Electronic Colloquium on Computational Complexity, Report No. 28 (2023)

An Algorithmic Approach to Uniform Lower Bounds

Rahul Santhanam
University of Oxford

Abstract

We propose a new family of circuit-based sampling tasks, such that non-trivial algorithmic solutions to certain tasks from this family imply frontier uniform lower bounds such as "NP not in uniform ACC^0" and "NP does not have polynomial-size depth-two threshold circuits". Indeed, the most general versions of our sampling tasks have implications for central open problems such as NP vs P and PSPACE vs P.

We argue the soundness of our approach by showing that the non-trivial algorithmic solutions we require do follow from standard cryptographic assumptions. In addition, we give evidence that a version of our approach for uniform circuits is *necessary* in order to separate NP from P or PSPACE from P. We give an *algorithmic characterization* for the PSPACE vs P question: $\mathsf{PSPACE} \neq \mathsf{P}$ iff either E has sub-exponential time non-uniform algorithms infinitely often or there are non-trivial space-efficient solutions to our sampling tasks for uniform Boolean circuits.

We show how to use our framework to capture uniform versions of known non-uniform lower bounds, as well as classical uniform lower bounds such as the space hierarchy theorem and Allender's uniform lower bound for the Permanent. We also apply our framework to prove new lower bounds: NP does not have polynomial-size uniform AC^0 circuits with a bottom layer of MOD 6 gates, nor does it have polynomial-size uniform AC^0 circuits with a bottom layer of threshold gates.

Our proofs exploit recently defined probabilistic time-bounded variants of Kolmogorov complexity [LOZ22, GKLO22, LO22].

1 Introduction

1.1 Background and Motivation

The NP vs P problem [Coo71, For09] is the central problem in theoretical computer science. Despite much effort over the years, we seem to be quite far from a solution. Theoretical computer science has had many successes over the years, but as far as NP vs P is concerned, it has been hard even to come up with viable approaches to the problem.

When the problem first received attention in the 1970s, a natural approach to it was to explore analogies with computability theory, and use simulation and diagonalization techniques to achieve a separation. For example, the uncomputability of the Halting Problem is a foundational result in computability theory proved using diagonalization. A *time-bounded* version of the Halting Problem for non-deterministic machines is NP-complete, so it makes sense to try resource-bounded variants of diagonalization to separate NP and P. However, all approaches using simulation and diagonalization have been fruitless so far, and a major reason for this was identified by Baker, Gill and Solovay [BGS75] in their paper on the relativization barrier. Classical techniques in computability theory *relativize*, meaning that they continue to hold relative to an arbitrary oracle, but no solution to the NP vs P can relativize - there is an oracle A such that $\mathsf{NP} = \mathsf{P}$ relative to A, and another oracle B such that $\mathsf{NP} \neq \mathsf{P}$ relative to B [BGS75].

After the relativization barrier was identified, attention shifted to the *non-uniform* version of the NP vs P problem. The non-uniform version asks if NP has polynomial-size Boolean circuits. A negative answer implies $\mathsf{NP} \neq \mathsf{P}$, since all problems in P have polynomial-size Boolean circuits. The hope in studying Boolean circuits was that they might be easier to analyze and understand using combinatorial and algebraic techniques than

ISSN 1433-8092

is the case for uniform algorithms. Indeed, the hope was fed by a spate of results in the 1980s showing super-polynomial lower bounds against weak circuit classes, including AC^0 circuits [Ajt83, FSS84, Yao85, Hås86] , $\mathsf{AC}^0[p]$ circuits for prime p [Raz87, Smo87] and monotone circuits [Raz85].

This spate of lower bound results slowed down to a trickle in the 1990s, and even the question of proving super-polynomial lower bounds against constant-depth circuits with composite modular gates remained unsolved. In an attempt to explain the stalled progress, Razborov and Rudich [RR97] identified a further barrier: the Natural Proofs barrier. While the relativization barrier applies to traditional simulation and diagonalization approaches, the Natural Proofs barrier applies to combinatorial and algebraic techniques that were the main source of hope for showing non-uniform lower bounds. The Natural Proofs barrier rules out *constructive* approaches to circuit lower bounds that involve identifying a complexity measure that is easy to compute, low for functions with small circuits and high for random functions, assuming standard cryptographic conjectures. Essentially all known non-uniform lower bound techniques at the time involved identifying such complexity measures, and so the Natural Proofs barrier did help to explain why progress was stalled.

Since the Natural Proofs barrier was identified, there has been pessimism about the prospect of proving lower bounds in the near future, and few promising lower bound approaches have been identified. The ambitious Geometric Complexity Theory program of Mulmuley and Sohoni [MS01, Mul11] seeks to solve the Permanent vs Determinant problem, which is an algebraic analogue of NP vs P, by using representation theory and algebraic geometry to analyze symmetries of the Determinant and Permanent. A version of their approach also applies to the NP vs P problem. While Geometric Complexity Theory has led to significant new insights in algebraic complexity theory, the original approach has also faced some obstacles [BIP16], and Mulmuley himself believes that it is likely to take a very long time for NP vs P to be solved using this approach [For09].

A rare success in the theory of lower bounds over the past couple of decades is the algorithmic method [Wil10, Wil11] of Ryan Williams. Somewhat paradoxically, Williams proposed to attack the question of circuit lower bounds, i.e., proving that no efficient non-uniform algorithms exist for some task, by finding improved algorithms for a different task. To be more specific, if we wish to prove that NEXP does not have polynomial-size $\mathfrak{C}$-circuits for some circuit class $\mathfrak{C}$ with natural closure properties, then all we need to do is to solve the Satisfiability problem for $\mathfrak{C}$-circuits in barely non-trivial time, i.e., in time $2^n \mathsf{poly}(m)/n^{\omega(1)}$, where m is the circuit size and n is the number of variables. Note that the trivial brute force search algorithm for Satisfiability takes time $2^n \mathsf{poly}(m)$, so what is required is just a *super-polynomial* improvement over this trivial algorithm.

One might wonder whether an algorithmic approach via improved algorithms for Satisfiability is feasible if we are interested in lower bounds for general Boolean circuits. Perhaps no non-trivial improvement over brute force search is possible for Circuit Satisfiability? This potential objection is addressed in [Wil10] by showing that circuit lower bounds hold even if we can estimate the acceptance probability of a $\mathfrak{C}$-circuit in barely non-trivial time. This acceptance probability estimation task is known to be doable in sub-exponential time under circuit lower bound assumptions, by using ideas from the theory of derandomization [NW94, BFNW93]. Thus, in a sense, the algorithmic approach is without loss of generality when it comes to circuit lower bounds for NEXP.

In a breakthrough, Williams showed how to solve a frontier question in circuit lower bounds using the algorithmic method. He showed that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$ [Wil11], by designing algorithms for ACC^0 Satisfiability that beat brute-force search. Since then, several other lower bounds for restricted classes of circuits have been shown using the algorithmic method [Wil14, Wil18, AC19, Che19, CLW20, MW20, CR20, BHPT20]. What is particularly appealing about the algorithmic method is that humans seem more suited to constructive algorithmic thinking than to proving impossibility results, and so the reformulation of a lower bound task as an algorithmic task is likely to stimulate progress.

The original formulation of the algorithmic method [Wil10] gave a connection between non-trivial Satisfiability algorithms and circuit lower bounds for NEXP. While NEXP lower bounds are interesting from a derandomization perspective, what we desire most in complexity theory is super-polynomial size lower bounds for NP. Murray and Williams [MW20] show how to scale down the algorithmic method to derive lower bounds for NQP (non-deterministic quasi-polynomial time) against polynomial-size $\mathfrak{C}$-circuits and lower bounds for NP against fixed polynomial size $\mathfrak{C}$-circuits (where the running time of the NP algorithm depends on the size lower bound) from circuit analysis algorithms for $\mathfrak{C}$. However, their techniques do not seem to be useful in deriving super-polynomial size lower bounds for NP - it is unclear what a corresponding circuit analysis task would be.

It seems very challenging even to prove that NP does not have polynomial-size ACC^0 circuits, and no approaches to this question are known. But what if we weaken our goal to lower bounds against polynomial-size *uniform* ACC^0 circuits? Note that from the perspective of making progress toward $\mathsf{NP} \neq \mathsf{P}$, uniform lower bounds are just as good as non-uniform ones. However, we do not have any cases so far of super-polynomial uniform lower bounds for NP against a class $\mathfrak{C}$ of circuits where a corresponding non-uniform lower bound is unknown. We also don't have any plausible approaches toward showing such uniform lower bounds.

This raises the following question, which is at the core of our work.

Question: Is there an algorithmic approach[1] to long-standing open questions about uniform lower bounds?

To make this question more precise, we state a couple of criteria that we require from an "algorithmic approach". First, we would like our algorithmic task to be as close to a conventional algorithmic task as possible - a task that takes an input and produces output that satisfies some desired property. For example, cryptographic pseudo-random generators imply $\mathsf{NP} \neq \mathsf{P}$, but we do not consider the construction of such generators as a standard algorithmic task, just as the construction of *complexity-theoretic* pseudo-random generators does not count as an algorithmic approach toward circuit lower bounds for EXP. Second, we would like our approach to be sound, meaning that there should be some evidence that the algorithmic task is indeed feasible. Note that an infeasible algorithmic task, such as designing polynomial-time algorithms for an EXP-complete problem, would imply anything at all, and in particular would imply $\mathsf{NP} \neq \mathsf{P}$.

When trying to design an algorithmic approach to uniform lower bounds, it is useful to keep in mind an informal but fundamental distinction between two regimes of lower bounds - the *complexity-theoretic* regime and the *cryptographic* regime. The complexity-theoretic regime refers to situations where we are trying to show a lower bound $\mathfrak{C} \not\subseteq \mathfrak{D}$ for complexity classes $\mathfrak{C}$ and $\mathfrak{D}$, and $\mathfrak{C}$ has enough computational resources to simulate $\mathfrak{D}$. For example, the case of $\mathsf{NEXP} \not\subseteq \mathsf{SIZE}(\mathsf{poly})$ falls into the complexity-theoretic regime, since exponential-time machines can simulate polynomial-size circuits. The cryptographic regime refers to situations where $\mathfrak{C}$ does not have the resources to simulate $\mathfrak{D}$. The case of super-polynomial size uniform circuit lower bounds for NP falls into the cryptographic regime, since we are trying to show that a *fixed* problem in NP does not have uniform circuits of *arbitrary* polynomial size, and so the NP machine does not have enough resources to simulate the circuits against which a lower bound is sought.

The common feature to all known applications of the algorithmic method [Wil10, Wil11, MW20] is that the corresponding lower bounds fall in the complexity-theoretic regime. The reason is that the proof technique for establishing the connection between algorithms and lower bounds involves indirect diagonalization, culminating in an application of *hierarchy theorems*. Since hierarchy theorems require the class for which we are showing a lower bound to have more resources than the class against which we are showing a lower bound, it seems unlikely that the algorithmic method can be directly adapted to the cryptographic regime of lower bounds.

In this work, we present a new family of circuit-based sampling tasks such that solutions to tasks in this family imply uniform circuit lower bounds in the cryptographic regime for classes such as NP and PSPACE. Since these algorithmic tasks have not been considered before, there isn't a clear path yet to solving them for general Boolean formulas or circuits. As such, this is not yet a full-fledged approach to strong uniform circuit lower bounds such as $\mathsf{NP} \neq \mathsf{P}$ or $\mathsf{PSPACE} \neq \mathsf{P}$. Nevertheless, our approach does allow us to recover state-of-the-art uniform lower bounds and to prove a couple of new ones, and we believe it might be useful

[1]Note that we are interested in this paper only in algorithmic approaches to *lower bounds*. We hold the conventional belief that $\mathsf{NP} \neq \mathsf{P}$ and wish to show this using an algorithmic approach. Of course, if one wishes to show that $\mathsf{NP} = \mathsf{P}$, an algorithmic approach is completely natural, i.e., designing and analyzing a polynomial-time algorithm for SAT.

to attack frontier uniform lower bound questions such as separating NP from uniform ACC^0 or uniform TC^0_2. Moreover, we believe that the connection from algorithms to lower bounds is interesting in itself, and hope that it will stimulate further research on the sampling tasks we define.

We now proceed to describe our approach.

1.2 The Approach

We describe our approach to lower bounds for PSPACE first. Suppose we seek to prove uniform lower bounds for PSPACE against some class $\mathfrak{C}$ of circuits. The only requirement we will make of $\mathfrak{C}$ is that it can be simulated by general Boolean circuits. In order to describe the algorithmic tasks in our approach, we first introduce some terminology.

Given a $\mathfrak{C}$-circuit C on n variables of size $\mathsf{poly}(n)$, we say that C is *dense* if C accepts at least a $2/3$ fraction of all inputs of length n. We are interested in the problem of sampling satisfying assignments of $\mathfrak{C}$. Satisfying assignments of C are plentiful, so a trivial randomized algorithm works with high probability, but it might not be easy for a deterministic algorithm to find satisfying assignments. In our setting, we will *allow* the use of randomness in the algorithm.

In order not to make the task trivial, we will require that the algorithm outputs a *fixed* satisfying assignment with probability at least $n^k/2^n$ for some large enough constant k. This requirement is related to the notion of pseudo-deterministic search defined by Gat and Goldwasser [GG11]. A pseudo-deterministic algorithm is a randomized algorithm for a search problem that outputs a fixed solution with high probability, say $2/3$. In contrast, we only require that a fixed assignment is output with probability $\mathsf{poly}(n)/2^n$, which is barely non-trivial.

It is not hard to see that this task is easy if the algorithm is given full access to C and is able to simulate C. The key restriction we impose is that the algorithm does not have enough resources to simulate C, though it does have full access to C. The algorithm is given random access to the representation of C, and must run in some fixed polynomial space bound n^d, where d is independent of the size of C. This restriction turns out to be enough to imply lower bounds for PSPACE against uniform $\mathfrak{C}$-circuits.

We say that there is *space-efficient non-trivial sampling for dense $\mathfrak{C}$-satisfiability* if the task described above is feasible, namely if for some large enough constant[2] k, there is a constant d and a probabilistic algorithm A running in space n^d such that, given an input $\mathfrak{C}$-circuit C on n variables of size $\mathsf{poly}(n)$ accepting at least a $2/3$ fraction of all inputs, A outputs a fixed satisfying assignment y of C with probability at least $n^k/2^n$. Our main result for PSPACE shows that feasibility of this task implies lower bounds for PSPACE against uniform $\mathfrak{C}$-circuits, where the notion of uniformity is $\mathsf{LOGSPACE}$-uniformity.

Theorem 1. *(Informal Statement) Let $\mathfrak{C}$ be any class of circuits that can be simulated by Boolean circuits of polynomial size. If there is space-efficient non-trivial sampling for dense $\mathfrak{C}$-satisfiability, then* PSPACE *does not have uniform $\mathfrak{C}$-circuits of polynomial size.*

As a corollary, when $\mathfrak{C}$ is the class of general Boolean circuits, space-efficient non-trivial sampling for dense $\mathfrak{C} - \mathsf{SAT}$ implies that $\mathsf{PSPACE} \neq \mathsf{P}$. Thus a solution to a purely algorithmic task separates PSPACE from P. We find this connection surprising, even if it's not apparent what sorts of algorithmic ideas might be useful in solving our task.

We note that the requirements of our algorithmic task are fairly relaxed in many ways. We are interested in randomized algorithms, while it is still open to derive non-uniform circuit lower bounds for NEXP from randomized algorithms for $\mathsf{CircuitSAT}$. A trivial linear-time randomized algorithm for our task simply samples a random $y \in \{0,1\}^n$ and outputs it - each satisfying assignment is output with probability 2^{-n} by this algorithm. We only require a *fixed polynomial advantage* in sampling probability over this trivial algorithm.

However, the restriction that the algorithm must operate in space a fixed polynomial independent of the circuit size of C is indeed a fairly strong requirement. Essentially, what this restriction means is that we can't simulate the circuit C when trying to solve our algorithmic task. Thus, in order to solve our algorithmic

[2]Our proof shows that $k > 3$ suffices, and we suspect that $k > 1$ might suffice if we optimise our parameters.

task, we need to have a rich structural understanding of $\mathfrak{C}$-circuits accepting at least a $2/3$ fraction of their inputs.

We argue heuristically that some such restriction on white-box access is necessary to derive lower bounds for PSPACE. Consider the case where $\mathfrak{C}$ is just the class of general Boolean circuits. Suppose we were able to derive $\mathsf{PSPACE} \neq \mathsf{P}$ from the success of some algorithmic task T that is defined with a Boolean circuit C as input, and suppose we had full white-box access to C. If T is solvable in PSPACE, then if $\mathsf{PSPACE} = \mathsf{P}$, T should be solvable efficiently. Since the efficient solvability of T implies $\mathsf{NP} \neq \mathsf{P}$, we get that $\mathsf{PSPACE} \neq \mathsf{P}$ unconditionally!

Of course it is possible that T is not solvable in PSPACE, but rather (say) in EXP. However, in this case, there might not be sufficient reason to believe T is solvable efficiently, and using T to derive a lower bound might not be a sound algorithmic approach.

We would like to emphasize the point made by the argument above: the restriction that the algorithm for our task cannot simulate the circuit C is not just an artifact or weakness of our approach. Rather, it seems unavoidable for any white-box task in the cryptographic regime, since the lower bound we are trying to show is quantitatively stronger than the upper bound.

Indeed, as mentioned before, if we have full white-box access to C, we can unconditionally solve the algorithmic task we propose, by repeatedly sampling strings of length n and outputting the lexicographically smallest string accepted by C. Theorem 31 in Section 4 establishes this formally. The ability of the algorithm to read the description of C as well as to simulate C makes this argument work.

Now suppose we wish to extend the approach of Theorem 1 to uniform lower bounds for NP. We can define an analogous notion of time-efficient non-trivial sampling for dense $\mathfrak{C}$-satisfiability, and prove a connection similar to Theorem 1 where the consequence is a lower bound for NP. However, this notion of time-efficient non-trivial sampling is very restrictive, as the sampling algorithm will not even be able to read the entire input circuit, let alone simulate it. Ideally, we would like the sampling algorithm to be able to read the entire input, even if it isn't able to perform a simulation, as in our setting for PSPACE.

Therefore, we consider a *succinct* version of our algorithmic task. Our input now is 1^n together with a $\mathfrak{C}$-circuit C of size at most n which is a compressed representation of a larger $\mathfrak{C}$-circuit C' of size $\mathsf{poly}(n)$ on n variables. Here, by saying that C is a compressed representation of C', we mean that we can recover any specific bit in the representation of C' by evaluating C on some input. Alternatively, one can think of C as a circuit for the *direct connection language* of C'.

We say that there is *efficient non-trivial sampling with* PH *oracle for the succinct version of dense* $\mathfrak{C}$-*satisfiability* if for some large enough constant k, there is a constant d and a probabilistic algorithm A running in time n^d with a PH oracle such that, given 1^n and $\mathfrak{C}$-circuit C as input, where C is a compressed representation of a $\mathfrak{C}$-circuit C' of size $\mathsf{poly}(n)$ on n variables accepting at least a $2/3$ fraction of inputs, A outputs some fixed satisfying assignment y of C' with probability at least $n^k/2^n$. The notion of uniformity we use for all of our results on lower bounds for NP is $\mathsf{LOGTIME}$-uniformity [BIS90].

Theorem 2. *(Informal Statement) Let $\mathfrak{C}$ be any class of circuits that can be simulated by Boolean circuits of polynomial size. If there is efficient non-trivial sampling with* PH *oracle for the succinct version of dense $\mathfrak{C}$-satisfiability, then* NP *does not have uniform $\mathfrak{C}$-circuits of polynomial size.*

Note that we allow the sampling algorithm oracle access to an arbitrary PH oracle in the hypothesis. This is intended to make the algorithmic task easier to solve.

Now we justify our claim that our algorithmic approach fulfils the two criteria we stated in Section 1.1. The first criterion is that the relevant algorithmic task should be a conventional one. This is true in our case since the algorithmic task we consider is that of efficiently sampling, with non-trivial probability, a fixed satisfying assignment to a circuit with many satisfying assignments. We observe that the second criterion holds as well, under standard cryptographic assumptions.

Theorem 3. *(Informal Statement) If one-way functions secure against super-polynomial size circuits exist, then there is efficient non-trivial sampling for the succint version of dense* $\mathsf{Circuit}$-*satisfiability.*

We show Theorem 3 by using cryptographic pseudo-random generators to give an efficient non-trivial sampling algorithm for the succinct version of dense $\mathsf{Circuit}$-satisfiability. Note that the sampling algorithm yielded by the assumption does not need access to a PH oracle. Also, an efficient non-trivial sampling algorithm for the succinct version of dense Circuit-satisfiability trivially implies a space-efficient non-trivial sampling algorithm, therefore Theorem 3 additionally evidences the feasibility of the approach toward PSPACE $\neq$ P.

One might still wonder if the algorithmic approach we present is far stronger than is actually necessary for uniform lower bounds. We show that under plausible complexity assumptions (which seem *morally* weaker than the lower bounds we are trying to prove), a uniform version of our approach[3] is in fact *necessary* in that the algorithmic tasks we propose become feasible.

Our ideas give an unconditional *algorithmic characterization* of PSPACE $\neq$ P.

Theorem 4. *(Informal Statement)* PSPACE $\neq$ P *iff* E *has circuits of size* $2^{o(n)}$ *on infinitely many input lengths, or if there is a space-efficient non-trivial algorithm for the uniform version of dense* Circuit-*satisfiability.*

In other words, a central uniform lower bound question in complexity theory reduces to either showing surprising non-uniform algorithms exist for E, or to solving our sampling task for general uniform Boolean circuits in a space-efficient non-trivial way. While there are such algorithmic characterizations of lower bounds for NEXP in the complexity-theoretic regime [IKW02, Wil10, Wil16], the characterization above seems to the first one in the cryptographic regime.

While the results above indicate that our approach is sound, it is unclear a priori whether our algorithmic approach is feasible, i.e., if it has any hope of yielding new lower bounds in the near future. We do believe that the connection from algorithms to uniform lower bounds is interesting in itself, but we would like the framework to be capable at least of proving some known uniform lower bounds.

We show that the framework does indeed have this capability. On the one hand, we use known unconditional results about hitting set generators for weak circuit classes to observe that our sampling tasks are solvable for these circuit classes. On the other hand, we show that our framework can be used to give alternative proofs of well-known uniform lower bounds such as versions of the space hierarchy theorem [SHL65] and Allenders' lower bound for Permanent [All99]. This evidences the flexibility of the framework - it accommodates techniques exploiting specific properties of circuit classes as well as techniques based on simulation and diagonalization.

Finally, we use our approach to prove a couple of new lower bounds, and hope that even stronger lower bounds will follow using more sophisticated algorithmic ideas.

Theorem 5. *(Informal Statement)* NP *does not have uniform polynomial-size* AC^0 *circuits with a bottom layer of* Mod$m$ *gates for any positive integer* m*, not does it have uniform polynomial-size* AC^0 *circuits with a bottom layer of threshold gates.*

We note that it is a longstanding open problem to prove non-uniform super-polynomial size lower bounds in NP against AC^0 circuits with a bottom layer of Mod6 gates or a bottom layer of threshold gates, despite much effort[4]. We show that uniformity can be exploited to prove lower bounds for these classes. As far as we are aware, this is the first case of a super-polynomial circuit lower bound for NP that holds for a uniform circuit class but is not known to hold for the corresponding non-uniform circuit class[5].

1.3 Discussion

In this sub-section, we discuss various features of our approach. Some of these have been mentioned before, but it might be useful for the reader to see them discussed together.

Another Algorithmic Approach to Lower Bounds Our work is the latest in the line of works which

[3]By this we mean that our algorithmic task is only required to be feasible on uniform sequences of circuits.

[4]However, such lower bounds are known for non-deterministic quasi-polynomial time NQP [Wil14, MW20], in the complexity-theoretic regime. We are interested here in lower bounds in the cryptographic regime.

[5]Allender's lower bound for Permanent is another example of this phenomenon, but Permanent is neither known nor believed to be in NP

apply algorithmic approaches to lower bound problems. However, it is the first to apply an algorithmic approach to lower bound problems in the cryptographic regime, and in particular with relevance to problems such as NP vs P and PSPACE vs P. The algorithmic method of [Wil10, Wil11] shows that non-trivial algorithms for $\mathsf{CircuitSAT}$ and Circuit Acceptance Probability Estimation imply super-polynomial circuit lower bounds for NEXP. Building on [FK09, KKO13], Oliveira and Santhanam [OS17] showed that non-trivial randomized learning algorithms with membership queries over the uniform distribution for a class $\mathfrak{C}$ of circuits implies lower bounds in BPEXP against polynomial-size $\mathfrak{C}$-circuits.

Previous works on algorithmic approaches require a non-trivial upper bound on the running time of the algorithm to derive lower bound consequences. In our case, in contrast, while it is important that the algorithm is efficient, what matters more is the probability of sampling some *fixed* solution - we need this to be non-trivial. Another difference between our work and previous works is that previous works all rely ultimately on hierarchy theorems. In contrast, we do not use hierarchy theorems, and this enables us to deal with the cryptographic regime of lower bounds.

Exploiting the Power of NP As mentioned in Section 1.1, there are several works beginning in the 1980s that establish super-polynomial circuit lower bounds for weak circuit classes using various combinatorial and algebraic techniques. An interesting feature of these results is that in most cases, the best lower bounds we know are for problems that are in P. For example, the strongest lower bounds we know for constant-depth circuits are for the Parity function, which is easily seen to be solvable by linear-size circuits. Clearly, any lower bound technique that yields lower bounds for problems in P is not capable of proving super-polynomial lower bounds for general Boolean circuits. Our approach, in contrast, uses the power of NP, and perhaps this suggests that the approach, or variants in it, might be useful in the long run to prove strong lower bounds.

The Importance of Uniformity Historically, there has been a divide in complexity theory between approaches to non-uniform lower bounds and approaches to uniform lower bounds. Approaches to non-uniform lower bounds are often tailored to the circuit class of interest, identifying a structural weaknesses of the circuit class (such as being simplified by random restrictions or being approximable by low-degree polynomials) and then exploiting the weakness mathematically or algorithmically. Approaches to uniform lower bounds tend to be more generic, employing clever combinations of simulation and diagonalization techniques. Our work bridges this divide to an extent in that it identifies stand-alone algorithmic tasks such that solutions for these tasks have implications for uniform lower bounds. The hope is that these algorithmic tasks can be solved efficiently and non-trivially by exploiting properties of the circuit class.

Generality of the Approach Our algorithmic approach is *general* in multiple respects. First, it is relevant to lower bounds for *any* circuit class $\mathfrak{C}$ contained in the class of Boolean circuits. We do not even require even weak closure properties of the circuit classes. In particular, this makes our approach potentially relevant to frontier lower bounds against *fixed-depth classes*, eg., the class of depth-two threshold circuits.

Secondly, our approach can be adapted to lower bounds for uniform classes other than NP, simply by modifying the resource requirements of the algorithm. Theorem 1 illustrates how this works in the context of lower bounds for PSPACE. The approach is also capable of being adapted to lower bounds for other problems such as $\mathsf{Permanent}$, as shown in Sections 3 and 5.

White-Box Algorithmic Tasks in the Cryptographic Regime Our approach puts the spotlight on white-box circuit-based algorithmic tasks in the cryptographic regime, where the algorithm does not have the resources to simulate the circuit it gets as input. To the best of our knowledge, these kinds of algorithms have not been considered before. We are specifically interested in sampling algorithms, and the extent to which sampling can outperform simulation. There are some known results about the power of low-complexity samplers, such as the work of Applebaum, Ishai and Kushilevitz[AIK04] using the technique of randomizing polynomials to show that in many contexts, NC^1-samplable distributions can be replaced by NC^0-samplable distributions, and the work of Viola [Vio12] initiating a line of research on the complexity of distributions. Perhaps ideas from these works or related works could be useful in approaching our algorithmic tasks.

Relevance to Known Barriers Several previous attempts to attack NP vs P and related problems have run into one or the other of various complexity barriers, including the relativization barrier [BGS75], the Natural Proofs barrier [RR97] and the algebrization barrier [AW08]. So it is important to examine how our approach fares against these barriers. As of now, we envision our approach as relevant mostly to frontier questions such as separating NP from uniform ACC^0 and uniform TC^0_2. It does not seem as though the relativization and algebraization barriers are relevant to such weak circuit classes, as existing lower bounds for weak circuit classes exploit weaknesses of the gate sets, and hence don't work when oracle gates with large fan-in occur in the circuit. The natural proofs barrier is not known to have any relevance to uniform circuit lower bounds. Indeed, even in the case of non-uniform circuit lower bounds against these classes, the natural proofs barrier would only be operative if there are pseudo-random functions in ACC^0 or TC^0_2, and no compelling evidence exists for the existence of such low-complexity pseudo-random functions.

1.4 Proof Ideas

We discuss here the ideas behind our approach and sketch the proofs of the connections from algorithms to lower bounds. We first discuss the proof ideas behind Theorem 2, and then the ideas behind Theorem 1.

Recall that our goal is to develop an *algorithmic approach* to uniform lower bounds for NP and PSPACE. We would like our algorithmic approach to involve a conventional algorithmic task, and for there to be complexity-theoretic evidence that the algorithmic task is feasible. Ideally, the algorithmic task should require only a marginal improvement over known algorithms.

Our starting point is an elegant idea of Hirahara [Hir20]. Hirahara was interested in the problem of proving uniform lower bounds for the problem R_{Kt} of determining whether an input string x has high Kt complexity. Recall that the Kt complexity of a string is the minimum of $|p| + \log(t)$ over programs p and time bounds t such that $U^t(p, \epsilon) = x$, i.e., a universal machine halts within t steps on input p and outputs x. R_{Kt} is known to be EXP-complete [ABK$^+$06] but only with respect to *non-uniform* truth-table reductions or NP Turing reductions. It is a long-standing open problem whether R_{Kt} is in P. Hirahara showed that R_{Kt} does not have P-uniform ACC^0 circuits of polynomial size.

His idea is as follows: suppose that there is a non-trivial algorithm for satisfiability for a circuit class $\mathfrak{C}$. Namely, we can find a satisfying assignment y of length n for a satisfiable $\mathfrak{C}$-circuit C on n variables in deterministic time $2^n/n^{\omega(1)}$. Then it is not too hard to show that y has Kt complexity $n - \omega(\log(n))$ *conditioned* on C. Now, if C itself were a uniform circuit generated by an efficient procedure given input 1^n, that implies that $\mathsf{Kt}(C) = O(\log(n))$, and by first generating C and then generating y conditioned on C, we get that y has Kt complexity $n - \Omega(\log(n))$.

We can use this to derive a contradiction to the assumption that R_{Kt} has P-uniform $\mathfrak{C}$-circuits. We consider the uniform circuit ACC^0 C_n assumed to solve R_{Kt}. The satisfying assignments of C_n are precisely the hard Kt strings, and in particular we can assume that every satisfying assignment has Kt complexity $n - \Omega(\log(n))$. But then the argument in the previous para yields a contradiction, since y is a satisfying assignment to C_n and has Kt complexity $n - \omega(\log(n))$.

Since we know that satisfiability of ACC^0 circuits can be solved in non-trivial time [Wil11], we derive a P-uniform ACC^0 lower bound for R_{Kt}. This is still quite far from the desired result showing $R_{\mathsf{Kt}} \notin \mathsf{P}$, but it constitutes some progress. Hirahara uses similar ideas to show that the set of K^t-random strings is not in P for any super-polynomial t.

While the idea of the proof is novel, the lower bound result for R_{Kt} is still in the complexity-theoretic regime of lower bounds, since we are trying to prove uniform super-polynomial lower bound for a language that is known to be complete for exponential time. However, we observe that this isn't *intrinsic* to the approach; rather it depends on which notion of resource-bounded Kolmogorov complexity we analyze. In this case, we analyzed Kt complexity, but in principle we could analyze some other resource-bounded Kolmogorov complexity measure.

In particular, let us imagine trying to upper-bound the $\mathsf{K}^{\mathsf{poly}}$ complexity of satisfying assignments to circuits. Our reason for doing this is that the language of hard $\mathsf{K}^{\mathsf{poly}}$ strings is in NP, hence if we are able to carry through an argument analogous to Hirahara's argument, we might be able to show a uniform circuit lower bound for NP. One obstacle that presents itself is that it is unclear what sort of algorithm we need to analyze in order to upper bound the $\mathsf{K}^{\mathsf{poly}}$ complexity of solutions. Another obstacle is that it is unclear even why there should be a solution of low $\mathsf{K}^{\mathsf{poly}}$ complexity.

Let us try to address the second obstacle first, and come up with an algorithmic task where there are likely to be solutions of non-trivial $\mathsf{K}^{\mathsf{poly}}$ complexity. Here we make a crucial observation: Hirahara considers the Satisfiability problem for general circuits, but in fact we can consider the Satisfiability problem for *dense* circuits instead. The reason is that if we are going to use the argument on a circuit that is presumed to decide the set of hard $\mathsf{K}^{\mathsf{poly}}$ strings correctly, the circuit will have many accepting inputs, since a random string is likely to be $\mathsf{K}^{\mathsf{poly}}$-hard.

Considering Dense Circuit Satisfiability makes our approach more plausible, since if we make cryptographic derandomization assumptions, we are at least likely to have solutions with low $\mathsf{K}^{\mathsf{poly}}$ complexity conditioned on the circuit. However, the first obstacle remains - it is not clear how to analyze $\mathsf{K}^{\mathsf{poly}}$ complexity of solutions for any natural algorithmic task.

Our idea is to use *probabilistic* notions of time-bounded Kolmogorov complexity. Several notions of probabilistic time-bounded Kolmogorov complexity have recently been defined and studied in [Oli19, LOS21, GKLO22, LOZ22, LO22]. A notion that is ideal for our purposes is the notion of $\mathsf{pK}^{\mathsf{poly}}$ [GKLO22, LOZ22]. Intuitively, a string x has low $\mathsf{pK}^{\mathsf{poly}}$ string if for most random strings r of a given polynomial length, there is a small description p_r from which x can be reconstructed in polynomial time given r. This notion of probabilistic Kolmogorov complexity has two very appealing features. First, $\mathsf{pK}^{\mathsf{poly}}$ complexity turns out to be closely tied to a very natural algorithmic task, namely sampling solutions of search problems. This allows us to define a natural algorithmic task that makes no reference to Kolmogorov complexity notions. Second, $\mathsf{pK}^{\mathsf{poly}}$ complexity has a so-called Optimal Coding Theorem, which implies that strings sampled efficiently with probability p have $\mathsf{pK}^{\mathsf{poly}}$ complexity very close to $\log(1/p)$. This allows us to define an algorithmic task that involves just a non-trivial improvement in success probability over existing efficient algorithms.

The price we pay is that the language of hard $\mathsf{pK}^{\mathsf{poly}}$ strings is no longer in NP. However, we can define a promise version of the language of hard strings which is in AM, and this turns out to be sufficient for our purposes, by using an additional idea.

However, our assumption in Theorem 2 is for the succinct version of Dense Circuit Satisfiability. In order to use this version, we observe that we use the assumption of feasibility of algorithmic tasks only on uniform circuits, which are succinctly representable. In fact, when we are arguing by contradiciton, we can efficiently recover a succinct description of the circuit, where the description itself belongs to the class of circuits that are being described. This additional step allows us to complete the proof of Theorem 2.

To extend this proof idea to lower bounds for PSPACE in Theorem 1, we show that it is enough to simply change the resource requirements of the algorithmic tasks to a fixed polynomial space bound rather than a fixed polynomial time bound. The natural idea for analyzing this would be to consider a notion of space-bounded Kolmogorov complexity, but our argument by contradiction finds a short-cut. We use an argument by contradiction again to observe that if indeed PSPACE had small uniform circuits, then $\mathsf{PSPACE} = \mathsf{P}$. This implies that our small-space sampling algorithm can be simulated by a time-efficient sampling algorithm, and then we can use the same machinery as in Theorem 2 to complete the proof.

For Theorem 3, we use the fact [HILL99] that one-way functions imply the existence of cryptographic pseudo-random generators (PRG) with seed length n^ϵ for any $\epsilon > 0$. A cryptographic PRG can be used to solve our sampling task efficiently by simply outputting a random element in the range of the PRG, which can be done in time a fixed polynomial in n. By the pseudo-randomness property, most elements of the range will be satisfying assignments of the dense circuit on which we are solving the sampling task, and each such element will be output with probability 2^{-n^ϵ}, which is non-trivial. Note that we do not even need to look at the circuit on which we are solving the sampling task.

For the algorithmic characterization of $\mathsf{PSPACE} \neq \mathsf{P}$, we use certain properties of the set of strings $L = \mathsf{MKSP}[\sqrt{n}]$ of Kolmogorov space-bounded complexity at most $\sqrt{n}$. It follows from [ABK$^+$06] that L is hard on average for polynomial time in a zero-error sense if $\mathsf{PSPACE} \neq \mathsf{BPP}$, and it can be shown using ideas in [San20] that the hardness on average of L implies the existence of a crytographic hitting-set generator against polynomial size circuits. A cryptographic hitting-set generator can be used to solve our sampling task space-efficiently using the observations in the previous paragraph. To complete our characterization, we use a standard result from derandomization [IW97], namely that either E has circuits of size $2^{o(n)}$ infinitely often or $\mathsf{BPP} = \mathsf{P}$.

The proof of Theorem 5 proceeds by designing efficient non-trivial sampling algorithms with PH oracle for the succinct version of Dense $\mathfrak{C}$ Satisfiability for $\mathfrak{C} = \mathsf{AC}^0 \circ (\mathsf{Mod}m)$ and $\mathfrak{C} = \mathsf{AC}^0 \circ \mathsf{Thr}$.

Several recent works in various areas of complexity theory, including learning theory, pseudorandomness, cryptography, structural complexity and proof complexity, have developed and exploited ideas from *meta-complexity*, i.e., the complexity of complexity. We refer to [All20] for a recent survey. The ideas of our proof are another illustration of this phenomenon. Like the recent work of Hirahara [Hir21] on average-case hardness of NP from exponential worst-case assumptions, our results use meta-complexity as a catalyst: the results make no reference to meta-complexity, yet the proofs use meta-complexity crucially.

2 Preliminaries

2.1 Standard Complexity Notions

The textbook by Arora and Barak [AB09] is an excellent reference for basic notions in complexity theory. Here we recall a few that are especially relevant to this paper.

Computational problems are typically modelled as decision problems, where each input is either accepted or rejected. Occasionally we are interested in *promise problems*, where the set of accepted inputs is disjoint from the set of rejected inputs, but some inputs might not belong to either category. Formally, a promise problem Γ over $\{0,1\}$ is a pair $(\Gamma_{YES}, \Gamma_{NO})$ where $\Gamma_{YES}, \Gamma_{NO} \subseteq \{0,1\}^*$ and $\Gamma_{YES} \cap \Gamma_{NO} = \varnothing$. The complement of a promise problem $(\Gamma_{YES}, \Gamma_{NO})$ is the promise problem $(\Gamma_{NO}, \Gamma_{YES})$. We say that a language L is consistent with a promise problem $\Gamma = (\Gamma_{YES}, \Gamma_{NO})$ if $\Gamma_{YES} \subseteq L$ and $\Gamma_{NO} \subseteq \bar{L}$.

We will need to be careful about which Turing machine model we consider, since we are often interested in computations that run in sub-linear time. We will use the random access Turing machine model, where each tape of a multi-tape Turing machine has a corresponding address tape. When the address tape for tape k has index i written on it, and the machine enters a special tape, the contents of the i'th tape cell of the k'th tape can be accessed in unit time. We also consider oracle Turing machines, where the random access Turing machine is provided with a separate oracle tape, on which queries to the oracle can be made, and answered in unit time.

We will be considering various standard circuit classes, including the class AC^0_d of constant-depth circuits of depth d with AND and OR gates, the class $\mathsf{AC}^0_d[p]$ of constant-depth circuits of depth d with AND, OR and MOD p gates for prime p, the class ACC^0 of constant-depth circuits of depth d with AND, OR and modular gates, the class TC^0_d of depth-d threshold circuits, the class $\mathsf{Formula}$ of Boolean formulas, and the class $\mathsf{Circuit}$ of Boolean circuit. By default, whenever we refer to a circuit class, we will mean the *non-uniform* version of the circuit class. However, as is standard terminology, NC^1 will refer to $\mathsf{LOGTIME}$-uniform circuits of logarithmic depth. We will occasionally abuse notation and use the name of a circuit class to refer to the circuit class as well as to the class of languages decided by the circuit class.

We will mainly be using two standard notions of uniformity for circuits: $\mathsf{LOGTIME}$-uniformity and $\mathsf{LOGSPACE}$-uniformity. We refer to [BIS90] for precise definitions of these notions as well as a detailed discussion on motivation. Briefly, $\mathsf{LOGTIME}$-uniformity means that the *direct connection language* of a sequence $\{C_n\}$ of circuits, encoding types of gates and connections between them in a natural way, is decidable in time logarithmic in the size of the circuit. $\mathsf{LOGSPACE}$-uniformity means that the direct connection language of $\{C_n\}$ is decidable in logarithmic space; equivalently, a description of C_n can be computed in logarithmic space given 1^n as input. The main property we will require of $\mathsf{LOGTIME}$-uniformity is that any given bit of the description of a $\mathsf{LOGTIME}$-uniform circuit C can be computed in time logarithmic in the size of the circuit. This is the case when a circuit is represented in a standard way, i.e., as a list of gate types and connections between gates in some pre-determined order.

We won't formally define direct connection languages, since the details depend on the circuit classes of interest, and we consider a wide variety of circuit classes. However, in each case, we will be able to answer questions about the gate type, about whether the i'th child of a gate is a certain other gate, and about whether a gate has more than i inputs using a single query to the direct connection language. Things get a bit subtle when considering circuits where the gates have weights, eg., threshold gates which check whether an integer-weighted sum of inputs is at least some integer value, or modular gates which check whether integer-weighted sums of the inputs belong to some set of values modulo a given integer. Indeed, we consider AC^0 circuits with a bottom layer of $\mathsf{Mod}m$ or Thr gates in Section 5, and we assume there that the gates are irredundant, i.e., that the weights aren't unnecessarily large. In particular, we assume that the weights for any $\mathsf{Mod}m$ gate are at most m, and that the weights for a threshold gate on n inputs are at most $n^{O(n)}$. Indeed, it is easy to see that any $\mathsf{Mod}m$ gate is equivalent to one where the weights are at most m, and it is known that any threshold gate is equivalent to one where the weights are at most $n^{O(n)}$[Mur71]. Using our assumption, the weights can be extracted using a fixed polynomial number of queries to the direct connection language - this will be crucial in the proofs of our new lower bounds.

We will also refer to $\mathsf{POLYLOG}$-uniformity, where the direct connection language is decidable in time poly-logarithmic in the size of the circuit.

We say that a circuit class $\mathfrak{C}$ is polynomially simulatable if there is a polynomial-time algorithm which, given a circuit C from $\mathfrak{C}$ and an input x to C, computes $C(x)$.

Proposition 6. *Suppose a circuit class $\mathfrak{C}$ is polynomially simulatable, and let L be a language that has $\mathsf{LOGTIME}$-uniform (resp. $\mathsf{LOGSPACE}$-uniform) $\mathfrak{C}$-circuits of polynomial size. Then L has $\mathsf{LOGTIME}$-uniform (resp. $\mathsf{LOGSPACE}$-uniform) Boolean circuits of polynomial size.*

Proof. Suppose $\mathfrak{C}$ is polynomially simulatable, and let M be a Turing machine that, given a $\mathfrak{C}$-circuit of size m on n variables and an input x to C, runs in time m^d and computes $C(x)$, where d is a constant. Let L be a language that has $\mathsf{LOGTIME}$-uniform $\mathfrak{C}$-circuits of polynomial size, and let $\{C_n\}$ be a sequence of $\mathsf{LOGTIME}$-uniform $\mathfrak{C}$-circuits of size n^c deciding L, where c is a constant. We derive $\mathsf{LOGTIME}$-uniform Boolean circuits of polynomial size deciding L as follows. By the standard simulation of polynomial time by polynomial size, we have that the computation of M can be represented by $\mathsf{LOGTIME}$-uniform circuits of size m^{2d} that take C and x as input. By fixing C to C_n, we derive $\mathsf{LOGTIME}$-uniform circuits $\{D_n\}$ of size n^{2cd} that take x as input and correctly compute $L(x)$. The fact that the direct connection language of $\{D_n\}$ is in $\mathsf{LOGTIME}$ follows from the fact that the direct connection language of the circuits simulating M is in $\mathsf{LOGTIME}$, together with the $\mathsf{LOGTIME}$-uniformity of $\{C_n\}$. $\square$

Recall that CH [Wag86, PS88] is the *counting hierarchy*, whose first level $\mathsf{CH}_1 = \mathsf{PP}$ and i'th level $\mathsf{CH}_i = \mathsf{PP}^{\mathsf{CH}_{i-1}}$. We will also need Toda's theorem [Tod91].

Theorem 7. *[Tod91]* $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\#\mathsf{P}}$.

2.2 Meta-Complexity

Here we define various notions of Kolmogorov complexity that will be needed in this work.

Throughout, we fix a time-efficient universal Turing machine U. Notions of Kolmogorov complexity are defined relative to this universal machine U, but since the notions and results we use are robust to the precise choice of the universal machine, we suppress the dependence on U.

Given a string x, the Kolmogorov complexity $\mathsf{K}(x)$ is defined to be the size of the smallest program p such that $U(p, \epsilon)) = x$. Given a time bound $t : \mathbb{N} \to \mathbb{N}$ and a string x, the t-time bounded Kolmogorov complexity of x is defined as follows: $\mathsf{K}^t(x)$ is the size of the smallest program p such that $U^{t(|x|}(p, \epsilon) = x$, where U^T means that that the universal machine is restricted to run for at most T steps.

K and K^t are *deterministic* notions of Kolmogorov complexity, in that a string is recovered from its compressed representation by a deterministic program. We require a probabilistic notion of Kolmogorov complexity recently introduced in [GKLO22].

Given a time bound $t : \mathbb{N} \to \mathbb{N}$, a string x and a number $\rho \in [0,1]$, we say that x has ρ-confidence pK^t complexity at most k if for at least ρ fraction of random strings r of length $t(|x|)$, there is a program $p_r, |p_r| \leqslant k$, for which $U^{t(|x|)}(p_r, r) = x$.

Proposition 8. *For any time bound $t : \mathbb{N} \to \mathbb{N}$, non-negative integers n, k and $\rho \in [0,1]$, at most $2^{k+1}/\rho$ strings have ρ-confidence pK^t-complexity at most k.*

Proof. Assume, for the sake of contradiction, there are more than $2^{k+1}/\rho$ strings of length n with ρ-confidence pK^t complexity at most k. Call this set of strings X_k; by assumption $|X_k| > 2^{k+1}/\rho$. We obtain a contradiction by counting program-randomness pairs (p, r) that witness membership in X_k, where $|p| \leqslant k$ and $|r| = t(n)$. Since each member of X_k corresponds to at least $\rho 2^{t(n)}$ distinct such pairs (by definition of ρ-confidence pK^t complexity), we have that the number of such witnessing pairs for X_k is greater than $2^{k+t(n)+1}$. However, note that there are at most 2^{k+1} programs of length at most k, hence the number of distinct (p, r) pairs where $|p| \leqslant k$ and $|r| = t(n)$ is at most $2^{k+1+t(n)}$. Hence we reach a contradiction to our assumption. □

Let $t : \mathbb{N} \to \mathbb{N}$ be a time bound. Given a complexity parameter $s : \mathbb{N} \to \mathbb{N}$ and real numbers $\rho, \delta \in [0,1]$ such that $\delta < \rho$, we define the meta-complexity promise problem $R_{\mathsf{pK}^t}[s, \rho, \delta] = (\Gamma_{YES}, \Gamma_{NO})$, with $\Gamma_{YES}, \Gamma_{NO} \subseteq \{0,1\}^*$ and $\Gamma_{YES} \cap \Gamma_{NO} = \varnothing$, as follows. A string $x \in \Gamma_{NO}$ if the ρ-confidence pK^t complexity of x is at most $s(|x|)$. A string $x \in \Gamma_{YES}$ if the δ-confidence pK^t complexity of x is not at most $s(|x|)$.

Proposition 9. $R_{\mathsf{pK}^t}[s, \rho, \delta] \in \mathsf{coAM}$. *Moreover, if $s(n) + \log(1/\delta) < n - 2$ for each $n \in \mathbb{N}$, then Γ_{YES} contains at least a $3/4$ fraction of all strings of length n.*

Proof. We give an Arthur-Merlin protocol for the complement of $R_{\mathsf{pK}^t}[s, \rho, \delta]$. In order to certify that a string x has ρ-confidence pK^t complexity at most $s(|x|)$, Arthur sends Merlin a random string r of length $t(|x|)$ and Merlin sends back a program p_r of size at most $s(|x|)$. Arthur accepts iff $U^{t(|x|)}(p_r, r) = x$. Clearly this protocol accepts with probability at least ρ iff x has ρ-confidence pK^t complexity at most $s(|x|)$. Thus the protocol has acceptance probability at least ρ on YES instances and at most δ on NO instances, which means that there is a gap between acceptance probabilities on YES and NO instances, as desired.

The fact that Γ_{YES} contains at least a $3/4$ fraction of all strings of length n follows from Proposition 8. □

2.3 One-Way Functions and Pseudorandomness

We need the standard cryptographic notion of a non-uniformly secure one-way function. In fact, we define the length-preserving variant of the notion, which is without loss of generality.

Definition 10. *Let $s : \mathbb{N} \to \mathbb{N}$. A function $f = \{f_n\}, f_n : \{0,1\}^n \to \{0,1\}^n$ is said to be an $s(\cdot)$-secure one-way function if for each sequence of circuits $\{D_n\}$ of size $\mathsf{poly}(s(n))$, we have that $\Pr_{x \sim \{0,1\}^n}[f(D(f(x))) = f(x)] = 1/n^{\omega(1)}$.*

Next we define the notion of a cryptographic pseudo-random generator. For ease of application, we define the notion slightly differently than the standard notion, with the computability of the PRG measured as a function of the output size.

Definition 11. *Let $\ell : \mathbb{N} \to \mathbb{N}$ be a function such that $\ell(n) \leqslant n$ for all $n \in \mathbb{N}$. A cryptographic pseudo-random generator with seed length ℓ is a function $G = \{G_n\}, G_n : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ computable in time $\mathsf{poly}(n)$ such that for any algorithm D running in time $\mathsf{poly}(n)$, $|\Pr_{y \in \{0,1\}^n} D(y) - \Pr_{z \in \{0,1\}^{\ell(n)}} D(G(z))| = 1/n^{\omega(1)}$.*

One of the foundational result in cryptography is that cryptographic pseudo-random generators with small seed length can be based on the existence of one-way functions with super-polynomial hardness.

Theorem 12. *[HILL99] Suppose there is an $n^{\omega(1)}$-secure one-way function. Then there is a cryptographic pseudo-random generator with seed length $n^{o(1)}$.*

We will also need the notion of a cryptographic hitting set generator useful against a circuit class $\mathfrak{C}$.

Definition 13. *Let $\ell : \mathbb{N} \to \mathbb{N}$ be a function such that $\ell(n) \leqslant n$ for all $n \in \mathbb{N}$, and let $\mathfrak{C}$ be a circuit class. A cryptographic hitting set generator with seed length ℓ against $\mathfrak{C}$ is a function $G = \{G_n\}, G_n : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ computable in time $\mathsf{poly}(n)$ such that for any sequence of $\mathfrak{C}$-circuits $\{C_n\}$ such that C_n accepts at least a $1/n$ fraction of n-bit inputs for large enough n, there exists a sequence $\{y_n\}$ with $y_n \in \{0,1\}^{\ell(n)}$ for each n such that $C(G_n(y_n)) = 1$.*

2.4 Search Problems and Samplers

The algorithmic tasks we consider will involve solving search problems. We first define the notion of a promise search problem.

Definition 14. *A promise search problem $\mathcal{S}$ is a pair (R, X) where $R \subseteq \{0,1\}^*$ is a polynomial-time computable binary relation and $X \subseteq \{0,1\}^*$ is a subset of inputs. A solution to the search problem $\mathcal{S}$ on input x is any string y such that $(x, y) \in R$. We say that an algorithm A solves the promise search problem $\mathcal{S}$ if for each $x \in X$, A outputs a solution to $\mathcal{S}$ on x if one exists.*

We will be interested in a specific promise search problem where the task is to find satisfying assignments of circuits that accept most of their inputs.

Definition 15. *Let $\mathfrak{C}$ be a circuit class. The promise search problem $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$ is defined by the following binary relation R and input set X. R consists of all pairs (C, x), where C is (the encoding of) a $\mathfrak{C}$-circuit, and x is a satisfying assignment of C. X consists of all $\mathfrak{C}$-circuit C such that C accepts at least a $2/3$ fraction of its inputs.*

For technical reasons, we will also need the notion of a sampler. Note that our notion of sampler simply models an algorithm that samples a distribution, and is not related to the notion of sampler in the theory of randomness extraction.

Definition 16. *A sampler is a polynomial-time randomized algorithm which, given 1^n as input for $n \in \mathbb{N}$, samples a distribution on n-bit outputs.*

3 From Algorithms to Uniform Lower Bounds

In this section, we prove our main results about connections from circuit sampling tasks to uniform lower bounds for NP, PSPACE and $\mathsf{Permanent}$.

3.1 An Algorithmic Approach to Uniform Lower Bounds for NP

We first define the algorithmic task we will consider in this sub-section.

Definition 17. *Given circuits C and C', we say that C encodes C' if the Boolean function computed by C is the direct connection language of C'.*

Definition 18. *Let $\mathfrak{C}$ be a circuit class. We say that there is efficient non-trivial sampling (resp. efficient non-trivial sampling with PH oracle) for the succinct version of $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$ if for every $k > 0$ there is a probabilistic algorithm A (resp. probabilistic algorithm A with PH oracle) which, given 1^n and a $\mathfrak{C}$-circuit C of size at most n, where C encodes a $\mathfrak{C}$-circuit C' of size at most n^b on n variables accepting at least a $2/3$ fraction of its inputs, runs in time n^d (for some constant d independent of b) and outputs some fixed satisfying assignment y to C' with probability at least $n^k/2^n$.*

The following is a version of the Optimal Coding Theorem for pK^t in [LOZ22] with slightly improved parameters[6] that are important in our application.

[6]Specifically, the coding theorem in [LOZ22] involves an additive term that is logarithmic in the time bound of the sampler, while the additive term in our bound is logarithmic in the input length of the sampler

Lemma 19. *Let S be a sampler that runs in time n^s on input of length n, where $s > 0$ is a constant. There is a constant β such that if S samples some $y \in \{0,1\}^n$ with probability p, then y has $3/4$-confidence $\mathsf{pK}^{n^{\beta s}}$ complexity at most $\log(1/p) + 3\log(n)$.*

Proof. We follow the proof strategy in [LOZ22]. Let S be a sampler that runs in time n^s and samples some $y \in \{0,1\}^n$ with probability p. The idea is to choose a random hash function h mapping $\log(1/p) + O(1)$ bits to n^s bits, and show that with high probability there exists a choice of input x for h such that S produces y when run with randomness $h(x)$. This argument is a simple probabilistic argument, but it is not sufficient to bound the $\mathsf{pK}^{\mathsf{poly}}$ complexity of y, since the randomness used to describe the hash function is exponential. We show instead that a *pseudo-random* hash function describable with $\mathsf{poly}(n)$ bits can be used instead, by showing that the test that h is a "good" hash function can be performed in AC^0, and then using known unconditional pseudo-random generators against AC^0.

In more detail, let h be a random function from $\log(1/p) + C$ bits to n^s bits, for some constant C to be chosen later. Given random string $R \in \{0,1\}^{n^s}$, let $S(R)$ denote the output of the sampler S when using randomness R. We say h is good for S and y if there is an x of size $\log(1/p) + \log\log(n)$ such that $S(h(x)) = y$. We upper bound the probability that h is *not* good for S and y. The probability that a random R does not satisfy $S(R) = y$ is at most $1 - p$, hence by a union bound the probability that no string R in the range of h satisies $h(R) = y$ is at most $(1-p)^{2^C/p} < 0.01$ for C chosen large enough.

Next we show that the test that h is good for S and x can be implemented by AC^0 circuits of size $2^{\mathsf{poly}(n)}$. We represent h by a concatenation of the values of h for all inputs in lexicographic order, i.e., a string of length $O(2^{n+n^s})$. Note that we can assume $p \geqslant 1/2^n$, otherwise the statement of the Lemma is trivially true since each string y of length n has $\mathsf{pK}^{n^{2s}}$ complexity at most $n + O(1)$.

Indeed, to check that h is good, we just need to take an OR over all strings x in the range of h of the condition that $S(h(x)) = y$, which can be checked by CNFs of size $2^{O(n^s)}$. Thus, the test that h is good can be implemented by depth-3 circuits of size $2^{O(n^s)}$. Now we use known constructions of pseudo-random generators against AC^0 [Nis91, NW94] which provide a PRG G with seed length $n^{\alpha s}$ for some fixed constant $\alpha > 0$, such that the i'th bit of $G(z)$ can be computed in time $\mathsf{poly}(z, \log(i))$ for $z \in \{0,1\}^{n^{\alpha s}}$. This yields a collection $\mathfrak{H}$ of hash functions indexed by the seed of G and evaluatable in polynomial time. The probability that $h \in \mathfrak{H}$ is good for S and y is close to the probability that a random h is good, since G is a PRG, and in particular we have that for each y sampled with probability p, $G(z) \in \mathfrak{H}$ is good for S and y with probability at least $3/4$ over choice of z.

To show that y has $\mathsf{pK}^{n^{\beta s}}$ complexity at most $\log(1/p) + 3\log(n)$, we describe y as follows. Let $\beta > 0$ be a constant to be determined later. Given $r \in \{0,1\}^{n^{\beta s}}$, we use the $n^{\alpha s}$-bit prefix z of r to index into $\mathfrak{H}$ and pick out a specific hash function $h = G(z)$. We have that with probability at least $3/4$ over the choice of r, h is good for S and y. For good h, we describe y by the string x of length $\log(1/p) + O(1)$ such that $S(h(x)) = y$, together with $2\log(n)$ bits to describe n as well as $O(1)$ bits to specify a program that given n and x, runs S on input 1^n and randomness $h(x)$ to obtain y. We choose β large enough so that the time taken to reconstruct y is at most $n^{\beta s}$ - such a constant β exists by the polynomial-time evaluability of G and the polynomial-time bound on S. This implies that the 3/4-confidence $\mathsf{pK}^{n^{\beta s}}$ complexity of y is at most $\log(p) + 3\log(n)$ for large enough $n \in \mathbb{N}$.

□

In fact it can be shown for any search problem that sampling of a fixed solution to the search problem with non-trivial probability is equivalent to the existence of solutions that have non-trivial *conditional* $\mathsf{pK}^{\mathsf{poly}}$ complexity, for an appropriately defined notion of conditional $\mathsf{pK}^{\mathsf{poly}}$ complexity. We do not pursue this direction here to avoid detracting from the focus of the paper on approaches to uniform lower bounds.

We need an easy lemma to allow us to deal with the issue of promise problems.

Lemma 20. *Let Γ be a promise problem in coAM. If $\mathsf{NP} = \mathsf{P}$, then there is a language L_Γ consistent with Γ such that $L_\Gamma \in \mathsf{P}$.*

Proof. Let $\Gamma = (\Gamma_{YES}, \Gamma_{NO})$ be a promise problem in coAM, with $\Gamma_{YES}, \Gamma_{NO} \subseteq \{0,1\}^*$ and $\Gamma_{YES} \cap \Gamma_{NO} = \varnothing$. Since $\mathsf{coAM} \subseteq \Pi_2^p$ [BM88][7], there is a Π_2 machine M running in polynomial time that accepts on all instances in Γ_{YES} and rejects on all instances in Γ_{NO}. Let $L_\Gamma \in \Pi_2^p$ be the language decided by M. Since M accepts all instances in Γ_{YES} and rejects all instances in Γ_{NO}, we have that L_Γ is consistent with Γ. If $\mathsf{NP} = \mathsf{P}$, then the Polynomial Hierarchy collapses to P and hence $\Pi_2^p = \mathsf{P}$, which implies that $L_\Gamma \in \mathsf{P}$. $\square$

We are now ready to establish the main result of this sub-section. The following is a more formal version of Theorem 2.

Theorem 21. *Let $\mathfrak{C}$ be a polynomially simulatable circuit class. Suppose that there is efficient non-trivial sampling with* PH *oracle for the succinct version of* $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$. *Then* NP *does not have* $\mathsf{LOGTIME}$-*uniform $\mathfrak{C}$-circuits of polynomial size.*

Proof. Suppose that there is efficient non-trivial sampling with PH oracle for the succinct version of $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$, and assume for the sake of contradiction that NP has $\mathsf{LOGTIME}$-uniform $\mathfrak{C}$-circuits of polynomial size. Since $\mathfrak{C}$ is polynomially simulatable, it follows from Proposition 6 that NP has $\mathsf{LOGTIME}$-uniform Boolean circuits of polynomial size, and hence that $\mathsf{NP} = \mathsf{P}$.

Suppose that the efficient sampling algorithm A for the succinct version of $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$ uses as oracle some language $O \in \mathsf{PH}$. Since $\mathsf{NP} = \mathsf{P}$, we have that $\mathsf{PH} = \mathsf{P}$, and hence $O \in \mathsf{DTIME}(n^c)$ for some constant c. Let d be a constant such that the oracle algorithm A runs in time n^d on any $\mathfrak{C}$-circuit C of size $\mathsf{poly}(n)$ on n variables. Since A runs in time n^d, it can make at most n^d oracle queries each of size at most n^d, and since $O \in \mathsf{DTIME}(n^c)$, each of these oracle queries can be simulated in time at most n^{cd}, yielding an equivalent algorithm A' which does not need access to the oracle and runs in time at most n^q, where $q = cd + d$.

Let β be the constant from Lemma 19. We now consider the promise problem $\Gamma = R_{\mathsf{pK}^{n^{2\beta q}}}[n-5, 3/4, 1/4]$. By Proposition 9, this problem is in coAM, and Γ_{YES} contains at least a 3/4 fraction of strings of length n. Since $\mathsf{NP} = \mathsf{P}$, we have by Lemma 20 that there is a language $L \in \mathsf{P}$ that is consistent with Γ. By the definition of consistency, all YES instances of Γ are YES instances of L and hence L accepts at least 3/4 fraction of inputs of length n, for large enough $n \in \mathbb{N}$.

By assumption, NP has $\mathsf{LOGTIME}$-uniform $\mathfrak{C}$-circuits of polynomial size, and since $L \in \mathsf{P}$, we have that L has $\mathsf{LOGTIME}$-uniform circuits of size n^ℓ for some constant $\ell > 0$. Let $\{C_n\}$ be a sequence of $\mathsf{LOGTIME}$-uniform $\mathfrak{C}$-circuits of size n^ℓ on n variables deciding L. Note that since $\{C_n\}$ decides L correctly, we have that C_n accepts at least a 3/4 fraction of inputs of length n for each large enough $n \in \mathbb{N}$.

We use the assumed efficient non-trivial sampling algorithm A' for the succinct version of $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$ to define a sampler S that runs in polynomial time. The sampler S operates as follows given input 1^n. It first computes n in binary.

Next consider the $\mathsf{LOGTIME}$-uniform sequence $\{C_n\}$ of circuits for L, where C_n is of size n^ℓ. We consider the *succinct* version of the direct connection language L_{dc} of $\{C_n\}$, where n as well as gate indices and types are represented in binary in the input to the language. Since L_{dc} is decidable in time $O(\log(n))$, it is decidable in linear time as a function of its input. Since by assumption P has $\mathsf{LOGTIME}$-uniform $\mathfrak{C}$-circuits of polynomial size, we have that there are $\mathsf{polylog}(n)$ size $\mathfrak{C}$-circuits for L_{dc}, and by the uniformity condition, these circuits can be computed in $\mathsf{polylog}(n)$ time. S computes the corresponding circuit D_n of size $\mathsf{polylog}(n)$ that decides the direct connection language of C_n, and feeds $(1^n, D_n)$ to the algorithm A'. Note that D_n encodes a $\mathfrak{C}$-circuit C_n of size $\mathsf{poly}(n)$ on n variables as required, and that C_n accepts at least a 3/4 fraction of inputs of length n. Note also that the size of D_n is at most n.

Hence A' does indeed perform efficient non-trivial sampling when given $(1^n, D_n)$ as input, and this implies that S halts in time $O(n^q)$ and outputs some fixed satisfying assignment y of C_n with probability at least $n^k/2^n$, where k is a constant we are free to choose.

By Lemma 19, we have that y has 3/4-confidence $\mathsf{pK}^{n^{2\beta q}}$ complexity at most $n - k\log(n) + 3\log(n)$, which for $k > 3$ and large enough n is at most $n - 5$. We use this to derive a contradiction.

[7]This simulation is usually stated for languages, but holds also for promise problems.

The upper bound on the $\mathsf{pK}^{n^{2\beta q}}$ complexity of y implies that y is a NO instance of Γ. Since L is consistent with Γ, y is a NO instance of L as well, and since C_n decides L correctly, y is rejected by C_n. However, by assumption on A', y is a satisfying assignment of C_n, which yields a contradiction.

□

Theorem 21 is very general in that there are no constraints on the circuit class $\mathfrak{C}$ apart from polynomial simulatability, and in particular we do not require any closure properties of $\mathfrak{C}$. We immediately obtain the following corollaries. The first two corollaries concern frontier questions in complexity theory, and the last two concern two of the central problems in the area.

Corollary 22. *Suppose that for any $d \in \mathbb{N}$ there is efficient non-trivial sampling with* PH *oracle for the succinct version of* $\mathsf{Dense} - \mathsf{ACC}^0_d - \mathsf{SAT}$*. Then* NP *does not have* $\mathsf{LOGTIME}$*-uniform* ACC^0*-circuits of polynomial size.*

Corollary 23. *Suppose that there is efficient non-trivial sampling with* PH *oracle for the succinct version of* $\mathsf{Dense} - \mathsf{TC}^0_2 - \mathsf{SAT}$*. Then* NP *does not have* $\mathsf{LOGTIME}$*-uniform* TC^0_2*-circuits of polynomial size.*

Corollary 24. *Suppose that there is efficient non-trivial sampling with* PH *oracle for the succinct version of* $\mathsf{Dense} - \mathsf{Formula} - \mathsf{SAT}$*. Then* $\mathsf{NP} \neq \mathsf{NC}^1$*.*

Corollary 25. *Suppose that there is efficient non-trivial sampling with* PH *oracle for the succinct version of* $\mathsf{Dense} - \mathsf{Circuit} - \mathsf{SAT}$*. Then* $\mathsf{NP} \neq \mathsf{P}$*.*

3.2 An Algorithmic Approach to Uniform Lower Bounds for PSPACE

Theorem 21 gives an algorithmic approach to showing uniform lower bounds for NP. It is natural to ask if there is a similar algorithmic approach involving an easier algorithmic task toward showing uniform lower bounds for larger classes such as PSPACE. We provide an affirmative answer in this sub-section.

We begin by defining a space-efficient notion of sampling.

Definition 26. *Let $\mathfrak{C}$ be a circuit class. We say that there is space-efficient non-trivial sampling for* $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$ *if for every $k > 0$ there is a $d > 0$ and a probabilistic algorithm A such that for every $b > 0$, when A is given an instance C of* $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$*, where C is of size $m = n^b$ on n variables, A has space and randomness complexity n^d for some fixed constant d, and outputs some fixed satisfying assignment y to C with probability at least $n^k/2^n$.*

We would like to show that space-efficient non-trivial sampling for $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$ leads to lower bounds for PSPACE against $\mathsf{LOGSPACE}$-uniform $\mathfrak{C}$-circuits of polynomial size. A natural strategy to achieve this is to define a new notion of probabilistic *space-bounded* Kolmogorov complexity and work in analogy to Section 3.2. But in fact we can short-circuit this process and adapt the argument in Section 3.2 more directly, while still working with time-bounded Kolmogorov complexity. We simply exploit the fact that our argument works by contradiction, and our initial assumption implies that $\mathsf{PSPACE} = \mathsf{P}$, which means that the space-bounded and time-bounded notions of Kolmogorov complexity essentially coincide.

The following is a more formal version of Theorem 1 from the Introduction.

Theorem 27. *Let $\mathfrak{C}$ be a polynomially simulatable circuit class, and suppose that there is space-efficient non-trivial sampling for* $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$*. Then* PSPACE *does not have* $\mathsf{LOGSPACE}$*-uniform $\mathfrak{C}$-circuits of polynomial size.*

Proof. Suppose that there is space-efficient non-trivial sampling for $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$, and assume for the sake of contradiction that PSPACE has $\mathsf{LOGSPACE}$-uniform $\mathfrak{C}$-circuits of polynomial size. Since $\mathfrak{C}$ is polynomially simulatable, it follows from Proposition 6 that PSPACE has $\mathsf{LOGSPACE}$-uniform Boolean circuits of polynomial size, and hence that $\mathsf{PSPACE} = \mathsf{P}$.

We now consider the promise problem $\Gamma = R_{\mathsf{pK}^{n^{2\beta q}}[n-5,3/4,1/4]}$, as in the proof of Theorem 21, where q is a constant to be determined later. By Proposition 9, this problem is in coAM, and Γ_{YES} contains at least a $3/4$ fraction of strings of length n. Since $\mathsf{PSPACE} = \mathsf{P}$ and since coAM is trivially contained in PSPACE, we have that there is a language $L \in \mathsf{P}$ that is consistent with Γ. By the definition of consistency, all YES instances of Γ are YES instances of L and hence L accepts at least $3/4$ fraction of inputs of length n, for large enough $n \in \mathbb{N}$.

By assumption, PSPACE has $\mathsf{LOGSPACE}$-uniform $\mathfrak{C}$-circuits of polynomial size, and since $L \in \mathsf{P}$, we have that L has $\mathsf{LOGSPACE}$-uniform circuits of size n^ℓ for some constant $\ell > 0$. Let $\{C_n\}$ be a sequence of $\mathsf{LOGSPACE}$-uniform $\mathfrak{C}$-circuits of size n^ℓ on n variables deciding L. Note that since $\{C_n\}$ decides L correctly, we have that C_n accepts at least a $3/4$ fraction of inputs of length n for each large enough $n \in \mathbb{N}$.

By assumption, we have a probabilistic algorithm A that, given any $\mathfrak{C}$-circuit C of size $\mathsf{poly}(n)$ on n variables that accepts at least a $2/3$ fraction of its inputs, has space and randomness complexity bounded by n^d for some fixed constant $d > 0$, and outputs some fixed satisfying assignment y of C with probability $n^k/2^n$, where $k > 0$ is a constant we are free to choose. We use A to define a sampler S that runs in fixed polynomial space as follows. Given input 1^n, S first computes n^ℓ in binary - this can be done in space $O(\log(n))$. It then simulates the operation of algorithm A' on input C_n, without computing C_n explicitly. Whenever A' requires some bit of the description of C_n (which it specifies by writing on the random access tape), S computes this bit in space $O(\ell \log(n)) = O(\log(n))$ by using the $\mathsf{LOGSPACE}$-uniformity of C_n. Thus the simulation can be done in space $O(n^d + \log(n))$, which is at most n^{2d} for large enough n and $d \geqslant 1$. If A' halts after outputting a string of length n, S halts after outputting the same string; if A' does not output a string or outputs a string of length other than n, S halts after outputting 0^n. Note that if A' outputs some satisfying assignment y of C with probability at least p, so does S.

Thus we have a sampler S running in space n^{2d} sampling a distribution over n-bit outputs. Now we use the fact that $\mathsf{PSPACE} = \mathsf{P}$ again to obtain an equivalent *polynomial-time* sampler S'. Let q be a constant such that S' runs in time n^q.

Since C_n accepts at least a $3/4$ fraction of all its inputs, C_n is a valid instance of $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$. Hence A, on input C_n, outputs some fixed satisfying assignment y of C_n with probability at least $n^k/2^n$. So we have that S' outputs y with probability at least $n^k/2^n$. By Lemma 19, we have that y has $3/4$-confidence $\mathsf{pK}^{n^{2\beta q}}$ complexity at most $n - k\log(n) + 3\log(n)$, which for $k > 3$ and large enough n is at most $n - 5$. We use this to derive a contradiction.

The upper bound on the $\mathsf{pK}^{n^{4q}}$ complexity of y implies that y is a NO instance of Γ. Since L is consistent with Γ, y is a NO instance of L as well, and since C_n decides L correctly, y is rejected by C_n. However, by assumption on A, y is a satisfying assignment of C_n, which yields a contradiction. $\square$

We immediately obtain the following corollary, which concerns a long-standing open problem in complexity theory . Note that analogues of Corollaries 22, 23, 24 are known to hold unconditionally for PSPACE by the space hierarchy theorem [SHL65].

Corollary 28. *Suppose that there is space-efficient non-trivial sampling for* $\mathsf{Dense} - \mathsf{Circuit} - \mathsf{SAT}$*. Then* $\mathsf{PSPACE} \neq \mathsf{P}$*.*

There are a couple of differences in the hypothesis of Theorem 27 as compared to Theorem 21. The first is that the sampling algorithm isn't given access to an oracle. However, this is an insignificant difference. Given a space-efficient sampling algorithm access to a PSPACE oracle doesn't really increase its power, as PSPACE is closed under polynomial-space reductions.

The second is that the sampling algorithm is given the entire circuit as input rather than a succinct representation of it. At first sight, this looks more like a hypothesis about a white-box algorithm rather than about a restricted white-box algorithm. But in fact, the hypothesis isn't fully white box, as the sampling algorithm doesn't have enough space to simulate the input circuit in general. The input circuit can be of size an arbitrary polynomial in n, while the sampling algorithm needs to run in space a fixed polynomial in n.

We could consider the succinct version of the sampling problem here too, and the connection would still go through, just as in the proof of Theorem 21. This would yield our desired conclusion under a weaker hypothesis, as space-efficient non-trivial sampling implies space-efficient non-trivial sampling for the succinct version. Indeed, suppose we have a space-efficient non-trivial sampling algorithm for the succinct version, where we are given a circuit C of size at most n encoding a circuit C' on n variables for which we want to sample a satisfying assignment. We could simulate any query to C' in the non-succinct version by running C in the succinct version, which costs at most a fixed polynomial overhead in space.

Our reason for stating the weaker result here (by using a stronger hypothesis) is that in some situations the stronger hypothesis seems more natural to attack, because it feels more similar in flavour to the white-box version. Indeed, when we reprove versions of the space hierarchy theorem using our approach in Section 5, we do establish the stronger hypothesis for the circuit class of interest there.

An analogue of the stronger hypothesis could also be defined and considered in the setting of uniform lower bounds for NP, but feels less achievable there, as it would involve solving the sampling task without even reading the entire input circuit. This might still be possible for weak circuit classes, such as depth-two circuits, but coming up with algorithmic approaches to the hypothesis for stronger circuit classes might be hard. In contrast, when considering the stronger hypothesis in the setting of uniform lower bounds for PSPACE, we are allowed to read the entire input circuit, just not to use a large amount of space when trying to sample from it.

An algorithmic approach to the problem of showing that $\mathsf{Permanent}$ is not in NC^1 can be developed along very similar lines to Theorem 21 and Theorem 27. Here $\mathsf{Permanent}$ is the problem of computing the permanent of a Boolean matrix over the integers, encoded in a standard way as a decision problem.

Theorem 29. *Let $\mathfrak{C}$ be a polynomially simulatable circuit class that is closed under projections. Suppose that there is efficient non-trivial sampling with* CH *oracle for the succinct version of* $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$*. Then* $\mathsf{Permanent}$ *does not have* $\mathsf{LOGTIME}$*-uniform $\mathfrak{C}$-circuits of polynomial size.*

Proof. Suppose that there is efficient non-trivial sampling with CH oracle for the succinct version of $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$, and suppose for the sake of contradiction that $\mathsf{Permanent}$ has $\mathsf{LOGTIME}$-uniform $\mathfrak{C}$-circuits of polynomial size. Let $\{C_n\}$ be this sequence of circuits, where C_n is of size n^b for some constant b and has n input variables. The second assumption implies that $\mathsf{Permanent} \in \mathsf{P}$. By Theorem 7 and the completeness of the Permanent for $\#\mathsf{P}$ [Val79], we have that $\mathsf{PP} = \mathsf{P}$, and hence $\mathsf{CH} = \mathsf{P}$. This means that we can eliminate the CH oracle for the sampling algorithm, to obtain an efficient non-trivial sampling algorithm A' for the succinct version of $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$. Let q be a constant such that A' runs in time at most n^q.

We proceed as in Theorem 21 to analyze the meta-complexity problem $\Gamma = R_{\mathsf{pK}^{n^{2\beta q}}}[n-5, 3/4, 1/4]$, where β is the constant in the statement of Lemma 19. Using Theorem 7 again, we have that $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PP}} = \mathsf{P}$. Since $\Gamma \in \mathsf{PH}$, this implies that there is a language $L \in \mathsf{P}$ consistent with Γ that accepts at least a $3/4$ fraction of inputs of each large enough input length.

We use the assumed efficient non-trivial sampling algorithm A' for the succinct version to define a sampler S. The sampler S operates as follows given input 1^n. It first computes n in binary.

Next consider the $\mathsf{LOGTIME}$-uniform sequence $\{C_n\}$ of circuits for L that exist by assumption, where C_n is of size n^b for some constant b and has n input variables. We consider the *succinct* version of the direct connection language L_{dc} of $\{C_n\}$, where n as well as gate indices and types are represented in binary in the input to the language. Since L_{dc} is decidable in time $O(\log(n))$, it is decidable in linear time as a function of its input. Since $\mathsf{Permanent}$ is hard for NP under $\mathsf{LOGTIME}$-uniform projections, and $\mathfrak{C}$ is closed under projections, the existence of $\mathsf{LOGTIME}$-uniform poly-size $\mathfrak{C}$-circuits for $\mathsf{Permanent}$ implies the existence of such circuits for NP and hence for P. Thus we have that there are $\mathsf{polylog}(n)$ size $\mathfrak{C}$-circuits for L_{dc}, and by the uniformity condition, these circuits can be computed in $\mathsf{polylog}(n)$ time. S computes the corresponding circuit D_n of size $\mathsf{polylog}(n)$ that decides the direct connection language of C_n, and feeds $(1^n, D_n)$ to the algorithm A'. Note that D_n encodes a $\mathfrak{C}$-circuit C_n of size $\mathsf{poly}(n)$ on n variables as required, and that C_n accepts at least a $3/4$ fraction of inputs of length n. Note also that the size of D_n is at most n.

Hence A' does indeed perform efficient non-trivial sampling when given $(1^n, D_n)$ as input, and this implies that S halts in time $O(n^q)$ and outputs some fixed satisfying assignment y of C_n with non-trivial probability. This can be used to derive a contradiction just as in the proof of Theorem 21 by applying Lemma 19.

□

4 Soundness of the Approach

4.1 Solving the Algorithmic Tasks under Standard Cryptographic Assumptions

As discussed, one of the most important criteria for an algorithmic approach to a lower bound problem is that the approach should be sound, i.e., there should ideally be evidence that the relevant algorithmic task is feasible. We begin by providing cryptographic evidence that the algorithmic tasks discussed in Section 3 are feasible. The following is a more formal version of Theorem 3.

Theorem 30. *Suppose there is an $n^{\omega(1)}$-secure one-way function. Let $\mathfrak{C}$ be any circuit class that is polynomially simulatable. Then there is efficient non-trivial sampling for* $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$. *Indeed, there is a probabilistic algorithm A which, given a $\mathfrak{C}$-circuit C of size $m = n^b$ on n variables accepting at least $2/3$ of its inputs, runs in time n^d (where d is independent of b) and outputs some fixed satisfying assignment of C with probability $2^{-n^{o(1)}}$.*

Proof. Suppose there is an $n^{\omega(1)}$-secure one-way function. By Theorem 12, there is a pseudo-random generator G with seed length $n^{o(1)}$. Consider the algorithm A that runs as follows given input C of size $m = n^b$ on n variables. It ignores its input C, instead running G_n on a random seed z and outputting $G_n(z)$. Since G is computable in time n^d for some fixed d, A runs in time n^d and is hence efficient. The non-trivial sampling property follows from the pseudo-randomness of G. Since C is polynomially simulatable, there is some polynomial-size Boolean circuit C' equivalent to C. By the pseudo-randomness of G, the probability that C' accepts on a random output of G is close to the probability that C' accepts on a random input of length n. Since at least a $2/3$ fraction of inputs of C' are accepted, most outputs of G are satisfying assignments to C and therefore also to C. Since G has seed length $n^{o(1)}$, each output of G is produced with probability at least $2^{-n^{o(1)}}$ by the algorithm A, and in particular there is a satisfying assignment y of C that is produced with probability at least $2^{-n^{o(1)}}$, as claimed. ☐

Note that the algorithm A in the proof of Theorem 30 is *oblivious*: it does not consult its input. Thus the standard cryptographic assumption of the existence of one-way functions implies an oblivious solution to our algorithmic task, while we only require a constrained white-box solution.

It turns out that if we are interested in a white-box solution to the algorithmic tasks that is not constrained, i.e., the algorithm is not required to be efficient, then the task is indeed solvable with non-trivial probability by a sampling argument.

Theorem 31. *Let $\mathfrak{C}$ be any polynomially simulatable circuit class. For each $k > 0$ there is a probabilistic algorithm A which, given a $\mathfrak{C}$-circuit $C$of size $\mathsf{poly}(n)$ on n variables accepting at least $2/3$ fraction of its inputs, runs in time $\mathsf{poly}(n)$ and outputs a fixed satisfying assignment y of C with probability at least $n^k/2^n$ for large enough $n \in \mathbb{N}$.*

Proof. We define a probabilistic algorithm A operating as follows given $\mathfrak{C}$-circuit C as input. It samples independently and uniformly at random n^{k+2} strings $y_i, i \in [n^{k+2}]$ each of length n. It checks for each y_i in polynomial time whether y_i is a satisfying assignment of C, using the polynomial simulatability of $\mathfrak{C}$. If none of the strings y_i is satisfying, A simply outputs 0^n and halts, otherwise it outputs the lexicographically smallest satisfying assignment y_i.

We argue that this algorithm outputs some satisfying assignment y of C with probability at least $n^k/2^n$. Indeed, we will argue that with probability close to 1, A outputs some assignment from the set of $2^n/n^{k+1}$ lexicographically smallest satisfying assignments of C. Note that C has at least this many satisfying assignments since it accepts at least a $2/3$ fraction of all inputs of length n. Let S be this set of $2^n/n^{k+1}$ lexicographically smallest satisfying assignments.

A randomly chosen y of length n is in S with probability at least $1/n^{k+1}$, hence the probability that none of the sampled strings y_i is in S is at most $(1 - 1/n^{k+1})^{n^{k+2}}$, which is $2^{-\Omega(n)}$. Note that if at least one of the strings $y_i \in S$, then the output of the algorithm belongs to S. Thus we have that the output of the algorithm belongs to S with probability at least $1 - 2^{-\Omega(n)}$, which means some string $y \in S$ is output with probability at least $1/(2|S|)$ for large enough n, which is at least $n^k/2^n$ for large enough n. ☐

4.2 Necessity of the Approach

We have argued that our algorithmic approach is sound, but could it be that what we require algorithmically is much stronger than what is needed? Next we show that under plausible complexity-theoretic assumptions, a version of our algorithmic approach to lower bounds for NP and PSPACE is in fact *necessary.* Specifically, we define a *uniform* version of our algorithmic approach, where efficient non-trivial sampling is only required for each uniform sequence of circuits, rather than for circuits given as input to an algorithm. We observe that our proofs in Section 3 go through if the uniform version is feasible, and then show that under our complexity assumptions, $\mathsf{NP} \neq \mathsf{P}$ and $\mathsf{PSPACE} \neq \mathsf{P}$ actually imply the feasibility of the uniform versions of our assumptions.

We need a standard complexity-theoretic derandomization assumption, as well as an additional assumption about NP-hardness of a meta-complexity problem in the case of uniform lower bounds for NP. We discuss the case of uniform lower bounds for NP first, and then move on to the case of uniform lower bounds for PSPACE. First we define a uniform version of our algorithmic approach for NP.

Definition 32. *Let $\mathfrak{C}$ be a circuit class. We say that there is efficient non-trivial sampling for the uniform version of* $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$ *if for every $k > 0$ and every* LOGTIME*-uniform sequence $\{C_n\}$ of $\mathfrak{C}$-circuits of size $\mathsf{poly}(n)$ on n variables such that C_n accepts at least $2/3$ fraction of inputs of length n, there is a probabilistic algorithm A which, given input 1^n, runs in time n^d (for some constant d independent of the exponent in the size of C_n) and outputs some fixed satisfying assignment y to C_n with probability at least $n^k/2^n$.*

Next we require a standard derandomization result.

Theorem 33. *[IW97] Suppose* E *requires exponential-size Boolean circuits. Then* $\mathsf{BPP} = \mathsf{P}$.

We need to define the meta-complexity problem MCSP and what it means for this problem to be average-case hard and to be hard to approximate.

Definition 34. *Given a size function $s : \mathbb{N} \to \mathbb{N}$ where $s(N) \leqslant N$ for each positive integer N, we define the problem* $\mathsf{MCSP}[s]$ *as follows. YES instances of* $\mathsf{MCSP}[s]$ *are strings y of length N where $N = 2^n$ for some integer n and f_y has Boolean circuits of size at most $s(N)$, where f_y is the Boolean function whose truth table is y.*

We say that $\mathsf{MCSP}[s]$ *is zero-error easy on average over the uniform distribution if there is a deterministic polynomial-time algorithm which, given input y, always outputs 0, 1 or '?'; always correctly classifies y with respect to* $\mathsf{MCSP}[s]$ *when it outputs a Boolean value; and outputs a non-'?' value with probability at least $1/poly(N)$ over $y \sim \{0,1\}^N$. We say that* $\mathsf{MCSP}[s]$ *is zero-error hard on average over the uniform distribution if it is not zero-error easy on average over the uniform distribution.*

Given a function $\gamma : \mathbb{N} \to \mathbb{N}$, we say that MCSP *is γ-hard to approximate (resp. γ-hard to approximate probabilistically) if there is no polynomial time algorithm (resp. probabilistic polynomial time algorithm) solving the following promise problem Γ. Γ_{YES} consists of tuples $(y, 1^s)$ such that y is the truth table of a function with Boolean circuits of size at most s. Γ_{NO} consists of tuples $(y, 1^s)$ such that y is the truth table of a function with no Boolean circuits of size at most $\gamma(|y|)s$.*

We say that it is NP*-hard to γ-approximate* MCSP *if there is a polynomial-time reduction from* SAT *to the promise problem Γ described above.*

We will use the following approximation to average-case reduction of Hirahara [Hir18].

Theorem 35. *[Hir18] Suppose that* MCSP *is $N^{1-\epsilon}$-hard to approximate probabilistically for each $\epsilon > 0$. Then for each $\delta > 0$,* $\mathsf{MCSP}[N^{\delta}]$ *is zero-error average-case hard over the uniform distribution.*

Now we are ready to prove our result about necessity of the uniform version of our algorithmic approach in the case of uniform lower bounds for NP.

Theorem 36. *Suppose that* E *requires exponential-size Boolean circuits, and moreover that* MCSP *is* NP*-hard to $N^{1-\epsilon}$-approximate for each $\epsilon > 0$. Then* $\mathsf{NP} \neq \mathsf{P}$ *iff there is efficient non-trivial sampling for the uniform version of* $\mathsf{Dense} - \mathsf{Circuit} - \mathsf{SAT}$.

Proof. We observe that the only circuits C for which we use sampling in the proof of Theorem 21 are uniform circuits, hence the conclusion of Theorem 21 holds even if the algorithmic assumption is that there is efficient non-trivial sampling for the uniform version of $\mathsf{Dense}-\mathsf{Circuit}-\mathsf{SAT}$. This shows the backward implication in the statement of Theorem 36. We will use the assumptions to argue the forward implication.

The proof is in several steps. We give a road-map and then show how to implement each step. We will first use the assumption that MCSP is NP-hard to approximate together with the assumption that $\mathsf{NP} \neq \mathsf{P}$ to show that MCSP is hard to approximate by polynomial-time algorithms. Then we use Theorem 35 to argue that MCSP is hard on average over the uniform distribution. Next we use a lemma from [San20] to argue that there are hitting set generators against polynomial time with small seed length. Finally we show how hitting set generators with small seed length imply efficient non-trivial sampling for the uniform version of $\mathsf{Dense}-\mathsf{Circuit}-\mathsf{SAT}$.

By the assumption that it is NP-hard to $N^{1-\epsilon}$-approximate MCSP, and since $\mathsf{NP} \neq \mathsf{P}$, we have immediately that MCSP is $N^{1-\epsilon}$-hard to approximate. Since E requires exponential-size Boolean circuits, and by using Theorem 33, we have that MCSP is $N^{1-\epsilon}$-hard to approximate probabilistically. By applying Theorem 35, we infer that for each $\delta > 0$, $\mathsf{MCSP}[N^{\delta}]$ is zero-error average-case hard over the uniform distribution.

By Proposition 10 in [San20], this implies that for each $\delta > 0$, there is a hitting set generator H_δ with seed length N^{δ} against polynomial time that is computable in time at most N^2 [8]. Here a hitting set generator against polynomial time is a function whose range intersects with any dense set in P, i.e., any set containing at least a $1/N^{O(1)}$ fraction of strings of length N for each N. Now consider the set $L \in \mathsf{P}$ such that $y \in L$ iff y is a satisfying assignment of $C_{|y|}$. This set is dense because C_N accepts at least $2/3$ fraction of length N. Hence the range of the hitting set generator intersects L.

We define an algorithm A_δ which, given input 1^N, samples a random element y of length N in the range of H_δ, and outputs it. Note that each element in the range of H_δ is output with probability at least $2^{-N^{\delta}}$. Since some element y in the range of H_δ is accepted by C_N, we have that some satisfying assignment y of C_N is output with probability at least $2^{-N^{\delta}}$. The algorithm A_δ uses fixed polynomial time independent of the size of C_N. Hence efficient non-trivial sampling for the uniform version of $\mathsf{Dense}-\mathsf{Circuit}-\mathsf{SAT}$ holds. $\square$

Given that the uniform version of the algorithmic approach suffices to show $\mathsf{NP} \neq \mathsf{P}$, one might ask why we do not highlight this version in Section 3. The reason is that this algorithmic task is not very naturally defined, since it has a unary input and refers to a uniform circuit family. We find the algorithmic tasks defined and studied in Section 3 more natural, in that an arbitrary circuit from the class $\mathfrak{C}$ is provided as input.

Next we tackle the generality question for our algorithmic approach to lower bounds for PSPACE. We first define a uniform version of the algorithmic approach.

Definition 37. *Let $\mathfrak{C}$ be a circuit class. We say that there is space-efficient non-trivial sampling for the uniform version of $\mathsf{Dense}-\mathfrak{C}-\mathsf{SAT}$ if for every $k > 0$ and every $\mathsf{LOGTIME}$-uniform sequence $\{C_n\}$ of $\mathfrak{C}$-circuits of size $\mathsf{poly}(n)$ on n variables such that C_n accepts at least $2/3$ fraction of inputs of length n, there is a probabilistic algorithm A which, given input 1^n, has space and randomness complexity at most n^d (for some constant d independent of the exponent in the size of C_n) and outputs some fixed satisfying assignment y to C_n with probability at least $n^k/2^n$.*

We require the notion of KS complexity and the corresponding meta-complexity problem MKSP.

Definition 38. *The KS complexity of a string x is defined as follows, relative to some space-efficient universal Turing machine U. $\mathsf{KS}(x)$ is the minimum over $|p| + s$ such that $U(p, \epsilon)$ halts and outputs x using space at most s.*

[8]The N^2 upper bound on time for computing the hitting set generator is not argued explicitly in [San20] but follows from the proof, since the seed of the hitting set generator is a circuit represented by N^{δ} bits on $\log(N)$ inputs and the output is the truth table of length N of the function computed by this circuit

Given a function $s : \mathbb{N} \to \mathbb{N}$, $\mathsf{MKSP}[s]$ is the set of strings x such that $\mathsf{KS}(x) \leqslant s(|x|)$. We say that $\mathsf{MCSP}[s]$ is zero-error easy on average over the uniform distribution if there is a deterministic polynomial-time algorithm which, given input y, always outputs 0, 1 or '?'; always correctly classifies y with respect to $\mathsf{MCSP}[s]$ when it outputs a Boolean value; and outputs a non-'?' value with probability at least $1/poly(n)$ over $y \sim \{0,1\}^n$. We say that $\mathsf{MCSP}[s]$ is zero-error hard on average over the uniform distribution if it is not zero-error easy on average over the uniform distribution.

We will use the following result which establishes that MKSP is PSPACE-hard even on average.

Theorem 39. *[ABK$^+$06] If $\mathsf{PSPACE} \neq \mathsf{BPP}$, then for each constant $\delta > 0$, $\mathsf{MKSP}[n^\delta]$ is zero-error hard on average under the uniform distribution.*

Finally we are ready to show our result in the case of uniform lower bounds for PSPACE.

Theorem 40. *Suppose that E requires exponential-size Boolean circuits. Then $\mathsf{PSPACE} \neq \mathsf{P}$ iff there is space-efficient non-trivial sampling for the uniform version of $\mathsf{Dense}-\mathsf{Circuit}-\mathsf{SAT}$.*

Proof. We observe that the only circuits C for which we use sampling in the proof of Theorem 27 are uniform circuits, hence the conclusion of Theorem 27 holds even if the algorithmic assumption is that there is space-efficient non-trivial sampling for the uniform version of $\mathsf{Dense}-\mathsf{Circuit}-\mathsf{SAT}$. This shows the backward implication in the statement of Theorem 40. We will use the assumptions to argue the forward implication.

The proof of the forward implication is roughly analogous to the proof of Theorem 36.

First we use the circuit lower bound assumption to argue that $\mathsf{BPP} = \mathsf{P}$ by applying Theorem 33. Since $\mathsf{PSPACE} \neq \mathsf{P}$, we have by Theorem 39 that for each $\delta > 0$, $\mathsf{MKSP}[n^\delta]$ is zero-error hard on average over the uniform distribution.

The proof of Proposition 10 in [San20] generalizes to show, based on the average-case hardness of MKSP, that for each $\delta > 0$, there is a hitting set generator H_δ with seed length n^δ against polynomial time that is computable in space $O(n^\delta)$. Now consider the set $L \in \mathsf{P}$ such that $y \in L$ iff y is a satisfying assignment of $C_{|y|}$. This set is dense because C_N accepts at least a $2/3$ fraction of strings of length n. Hence the range of the hitting set generator intersects L.

We define a space–efficient algorithm A_δ which, given input 1^n, that samples a random element y of length n in the range of H_δ and outputs it. Note that each element in the range of H_δ is output with probability at least 2^{-n^δ}. Since some element y in the range of H_δ is accepted by C_n, we have that some satisfying assignment y of C_n is output with probability at least 2^{-n^δ}. The algorithm A_δ uses space only $O(n^\delta)$. Hence space-efficient non-trivial sampling for the uniform version of $\mathsf{Dense}-\mathsf{Circuit}-\mathsf{SAT}$ holds. □

As an easy corollary of Theorem 40, we derive an *algorithmic characterization* of $\mathsf{PSPACE} \neq \mathsf{P}$. We show that separating PSPACE and P, which is a lower bound question, is equivalent to the existence of at least one of two kinds of algorithms: a sub-exponential time non-uniform algorithm for E that works on infinitely many input lengths, or a space-efficient non-trivial sampling algorithm for the unfiorm version of $\mathsf{Dense}-\mathsf{Circuit}-\mathsf{SAT}$.

The following is a re-statement of Theorem 4.

Corollary 41. *$\mathsf{PSPACE} \neq \mathsf{P}$ iff E has circuits of size $2^{o(n)}$ infinitely often or there is space-efficient non-trivial sampling for the uniform version of $\mathsf{Dense}-\mathsf{Circuit}-\mathsf{SAT}$.*

Proof. The forward direction follows directly from Theorem 40.

We next prove the backward direction. Suppose E has circuits of size $2^{o(n)}$ infinitely often. Then we have by the Karp-Lipton theorem for E that E is in space $2^{o(n)}$ infinitely often. This implies that $\mathsf{PSPACE} \neq \mathsf{P}$, as otherwise we would have that E is in time $2^{o(n)}$ infinitely often by a padding argument, which contradicts the time hierarchy theorem.

On the other hand, if there is space-efficient non-trivial sampling for the uniform version of $\mathsf{Dense}-\mathsf{Circuit}-\mathsf{SAT}$, $\mathsf{PSPACE} \neq \mathsf{P}$ follows as observed in the proof of the backward direction of Theorem 40.

□

5 Feasibility of the Approach

In this section, we argue for the feasibility of our approach. We first show that uniform versions of most super-polynomial circuit lower bounds for NP can be captured within the framework, and then that some of the best-known uniform lower bounds proved using diagonalization, such as the space hierarchy theorem and Allender's lower bound for the Permanent, can be reproved using our approach. Then we show how to prove a couple of new lower bounds: NP does not have uniform polynomial-size AC^0 circuits with a bottom layer of Mod m gates, for any composite m, nor uniform polynomial-size AC^0 circuits with a bottom layer of threshold gates.

5.1 Capturing Known Lower Bounds

Complexity theorists have had success proving super-polynomial circuit lower bounds for NP against a variety of weak circuit classes, such as AC^0 and $\mathsf{AC}^0[p]$ (for primes p). We observe that the proofs of these lower bounds imply efficient non-trivial sampling for the corresponding circuit classes, and hence our framework applies. Of course our framework only yields *uniform* lower bounds, which are weaker than the non-uniform lower bounds already known for these classes. However our hope is that the framework might be useful even for stronger circuit classes where non-uniform lower bounds in NP are *not* known, and in order for this to be credible, the framework should at least apply in cases where lower bounds *are* known.

We use the fact that super-polynomial size lower bounds for NP against AC^0 and $\mathsf{AC}^0[p]$ yield cryptographic hitting set generators against these classes with non-trivial seed length.

Theorem 42. *[Nis91, All01, FSUV13, HS17] There is a cryptographic hitting set generator with seed length* $\mathsf{polylog}(n)$ *useful against* AC^0 *and, for any prime p, a cryptographic hitting set generator with seed length* $n - \sqrt{n}$ *useful against* $\mathsf{AC}^0[p]$.

In fact the work of [Nis91, All01, FSUV13, HS17] gives cryptographic pseudo-random generators rather then just cryptographic hitting-set generators, but the weaker hitting property satisfies for our application.

Corollary 43. *There is efficient non-trivial sampling for the succinct versions of* $\mathsf{Dense} - \mathsf{AC}^0 - \mathsf{SAT}$ *and* $\mathsf{Dense} - \mathsf{AC}^0[p] - \mathsf{SAT}$.

The corollary follows from Theorem 42 by just sampling a random output of the generator, which can be done in fixed polynomial time in n independent of the size of the circuit on n bits for which we are solving the sampling problem. Note that by the hitting property, at least one of the outputs of the generator will satisfy the circuit, and each such output is sampled with probability $2^{-\ell(n)}$, where $\ell(n)$ is the seed length of the generator, which is non-trivial by Theorem 42.

We next show that versions of some of the classical results on uniform lower bounds in the literature, shown using direct or indirect diagonalization, can be shown using our framework. First, we consider versions of the space hierarchy theorem.

Theorem 44. *Let $\mathfrak{C}$ be the class of branching programs of polynomial size. There is space-efficient non-trivial sampling for* $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$.

Proof. Let $\mathfrak{C}$ be the class of polynomial-size branching programs. To show space-efficient non-trivial sampling for $\mathfrak{C}$, we use the approach in the proof of Theorem 31. Let $C \in \mathfrak{C}$ be a branching program of size $\mathsf{poly}(n)$ on n variables that accepts at least a $2/3$ fraction of assignments. Fix an integer $k \geqslant 0$. We define an algorithm A with space and randomness complexity at most $n^{k+O(1)}$ that samples a fixed satisfying assignment of C with probability at least $n^k/2^n$.

As in the proof of Theorem 31, A samples independently and uniformly at random n^{k+2} strings $y_i, i \in [n^{k+2}]$ each of length n. It checks for each y_i whether y_i is a satisfying assignment of C. If none of the strings y_i is satisfying, A simply outputs 0^n and halts, otherwise it outputs the lexicographically smallest satisfying assignment y_i.

The argument that A outputs a fixed satisfying assignment with probability at least $n^k/2^n$ is the same as in the proof of Theorem 31. We need to argue in addition that A uses fixed polynomial space and randomness, independent of the size of C. Indeed, the randomness required to generate n^{k+2} uniformly random n-bit strings is n^{k+3}. In terms of space complexity, these strings can be stored in space n^{k+3}. Crucially, the check for any fixed y_i that y_i satisfies C costs space $O(\log(n))$, since we only need to maintain the current state of the branching program. Maintaining the current index i also only costs space $O(\log(n))$, and so too the index j (initialised to 0) of the lexicographically smallest satisfying assignment so far. Thus the total space required is $n^{k+O(1)}$. □

Corollary 45. *[SHL65]* $\mathsf{PSPACE} \neq \mathsf{LOGSPACE}$.

Corollary 45 follows from Theorem 44 by applying Theorem 27, since the class of branching programs of polynomial size is polynomially simulatable.

We remark that the standard proof of the space hierarchy theorem is a fairly simple direct diagonalization, so the proof of Corollary 45 does not have any advantage in terms of simplicity. In addition, Corollary 45 does not give the tight parameters of the space hierarchy theorem, i.e., separating space S from space S' for any space-constructible bounds S, S' where $S = o(S')$. The reason that Theorem 27 does not give the tight space hierarchy is that tighter separations correspond to simulation of circuit classes $\mathfrak{C}$ that are not known to be polynomially simulatable, i.e., branching programs of super-polynomial size. However, the tight space hierarchy can be recovered using the *ideas* of the proof of Theorem 27, by defining and applying an appropriate space-bounded version of Kolmogorov complexity. We omit the details, as we do not see how to obtain a new result on space hierarchies using these ideas.

Next, we show how to rederive Allender's celebrated uniform lower bound for the Permanent using our approach [All99]. To show the efficient non-trivial sampling required to derive this lower bound, we need a lemma about the evaluation of succinctly described threshold circuits. For convenience, we will work with Majority circuits instead, and then use the fact that uniform depth d threshold circuits can be simulated by uniform depth $d+1$ Majority circuits.

For any positive integer d, define the language Succinct-Maj_d^0-Eval to be the set of pairs $< C, x >$, where C is a Maj_d circuit of size n encoding a Maj_d circuit C' on n variables, and x is an input of length n, satisfying the condition that $C'(x) = 1$.

Lemma 46. *Let d be any positive integer. Succinct-Maj_d-Eval is in CH.*

Proof. We show how to solve Succinct-Maj_d^0-Eval efficiently on a threshold Turing machine [PS88] with a constant number of thresholds. It is shown in [PS88] that $L \in \mathsf{CH}$ iff L can be solved efficiently on a threshold Turing machine with a constant number of thresholds. A threshold Turing machine is defined analogously to an alternating Turing machine, but instead of designated existential (resp. universal) states which are accepting if at least one succeeding configuration is accepting (resp. all succeeding configurations are accepting), we have a designated threshold state, which are accepting iff a certain number m (stored on a separate write-only threshold tape) of succeeding configurations are accepting. The number of succeeding configurations at any point when the machine is in a threshold state is specified by a position of the head on a special tape known as the guess tape - this head position can change in each deterministic transition just as with a regular Turing machine. The machine is said to have a constant number of thresholds if on any computation, the threshold state is only entered a constant number of times.

Threshold Turing machines clearly generalize alternating Turing machines. It will be more convenient for us to work with the equivalent model of majority Turing machines, where instead of a single threshold state, we have a majority state and a minority state, with the majority state (resp. the minority state) accepting iff a majority (resp. a minority) of succeeding configurations are accepting. Note that we can simulate the majority state (resp. a minority state) on a threshold Turing machine by existentially guessing a number m that is at least $K/2$ (resp. less than $K/2$), where K is the number of succeeding configurations, writing m on the threshold state, and moving into the threshold state. Here, and later, we use the fact that existential guesses can be simulated by a single threshold.

Let C be a Maj_d circuit of size n encoding a Maj_d circuit C' on n inputs, and x be an input of length n. It will not be important for us that C is a Maj_d circuit, but it will be important that C' is. The idea is to simulate C' in a top-down manner on input x. We do not have time to write C' down as our machine needs to operate in fixed polynomial time in n, while C' could be of arbitrary polynomial size, or even of size $2^{O(n)}$. So we will use C to efficiently obtain information about the local structure of the circuit C', and the majority states of our majority Turing machine M to simulate the gates of C.

We first show how M simulates the output gate of C', and then describe how it recursively simulates other gates of the circuit.

M first existential guesses the index of the output gate g_0 of C', and verifies this guess by running C on the appropriate tuple of the direct connection language for C'. The existential guessing requires a single threshold in the computation of M, and the verification can be done in time quasi-linear in the size of C, and therefore in time $O(n^2)$. M runs C to determine whether g_0 is a majority gate or a minority gate, setting a bit b to 1 if the former is the case and to 0 otherwise. M then computes the number of children k_0 of g_0, again by guessing this number exisentially, and running C to verify the guess. Let $p_0 = \lceil \log(k_0) \rceil$. M moves the head of its guess tape to the p_0'th cell, indicating that there will be 2^{p_0} successor configurations, and moves into a Majority state if $b = 1$ and into a Minority state otherwise. By the rules of operation of threshold Turing machines [PS88], any given successor configuration is encoded by a bit-string y of length p_0 written on the guess tape at this point. M will interpret the successor configuration corresponding to y as follows. If y represents a number $k(y)$ in binary such that $k(y) \leqslant k_0$, M simulates the $k(y)$'th child of g_0. If y represents a number $k(y)$ such that $k(y) > k_0$, M rejects immediately if $k_0 - k(y)$ is odd, and accepts if $k_0 - k(y)$ is even.

To simulate a child g_1 of g_0, M repeats the process above for g_1 in case g_1 is a gate, while if g_1 is an input bit x_i, M reads x_i and accepts iff $x_i = 1$. Note that the depth of recursion of the process above is at most the depth of the circuit C', which is a constant d. Each recursion step requires a constant number of thresholds, and takes time $O(n^2)$ (corresponding to a constant number of invocations of C and other simple computations that can be done very efficiently). Thus, the threshold complexity of M is indeed constant, and its running time is $O(n^2)$. It can be seen inductively that M does correctly simulate any given gate g of C', and hence outputs the correct answer for C' on x. □

Theorem 47. *Let d be any positive integer.There is efficient non-trivial sampling with* CH *oracle for the succinct version of* $\mathsf{Dense} - \mathsf{Maj}_d - \mathsf{SAT}$.

Proof. As in the proofs of Theorems 31 and 44, the idea is to sample several independently random strings, and output the lexicographically first one that satisfies the succinctly represented Maj_d circuit.

Let k be any positive integer. We show that there is an algorithm A with CH oracle and a constant $c > 0$ such that for all constants $b > 0$ the following holds: Given a Maj_d circuit C of size at most n that encodes a TC^0 circuit C' of size n^b on n variables accepting at least a $2/3$ fraction of assignments, A outputs some fixed satisfying assignment of C' with probability at least $n^k/2^n$ in time $O(n^c)$.

A generates uniformly random strings $x_1, \ldots, x_t$ for $t = n^{k+2}$, each of length n. For each $i \in [t]$, A makes an oracle call to Succinct-Maj_d-Eval to evaluate C' on x_i, and receives an answer b_i. A outputs the lexicographically smallest x_i such that $b_i = 1$, and the string 0^n otherwise. The argument that A outputs a fixed satisfying assignment with probability at least $n^k/2^n$ is the same as in the proof of Theorem 31. Clearly A runs in time $O(n^{k+3})$, and by setting $c = k + 3$, independent of b, we complete the proof.

□

Corollary 48. *[All99]* $\mathsf{Permanent}$ *does not have* $\mathsf{LOGTIME}$*-uniform* TC^0 *circuits of polynomial size.*

Proof. If $\mathsf{Permanent}$ has $\mathsf{LOGTIME}$-uniform TC^0_d circuits of polynomial size for some constant depth d, then $\mathsf{Permanent}$ has $\mathsf{LOGTIME}$-uniform Maj_{d+1} circuits of polynomial size [Hof96, GK98]. Since the class of Maj_{d+1} circuits of polynomial size is polynomially simulatable and closed under projections, we can apply Theorem 29 with the sampling algorithm given by Theorem 47, and derive a contradiction. □

We explain briefly how the proofs of Corollary 45 and Corollary 48 differ from the standard proofs. The standard proof of the space hierarchy [SHL65] combines two ingredients: (i) The existence of a space-efficient universal Turing machine U that can simulate any Turing machine M with at most a constant factor overhead in space, and (ii) The idea of directly diagonalizing against all Turing machines operating in a given space bound by mapping inputs x to Turing machines M_x in a surjective way and doing the opposite of M_x on x. The proof of Corollary 45 replaces the simulation ingredient (i) with the existence of a non-trivial space-efficient sampling algorithm, and the direct diagonalization part (ii) with a different diagonalization argument based on resource-bounded Kolmogorov complexity.

The standard proof of Allender's lower bound [All99] is an indirect diagonalization argument with the following parts: (i) A time hierarchy theorem for threshold Turing machines proved in an analogous way to the space hierarchy theorem, with a simulation step and a direct diagonalization step, and (ii) An inductive argument showing that if the Permanent has small uniform threshold circuits, so does every level of the counting hierarchy; then combining (i) and (ii) to derive a contradiction. The proof of Corollary 48 still uses the ingredient (ii), but replaces the simulation step of (i) with a sampling step, and the direct diagonalition step with a diagonalization argument based on resource-bounded Kolmogorov complexity.

The broader point we wish to make is that our framework is capable of capturing both direct and indirect diagonalization arguments, since the sampling condition we require seems weaker than the simulation conditions in previous diagonalization arguments, and hence potentially capable of proving a broader class of lower bounds. This is related to Theorem 31, which shows that efficient non-trivial sampling exists unconditionally in a white box setting.

5.2 New Lower Bounds

Ideally, we would like to be able to use our new approach to attack frontier open problems in uniform circuit lower bounds, such as separating NP from $\mathsf{LOGTIME}$-uniform ACC^0 and separating NP from $\mathsf{LOGTIME}$-uniform TC^0_2. While we are unable to do this, we are able to show lower bounds in NP against interesting subclasses of these circuit classes, namely against $\mathsf{LOGTIME}$-uniform AC^0 circuits with $\mathsf{Mod}m$ gates at the bottom (for an arbitrary positive integer m), and against $\mathsf{LOGTIME}$-uniform AC^0 circuits with Thr gates at the bottom. To show lower bounds in NP against the non-uniform versions of these classes is a longstanding open problem (though we do know lower bounds in NQP [Wil14, MW20]), and to the best of our knowledge, lower bounds in NP against the uniform versions have also been open so far.

Theorem 49. *Let d, m be any positive integers.* NP *does not have polynomial-size* $\mathsf{LOGTIME}$*-uniform* $\mathsf{AC}^0_d \circ (\mathsf{Mod}m)$ *circuits.*

Proof. Let $\mathfrak{C}$ be the class of $\mathsf{AC}^0_d \circ (\mathsf{Mod}m)$ circuits of polynomial size. We will show that there is efficient non-trivial sampling for the succinct version of $\mathsf{Dense} - \mathfrak{C} - \mathsf{SAT}$, and then apply Theorem 21, since $\mathfrak{C}$ is polynomially simulatable. This implies that for each d, NP does not have $\mathsf{LOGTIME}$-uniform $\mathsf{AC}^0_d \circ (\mathsf{Mod}m)$ circuits of polynomial size.

Our efficient non-trivial sampling algorithm employs the same idea and proof as in Theorem 47, of picking several independently random assignments and outputting the lexicographically first one that satisfies the succinctly represented circuit. Thus our task reduces to giving a PH algorithm for Succinct-$\mathfrak{C}$-Eval, analogous to Lemma 46.

Let C be an $\mathsf{AC}^0_d \circ (\mathsf{Mod}m)$ circuit of size n encoding an $\mathsf{AC}^0_d \circ (\mathsf{Mod}m)$ circuit C' on n input bits that accepts at least $2/3$ of its satisfying assignments, and x be an input of length n. We define an alternating Turing machine M running in time $\mathsf{poly}(n)$ which, on input $< C, x >$, accepts iff C' accepts x.

M simulates C' implicitly in a top-down fashion, as in the proof of Lemma 46. To verify that $C'(x)$ is 1, M first existentially guesses the index of the top gate g of C', and verifies this guess by simulating C in time $O(n^2)$. It then runs C again to determine whether g is an AND or OR gate, setting a bit b to 1 if the gate is AND and to 0 otherwise. It then finds the arity of g by guessing a number k such that the k'th input to g is defined but the $k+1$'th input is not - this guess can be verified using 2 simulations of C. If $b = 1$, it universally guesses a $j \in [k]$ and verifies recursively that the j'th input gate of g (whose index it can identify by another simulation of C) evaluates to 1. If $b = 0$, it existentially guesses a $j \in [k]$ and verifies recursively that the j'th input gate of g evaluates to 1.

The evaluation of any gate proceeds in the same way as for the top gate, as long as the gate is not a $\mathsf{Mod}m$ gate. For $\mathsf{Mod}m$ gates, which are only found in the bottom layer, we proceed differently. By our definition of $\mathsf{Mod}m$ gates, any such gate g is of the form $\Sigma_{i=1}^{n} a_i x_i \in S$ modulo m, where each $0 \leqslant a_i < m$, and $S \subseteq \{0, 1, \ldots, m-1\}$. By our assumption on uniformity of $\mathsf{Mod}m$ gates, the numbers a_i and the set S can all be determined by $O(n)$ simulations of C, and hence in time $O(n^3)$. Once this information has been determined, M simply evaluates g on x, which can be done in time $O(n^2)$.

The number of alternations on any computation of M is at most the depth of C', i.e., d, and the time taken by M is $O(n^3)$. It is straightforward to verify inductively that M does indeed accept x iff C' does.

□

Examining the proof of Theorem 49, the only fact we used about the bottom layer of gates is that they can be evaluated in fixed polynomial time in n. Hence the same proof also gives the following result.

Theorem 50. *Let d be any positive integer.* NP *does not have polynomial-size* LOGTIME*-uniform* $\mathsf{AC}^0_d \circ \mathsf{Thr}$ *circuits.*

Theorems 49 and 50 above together capture the content of Theorem 5 from the Introduction.

We note that the simulation arguments used for the non-trivial sampling in the proofs of Theorem 49 and Theorem 50 are fairly generic. This leads us to believe that there might be alternate proofs of these results using a more standard indirect diagonalization approach. However, these results already seem new, and exploiting the fact that we only need *sampling* rather than *simulation* to apply Theorem 21 might lead to even stronger lower bounds.

6 Future Work

We describe here some directions for future work.

The main direction is to develop new algorithmic ideas for the sampling problems we consider, and use these to prove new lower bounds. In particular, it would be interesting to explore if the ideas and techniques of [AIK04] and [Vio12] are useful here. A particular circuit class of interest is the class of quasi-polynomial size SYM^+ circuits, i.e., depth-two circuits with a top symmetric gate and polylogarithmic fan-in AND gates at the bottom. Efficient non-trivial sampling for the succinct version of this class would imply that NP does not have LOGTIME-uniform ACC^0 circuits of polynomial size.

Another question is whether there is an analogous approach to separating NP and PSPACE from probabilistic uniform classes. For example, are there natural sampling tasks or similar tasks such that efficient solutions imply $\mathsf{NP} \not\subseteq \mathsf{BPP}$?

While we provide one potential algorithmic approach to uniform lower bounds, there might be others. It would be interesting to look into this, especially if these other approaches are more feasible with the algorithmic techniques we have at present.

7 Acknowledgments

I am grateful to Eric Allender, Arkadev Chattopadhyay, Hanlin Ren, Zhenjian Lu, Igor Oliveira, Ronen Shaltiel and Srikanth Srinivasan for useful discussions.

References

[AB09] S. Arora and B. Barak. *Complexity Theory: A Modern Approach.* Cambridge University Press, Cambridge, 2009.

[ABK+06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35(6):1467–1493, 2006.

[AC19] Josh Alman and Lijie Chen. Efficient construction of rigid matrices using an NP oracle. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 1034–1055. IEEE Computer Society, 2019.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 166–175. IEEE Computer Society, 2004.

[Ajt83] Miklos Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[All99] Eric Allender. The permanent requires large uniform threshold circuits. *Chic. J. Theor. Comput. Sci.*, 1999, 1999.

[All01] Eric Allender. When worlds collide: Derandomization, lower bounds, and kolmogorov complexity. *Foundations of Software Technology and Theoretical Computer Science*, 21, 2001.

[All20] Eric Allender. The new complexity landscape around circuit minimization. In *Language and Automata Theory and Applications - 14th International Conference, LATA 2020*, volume 12038 of *Lecture Notes in Computer Science*, pages 3–16. Springer, 2020.

[AW08] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*, 2008. To appear.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[BHPT20] Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal. Rigid matrices from rectangular pcps or: Hard claims have complex proofs. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 858–869. IEEE, 2020.

[BIP16] Peter Bürgisser, Christian Ikenmeyer, and Greta Panova. No occurrence obstructions in geometric complexity theory. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 386–395. IEEE Computer Society, 2016.

[BIS90] David Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41, 1990.

[BM88] László Babai and Shlomo Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[Che19] Lijie Chen. Non-deterministic quasi-polynomial time is average-case hard for ACC circuits. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 1281–1304. IEEE Computer Society, 2019.

[CLW20] Lijie Chen, Xin Lyu, and R. Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 1–12. IEEE, 2020.

[Coo71] Stephen Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[CR20] Lijie Chen and Hanlin Ren. Strong average-case lower bounds from non-trivial derandomization. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1327–1334. ACM, 2020.

[FK09] Lance Fortnow and Adam R. Klivans. Efficient learning algorithms yield circuit lower bounds. *Journal of Computer and System Sciences*, 75(1):27–36, 2009.

[For09] Lance Fortnow. The status of the P versus NP problem. *Communications of the ACM*, 52(9):78–86, 2009.

[FSS84] Merrick Furst, James Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

[FSUV13] Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. On beating the hybrid argument. *Theory Comput.*, 9:809–843, 2013.

[GG11] Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electron. Colloquium Comput. Complex.*, TR11-136, 2011.

[GK98] Mikael Goldmann and Marek Karpinski. Simulating threshold circuits by majority circuits. *SIAM J. Comput.*, 27(1):230–246, 1998.

[GKLO22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor Carboni Oliveira. Probabilistic kolmogorov complexity with applications to average-case complexity. In *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 16:1–16:60. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 6–20, 1986.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 247–258. IEEE Computer Society, 2018.

[Hir20] Shuichi Hirahara. Unexpected hardness results for kolmogorov complexity under uniform reductions. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 1038–1051. ACM, 2020.

[Hir21] Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing,*, pages 292–302. ACM, 2021.

[Hof96] Thomas Hofmeister. A note on the simulation of exponential threshold weights. In Jin-yi Cai and C. K. Wong, editors, *Computing and Combinatorics, Second Annual International Conference, COCOON '96, Hong Kong, June 17-19, 1996, Proceedings*, volume 1090 of *Lecture Notes in Computer Science*, pages 136–141. Springer, 1996.

[HS17] Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 220–229, 1997.

[KKO13] Adam R. Klivans, Pravesh Kothari, and Igor Carboni Oliveira. Constructing hard functions using learning algorithms. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013*, pages 86–97. IEEE Computer Society, 2013.

[LO22] Zhenjian Lu and Igor Carboni Oliveira. Theory and applications of probabilistic kolmogorov complexity. *Bull. EATCS*, 137, 2022.

[LOS21] Zhenjian Lu, Igor Carboni Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 303–316. ACM, 2021.

[LOZ22] Zhenjian Lu, Igor Carboni Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded kolmogorov complexity. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 92:1–92:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[MS01] Ketan Mulmuley and Milind Sohoni. Geometric complexity theory I: an approach to the P vs. NP and related problems. *SIAM Journal on Computing*, 31(2):496–526, 2001.

[Mul11] Ketan Mulmuley. On p vs np and geometric complexity theory: dedicated to sri ramakrishna. *Journal of the Association of Computing Machinery*, 58(2), 2011.

[Mur71] Saburo Muroga. *Threshold logic and its applications.* Wiley, 1971.

[MW20] Cody D. Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM J. Comput.*, 49(5), 2020.

[Nis91] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[Oli19] Igor Carboni Oliveira. Randomness and intractability in kolmogorov complexity. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019*, volume 132 of *LIPIcs*, pages 32:1–32:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[OS17] Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *32nd Computational Complexity Conference, CCC 2017*, volume 79 of *LIPIcs*, pages 18:1–18:49. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[PS88] Ian Parberry and Georg Schnitger. Parallel computation with threshold functions. *J. Comput. Syst. Sci.*, 36(3):278–302, 1988.

[Raz85] Alexander Razborov. Lower bounds for the monotone complexity of some boolean functions. *Soviet Mathematics Doklady*, 31:354–357, 1985.

[Raz87] Alexander Razborov. Lower bounds on the size of bounded-depth networks over the complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[RR97] Alexander Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

[San20] Rahul Santhanam. Pseudorandomness and the minimum circuit size problem. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020*, volume 151 of *LIPIcs*, pages 68:1–68:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[SHL65] Richard Stearns, Juris Hartmanis, and Philip Lewis. Hierarchies of memory limited computations. In *Proceedings of the Sixth Annual Symposium on Switching Circuit Theory and Logical Design*, pages 179–190. IEEE, 1965.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual Symposium on Theory of Computing*, pages 77–82, 1987.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Val79] Leslie Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Vio12] Emanuele Viola. The complexity of distributions. *SIAM J. Comput.*, 41(1):191–218, 2012.

[Wag86] Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3):325–356, 1986.

[Wil10] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*, pages 231–240, 2010.

[Wil11] Ryan Williams. Non-uniform ACC circuit lower bounds. In *Proceedings of 26th Annual IEEE Conference on Computational Complexity*, pages 115–125, 2011.

[Wil14] Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 194–202, 2014.

[Wil16] R. Ryan Williams. Natural proofs versus derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016.

[Wil18] Richard Ryan Williams. Limits on representing boolean functions by linear combinations of simple functions: Thresholds, relus, and low-degree polynomials. In *33rd Computational Complexity Conference, CCC 2018*, volume 102 of *LIPIcs*, pages 6:1–6:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *26th Annual Symposium on Foundations of Computer Science*, pages 1–10. IEEE Computer Society, 1985.