM18] Marco L. Carmosino, Russell Impagliazzo, Shachar Lovett, and Ivan Mihajlin. Hardness amplification for non-commutative arithmetic circuits. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 12:1–12:16, 2018.

[CKLM19] 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 2019, July 9-12, 2019, Patras, Greece.*, pages 39:1–39:14, 2019.

[CMMW19] Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Relations and equivalences between circuit lower bounds and karp-lipton theorems. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA.*, pages 30:1–30:21, 2019.

[CRVW02] Michael R. Capalbo, Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 659–668, 2002.

[CS76] Ashok K. Chandra and Larry J. Stockmeyer. Alternation. In *17th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 98–108, 1976.

[5] Note that for such variants, better concrete lower bounds are known [CKLM19].

[CT19] Lijie Chen and Roei Tell. Bootstrapping results for threshold circuits ”just beyond” known lower bounds. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 34–41, 2019.

[DM18] Irit Dinur and Or Meir. Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity. *Computational Complexity*, 27(3):375–462, 2018.

[FGHK16] Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A better-than-3n lower bound for the circuit complexity of an explicit function. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 89–98, 2016.

[FKS84] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with 0(1) worst case access time. *J. ACM*, 31(3):538–544, June 1984.

[FLvMV05] Lance Fortnow, Richard Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *Journal of the ACM (JACM)*, 52(6):835–865, 2005.

[For79] Steven Fortune. A note on sparse complete sets. *SIAM J. Comput.*, 8(3):431–433, 1979.

[For09] Lance Fortnow. A simple proof of toda’s theorem. *Theory of Computing*, 5(7):135–140, 2009.

[FvM00] Lance Fortnow and Dieter van Melkebeek. Time-space tradeoffs for nondeterministic computation. In *Proceedings 15th Annual IEEE Conference on Computational Complexity*, pages 2–13. IEEE, 2000.

[GG81] Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences*, 22(3):407–420, 1981.

[GHK$^+$13] Anna Gál, Kristoffer Arnsfelt Hansen, Michal Koucký, Pavel Pudlák, and Emanuele Viola. Tight bounds on computing error-correcting codes by bounded-depth circuits with arbitrary gates. *IEEE Trans. Information Theory*, 59(10):6611–6627, 2013.

[GI02] Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 812–821, 2002.

[Gil98] David Gillman. A chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[Hås98] Johan Håstad. The shrinkage exponent of de morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998.

[HS65] Juris Hartmanis and Richard E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[HS17] Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 7:1–7:20, 2017.

[IPS97] Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. Size-depth tradeoffs for threshold circuits. *SIAM J. Comput.*, 26(3):693–707, 1997.

[Juk12] Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

[Lev84] Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15 – 37, 1984.

[LW13] Richard J. Lipton and Ryan Williams. Amplifying circuit lower bounds against polynomial time, with applications. *Computational Complexity*, 22(2):311–343, 2013.

[Mah80] Stephen R. Mahaney. Sparse complete sets for NP: solution of a conjecture of berman and hartmanis. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*, pages 54–60, 1980.

[MMW19] 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.

[MP17] Moritz Müller and Ján Pich. Feasibly constructive proofs of succinct weak circuit lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:144, 2017.

[Neč66] Eduard I. Nečiporuk. On a boolean function. *Soviet Math. Dokl*, 7(4):999–1000, 1966.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

[NSS95] M. Naor, L. J. Schulman, and A. Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, FOCS '95, pages 182–, Washington, DC, USA, 1995. IEEE Computer Society.

[Oli19] Igor Carboni Oliveira. Randomness and intractability in kolmogorov complexity. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 32:1–32:14, 2019.

[OPS19] Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA.*, pages 27:1–27:29, 2019.

[OS18] Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 65–76, 2018.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[RVW02] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of mathematics*, pages 157–187, 2002.

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Information Theory*, 42(6):1723–1731, 1996.

[Sri03] Aravind Srinivasan. On the approximability of clique and related maximization problems. *J. Comput. Syst. Sci.*, 67(3):633–651, 2003.

[SS90] Jeanette P. Schmidt and Alan Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM J. Comput.*, 19(5):775–786, September 1990.

[Tal14] Avishay Tal. Shrinkage of de morgan formulae by spectral techniques. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 551–560, 2014.

[Tal16] Avishay Tal. The bipartite formula complexity of inner-product is quadratic. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:181, 2016.

[Tel18] Roei Tell. Quantified derandomization of linear threshold circuits. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 855–865, 2018.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[Wil06] Ryan Williams. Inductive time-space lower bounds for sat and related problems. *Computational Complexity*, 15(4):433–470, 2006.

[Wil07] Ryan Williams. *Algorithms and resource requirements for fundamental problems*. PhD thesis, Ph. D. Thesis, Carnegie Mellon University, CMU-CS-07-147, 2007.

[Wil08] R. Ryan Williams. Time-space tradeoffs for counting NP solutions modulo integers. *Computational Complexity*, 17(2):179–219, 2008.

A Time Hierarchy Theorem Made Sparse

Here we show that an adaptation of the classical deterministic time hierarchy theorem [HS65] already gives us the required lower bound in Theorem 1.1, although it is (unfortunately) not as sparse as we would like for lower bound applications.

Reminder of Theorem 1.3 *For all $\varepsilon \in (0,1)$, there is a language L decidable in $n^{1+\varepsilon} \cdot \operatorname{poly}\log(n)$ time that is $(2^{n^\varepsilon} \cdot n)$-sparse but not computable in $n^{1+\varepsilon}$ time with n^ε bits of advice.*

Proof. This is a simple adaptation of the classical time hierarchy theorem.

Consider the following algorithm M: given an input $x \in \{0,1\}^n$, it treats the first n^ε bits as an advice $\alpha \in \{0,1\}^{n^\varepsilon}$, and the next $\log n$ bits as an integer $i \in [n]$. If the rest $n - n^\varepsilon - \log n$ bits are not all-zero, M outputs 0 immediately. Otherwise, $M(x)$ simulates M_i (the i-th RAM machine) on the input x for $n^{1+\varepsilon}$ steps, if M_i terminates with an output t, it outputs $1-t$, and otherwise outputs 0.

Now, for any $n^{1+\varepsilon}$ time deterministic algorithm with advice sequence $\{\alpha_n\}_{n\in\mathbb{N}}$ such that $|\alpha_n| = n^\varepsilon$, suppose it is implemented by the i-th RAM machine. Then for sufficiently large input length n, we know that for the input $z_n = \alpha_n \circ \mathsf{bin}(i) \circ 0^{(n-n^\varepsilon-\log n)}$ ($\mathsf{bin}(i)$ is the binary representation of i), we have that $M(z_n) = 1 - M_i(z_n)$ by the way M is constructed. Therefore, M cannot be computed by any $n^{1+\varepsilon}$ deterministic algorithm with n^ε bits of advice.

It is easy to see that the simulation runs in $n^{1+\varepsilon} \cdot \operatorname{poly}\log(n)$ time, and L is $2^{n^\varepsilon} \cdot n$ sparse. □

B Time-space Trade-off Lower Bounds Made 1-Sparse

In this section we prove Theorem 1.5 (restated below). The proof follows the same structure of [Wil07, Section 4.2], with the observation that the proofs there also work with 1-sparse languages.

Reminder of Theorem 1.5 *For every $c < \phi$, there is a 1-sparse language $L \in \mathsf{coNTIME}[n^{1+o(1)}]$, such that $L \notin \mathsf{TS}[n^c, n^{o(1)}]$. Here $\phi = \frac{1+\sqrt{5}}{2} = 1.618\cdots$ is the golden ratio.*

Let 1-SPARSE denote the class of 1-sparse languages (languages with at most one yes-instance per input length). Let $\mathsf{DTS}[t]$ denote $\mathsf{TS}[t^{1+o(1)}, t^{o(1)}]$. For a complexity class $\mathcal{C}$ and a time constructible function f, define $(\exists f(n))\mathcal{C}$ to be the class of problems that can be solved by some nondeterministic machine N that, on input x, writes an $f(n)^{1+o(1)}$ bit string nondeterministically to a special tape, then feeds the input $\langle x, y\rangle$ to a machine from class $\mathcal{C}$. The class $(\forall f(n))\mathcal{C}$ is defined similarly (with co-nondeterministic machines).

We first show that the "No Complementary Speedup" theorem [CS76] (see also [Wil07, Theorem 3.3.1]) can be adapted to 1-sparse languages.

Lemma B.1 ("No Complementary Speedup" for 1-sparse languages)**.** *For time constructible $t(n) \ge n$,*

$$\text{1-SPARSE} \cap \mathsf{coNTIME}[t] \not\subseteq \mathsf{NTIME}[o(t)].$$

Proof Sketch. On unary input 1^n, diagonalize against the NTIME machine running in time $t(n)$, encoded by n. □

Next, we show that the "Speedup lemma" [FvM00] can also be adapted to 1-sparse languages.

Lemma B.2 ("Speedup lemma" for 1-sparse languages)**.** *For $b = t^{\Omega(1)}$,*

$$\text{1-SPARSE} \cap \mathsf{DTS}[t] \subseteq (\exists b)(\text{1-SPARSE} \cap (\forall \log t)\mathsf{DTS}[t/b]).$$

Proof Sketch. For a 1-sparse language L solved by a machine M_L in $\mathsf{DTS}[t]$, on an input x we guess b strings $s_1, \ldots, s_b$ of length $t^{o(1)}$, where s_i is intended to be the memory state of M_L at the $(i \cdot t/b)$-th step of M_L running on x. (Without loss of generality, we assume b divides t.) Since L is 1-sparse and M_L is deterministic, there exists at most one sequence $(x, s_1, \ldots, s_b)$ such that s_i's are valid memory states leading to the accepting state, given input x. To check that a guessed sequence is correct, we universally guess $i = 1, \ldots, b$ and simulate the computation from s_{i-1} for t/b steps to see if s_i is reached. □

With the above adapted speedup lemma, we can show that the "Conditional Speedup Theorem" of [Wil06, Wil07] can also be adapted to hold for 1-sparse languages.

Lemma B.3 (adaptation of [Wil07, Lemma 4.2.1])**.** *Let $c \in (1, 2)$. Define the sequence $d(1) := 2, d(k) := 1 + d(k-1)/c$. If $\text{1-SPARSE} \cap \mathsf{coNTIME}[n^{1+o(1)}] \subseteq \mathsf{DTS}[n^c]$ then for all $k \ge 1$,*

$$\text{1-SPARSE} \cap \mathsf{DTS}[n^{d(k)}] \subseteq (\exists n)\big(\text{1-SPARSE} \cap (\forall \log n)\mathsf{DTS}[n]\big).$$

Proof. We proceed by induction on k. The $k = 1$ case $\text{1-SPARSE} \cap \mathsf{DTS}[n^2] \subseteq (\exists n)(\text{1-SPARSE} \cap (\forall \log n)\mathsf{DTS}[n])$ directly follows from the Speedup Lemma (Lemma B.2). For the inductive step, assume $\text{1-SPARSE} \cap \mathsf{coNTIME}[n^{1+o(1)}] \subseteq \mathsf{DTS}[n^c]$ and $\text{1-SPARSE} \cap \mathsf{DTS}[n^{d(k)}] \subseteq (\exists n)(\text{1-SPARSE} \cap (\forall \log n)\mathsf{DTS}[n])$. Observe that for $c < 2$, $d(k) \ge c$. By padding and the inductive hypothesis,

$$\text{1-SPARSE} \cap \mathsf{coNTIME}[n^{d(k)/c}] \subseteq \text{1-SPARSE} \cap \mathsf{DTS}[n^{d(k)}] \subseteq (\exists n)(\text{1-SPARSE} \cap (\forall \log n)\mathsf{DTS}[n]). \quad (1)$$

By the Speedup Lemma (Lemma B.2),

$$\text{1-SPARSE} \cap \mathsf{DTS}[n^{1+d(k)/c}] \subseteq (\exists n)(\text{1-SPARSE} \cap (\forall \log n)\mathsf{DTS}[n^{d(k)/c}]),$$

where the $(\forall \log n)\mathsf{DTS}[n^{d(k)/c}]$ part corresponds to a $\mathsf{coNTIME}$ computation that takes an input of length $n^{1+o(1)}$ (the input together with a list of guessed memory states) and runs in $n^{d(k)/c+o(1)}$ time. By equation (1) above, this co-nondeterministic computation can be replaced with a Σ_2 computation running in $n^{1+o(1)}$ time. Therefore

$$\begin{aligned}\text{1-}\mathsf{SPARSE} \cap \mathsf{DTS}[n^{1+d(k)/c}] &\subseteq (\exists n)(\exists n)(\text{1-}\mathsf{SPARSE} \cap (\forall \log n)\mathsf{DTS}[n]) \\ &\subseteq (\exists n)(\text{1-}\mathsf{SPARSE} \cap (\forall \log n)\mathsf{DTS}[n]).\end{aligned}$$

□

Theorem B.4 ("Conditional Speedup" for 1-sparse languages, adapting [Wil06], [Wil07, Theorem 4.2.1]). *Let $c \in (1,2)$. If* $\text{1-}\mathsf{SPARSE} \cap \mathsf{coNTIME}[n^{1+o(1)}] \subseteq \mathsf{DTS}[n^c]$ *then for all $\varepsilon > 0$,*

$$\text{1-}\mathsf{SPARSE} \cap \mathsf{DTS}[n^{\frac{c}{c-1}-\varepsilon}] \subseteq (\exists n)\big(\text{1-}\mathsf{SPARSE} \cap (\forall \log n)\mathsf{DTS}[n]\big).$$

Proof Sketch. The proof directly follows from the observation that sequence $d(k)$ defined in Lemma B.3 monotonically converges to $c/(c-1)$. □

Finally, we are ready to prove Theorem 1.5.

Proof of Theorem 1.5. Let $c \in (1, \phi)$, and assume for contradiction that

$$\text{1-}\mathsf{SPARSE} \cap \mathsf{coNTIME}[n^{1+o(1)}] \subseteq \mathsf{DTS}[n^c].$$

For any $\varepsilon > 0$, by padding,

$$\text{1-}\mathsf{SPARSE} \cap \mathsf{coNTIME}[n^{\frac{1}{c-1}-\varepsilon}] \subseteq \text{1-}\mathsf{SPARSE} \cap \mathsf{DTS}[n^{\frac{c}{c-1}-c\varepsilon}].$$

By Theorem B.4,

$$\begin{aligned}\text{1-}\mathsf{SPARSE} \cap \mathsf{DTS}[n^{\frac{c}{c-1}-c\varepsilon}] &\subseteq (\exists n)\big(\text{1-}\mathsf{SPARSE} \cap (\forall \log n)\mathsf{DTS}[n]\big) \\ &\subseteq (\exists n)\big(\text{1-}\mathsf{SPARSE} \cap \mathsf{coNTIME}[n^{1+o(1)}]\big).\end{aligned}$$

Combining the above two, we have

$$\begin{aligned}\text{1-}\mathsf{SPARSE} \cap \mathsf{coNTIME}[n^{\frac{1}{c-1}-\varepsilon}] &\subseteq (\exists n)\big(\text{1-}\mathsf{SPARSE} \cap \mathsf{coNTIME}[n^{1+o(1)}]\big) \\ &\subseteq (\exists n)\mathsf{DTS}[n^c] \subseteq \mathsf{NTIME}[n^c],\end{aligned}$$

which contradicts "No Complementary Speedup" (Lemma B.1) when $1/(c-1) - \varepsilon > c$. For $c \in (1, \phi)$, setting $\varepsilon = (1/(c-1)-c)/2 > 0$ suffices. Hence, there is a 1-sparse language L solvable in $\mathsf{coNTIME}[n^{1+o(1)}]$, such that $L \notin \mathsf{DTS}[n^c]$. □

We cannot seem to extend the full argument of [Wil07, Wil08] to improve the time lower bound to $n^{2\cos(\pi/7)}$. Moreover, we only seem to be able to prove our time-space lower bound for 1-sparse languages in coNP, rather than sparse languages in NP. Given the results of this paper, we consider both of these stumbling blocks to be interesting open problems which may have further consequences for hardness magnification.