Hardness Magnification for Natural Problems

Igor C. Oliveira
Department of Computer Science
University of Oxford
Oxford, United Kingdom
igor.carboni.oliveira@cs.ox.ac.uk

Rahul Santhanam
Department of Computer Science
University of Oxford
Oxford, United Kingdom
rahul.santhanam@cs.ox.ac.uk

***Abstract*—We show that for several natural problems of interest, complexity lower bounds that are barely non-trivial imply super-polynomial or even exponential lower bounds in strong computational models. We term this phenomenon "hardness magnification". Our examples of hardness magnification include:**

1) **Let $\mathsf{MCSP}[s]$ be the decision problem whose YES instances are truth tables of functions with circuit complexity at most $s(n)$. We show that if $\mathsf{MCSP}[2^{\sqrt{n}}]$ cannot be solved on average with zero error by formulas of linear (or even sub-linear) size, then NP does not have polynomial-size formulas. In contrast, Hirahara and Santhanam [1] recently showed that $\mathsf{MCSP}[2^{\sqrt{n}}]$ cannot be solved in the worst case by formulas of nearly quadratic size.**
2) **If there is a $c > 0$ such that for each positive integer d there is an $\varepsilon > 0$ such that the problem of checking if an n-vertex graph in the adjacency matrix representation has a vertex cover of size $(\log n)^c$ cannot be solved by depth-d AC^0 circuits of size $m^{1+\varepsilon}$, where $m = \Theta(n^2)$, then NP does not have polynomial-size formulas.**
3) **Let (α, β)-$\mathsf{MCSP}[s]$ be the promise problem whose YES instances are truth tables of functions that are α-approximable by a circuit of size $s(n)$, and whose NO instances are truth tables of functions that are not β-approximable by a circuit of size $s(n)$. We show that for arbitrary $1/2 < \beta < \alpha \leq 1$, if (α, β)-$\mathsf{MCSP}[2^{\sqrt{n}}]$ cannot be solved by randomized algorithms with random access to the input running in sublinear time, then $\mathsf{NP} \not\subseteq \mathsf{BPP}$.**
4) **If for each probabilistic quasi-linear time machine M using poly-logarithmic many random bits that is claimed to solve Satisfiability, there is a deterministic polynomial-time machine that on infinitely many input lengths n either identifies a satisfiable instance of bitlength n on which M does not accept with high probability or an unsatisfiable instance of bitlength n on which M does not reject with high probability, then $\mathsf{NEXP} \neq \mathsf{BPP}$.**
5) **Given functions $s, c\colon \mathbb{N} \to \mathbb{N}$ where $s \geq c$, let $\mathsf{MKtP}[c, s]$ be the promise problem whose YES instances are strings of Kt complexity [2] at most $c(N)$ and NO instances are strings of Kt complexity greater than $s(N)$. We show that if there is a $\delta > 0$ such that for each $\varepsilon > 0$, $\mathsf{MKtP}[N^{\varepsilon}, N^{\varepsilon} + 5\log(N)]$ requires Boolean circuits of size $N^{1+\delta}$, then $\mathsf{EXP} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$.**

For each of the cases of magnification above, we observe that standard hardness assumptions imply much stronger lower bounds for these problems than we require for magnification.

We further explore magnification as an avenue to proving strong lower bounds, and argue that magnification circumvents the "natural proofs" barrier of Razborov and Rudich [3]. Examining some standard proof techniques, we find that they fall just short of proving lower bounds via magnification. As one of our main open problems, we ask whether there are other meta-mathematical barriers to proving lower bounds that rule out approaches combining magnification with known techniques.

***Keywords*-computational complexity, lower bounds; circuit complexity; hardness magnification; minimum circuit size problem; vertex cover; satisfiability; time-bounded Kolmogorov complexity**

I. Introduction

A. Overview

It is a truth universally acknowledged that the state of the art in complexity lower bounds is very primitive indeed. The main open problems in complexity theory, such as NP vs. P and EXP vs. $\mathsf{SIZE}(\mathrm{poly})$ ask about *super-polynomial* lower bounds against *strong* computational models. In contrast, the lower bounds we can actually show for explicit problems are either for weak models such as constant-depth circuits [4, 5] and monotone circuits [6], or are weak in magnitude, such as the $5n$ circuit size lower bound [7] or sub-cubic formula size lower bound for Andreev's function [8]. Despite decades of effort, the frontier of complexity lower bounds tends to advance very slowly or not at all. Advances often require substantially new ideas, such as the algorithmic paradigm introduced by Williams [9] and used to show [10] that NEXP does not have polynomial-size constant-depth circuits with composite modular gates.

Is the lack of progress on lower bounds due to our own lack of imagination, or is there an inherent difficulty? Some evidence for the inherent difficulty of the problem is given by meta-mathematical barriers such as the relativization barrier [11], natural proofs barrier [3] and algebrization barrier [12]. These barriers seek to understand and explain the limits of various classes of techniques for proving lower bounds.

The natural proofs barrier, in particular, suggests a certain dichotomy between "weak" circuit classes such as AC^0 circuits of sub-exponential size, $\mathsf{AC}^0[p]$ (for prime p) circuits of sub-exponential size, branching programs of sub-quadratic size and Boolean formulas of sub-cubic size on the one hand,[1] and "strong" circuit classes such as Boolean circuits of polynomial size or Boolean formulas of polynomial size which are believed to be able to implement pseudo-random functions (see e.g. [14]). For weak circuit classes, we already have lower bounds via current techniques using "naturalizing" proofs. For strong circuit classes, which are the ones that primarily interest us, lower bounds are believed to be far out of our reach, as there are not even good candidate techniques for proving such lower bounds. There are certain liminal classes such as ACC^0 and TC^0 circuits of depth 2 which don't yet fit clearly into this dichotomy, but by and large the dichotomy seems to capture current beliefs about the hardness of proving lower bounds.

We argue that this gap between weak and strong classes is somewhat illusory, or at the very least depends on the specific problems for which we try to show lower bounds. For various natural problems of interest, and for a range of computational models, we show that lower bounds against weak circuit classes *imply* lower bounds for strong circuit classes. We term this phenomenon "hardness magnification".

Hardness magnification in *computational complexity* is related to the somewhat recent work of Allender and Koucký [15], who showed how to amplify lower bounds for NC^1 against constant-depth circuit classes from size $n^{1+\varepsilon}$ for fixed $\varepsilon > 0$ to super-polynomial size using self-reducibility. However, while their work is focused on lower bounds for "strongly self-reducible" problems in NC, our results are much broader, addressing problems such as SAT for which their techniques do not seem to yield interesting consequences, and also addressing a greater variety of computational models such as general Boolean circuits, Boolean formulas, branching programs, sub-linear time algorithms, etc. Moreover, while the lower bounds they "amplify" are not known to hold for any explicit problems, the lower bounds from which we show magnification are in many cases known for explicit problems – just not for the specific problems we consider.[2] (We discuss additional related work in Section I-C.)

On the other hand, a form of hardness magnification has been recently observed by Müller and Pich [17, Proposition 4.14] in the context of *proof complexity*. In more detail, their argument establishes that barely non-trivial lower bounds for bounded-depth Frege systems for certain tautologies of interest imply super-polynomial lower bounds for Frege systems.[3] The interesting aspect of this result compared to [15] is that lower bounds against bounded-depth Frege systems *are* known (but not for the same tautologies). Moreover, the tautologies from [17] have been studied in the past, and lower bounds in weaker proof systems have been obtained for the same class of tautologies. While our results are incomparable to their work in proof complexity, it served as a source of inspiration for our investigations.

The general template for our results is as follows. We consider various natural problems Q that are believed to be hard. For each such problem Q, we show:

(a) An implication from a weak lower bound for Q (i.e., a lower bound that is weak in magnitude and/or against a weak model) to a strong lower bound for a problem in NP or EXP.

(b) Under a standard hardness hypothesis, the weak lower bound for Q (and indeed a stronger lower bound) does hold. Thus magnification is a sound approach to proving strong lower bounds under standard hardness hypotheses.

(c) The weak lower bound is "barely non-trivial", in that a trivial lower bound that is only slightly smaller can be shown to hold. Moreover, in several cases, the weak bound is known for some explicit problem, just not for the problem Q we consider.

The details of the magnification results depend on the problems Q we consider and on the involved computational models. To establish magnification as a fairly general phenomenon, we consider a range of such problems:

- Variants of the Minimum Circuit Size Problem (MCSP), where the input is the truth table of a Boolean function and the question is whether the function has small circuits.
- Variants of the Minimum Kt Complexity Problem (MKtP), where the input is a string and the question is if the string has low Kt complexity, where Kt complexity is Levin's notion of Kolmogorov time bounded complexity.
- The Vertex Cover problem.
- Satisfiability (3-SAT).

Our most interesting magnification results are for "meta-computational" problems such as MCSP, MKtP and SAT, where the instance of the problem itself encodes a computation. This re-inforces the message of several works in complexity theory (e.g., [18, 3, 9]) that understanding the complexity of meta-computational problems is closely associated with making progress on complexity lower bounds.

Our results can be interpreted in different ways. For an optimist, they might give hope that strong lower bounds are achievable. First, the weak lower bounds from which we magnify are in several cases already known for explicit problems, so it's "merely" a question of showing similar lower bounds for the problems we consider. Second, approaches via magnification appear to avoid the natural proofs barrier to proving lower bounds. Recall that the natural proofs barrier [3] rules out circuit lower bounds given by dense and constructible properties that are useful against the circuit

[1] We refer to a textbook such as [13] for an exposition of these circuit classes and corresponding lower bounds.

[2] Note that some researchers [16] do not see compelling evidence that the weak lower bounds required by Allender and Koucký [15] hold.

[3] The result is not stated in this way, but the proof shows that weak lower bounds are sufficient.

class, modulo standard cryptographic assumptions. Even if the weak lower bound that is required for magnification is shown via a natural property, the magnification step seems to destroy the density of the corresponding property, as magnification only seems to hold for certain special structured problems and not for generic ones. The argument here is similar to the argument of Allender and Koucký [15, Section 8] that "amplifying lower bounds via self-reducibility" evades the natural proofs barrier.

For the pessimist, magnification might simply be an indication that natural proofs and other barriers do not capture all obstacles to proving circuit lower bounds. In this view, magnification is simply an invitation to refine and extend our meta-mathematical understanding of circuit lower bounds so that we have compelling explanations of why lower bounds via magnification would be hard to achieve.

We see this as a win-win situation: either strong lower bounds can be shown via magnification, or magnification and similar phenomena will motivate us to gain a better understanding of the limitations of lower bound techniques.

B. Results and techniques

This section describes in more detail our main results and techniques. For the required background in complexity theory, we refer to standard textbooks such as [13, 19].

We often use a concrete choice of parameters to simplify the exposition. More general formulations of our statements and the majority of proofs are deferred to the full version of the paper [20].

A magnification phenomenon around quasi-linear size lower bounds. We say that a function $f\colon \{0,1\}^n \to \{0,1\}$ is γ-approximable by a boolean circuit C if $\Pr_x[f(x) = C(x)] \geq \gamma$, where $x \sim \{0,1\}^n$. Let (α,β)-$\mathsf{MCSP}[s]$ be the promise problem whose YES instances are truth tables of functions that are α-approximable by a circuit of size $s(n)$, and whose NO instances are truth tables of functions that are not β-approximable by a circuit of size $s(n)$. We use $N = 2^n$ to denote the input length of (α,β)-$\mathsf{MCSP}[s]$, and consider the problem with $\alpha = 1$ and $\beta = 1-\delta$, where $\delta = \delta(n) > 0$ is a parameter that measures the gap between YES and NO instances of $(1, 1-\delta)$-$\mathsf{MCSP}[s]$.

We use $\mathsf{Formula}[t]$ to denote the set of functions that can be computed by Boolean formulas of size at most t. Our first result can be stated as follows.

Theorem 1. *Let $\delta\colon \mathbb{N} \to \mathbb{R}$ and consider the problem $(1, 1-\delta)$-$\mathsf{MCSP}[s]$. The following results hold.*

(i) *Let $s(n) = n^k$ and $\delta(n) = n^{-k}$, where $k \in \mathbb{N}$. If $(1, 1-\delta)$-$\mathsf{MCSP}[s] \notin \mathsf{Formula}[N \cdot (\log N)^{O(1)}]$ then there is $L \in \mathsf{NP}$ over m-bit inputs such that $L \notin \mathsf{Formula}[\mathsf{poly}(m)]$.*

(ii) *For the same choice of parameters, if $(1, 1-\delta)$-$\mathsf{MCSP}[s] \notin \mathsf{Formula}[N^{1+\varepsilon}]$ for some $\varepsilon > 0$, then there is $L \in \mathsf{NP}$ over m-bit inputs and $\delta > 0$ such that $L \notin \mathsf{Formula}[2^{m^\delta}]$.*

(iii) *Let $s(n) = 2^{o(n)}$ and $\delta(n) = 2^{-o(n)}$. if $(1, 1-\delta)$-$\mathsf{MCSP}[s] \notin \mathsf{Formula}[N^{1+\varepsilon}]$ for some $\varepsilon > 0$, then there is $L \in \mathsf{NP}$ over m-bit inputs such that $L \notin \mathsf{Formula}[\mathsf{poly}(m)]$.*

Theorem 1 is a consequence of a more general result presented in Section II. It magnifies super-linear size formula lower bounds between $N \cdot \mathsf{poly}(\log N)$ and $N^{1+\varepsilon}$ to much stronger lower bounds against formulas for a function in NP. As a consequence, minor improvements in lower bounds for $(1, 1-\delta)$-$\mathsf{MCSP}[s]$ around the linear-size regime would have major implications in our understanding of formula lower bounds for explicit problems.

Note that Theorem 1 asks for barely super-linear formula size lower bounds. We observe that there are no known algorithms for $(1, 1-\delta)$-$\mathsf{MCSP}[s]$ over N-bit inputs running in time $2^{o(s(n))}$. In particular, for $\delta > 1/2$ it is not known how to solve this problem in time polynomial over the input length N for any $n \ll s(n) \ll 2^{o(n)}$. Under standard cryptographic assumptions, it is possible to use the theory of natural proofs [3] to prove that $(1, 1-\delta)$-$\mathsf{MCSP}[s]$ cannot be computed by circuits of polynomial size if $s(n) = n^k$ and k is sufficiently large (Proposition 12).

Let $\mathsf{AC}^0_d[t]$ denote depth-d AC^0 circuits of size t. We establish the following additional hardness magnification result.

Theorem 2. *Suppose there exists $k \geq 1$ such that for every $d \geq 1$ there is $\varepsilon_d > 0$ such that $(1, (1-\delta))$-$\mathsf{MCSP}[s] \notin \mathsf{AC}^0_d[N^{1+\varepsilon_d}]$, where $s(n) = n^k$ and $\delta(n) = n^{-k}$. Then $\mathsf{NP} \nsubseteq \mathsf{NC}^1$.*

This result can be seen as an analogue in circuit complexity of the hardness magnification phenomenon observed in proof complexity by Müller and Pich [17, Proposition 4.14].

Perhaps intriguingly, much stronger formula size lower bounds are known for problems considerably simpler than $(1, 1-\delta)$-$\mathsf{MCSP}[s]$. For instance, it is well-known that the parity function over N input variables requires formulas of size $\Omega(N^2)$, and there are several techniques able to prove strong lower bounds against AC^0 circuits. Can we use these existing approaches to establish lower bounds for $(1, 1-\delta)$-$\mathsf{MCSP}[s]$?

We sketch in the full version of the paper [20] how known results and techniques imply the following lower bounds. Let $1/2 \leq \gamma < 1$ be an arbitrary constant. Then, for some choice of $s(n) = 2^{\Theta(n^\gamma)}$ and $\delta(n) = 2^{-n+\Theta(n^\gamma)}$, we have $(1, 1-\delta)$-$\mathsf{MCSP}[s] \notin \mathsf{Formula}[N^{2-o(1)}]$. Note that this lower bound is not good enough for hardness magnification (i.e., Theorem 1) because the parameter $\delta(n)$ is not large enough. On the other hand, we note that if $s(n) = n^{\omega(1)}$ and $\delta < 1/2$ is fixed, then for every $d \geq 1$ and every $\ell \geq 1$ we have $(1, 1-\delta)$-$\mathsf{MCSP}[s] \notin \mathsf{AC}^0_d[N^\ell]$. Observe that this lower bound is not good enough for hardness magnification (i.e., Theorem 2) because the parameter $s(n)$ is too large.

We briefly discuss the techniques behind the proof of Theorem 1, which is not technically involved but requires certain crucial insights. The argument is by contrapositive, so we assume that every problem in NP on m input bits has formulas of bounded size. In order to solve $(1, 1-\delta)$-$\mathsf{MCSP}[s]$ by almost-linear size formulas, we proceed as follows. We take a projection of the input truth table on a certain number of random locations, and define a "compressed" language L in NP over m input bits, where $m \approx s(n)/\delta(n) \ll N = 2^n$, which captures a succinct version of the initial problem. Using the gap parameter δ, it is possible to show that if the input truth table admits small circuits, so does its random projection. On the other hand, a probabilistic argument proves that if the input truth table cannot be approximated by small circuits, then with high probability the projected truth table does not admit small approximating circuits. This allows us to employ small formulas for L to solve the original problem. However, this argument is probabilistic, and the reduction sketched before is randomized. In order to get almost-linear size formulas for $(1, 1-\delta)$-$\mathsf{MCSP}[s]$, we derandomize this construction in an elementary but careful way using non-uniform formulas. While the formulas for L have small size as a function of m when compared to N, the non-constructive derandomization argument is responsible for the almost-linear size bounds (in N) for the formulas obtained for $(1, 1-\delta)$-$\mathsf{MCSP}[s]$.

We observe that the argument is inspired in part by Occam's Razor, an idea from learning theory (see e.g. [21]). The details appear in Section II, where the proof of Theorem 2 is also presented.

Results similar to Theorem 1 can be proved with respect to boolean circuits and indeed for a variety of boolean devices. The main advantage of presenting these ideas in the context of formula complexity is because for formulas there are non-trivial polynomial lower bounds for several explicit problems, and a large number of techniques have been developed for establishing such lower bounds (cf. the discussion in Section I-D).

Kt Complexity and lower bounds for EXP. Note that the previously considered problem concerns the *average-case* complexity of strings, viewed as truth tables. We show next a magnification result for a computational problem related to the *worst-case* complexity of strings.

Let U be a fixed universal machine. For a non-empty string $x \in \{0,1\}^*$, $Kt(x)$ is the minimum over $|p| + \log(t)$ such that $U(p)$ outputs x within t steps. Given functions $s, c \colon \mathbb{N} \to \mathbb{N}$ where $s \geq c$, let $\mathsf{MKtP}[c, s]$ be the promise problem over N-bit inputs whose YES instances are strings of Kt complexity [2] at most $c(N)$ and NO instances are strings of Kt complexity greater than $s(N)$.

Let $\mathsf{SIZE}[t]$ denote the class of boolean circuits of size at most t. We establish the following result.

Theorem 3. *If there is a fixed $\delta > 0$ such that for each $\varepsilon > 0$, $\mathsf{MKtP}[N^\varepsilon, N^\varepsilon + 5\log(N)]$ does not have circuits of size $N^{1+\delta}$, then $\mathsf{EXP} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$.*

It is not hard to see that $\mathsf{MKtP}[N^\varepsilon, N^\varepsilon + 5\log(N)] \in \mathsf{EXP}$. Moreover, results from [22] show that this problem is hard for EXP under non-uniform reductions. In particular, if EXP does not have polynomial size circuits, neither does this problem. This allows us to get the following *equivalence* as a consequence.

Corollary 4. *$\mathsf{MKtP}[N^\varepsilon, N^\varepsilon + 5\log(N)]$ has polynomial-size circuits for each $\varepsilon > 0$ iff for all $\delta > 0$ there is an $\varepsilon > 0$ such that $\mathsf{MKtP}[N^\varepsilon, N^\varepsilon + 5\log(N)]$ has circuits of size $N^{1+\delta}$.*

In terms of techniques, the proof of Theorem 3 adapts the ideas behind Theorem 1 and combines them with a new ingredient: highly efficient error-correcting codes that can be encoded and decoded by algorithms of quasi-linear complexity [23]. Such error-correcting codes can be used together with the notion of Kt complexity to reduce the proof of Theorem 3 to a scenario that is similar to the one in the proof of Theorem 1. An important distinction is that we reduce to a certain compressed language that is in EXP instead of NP, and the argument uses boolean circuits instead of boolean formulas.

The proofs of Theorem 3 and Corollary 4 can be found in the full version of our work [20].

Magnification from sub-linear average-case lower bounds. The results discussed above require lower bounds that are super-linear in the number N of input bits. Our next theorem shows that in some settings hardness magnification also follows from *sub-linear* lower bounds.

Let $\mathsf{MCSP}[s]$ be the (standard) Minimum Circuit Size Problem over N-bit inputs whose YES instances are truth tables of functions with circuit complexity at most $s(n)$. We consider the most natural notion of average-case complexity for MCSP over the uniform distribution, as introduced by [1]. A (zero-error) average-case algorithm or device A for $\mathsf{MCSP}[s]$ is a deterministic procedure that always outputs a value in $\{0, 1, \text{“?”}\}$, and that satisfies the following conditions: (1) A is never incorrect; and (2) $\Pr_x[A(x) \neq \text{“?”}] \geq 1/2$. (The constant $1/2$ is arbitrary.) In other words, the procedure provides a 0/1 answer for a non-trivial fraction of input truth tables, and never makes a mistake.

Theorem 5. *If $\mathsf{MCSP}[2^{\sqrt{n}}]$ on N-bit inputs cannot be solved on average with zero error by formulas of size $o(N)$, then NP does not have polynomial size formulas.*

It is not hard to show that $\mathsf{MCSP}[s]$ is not in $\mathsf{SIZE}[\mathsf{poly}(N)]$ on average under standard cryptographic assumptions. Indeed, as observed in [1], the existence of average-case algorithms for MCSP is equivalent to the existence of natural proofs.

This magnification result is in sharp contrast to a recent lower bound from [1] showing that $\mathsf{MCSP}[2^{\sqrt{n}}]$ cannot be solved in the worst case by formulas of nearly quadratic size. As a consequence, there is a dramatic distinction between establishing *worst-case super-linear* lower bounds and establishing *average-case sub-linear* lower bounds for MCSP. Also note that much stronger $N^{3-o(1)}$ average-case formula size lower bounds are known for explicit problems [24].

The proof Theorem 5 is elementary, and is reminiscent of an idea used in the proof of the main result from [3]. Our new conceptual insight is in exploring this argument from the perspective of hardness magnification. Details appear in the full version [20].

Sub-linear randomized lower bounds and NP vs. BPP. Our next result in the sub-linear regime is with respect to worst-case probabilistic computations. We say that a randomized algorithm A computes a function f if on every input x, $\Pr[A(x) = f(x)] \geq 2/3$. For the definition of (sub-linear) randomized computation, we take any uniform model of computation that allows random-access to the input string.

Recall the definition of the problem (α, β)-$\mathsf{MCSP}[s]$ introduced above. We show the following magnification result in the setting of sub-linear randomized computations.

Theorem 6. *Let $1/2 < \beta < \alpha \leq 1$ be constants, $c \in \mathbb{N}$, and $s(n) \leq 2^{n^\gamma}$, where $\gamma < 1$. If there is $\varepsilon > 0$ such that (α, β)-$\mathsf{MCSP}[s]$ on N-bit inputs cannot be computed by a (two-sided error) randomized algorithm running in time $2^{(\log N)^{1-\varepsilon}}$, then* $\mathsf{NP} \nsubseteq \mathsf{BPP}$.

We note that stronger lower bounds are known against randomized computations with random-access to the input string for other (explicit) computational problems. Indeed, any reasonable model of computation of this form can be simulated by (non-uniform) randomized branching programs. If the original randomized algorithm uses time T and memory S, so does the corresponding distribution of deterministic branching programs (i.e., the longest path has length $\leq T$ and there are $\leq 2^S$ nodes in any branching program in the support of the distribution). In [25, Corollary 6.7], it is proved that an explicit N-bit boolean function in P requires two-sided error randomized branching programs running in super-linear time $T(N) = \omega(N)$ if the number of nodes in the branching program is $\leq 2^{N^{1-\Omega(1)}}$. This lower bound is much stronger than the one required in Theorem 6.

The proof of Theorem 6 is along the lines of the random projection argument sketched for the proof of Theorem 1.

Vertex Cover and Magnification. The results discussed before concerned meta-computational problems such as MCSP and its variants. We discuss now a hardness magnification result for Vertex Cover, a well-known graph-theoretic problem in NP.

Recall that in the Vertex-Cover problem, the input is a pair (G, w), where $G \in \{0,1\}^{\binom{n}{2}}$ encodes the adjacency matrix of an undirected n-vertex graph $G = (V, E)$, and $w \in \{0,1\}^{\log n}$ encodes in binary an integer parameter $k \geq 0$. The pair (G, w) is a positive instance of Vertex-Cover if there exists $S \subseteq V$, $|S| \leq k$, such that every $e = \{u, v\} \in E$ intersects S. Similarly, for $k = k(n)$ we let k-Vertex-Cover be the same computational problem with the exception that the parameter k is fixed in advance and is not part of the input. We use $m = \Theta(n^2)$ to denote the total input length of an instance of k-Vertex-Cover on n-vertex graphs.

Theorem 7. *Let $k(n) = (\log n)^C$, where $C \in \mathbb{N}$ is arbitrary. If for every $d \geq 1$ there exists $\varepsilon > 0$ such that k-Vertex-Cover $\notin \mathsf{AC}^0_d[m^{1+\varepsilon}]$, then* $\mathsf{NP} \nsubseteq \mathsf{NC}^1$.

Under ETH, k-Vertex-Cover cannot be solved in time $2^{o(k)} \cdot \mathsf{poly}(m)$ ([26]; see e.g. [27, Theorem 29.5.9]). Therefore, it is plausible that k-Vertex-Cover is not in P if $k = \omega(\log n)$. Moreover, using the NP-completeness of Vertex-Cover and a simple translation argument, for any $\varepsilon > 0$ there is a constant C such that if $k = (\log n)^C$, then k-Vertex-Cover is not in P unless CNF-SAT $\in \mathsf{DTIME}[2^{n^\varepsilon}]$. Consequently, while Theorem 7 asks for a barely non-trivial lower bound in a very weak circuit model, standard assumptions imply much stronger results.

For larger $k(n)$, we can establish a connection to time-space trade-off results (see e.g. [28, 29]). The next statement assumes a model where the algorithm has random-access to the input string. This is compatible with existing time-space trade-offs. For concreteness, one can use Turing Machines with a read-only input tape, a constant number of working tapes, and a special tape used to address the input string. Recall that $\mathsf{DTISP}[t(n), s(n)]$ denotes the class of languages that can be decided by an algorithm that simultaneously runs in time $O(t(n))$ and space $O(s(n))$.

Theorem 8. *Let $k(n) = n^{o(1)}$. If there exists $\varepsilon > 0$ such that k-Vertex-Cover $\notin \mathsf{DTISP}[m^{1+\varepsilon}, m^{o(1)}]$, where the input is an n-vertex graph represented by an adjacency matrix of bit length $m = \Theta(n^2)$, then* $\mathsf{P} \neq \mathsf{NP}$.

The reader familiar with time-space trade-offs might recall that it is known unconditionally that the Vertex Cover problem is not in $\mathsf{DTISP}[m^{1.8}, m^{o(1)}]$ (see e.g. [29]). However, such statement has two important differences in comparison to Theorem 8. First, the input encoding is different. Existing lower bounds employ a list of vertices followed by a list of edges. Secondly, the parameter $k(n)$ is much closer to n.

The techniques employed in the proof of these two results explore a connection to *kernelization*, a widely-investigated method from parameterized complexity (see e.g. [27]). Recall that a kernelizable (parameterized) problem can be reduced in polynomial time to a much smaller instance of itself, where the new input size m' can be upper bounded by a function of k. It is well-known that k-Vertex-Cover is kernelizable. We explore this idea from the point of view of *hardness magnification*. Indeed, while kernelizablity has been used mainly algorithmically, our insight here is that an extremely efficient implementation has consequences for lower bounds.

In order to explain the approach, we sketch some high-level ideas used in the proof of Theorem 7. The result is established in the contrapositive. In other words, under the assumption that $\mathsf{NP} \subseteq \mathsf{NC}^1$, we must compute k-Vertex-Cover using almost-linear size constant-depth circuits. Let T be a kernelization routine for k-Vertex-Cover which maps inputs represented by m bits and with a parameter k to an instance of Vertex-Cover whose size is $\mathrm{poly}(k)$. Since by assumption NP is contained in NC^1, standard results imply that NP can be computed by sub-exponential size circuits of sufficiently large constant depth. Consequently, using that Vertex-Cover $\in$ NP and $k(n) = (\log n)^C$, we can solve the new instance obtained via kernelization by constant-depth circuits of size at most m (where $m = \Theta(n^2)$ is the original input length). Our argument would be complete if we could also implement the routine T using almost-linear size AC^0 circuits. This part of the argument requires low-level implementations, and relies on fairly efficient computations that can be done by such circuits with a sub-linear number of (unbounded fan-in) gates.

The reader is referred to the full version [20] for proofs of Theorems 7 and 8.

Nontrivial lower bounds for Satisfiability and NEXP vs. BPP. Our last hardness magnification example holds for the standard formulation of the 3-SAT problem. However, unlike the earlier results which magnify from worst-case or average-case lower bounds, the main result in this section magnifies from lower bounds against polynomial-time refuters, a notion defined by Kabanets [30].

Intuitively, the theorem says that if any probabilistic quasi-linear time algorithm for SAT can be efficiently refuted in the sense that there is a polynomial-time algorithm that infinitely often produces either YES instances on which the algorithm does not accept with high probability or NO instances for which the algorithm does not reject with high probability, then NEXP is different from BPP. The antecedent here is a "mild" lower bound against quasi-linear time algorithms, while the consequent is a more significant lower bound against any polynomial-time algorithm, but for a larger class. While the requirement to prove a lower bound against refuters might appear much stronger than the requirement to prove a worst-case lower bound, the work of [31] (see also [32]) shows that these requirements are *essentially equivalent* when the refuter has more resources than the algorithm, as in our case. The reason we are unable to relax our requirement to a worst-case lower bound is that it is important in our argument that the refuter is deterministic while the algorithm is allowed to be randomised.

We use $|\phi|$ to denote the bitlength complexity of a 3-SAT formula ϕ.

Theorem 9. *Suppose that for every probabilistic machine M that on inputs of length m halts in time $m \cdot \mathrm{polylog}(m)$ and uses $\mathrm{polylog}(m)$ many random bits, there is a deterministic polynomial-time machine N such that for infinitely many m, $N(1^m)$ outputs a* 3-CNF *formula ϕ_m where $|\phi_m| = m$ for which either ϕ_m is satisfiable and $M(\phi_m)$ rejects with probability $> 1/m$ or ϕ_m is unsatisfiable and $M(\phi_m)$ accepts with probability $> 1/m$. Then* $\mathsf{NEXP} \nsubseteq \mathsf{BPP}$.

Modulo the use of refuters, Theorem 9 provides a bridge between two frontiers in complexity theory: establishing non-trivial lower bounds for Satisfiability against unrestricted algorithms, and understanding the power of randomness in computation. Note that much stronger running time lower bounds are know for Satisfiability in the standard sense (i.e., without refuters), but they only hold under the assumption that the algorithm uses a sub-linear amount of memory (see e.g. [28, 29]).

This is our most technically demanding proof. In terms of results, the argument relies on efficient versions of the PCP theorem where the reduction runs in quasi-linear time, and on the Easy Witness Lemma of [33]. Other concepts that play a role are the notion of Kt complexity discussed above, a related notion of KT Kolmogorov complexity introduced by [34], and a random projection of clauses that defines a randomized reduction to a certain compressed satisfiability problem. Instead of presenting a high-level exposition of the proof, we discuss some analogies with the proof of Theorem 1, which motivated our main ideas. (The argument sketched below assumes background in complexity theory.)

The proof of Theorem 9 is also by contrapositive. Suppose that $\mathsf{NEXP} = \mathsf{BPP}$. Recall that for Theorem 1 we considered a version of the MCSP problem with a *gap* between positive and negative instances. One should think of each clause of the 3-SAT instance ϕ as an entry in the input truth table for $(1, 1-\delta)$-MCSP$[s]$. If we were able to create a gap between positive (satisfiable) and negative (unsatisfiable) instances of 3-SAT, we could try to exploit some of the ideas employed in the MCSP context. Under this analogy, this turns out to be precisely what is granted by the PCP theorem. (We use a very efficient version of the PCP theorem that does not affect our barely super-linear complexity requirements.) Assume from now on that we need to solve 3-SAT on instances with a gap: YES instances are satisfiable, while in any NO instance at most a $(1-\varepsilon)$-fraction of the clauses can be simultaneously satisfied.

For the next step of the analogy, the reader should relate circuits to assignments, and small circuits to assignments that are encoded by strings of low complexity. From this perspective, a random projection in the context of $(1, 1-\delta)$-$\mathsf{MCSP}[s]$ corresponds here to a random projection of input clauses. Similarly to the proof of Theorem 1, we reduce the problem to a certain compressed language L over smaller input strings, and use the complexity collapse to solve L very efficiently. There is however a crucial difference: while in the $(1, 1-\delta)$-$\mathsf{MCSP}[s]$ setting we only had to care about potential small circuits computing the input truth table, for 3-SAT we care about *all* possible satisfying assignments, and not only about those of "low complexity". In order to address this, the Easy Witness Lemma and the notion of refuters play a fundamental role.

The high-level idea is that the refuter either produces (1) a unsatisfiable formula; (2) a satisfiable formula that admits a satisfying assignment of low complexity; or (3) a satisfiable formula such that every satisfying assignment has high complexity. Since we are arguing in the contrapositive, we need to claim that no refuter succeeds. By carefully designing the compressed/sketched version of the 3-SAT problem used in our reduction, it is possible to rule out cases (1) and (2) using an analysis that is similar to the proof of Theorem 1. On the other hand, in the remaining case (3) we get that the (deterministic) refuter produces a sequence of infinitely many formulas that are satisfiable but do not admit satisfying assignments of low-complexity. By appropriately defining the notions of complexity involved in the argument, we are able to explore this refuter to define a language in NEXP that does not admit succinct witnesses in the sense of [33]. By their Easy Witness Lemma, this implies (in particular) that $\mathsf{NEXP} \neq \mathsf{BPP}$, which contradicts our initial assumption.

A detailed exposition of the proof of Theorem 9 appears in Section III.

C. Further remarks and related work

Artificial Constructions. The magnification theorems presented in the previous section hold for several *natural* problems. We note that in the deterministic worst-case setting more dramatic magnification theorems can be established for problems that are defined in an artificial way. In particular, under plausible hardness assumptions, there are explicit problems over n-bit inputs that are not in P, but showing that they require circuits of size $(1+\varepsilon)n$ in the worst-case implies $\mathsf{NP} \not\subseteq \mathsf{SIZE}[2^{n^{o(1)}}]$.[4] Unfortunately, this does not offer a better approach to lower bounds. We are not aware of a magnification result from $C \cdot n$ size lower bounds for problems of practical or theoretical interest, where C is a constant (in the deterministic worst-case setting).

Terminology. A few comments on the "magnification" terminology are in order. As opposed to other similar names considered in the literature, we use this notation to refer to results where a lower bound that is small in value with respect to a certain measure is magnified to a lower bound of larger value with respect to the same complexity measure. In some cases, our results also "escalate" to a more expressive computational model, such as in Theorems 2 and 7 (this phenomenon is observed in the so-called "lifting" theorems from communication complexity and proof complexity). Hardness magnification is in particular different from standard hardness "amplification" results such as the well-known XOR Lemma and its variants (where average-case hardness is amplified, but size bounds slightly decrease). Due to these distinctions, we believe that the term "magnification" describes our set of results in a more appropriate way.

Related work on magnification. In addition to the papers [15] and [17] mentioned above, we are aware of three other works showing magnification theorems in our sense.

Srinivasan [35] considered the problem of $n^{1-o(1)}$-approximating the size of the maximum clique in a graph, and showed that barely super-linear lower bounds against randomized algorithms imply separations such as $\mathsf{NP} \not\subseteq \mathsf{BPP}$. We refer to his work for several variants of this result. In terms of techniques, Srinivasan's proof also uses random projections, though the technical details are different compared for instance to our Theorem 1. A drawback of his magnification theorem is that it seems to be very hard to prove unconditional lower bounds for this approximating problem, not to mention the difficulty of analysing the corresponding computational model. To our knowledge, [35] was the first to show a magnification phenomenon for a natural problem previously considered in the literature.

Motivated by [15], Lipton and Williams [36] established a magnification result for the Circuit Evaluation Problem in the context of almost-linear time and sub-linear space algorithms (similar in spirit to Theorem 8, but with different parameters). In contrast, their result offers an approach to separating P from NC. The proof uses different ideas compared to our Theorem 8, and we refer to their work for more details and related results.

More recently, [37] (see also [38]) established a magnification theorem in non-commutative arithmetic circuit complexity. Interestingly, their results show the existence of a generic magnification phenomenon in this setting, where any explicit lower bound can be magnified. The argument seems to explore non-commutativity in a fundamental way.

D. Directions and open problems

We list below some questions and directions motivated by our results:

1. Formula lower bounds. There are several techniques known to yield super-linear size formula lower bounds

[4] This can be shown, for instance, using a padding argument and assuming ETH (Exponential Time Hypothesis).

(e.g. [39], [40, 41], [13, Chapter 6], [42, Chapter 4], [43, Chapter 8]), and an obvious direction (see Theorem 1) is to try to adapt them to $(1, 1-\delta)$-$\mathsf{MCSP}[s]$. A necessary step is to use the technique to prove lower bounds against the standard $\mathsf{MCSP}[s]$ problem, as done by [1]. Similarly, one can try to prove super-linear bounded-depth lower bounds for $(1, 1-\delta)$-$\mathsf{MCSP}[s]$, a direction motivated by Theorem 2.[5] While there are techniques tailored to such results (see [20] for three examples), it seems unlikely that they can be combined with magnification without substantial new ideas.

Perhaps a more reasonable goal is to prove super-linear lower bounds that, while not strong enough for hardness magnification, are at least compatible with the parameters $s(n)$ and $\delta(n)$ required for magnification. In particular, with respect to formulas and bounded-depth circuits, we pose the challenge of establishing a $\omega(N)$ lower bound. (Stronger lower bounds appear in the full version of the paper [20], but they hold for values of $s(n)$ and $\delta(n)$ that do not seem to be enough for magnification.)

2. Barriers for magnification-based lower bounds? Another pressing question is whether there are barriers such as natural proofs [3] for this technique. A related discussion appears in [15, Section 8]. It seems that the magnification approach in its general form is not naturalizable.

3. The complexity of k-Vertex-Cover. Investigate the computational complexity of this problem in weak circuit models in the regime where $\log n \leq k \leq \mathsf{poly}(\log n)$, as suggested by Theorem 7. (Note that there has been progress in understanding the circuit complexity of other parameterized graph problems such as k-Clique [44], when k is small.) Another related direction is to improve existing time-space trade-offs, in connection to Theorem 8.

4. Magnification of existing lower bounds? Our results follow a general strategy of reducing a problem of size n to a smaller instance of another explicit problem. The reduction is computed in a very efficient way, measured according to the computational model. Can we follow this strategy to magnify existing lower bounds in computational models such as formulas and bounded-depth circuits? Note that compression lower bounds in the sense of [45, 46] can be interpreted as "barriers" to magnification. But in boolean devices such as formulas, monotone circuits, and circuits of linear size compression is less understood, and magnification of existing lower bounds might perhaps be feasible.

II. Magnification for (α, β)-MCSP

In this section, we sketch the proof of a generalisation of Theorem 1 and the proof of Theorem 2. We use number of leaves to measure formula size, and let $\mathsf{Formula}[s]$ denote the set of functions computed by formulas of size at most s. We assume negations appear at the input leaves only, and that constants 0 and 1 are available in addition to the input literals. We will need the following standard result.

Proposition 10 (Bound on the Number of Circuits). *Let $s \geq n$ and $n \geq c$, where c is a large enough constant. There are at most $2^{3s \log s}$ different circuits on n input variables of size at most s.*

For string $w \in \{0,1\}^N$ and $N = 2^n$, we use $\mathsf{fn}(w)\colon \{0,1\}^n \to \{0,1\}$ to denote the boolean function encoded by w.

Lemma 11 (Magnification Lemma for Formulas). *Let $s\colon \mathbb{N} \to \mathbb{N}$ and $\delta\colon \mathbb{N} \to [0, 1/2)$ be constructive functions, where $s(n) \geq n$. Consider the promise problem $\Pi[s, \delta] = (1, 1-\delta)$-$\mathsf{MCSP}[s]$, where for every $N = 2^n$,*

$$\Pi_N^{\mathsf{yes}} = \{y \in \{0,1\}^N \mid \exists \textit{ circuit of size } \leq s(n) \textit{ for } \mathtt{fn}(y)\},$$

$$\Pi_N^{\mathsf{no}} = \{y \in \{0,1\}^N \mid \nexists \textit{ circuit of size } \leq s(n) \textit{ that } (1-\delta(n))\textit{-approximates } \mathtt{fn}(y)\}.$$

Then, for every function $\ell\colon \mathbb{N} \to \mathbb{N}$, if $\Pi[s, \delta]$ cannot be computed by formulas of size $N \cdot \ell(n)$, then there is a sequence $\{f_{m(n)}\}_{n \geq 1}$ of $m(n)$-bit functions in NP that cannot be computed by formulas of size $\leq \ell(n)$, where $m(n) = \Theta((n \cdot s(n) \cdot \log s(n))/\delta(n))$.

Proof: We consider the computational problem $\mathtt{Succinct}$-$\Pi[s,t]$, defined next. The input instances are of the form $\langle 1^n, 1^{s(n)}, (x_1, b_1), \ldots, (x_{t(n)}, b_{t(n)})\rangle$, where $x_i \in \{0,1\}^n$ and $b_i \in \{0,1\}$, $i \in [t(n)]$. Any instance of this form can be encoded by a string of length exactly $m(n) = n + 1 + s(n) + 1 + t(n) \cdot (n+1)$. The function $\mathtt{Succinct}$-$\Pi[s,t]\colon \{0,1\}^* \to \{0,1\}$ evaluates to 1 on an input string if and only if it is of the form above and there exists a circuit C over n input variables and of size at most s such that $C(x_i) = b_i$ for all $i \in [t(n)]$. Note that the problem is in NP as a function of its total input length m, provided that s and t are constructive functions.

From now on, let $t(n) \stackrel{\text{def}}{=} (3s(n) \log s(n))/\delta(n)$. Assume there exists a sequence $\{F_{m(n)}\}_{n \geq 1}$ of formulas of size $\ell(n)$ computing $\mathtt{Succinct}$-$\Pi[s,t]$, where $F_{m(n)}\colon \{0,1\}^{m(n)} \to \{0,1\}$. Recall that $N(n) = 2^n$. We construct randomized formulas E_N that separate Π_N^{yes} and Π_N^{no}. More precisely, each formula $E_N\colon \{0,1\}^N \times \{0,1\}^{n \cdot t(n)} \to \{0,1\}$ will satisfy the following conditions:

$$\text{If } w \in \Pi_N^{\mathsf{yes}} \implies \Pr_{\boldsymbol{y}}[E_N(w, \boldsymbol{y}) = 1] \;=\; 1, \qquad (1)$$

$$\text{If } w \in \Pi_N^{\mathsf{no}} \implies \Pr_{\boldsymbol{y}}[E_N(w, \boldsymbol{y}) = 1] \;<\; 1/2. \qquad (2)$$

Each formula E_N computes as follows. It interprets its random input y as a sequence of $t(n)$ strings $y_1, \ldots, y_{t(n)} \in \{0,1\}^n$. From such strings and the input string $w \in \{0,1\}^N$, E_N produces an instance

[5] Note however that the regime of s is different in the corresponding magnification results.

$z_{w,y} \in \{0,1\}^{m(n)}$ of $\texttt{Succinct-}\Pi[s,t]$ given by $z_{w,y} \overset{\text{def}}{=} \langle 1^n, 1^{s(n)}, (y_1, (\mathsf{fn}(w))(y_1)), \ldots, (y_{t(n)}, (\mathsf{fn}(w))(y_{t(n)}))\rangle$, where $(\mathsf{fn}(w))(y_i)$ is the value of the function represented by w on the input y_i, i.e., the bit of w indexed by y_1. Finally, the formula E_N outputs $F_{m(n)}(z_{w,y})$.

We argue next that conditions (1) and (2) are satisfied by E_N. If $w \in \Pi_N^{\mathsf{yes}}$, then w is computed by some circuit C of size at most $s(n)$. Consequently, for every choice y of the random string $\boldsymbol{y}$, C is also consistent with the input pairs encoded by $z_{w,y}$. It follows that $E_N(w,y) = F_{m(n)}(z_{w,y}) = 1$. On the other hand, let $w \in \Pi_N^{\mathsf{no}}$ be a negative instance, and fix an arbitrary circuit C on n input variables and of size at most $s(n)$. Since $\Pr_{\boldsymbol{x}}[C(\boldsymbol{x}) \neq (\mathsf{fn}(w))(\boldsymbol{x})] \geq \delta(n)$, it follows that the probability that C agrees with $\mathsf{fn}(w)$ on $t(n)$ independent random inputs $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t(n)}$ is less than $(1-\delta(n))^{t(n)} \leq e^{-\delta(n)t(n)} \leq e^{-3s(n)\log(s(n))}$, using our choice of $t(n)$. Therefore, by a union bound on the number of circuits of size at most $s(n)$ (Proposition 10), it follows that condition (2) is also satisfied.

Our next step is a derandomization of the formulas E_N using a standard argument. Let G_N be the formula that computes the conjunction of N independent copies of E_N. In other words, $G_N \colon \{0,1\}^N \times (\{0,1\}^{n \cdot t(n)})^N \to \{0,1\}$, and $G_N(w, y^{(1)}, \ldots, y^{(N)}) \overset{\text{def}}{=} \bigwedge_{i=1}^N E_N(w, y^{(i)})$. Clearly, $G_N(w, \vec{\boldsymbol{y}})$ accepts a string $w \in \Pi_N^{\mathsf{yes}}$ with probability 1. On the other hand, it accepts a string $w \in \Pi_N^{\mathsf{no}}$ with probability strictly less than 2^{-N}. Fix $\alpha \in \{0,1\}^{N \cdot (n \cdot t(n))}$ to be a string that correctly derandomizes G_N on inputs from $\Pi_N^{\mathsf{yes}} \cup \Pi_N^{\mathsf{no}} \subseteq \{0,1\}^N$, and let $G_N^\star$ be a formula computing the corresponding function $G_N^\star \colon \{0,1\}^N \to \{0,1\}$.

In order to complete the proof, it is enough to upper bound the formula size of $G_N^\star$. Recall that $F_{m(n)}$ has formula size $\ell(n)$. Each component $E_N(w, y^{(i)})$ of $G_N^\star$ has been fixed to $E_N(w, \alpha^{(i)})$, where $\alpha^{(i)} \in \{0,1\}^{n \cdot t(n)}$ denotes the i-th section of the string α. Therefore, it is enough to replace the leaves of the formula $F_{m(n)}$ by constants and appropriate literals formed from w in order to compute $E_N(w, \alpha^{(i)})$. It follows that each $E_N(w, \alpha^{(i)})$ can be computed by a formula of size at most $\ell(n)$. Consequently, $G_N^\star$ can be computed by a formula containing at most $N \cdot \ell(n)$ leaves. □

Note that Lemma 11 immediately implies Theorem 1.

Let $\mathsf{AC}_d^0[M]$ denote the class of unbounded fan-in depth-d size-M circuits. We can use a similar argument to establish Theorem 2, as described next.

Proof Sketch: We employ the same approach and notation of Lemma 11. Suppose that $\mathsf{NP} \subseteq \mathsf{NC}^1$, and let $\{F_m\}$ be a family of polynomial size formulas for the corresponding problem $\texttt{Succinct-}\Pi[s,t]$. Take $d' \in \mathbb{N}$ large enough so that each F_m can be computed by a depth-d' circuit of size, say, $2^{O(\sqrt{n})}$ (this is possible thanks to a folklore simulation of formulas by bounded-depth circuits; see [20] for details). Note that a fixed d' independent of n with this property must exist, since for our choice of parameters each formula F_m has $\mathsf{poly}(n)$ input variables. Proceeding exactly as in the proof of Lemma 11 but using bounded-depth circuits instead of formulas, we get that for some $d \in \mathbb{N}$, $\Pi[s,\delta]$ can be computed by depth-d circuits of size $N^{1+o(1)}$. This completes the proof. □

We note that one can show under standard cryptographic assumptions and an appropriate choice of parameters that $(1, 1-\delta)$-$\mathsf{MCSP}[s]$ is hard even for polynomial size circuits. The next proposition follows ideas and notation from [3].

Proposition 12 (Hardness of $(1, 1-\delta)$-$\mathsf{MCSP}[s]$)**.** *Suppose there exists a polynomial time one-way function $f \colon \{0,1\}^* \to \{0,1\}^*$ secure against circuits of size 2^{n^ε}, where $\varepsilon > 0$. Then, if $k \in \mathbb{N}$ is large enough, $(1, 2/3)$-$\mathsf{MCSP}[n^k]$ cannot be computed by circuits of size $N^{O(1)}$.*

III. Magnification for Satisfiability

In this section, we prove Theorem 9. We fix a computational model with random-access to the input string. The particular details of the model are not so relevant, since we will not distinguish polylogarithmic factors in the context of uniform computations.

First, we recall some notions of time-bounded Kolmogorov complexity needed in the proof (cf. [34]).

Definition 13. *Let U be a universal machine. For any string $x \in \{0,1\}^*$, $Kt(x)$ is the minimum over $|p| + \log(t)$ such that $U(p)$ outputs x within t steps.*

Definition 14. *Let U be a universal machine. For any string $x \in \{0,1\}^*$, $KT(x)$ is the minimum over $|p| + t$ such that $U(p,i) = x_i$ in at most t steps for each $i, 1 \leq i \leq |x|$, and moreover $U(p,i) = \perp$ for all $i > |x|$.*

The following inequality is Proposition 1 in [34], where KT-complexity was introduced.

Proposition 15 ([34])**.** *For all $x \in \{0,1\}^*$, $Kt(x) \leq 3KT(x)$.*

We will employ an efficient version of the PCP theorem where the reduction runs in quasi-linear time.

Theorem 16 ([47, 48, 49])**.** *For some constant $\epsilon > 0$, there is a function f computable in time $O(m \operatorname{polylog}(m))$ and a function g computable in polynomial time such that:*

1) *If ϕ is a satisfiable 3-CNF, $f(\phi)$ is a satisfiable 3-CNF.*
2) *If ϕ is an unsatisfiable 3-CNF, then at most a $(1-\epsilon)$ fraction of clauses of $f(\phi)$ are simultaneously satisfiable.*
3) *If ϕ is a satisfiable 3-CNF and w is a satisfying assignment to ϕ, then $g(\phi, w)$ is a satisfying assignment to $f(\phi)$.*

Finally, we will need the Easy Witness Lemma of [33]. The lemma is usually stated in terms of witnesses of YES instances having small circuit complexity when interpreted as truth tables, but we find it more convenient to use an equivalent formulation in terms of KT complexity.

Lemma 17 ([33]). *Let $L \in \mathsf{NEXP}$ be any language, and let R be a polyomial-time computable relation such that $x \in L$ iff there is a string $y, |y| = 2^{|x|^k}$ such that $R(x, y) = 1$, where k is a fixed constant. If $\mathsf{NEXP} \subseteq \mathsf{SIZE}(poly)$, then for each $x \in L$, there is $y, |y| = 2^{|x|^k}$ such that $R(x, y) = 1$ and $KT(y) = \mathrm{poly}(|x|)$.*

We are now ready to prove the result, stated again for convenience of the reader.

Theorem 18 (Magnification for SAT). *Suppose that for every probabilistic machine M halting in time $m \,\mathrm{polylog}(m)$ and using $\mathrm{polylog}(m)$ many random bits, there is a deterministic polynomial-time machine N such that for infinitely many m, $N(1^m)$ outputs a 3-CNF formula $\phi_m, |\phi_m| = m$ for which either ϕ_m is satisfiable and $M(\phi_m)$ rejects with probability $> 1/m$ or ϕ_m is unsatisfiable and $M(\phi_m)$ accepts with probability $> 1/m$. Then $\mathsf{NEXP} \neq \mathsf{BPP}$.*

Proof: Assume, for the sake of contradiction, that $\mathsf{NEXP} = \mathsf{BPP}$. Let c be a constant such that $\mathsf{NTIME}(2^n) \subseteq \mathsf{BPTIME}(n^c)$ – the existence of such a constant c follows from $\mathsf{NEXP} = \mathsf{BPP}$ using the fact that there is a language complete for $\mathsf{NTIME}(2^n)$ under deterministic linear-time reductions.

We describe a probabilistic machine M that attempts to solve 3-SAT, and show that any deterministic polynomial-time refuter for M yields a contradiction.

Let ϕ be a 3-CNF formula with $|\phi| = m$ (we always use $|\phi|$ to denote the bitlength of ϕ in a standard representation) and q variables. Let $k = \log(m)^a$, where a is a constant to be determined later. M behaves as follows on ϕ. It applies the reduction f from Theorem 16 to ϕ to produce a 3-CNF formula ϕ'. Since f runs in time $O(m \,\mathrm{polylog}(m))$, we have that $|\phi'| = O(m \,\mathrm{polylog}(m))$. M then samples k^2 clauses from ϕ' independently and uniformly at random, and forms their conjunction ϕ''. Note that not all variables in ϕ' will appear in ϕ'' if k^2 is much smaller than q – the variables retain their original indices in ϕ''.

Define an auxiliary language L' as follows. Consider an input $w = (\psi, 1^\ell, q')$, where ψ is a 3-CNF formula such that all indices of variables appearing in ψ belong to $[q']$ and ℓ and q' are positive integers with $\ell \leq q' \leq 2^\ell$. This string belongs to L' if and only if there is a string $w' \in \{0,1\}^{q'}$ such that $Kt(w') \leq \ell$ and w' satisfies ψ when interpreted as a Boolean assignment to variables with indices in $[q']$. L' can be solved in deterministic time $2^{O(\ell)}|\psi| \,\mathrm{polylog}(|\psi|)$ simply by enumerating all strings of length q' with Kt complexity at most ℓ, which can be done in time $2^{O(\ell)}$, and checking for each one if it is a satisfying assignment of ψ. Each such check can be done in time $|\psi| \,\mathrm{polylog}(|\psi|)$. In particular, $L' \in \mathsf{DTIME}(2^{O(n)})$, where n is the total length of the input to L', and hence by assumption $L' \in \mathsf{BPTIME}(n^c)$. Let M' be a probabilistic machine deciding L' with error 2^{-n}.

M simulates M' on input $(\phi'', 1^{2k}, q')$, where q' is the number of variables in ϕ', accepting if M' accepts and rejecting if M' rejects. The total time taken by M is $O(m \,\mathrm{polylog}(m))$, since applying the reduction f takes time $m \,\mathrm{polylog}(m)$ and the simulation of the machine M' takes time $\mathrm{poly}(k) = \mathrm{polylog}(m)$. Thus M is a probabilistic machine running in time $m \,\mathrm{polylog}(m)$ that attempts to solve 3-SAT. In addition, it is easy to see that M employs at most poly-logarithmic many random bits, given our choice of k and the input length on which the machine M' is invoked. We would like to show that any deterministic polynomial-time refuter for M, i.e., any deterministic polynomial-time machine that infinitely often outputs an instance ϕ on which M does not give the correct answer with high probability, can be used to derive a contradiction.

Indeed, suppose there is such a polynomial-time machine N, that for infinitely many m on input 1^m, outputs a formula ϕ_m such that either ϕ_m is satisfiable and M rejects ϕ_m with probability $> 1/m$, or ϕ_m is unsatisfiable and M accepts ϕ_m with probability $> 1/m$.

Note that each formula ϕ_m output by N on input 1^m has Kt complexity $O(\log(m))$ (when interpreted as a string). The reason is that the code for the machine N together with the standard $\log(m)$-bit description of the number m can be used as a program p to generate ϕ_m; moreover the generation is accomplished in time $t = \mathrm{poly}(m)$. Recall that the Kt complexity of the string ϕ_m is the minimum over $|p| + \log(t)$ such that the universal machine U generates ϕ_m from p within t steps. In our case, we have $|p| = O(\log(m))$ and $t = \mathrm{poly}(m)$, hence the Kt complexity of ϕ_m is $O(\log(m))$.

We will argue that for any input $\phi_m, |\phi_m| = m$ such that $Kt(\phi_m) = O(\log(m))$, the machine M solves ϕ_m correctly if ϕ_m is unsatisfiable, and also solves ϕ_m correctly if ϕ_m is satisfiable and has a satisfying assignment w such that $Kt(w) \leq k$. We will then use these two facts to derive a contradiction to the existence of the refuter N.

We first argue that M solves ϕ_m correctly if ϕ_m is unsatisfiable, when $|\phi_m| = m$ and $Kt(\phi_m) = O(\log(m))$. Let ϕ' be the 3-CNF produced by M from applying the reduction f to ϕ_m and ϕ'' be the 3-CNF produced by then randomly sampling k^2 clauses and taking their conjunction. We argue that with overwhelmingly high probability, ϕ'' does not have a satisfying assignment w' such that $Kt(w') \leq 2k$.

Indeed, we have that for any fixed assignment y to the variables of ϕ', y satisfies at most a $(1 - \epsilon)$ fraction of the clauses of ϕ', where ϵ is the constant given by Theorem 16. Thus the probability that y satisfies a randomly sampled clause is at most $(1 - \epsilon)$. It follows that the probability that y satisfies all k^2 randomly sampled clauses occurring in ϕ'' is at most $(1-\epsilon)^{k^2} = 2^{-\Omega(k^2)}$, since the clauses are sampled independently and uniformly at random.

Now, by a union bound over all assignments w' such that $Kt(w') \leq 2k$, we have that the probability that there exists such a w' satisfying all clauses in ϕ'' is at most $2^{2k}2^{-\Omega(k^2)} = 2^{-\Omega(k^2)}$. Hence with probability at least $1-2^{-\Omega(k^2)}$ over the internal randomness of the machine M, the formula ϕ'' has no satisfying assignment w' such that $Kt(w') \leq 2k$. Note that for every such ϕ'', by correctness of the machine M', we have that M' rejects with probability at least $1-2^{-n}$, where n is the length of the input to M'. This probability is at least $1-2^{-k}$, hence by using Bayes' theorem, we have that with probability at least $1-2^{-\Omega(k)} > 1-1/m$ (for a chosen sufficiently large), M rejects as claimed.

We next argue that M solves ϕ_m correctly if ϕ_m is satisfiable and has a satisfying assignment w such that $Kt(w) \leq k$. Consider such a formula ϕ_m. The ordered pair $< \phi_m, w >$ has Kt complexity at most $k + O(\log(m))$ by sub-additivity of Kt complexity, which is at most $1.5k$ for large enough m when a is chosen to be greater than 1.

Now consider the formula ϕ' produced by applying the reduction f to ϕ_m. By Theorem 16, $g(\phi_m, w)$ is a satisfying assignment to ϕ'. Since (ϕ_m, w) has Kt complexity at most $1.5k$, and since g runs in $\mathrm{poly}(m)$ time, we have that the Kt complexity of $w' = g(\phi_m, w)$ is at most $2k$, for large enough m and when a is chosen to be greater than 1. Note that since w' satisfies ϕ', it also satisfies each sub-sampled formula ϕ''. Hence M' is always simulated on a positive instance of L', regardless of the internal randomness of L, and consequently M accepts with probability greater than $1-1/m$, using the correctness of M'.

Thus the only case where M does not solve ϕ_m correctly with error $< 1/m$ is when ϕ_m is satisfiable and does not have a satisying assignment of Kt complexity at most k. We will show that this case leads to a contradiction. We sketch the definition of a relation $R(u, y)$ as follows. The first input u is viewed as a pair (D, x), where D is a machine, and x is a number. We assume that the sizes of D and x are roughly $\log m$, where $\widetilde{\Theta}(m)$ is the total input length expected by R on a valid input pair (u, y). The relation R accepts its input pair (u, y) iff $D(1^x)$ when executed for $m^{O(\log m)}$ steps outputs a formula ϕ_x such that $\phi_x(y) = 1$. Note that R is an NEXP-verifier. Consequently, by Lemma 17 and using the assumption that $\mathsf{NEXP} = \mathsf{BPP} \subseteq \mathsf{SIZE}(\mathsf{poly})$, there is a constant b such that for every $u = (D, x)$, if $D(1^x)$ outputs a satisfiable formula ϕ_x during its computation then there is a satisfying assignment y for ϕ_x such that $KT(y) \leq \log(m)^b$. But now using Proposition 15 and choosing $a > b$, if fix the input D to be the code of the refuter, we have a contradiction to the assumption that there are infinitely many m for which ϕ_m is satisfiable and does not have a satisfying assignment of Kt complexity at most k. □

Acknowledgements

We are grateful to Jan Krajíček, Ján Pich and Ninad Rajgopal for helpful discussions and comments.

References

[1] S. Hirahara and R. Santhanam, "On the average-case complexity of MCSP and its variants," in *Computational Complexity Conference* (CCC), 2017, pp. 7:1–7:20.

[2] L. Levin, "Randomness conservation inequalities; information and independence in mathematical theories," *Information and Control*, vol. 61, pp. 15–37, 1984.

[3] A. A. Razborov and S. Rudich, "Natural proofs," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 24–35, 1997.

[4] M. Ajtai, "$\sum_1^1$-formulae on finite structures," *Annals of Pure and Applied Logic*, vol. 24, no. 1, pp. 1–48, 1983.

[5] M. L. Furst, J. B. Saxe, and M. Sipser, "Parity, circuits, and the polynomial-time hierarchy," *Mathematical Systems Theory*, vol. 17, no. 1, pp. 13–27, 1984.

[6] A. A. Razborov, "Lower bounds on the monotone complexity of some Boolean functions," *Doklady Akademii Nauk SSSR*, vol. 281, pp. 798–801, 1985, english translation in: Soviet Mathematics Doklady 31:354–357, 1985.

[7] K. Iwama and H. Morizumi, "An explicit lower bound of $5n - o(n)$ for Boolean circuits," in *Symposium on Mathematical Foundations of Computer Science* MFCS, 2002, pp. 353–364.

[8] J. Håstad, "The shrinkage exponent of de Morgan formulas is 2," *SIAM J. Comput.*, vol. 27, no. 1, pp. 48–64, 1998.

[9] R. Williams, "Improving exhaustive search implies superpolynomial lower bounds," *SIAM J. Comput.*, vol. 42, no. 3, pp. 1218–1244, 2013.

[10] ——, "Nonuniform ACC circuit lower bounds," *J. ACM*, vol. 61, no. 1, pp. 2:1–2:32, 2014.

[11] T. P. Baker, J. Gill, and R. Solovay, "Relativizatons of the $\mathsf{P} =? \mathsf{NP}$ question," *SIAM J. Comput.*, vol. 4, no. 4, pp. 431–442, 1975.

[12] S. Aaronson and A. Wigderson, "Algebrization: A new barrier in complexity theory," *Transactions on Computation Theory*, vol. 1, no. 1, 2009.

[13] S. Jukna, *Boolean Function Complexity - Advances and Frontiers*. Springer, 2012.

[14] A. Bogdanov and A. Rosen, "Pseudorandom functions: Three decades later," in *Tutorials on the Foundations of Cryptography*, 2017, pp. 79–158.

[15] E. Allender and M. Koucký, "Amplifying lower bounds by means of self-reducibility," *J. ACM*, vol. 57, no. 3, pp. 14:1–14:36, 2010.

[16] R. Williams, "Some estimated likelihoods for computational complexity," *Springer LNCS 10,000 Volume*, 2018.

[17] M. Müller and J. Pich, "Feasibly constructive proofs of succinct weak circuit lower bounds," *Electronic Colloquium on Computational Complexity* (ECCC), TR-17-144, 2017.

[18] N. Nisan and A. Wigderson, "Hardness vs randomness," *J. Comput. Syst. Sci.*, vol. 49, no. 2, pp. 149–167, 1994.

[19] S. Arora and B. Barak, *Complexity Theory: A Modern Approach*. Cambridge University Press, 2009.

[20] I. C. Oliveira and R. Santhanam, "Hardness magnification for natural problems," *Electronic Colloquium on Computational Complexity* (ECCC), TR-18-139, 2018.

[21] M. Kearns and U. Vazirani, *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[22] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, and D. Ronneburger, "Power from random strings," *SIAM J. Comput.*, vol. 35, no. 6, pp. 1467–1493, 2006.

[23] D. A. Spielman, "Linear-time encodable and decodable error-correcting codes," *IEEE Trans. Information Theory*, vol. 42, no. 6, pp. 1723–1731, 1996.

[24] I. Komargodski, R. Raz, and A. Tal, "Improved average-case lower bounds for de Morgan formula size: Matching worst-case lower bound," *SIAM J. Comput.*, vol. 46, no. 1, pp. 37–57, 2017.

[25] P. Beame, M. E. Saks, X. Sun, and E. Vee, "Time-space trade-off lower bounds for randomized computation of decision problems," *J. ACM*, vol. 50, no. 2, pp. 154–195, 2003.

[26] R. Impagliazzo, R. Paturi, and F. Zane, "Which problems have strongly exponential complexity?" *J. Comput. Syst. Sci.*, vol. 63, no. 4, pp. 512–530, 2001.

[27] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, ser. Texts in Computer Science. Springer, 2013.

[28] L. Fortnow, R. J. Lipton, D. van Melkebeek, and A. Viglas, "Time-space lower bounds for satisfiability," *J. ACM*, vol. 52, no. 6, pp. 835–865, 2005.

[29] R. Williams, "Algorithms and resource requirements for fundamental problems," Ph.D. dissertation, Carnegie Mellon University, 2007.

[30] V. Kabanets, "Easiness assumptions and hardness tests: Trading time for zero error," *J. Comput. Syst. Sci.*, vol. 63, no. 2, pp. 236–252, 2001.

[31] D. Gutfreund, R. Shaltiel, and A. Ta-Shma, "If NP languages are hard on the worst-case, then it is easy to find their hard instances," *Computational Complexity*, vol. 16, no. 4, pp. 412–441, 2007.

[32] A. Atserias, "Distinguishing SAT from polynomial-size circuits, through black-box queries," in *Conference on Computational Complexity* (CCC), 2006, pp. 88–95.

[33] R. Impagliazzo, V. Kabanets, and A. Wigderson, "In search of an easy witness: exponential time vs. probabilistic polynomial time," *J. Comput. Syst. Sci.*, vol. 65, no. 4, pp. 672–694, 2002.

[34] E. Allender, "When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity," in *Foundations of Software Technology and Theoretical Computer Science* FSTTCS, 2001, pp. 1–15.

[35] A. Srinivasan, "On the approximability of clique and related maximization problems," *J. Comput. Syst. Sci.*, vol. 67, no. 3, pp. 633–651, 2003.

[36] R. J. Lipton and R. Williams, "Amplifying circuit lower bounds against polynomial time, with applications," *Computational Complexity*, vol. 22, no. 2, pp. 311–343, 2013.

[37] M. L. Carmosino, R. Impagliazzo, S. Lovett, and I. Mihajlin, "Hardness amplification for non-commutative arithmetic circuits," in *Computational Complexity Conference* (CCC), 2018, pp. 12:1–12:16.

[38] P. Hrubeš, A. Wigderson, and A. Yehudayoff, "Non-commutative circuits and the sum-of-squares problem," *Journal of the American Mathematical Society*, vol. 24, no. 3, pp. 871–898, 2011.

[39] A. Tal, "Formula lower bounds via the quantum method," in *Symposium on Theory of Computing* (STOC), 2017, pp. 1256–1268.

[40] I. Dinur and O. Meir, "Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity," in *Conference on Computational Complexity* (CCC), 2016, pp. 3:1–3:51.

[41] D. Gavinsky, O. Meir, O. Weinstein, and A. Wigderson, "Toward better formula lower bounds: The composition of a function and a universal relation," *SIAM J. Comput.*, vol. 46, no. 1, pp. 114–131, 2017.

[42] P. E. Dunne, *The Complexity of Boolean Networks*. Academic Press, 1988.

[43] I. Wegener, *The complexity of Boolean functions*. Wiley, 1987.

[44] B. Rossman, "Average-case complexity of detecting cliques," Ph.D. dissertation, Massachusetts Institute of Technology, 2010.

[45] A. Chattopadhyay and R. Santhanam, "Lower bounds on interactive compressibility by constant-depth circuits," in *Symposium on Foundations of Computer Science* (FOCS), 2012, pp. 619–628.

[46] I. C. Oliveira and R. Santhanam, "Majority is incompressible by $\mathrm{AC}^0[p]$ circuits," in *Conference on Computational Complexity* (CCC), 2015, pp. 124–157.

[47] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. P. Vadhan, "Robust PCPs of proximity, shorter PCPs, and applications to coding," *SIAM J. Comput.*, vol. 36, no. 4, pp. 889–974, 2006.

[48] E. Ben-Sasson and M. Sudan, "Short PCPs with polylog query complexity," *SIAM J. Comput.*, vol. 38, no. 2, pp. 551–607, 2008.

[49] I. Dinur, "The PCP theorem by gap amplification," *J. ACM*, vol. 54, no. 3, p. 12, 2007.