Constructive Separations and Their Consequences

Lijie Chen
Elect. Eng. & Comp. Sci.
MIT
Cambridge, MA USA
lijieche@mit.edu

Ce Jin
Elect. Eng. & Comp. Sci.
MIT
Cambridge, MA USA
cejin@mit.edu

Rahul Santhanam
Dept. of Computer Science
University of Oxford
Oxford, UK
rahul.santhanam@cs.ox.ac.uk

R. Ryan Williams
Elect. Eng. & Comp. Sci.
MIT
Cambridge, MA USA
rrw@mit.edu

2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS) | 978-1-6654-2055-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/FOCS52979.2021.00069

***Abstract*—For a complexity class C and language L, a *constructive separation of "L is not in C"* gives an efficient algorithm (also called a *refuter*) to find counterexamples (bad inputs) for every C-algorithm attempting to decide L. We study the questions: Which lower bounds can be made constructive? What are the consequences of constructive separations? We build a case that "constructiveness" serves as a dividing line between many weak lower bounds we know how to prove, and strong lower bounds against P, ZPP, and BPP. Put another way, constructiveness is the opposite of a complexity barrier: it is a property we want lower bounds to have. Our results fall into three broad categories.**

1. For many separations, making them constructive would imply breakthrough lower bounds. Our first set of results shows that, for many well-known lower bounds against streaming algorithms, one-tape Turing machines, and query complexity, as well as lower bounds for the Minimum Circuit Size Problem, making these lower bounds constructive would imply breakthrough separations ranging from "EXP not equal to BPP" to even "P not equal to NP".

2. Most conjectured uniform separations can be made constructive. Our second set of results shows that for most major open problems in lower bounds against P, ZPP, and BPP, including "P not equal to NP", "P not equal to PSPACE", "P not equal to PP", "ZPP not equal to EXP", and "BPP not equal to NEXP", any proof of the separation would further imply a *constructive* separation. Our results generalize earlier results for "P not equal to NP" [Gutfreund, Shaltiel, and Ta-Shma, CCC 2005] and "BPP not equal to NEXP" [Dolev, Fandina and Gutfreund, CIAC 2013]. Thus any proof of these strong lower bounds must also yield a constructive version, compared to many weak lower bounds we currently know.

3. Some separations cannot be made constructive. Our third set of results shows that certain complexity separations cannot be made constructive. We observe that for all super-polynomially growing functions t, there are no constructive separations for detecting high t-time Kolmogorov complexity (a task which is known to be not in P) from any complexity class, unconditionally. We also show that under plausible conjectures, there are languages in NP - P for which there are no constructive separations from any complexity class.

***Keywords*-computational complexity; circuit complexity; lower bounds; barriers; refuters**

I. Introduction

A primary goal of complexity theory is to derive strong complexity lower bounds for natural computational problems. When a lower bound holds for a problem Π against a model $\mathcal{M}$ of algorithms, this implies that for each algorithm A from $\mathcal{M}$, there is an infinite sequence of *counterexamples* $\{x_i\}$ for which A fails to solve Π correctly.[1] In this paper, we study the question: can such a family of counterexamples be constructed efficiently, for fixed Π and a given algorithm A in $\mathcal{M}$? We call a positive answer to this question a *constructive* separation of Π from $\mathcal{M}$.

There are several motivations for studying this question in a systematic way for natural problems Π and models $\mathcal{M}$. Computer science is inherently a constructive discipline, and it is natural to ask if a given lower bound can be made constructive. Indeed, this can be seen as an "explicit construction" question of the kind that is studied intensively in the theory of pseudorandomness, where we may have a proof of existence of certain objects with optimal parameters, e.g., extractors, and would like to construct such objects efficiently. At a high level, cryptography is based on the constructiveness of lower bounds: we need lower bounds to exist, and we also need to sample hard instances efficiently.[2]

Our primary motivation is to understand the general lower bound problem better! Constructive lower bounds have led to some recent resolutions of lower bound problems in complexity theory, and we believe they will lead to more. In his Geometric Complexity Theory approach, Mulmuley [2] suggests that in order to break the "self referential paradox" of P vs NP and related problems[3], one has to shoot for **algorithms** which can efficiently find counterexamples for any algorithms claiming to solve the conjectured hard language. This view has been dominant in the GCT approach towards the VNP vs. VP problem [3], [4], [5].

The ability to "construct bad inputs for a hard function" has also been critical to some recent developments in

[1]If the family of counterexamples was finite, we could hard-code them into the algorithm A to give a new algorithm A' that solves Π correctly, for most "reasonable" models $\mathcal{M}$.

[2]A superficial difference is that in cryptography, we would like to construct instances that are hard for *any* efficient algorithm, whereas in our setting, there is a fixed algorithm, and we would like to construct instances that are hard for it. This difference vanishes when the problem Π has an optimal algorithm in the sense of Levin [1], since any instance that is hard for the optimal algorithm is hard for *all* efficient algorithms.

[3]Namely, since the P vs. NP problem is a universal statement about mathematics that says that discovery is hard, why could it not preclude its own proof and hence be independent of the axioms of set theory?

(Boolean) complexity theory. Chen, Jin, and Williams [6] studied a notion of constructive proof they called *explicit obstructions*. They show several "sharp threshold" results for explicit obstructions, demonstrating (for example) that explicit obstructions unconditionally exist for $n^{2-\varepsilon}$-size DeMorgan formulas, but if they existed for $n^{2+\varepsilon}$-size formulas then one could prove the breakthrough lower bound $\mathsf{EXP} \not\subset \mathsf{NC}^1$. (We discuss the differences between their work and ours in Section II-E, along with other related work.)

Constructive lower bounds have also been directly useful in proving recent lower bounds. Chen, Lyu, and Williams [7] recently showed how to strengthen several prior lower bounds for E^{NP} based on the algorithmic method to hold *almost everywhere*. A key technical ingredient in this work was the development of an *constructive* version of a nondeterministic time hierarchy that was already known to hold almost everywhere [8]. The "refuter" in the constructive lower bound (the algorithm producing counterexamples) is used directly in the design of the hard function in E^{NP}. This gives a further motivation to study when lower bounds can be made constructive.

The Setup: More formally, for a function $f\colon \{0,1\}^\star \to \{0,1\}$ and algorithm A, we define the *search problem $D_{f,A}$ of counterexamples* to be $D_{f,A} := \{(1^n, x) \mid x \in \{0,1\}^n \wedge f(x) \neq A(x)\}$. Intuitively, a *refuter* for f against A is an algorithm for the search problem $D_{f,A}$, proving in an algorithmic way that the algorithm A cannot compute f. (This notion seems to have first been introduced by Kabanets [9] in the context of derandomization; see Section II-E for more details.)

Definition I.1 (Refuters and Constructive Separations)**.** An algorithm R is a *refuter for f against A* if there are infinitely many n such that $(1^n, R(1^n)) \in D_{f,A}$. For complexity classes $\mathcal{C}$ and $\mathcal{D}$, we say there is a *$\mathcal{D}$-constructive separation of $f \notin \mathcal{C}$* if for every algorithm A computable in $\mathcal{C}$ there is a refuter for f against A that is computable in $\mathcal{D}$.

Note that we allow the refuter algorithm to depend on the algorithm A. The notion of refuter can also be extended naturally to randomized algorithms. Formally, we say a randomized algorithm R *solves* $D_{L,A}$ infinitely often, if for infinitely many integers n, $(1^n, R(1^n)) \in D_{L,A}$ with probability at least $2/3$. If for these infinitely many integers n, it holds in addition that $R(1^n)$ either outputs $\perp$ or a counterexample such that $(1^n, R(1^n)) \in D_{L,A}$, we say R is a *zero-error randomized algorithm solving $D_{L,A}$*.

At this point it is natural to ask:

Question 1: Which lower bounds imply a corresponding *constructive* lower bound?

Naively, one might expect that the answer to Question 1 is positive when the lower bound is relatively easy to prove. We show that this intuition is wildly inaccurate. On the one hand, we show that for many natural examples of problems Π and weak models $\mathcal{M}$, a lower bound is easily provable (and well-known), but *constructivizing* the *same* lower bound would imply a breakthrough separation in complexity theory (a much stronger type of lower bound). On the other hand, we show that for many "hard" problems Π and strong models $\mathcal{M}$, a lower bound for Π against $\mathcal{M}$ *automatically* constructivizes: the existence of the lower bound alone can be used to derive an algorithm that produces counterexamples. So, in contrast with verbs such as "relativize" [10], "algebrize" [11], and "naturalize" [12], we *want* to prove lower bounds that constructivize! We are identifying a *desirable* property of lower bounds.

We now proceed to discuss our results in more detail, and then give our interpretation of these results.

A. Most Conjectured Poly-Time Separations Can Be Made Constructive

Generalizing prior work [13], [14], we show that for most major open lower bound problems regarding polynomial time, their resolution implies corresponding *constructive* lower bounds for most complete problems.

Theorem I.2. *Let $\mathcal{C} \in \{\mathsf{P}, \mathsf{ZPP}, \mathsf{BPP}\}$ and let $\mathcal{D} \in \{\mathsf{NP}, \Sigma_2\mathsf{P}, \ldots, \Sigma_k\mathsf{P}, \ldots, \mathsf{PP}, \mathsf{PSPACE}, \mathsf{EXP}, \mathsf{NEXP}, \mathsf{EXP}^{\mathsf{NP}}\}$. Then $\mathcal{D} \not\subseteq \mathcal{C}$ implies that for every paddable $\mathcal{D}$-complete language L, there is a $\mathcal{C}$-constructive[4]separation of $L \notin \mathcal{C}$.[5] Furthermore, $\oplus\mathsf{P} \not\subseteq \mathcal{C}$ implies that for every paddable $\oplus\mathsf{P}$-complete language L, there is a* BPP*-constructive separation of $L \notin \mathcal{C}$.*

In other words, for many major separation problems such as $\mathsf{PP} \neq \mathsf{BPP}$, $\mathsf{EXP} \neq \mathsf{ZPP}$, and $\mathsf{PSPACE} \neq \mathsf{P}$, proving the separation automatically implies constructive algorithms that can produce counterexamples to any given weak algorithm. We find Theorem I.2 to be mildly surprising: intuitively it seems that proving a constructive lower bound should be strictly stronger than simply proving a lower bound. (Indeed, we will later see other situations where making *known* lower bounds constructive would have major consequences!) Moreover, for separations beyond $\mathsf{P} \neq \mathsf{NP}$, the polynomial-time refuters guaranteed by Theorem I.2 are producing hard instances for a problem that does not have short certificates, in general.

B. Unexpected Consequences of Making Some Separations Constructive

Given Theorem I.2, we see that most of the major open problems surrounding polynomial-time lower bounds would yield constructive separations. Can *all* complexity separations can be made constructive? It turns out that

[4]When we use the term $\mathcal{C}$-constructive for a class $\mathcal{C}$ such as P or BPP, we mean it is the functional version of the class

[5]Throughout this paper when we say a language L is $\mathcal{D}$-complete, we mean it is $\mathcal{D}$-complete under polynomial-time many-one reductions. A language L is *paddable* if there is a deterministic polynomial-time algorithm that receives $(x, 1^n)$ as input, where string x has length at most $n-1$, and then outputs a string $y \in \{0,1\}^n$ such that $L(x) = L(y)$.

for several "weak" lower bounds proved by well-known methods, making them constructive requires proving *other* breakthrough lower bounds!

Thus, there seems to be an algorithmic "dividing line" between many lower bounds we are able to prove, and many of the longstanding lower bounds that seem perpetually out of reach. The longstanding separation questions (as seen in Theorem I.2) *require* a constructive proof: an efficient algorithm that can print counterexamples. Here we show that many lower bounds we are able to prove do not require constructivity, but if they could be made constructive then we would prove a longstanding separation! In our minds, these results confirm the intuition of Mulmuley that we should "go for explicit proofs" in order to make serious progress on lower bounds, especially uniform ones.

Constructive Separations for (Any) Streaming Lower Bounds Imply Breakthroughs: It is well-known that various problems are *unconditionally* hard for low-space randomized streaming algorithms. For example, from the randomized communication lower bound for the Set-Disjointness (DISJ) problem [15], [16], [17], it follows that no $n^{1-\varepsilon}$-space randomized streaming algorithm can solve DISJ on $2n$ input bits.[6]

Clearly, every $n^{o(1)}$-space streaming algorithm for DISJ *must* fail to compute DISJ on some input (indeed, it must fail on many inputs). We show that efficient refuters against streaming algorithms attempting to solve *any* NP problem would imply a breakthrough lower bound on *general* randomized algorithms, not just streaming algorithms.

Theorem I.3. *Let $f(n) \geq \omega(1)$. For every language $L \in \mathsf{NP}$, a P^{NP}-constructive separation of L from uniform randomized streaming algorithms with $O(n \cdot (\log n)^{f(n)})$ time and $O(\log n)^{f(n)}$ space[7] implies $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{BPP}$.*

Essentially every lower bound proved against streaming algorithms in the literature holds for a problem whose decision version is in NP. Theorem I.3 effectively shows that if *any* of these lower bounds can be made constructive, even in a P^{NP} sense, then we would separate randomized polynomial time from $\mathsf{EXP}^{\mathsf{NP}}$, a longstanding open problem in complexity theory. A more constructive separation (with an algorithm in a lower complexity class than P^{NP}) would imply a stronger separation. We are effectively showing that the counterexamples printed by such a refuter algorithm must encode a function that is hard for *general* randomized algorithms.

Stronger lower bounds follow from more efficient refuters for DISJ against randomized streaming algorithms. At the extreme end, we find that uniform circuits refuting DISJ against randomized streaming algorithms would even imply $\mathsf{P} \neq \mathsf{NP}$.

Theorem I.4. *Let $f(n) \geq \omega(1)$. A polylogtime-uniform-AC^0-constructive separation of DISJ from randomized streaming algorithms with $O(n \cdot (\log n)^{f(n)})$ time and $O(\log n)^{f(n)}$ space[8] implies $\mathsf{P} \neq \mathsf{NP}$.*

To recap, it is well-known that DISJ does not have randomized streaming algorithms with $O(n \cdot (\log n)^{f(n)})$ time and $O(\log n)^{f(n)}$ space, even for $f(n) \leq o(\log n / \log\log n)$, by communication complexity arguments. We are saying that, if (given the code of such an algorithm) we can efficiently construct hard instances of DISJ for that algorithm, then strong lower bounds follow. *That is, making communication complexity arguments constructive would imply strong unconditional lower bounds.*

Constructive Separations for One-Tape Turing Machines Imply Breakthroughs: Next, we show how making some rather old lower bounds constructive would imply a circuit complexity breakthrough. It has been known at least since Maass [18] that nondeterministic one-tape Turing machines require $\Omega(n^2)$ time to simulate nondeterministic multitape Turing machines. However, those lower bounds are proved by non-constructive counting arguments. We show that if there is a P^{NP} algorithm that can produce bad inputs for a given one-tape Turing machine, then E^{NP} requires exponential-size circuits. This in turn would imply $\mathsf{BPP} \subseteq \mathsf{P}^{\mathsf{NP}}$, a breakthrough simulation of randomized polynomial time.

Theorem I.5. *For every language L computable by a nondeterministic $n^{1+o(1)}$-time RAM, a P^{NP}-constructive separation of L from nondeterministic $O(n^{1.1})$-time one-tape Turing machines implies $\mathsf{E}^{\mathsf{NP}} \not\subset \mathsf{SIZE}[2^{\delta n}]$ for some constant $\delta > 0$.*

Constructive Separations for Query Lower Bounds Imply Breakthroughs: Now we turn to query complexity. Consider the following basic problem $\mathsf{PromiseMAJORITY}_{n,\varepsilon}$ for a parameter $\varepsilon < 1/2$.

$\mathsf{PromiseMAJORITY}_{n,\varepsilon}$: *Given an input $x \in \{0,1\}^n$, letting $p = \frac{1}{n}\sum_{i=1}^{n} x_i$, distinguish between the cases $p < 1/2 - \varepsilon$ or $p > 1/2 + \varepsilon$.*

This is essentially the "coin problem" [19]. It is well-known that every randomized query algorithm needs $\Theta(1/\varepsilon^2)$ queries to solve $\mathsf{PromiseMAJORITY}_{n,\varepsilon}$ with constant success probability (uniform random sampling is the best one can do). That is, any randomized query algorithm making $o(1/\varepsilon^2)$ must make mistakes on some inputs, with high

[6]Recall in the DISJ problem, Alice is given an n-bit string x, Bob is given an n-bit string y, and the goal is to determine whether their inner product $\sum_{i=1}^{n} x_i y_i$ is nonzero.

[7]That is, for every such randomized streaming algorithm A, there is a P^{NP} refuter B such that $B(1^n)$ prints an input x of length n such that A decides whether $x \in L$ incorrectly, for infinitely many n.

[8]That is, for every such randomized streaming algorithm A, there is a polylogtime-uniform AC^0 circuit family $\{C_n\}$ such that A fails to solve DISJ on $2n$-bit inputs correctly on the output $C_n(1^n)$ for infinitely many n.

probability. We show that constructing efficient refuters for this simple sampling lower bound would imply $\mathsf{P} \neq \mathsf{NP}$!

Theorem I.6. *Let ε be a function of n satisfying $\varepsilon(n) \leq 1/(\log n)^{\omega(1)}$.*

- *If there is a polylogtime-uniform-AC^0-constructive separation of $\mathsf{PromiseMAJORITY}_{n,\varepsilon}$ from randomized query algorithms A using $o(1/\varepsilon^2)$ queries and $\mathrm{poly}(1/\varepsilon)$ time, then $\mathsf{NP} \neq \mathsf{P}$.*
- *If there is a polylogtime-uniform-NC^1-constructive separation of $\mathsf{PromiseMAJORITY}_{n,\varepsilon}$ from randomized query algorithms A using $o(1/\varepsilon^2)$ queries and $\mathrm{poly}(1/\varepsilon)$ time, then $\mathsf{PSPACE} \neq \mathsf{P}$.*

Note that $\mathsf{PromiseMAJORITY}_{n,\varepsilon}$ can be easily computed in NC^1. If for every randomized query algorithm A running in n^α time and making n^α queries for some $\alpha > 0$, we can always find inputs in NC^1 on which A makes mistakes, then would separate P from PSPACE.

Constructive Separations for MCSP *Against* AC^0 *Imply Breakthroughs:* Informally, the Minimum Circuit Size Problem (MCSP) is the problem of determining the circuit complexity of a given 2^n-bit truth table. Recent results on the phenomenon of hardness magnification [20], [21], [22], [23], [6] show that, for various restricted complexity classes $\mathcal{C}$:

- Strong lower bounds against $\mathcal{C}$ are known for explicit languages.
- Standard complexity-theoretic hypotheses imply that such lower bounds should hold also for MCSP (and its variants).
- However, actually proving that $\mathsf{MCSP} \notin \mathcal{C}$ would imply a breakthrough complexity separation.

In such situations, there is also often a slightly weaker lower bound against $\mathcal{C}$ that can be shown for MCSP, suggesting that we are quantitatively "close" to a breakthrough separation in some sense.

We show that a similar phenomenon holds for constructive separations. It is well known that versions of MCSP are not in AC^0 [24], but strongly constructive separations are not known. We show that strongly constructive separations would separate P from NP, and that they exist under a standard complexity hypothesis. Moreover, we show that slightly weaker constructive separations *do* exist, and the strong constructive separations we seek do hold for other hard problems such as Parity.

In the following, $\mathsf{MCSP}[f(n)]$ is the computational problem that asks whether a Boolean function on n bits, represented by its truth table, has circuits of size at most $f(n)$.

Theorem I.7. *Let $f(n) \geq n^{\log(n)^{\omega(1)}}$ be any time-constructive super-quasipolynomial function. The following hold:*

1) *(Major Separation from Constructive Lower Bound) If there is a polylogtime-uniform $\mathsf{AC}^0[\text{quasipoly}]$ refuter for $\mathsf{MCSP}[f(n)]$ against every polylogtime-uniform AC^0 algorithm, then $\mathsf{P} \neq \mathsf{NP}$.*
2) *(Constructive Lower Bound Should Exist) If $\mathsf{PH} \not\subset \mathsf{SIZE}(f(n)^2)$, then there is a polylogtime-uniform-$\mathsf{AC}^0[\text{quasipoly}]$ refuter for $\mathsf{MCSP}[f(n)]$ against every polylogtime-uniform AC^0 algorithm.*
3) *(Somewhat Constructive Lower Bound) There is a polylogtime-uniform-$\mathsf{AC}^0[2^{\mathrm{poly}(f(n))}]$ refuter for $\mathsf{MCSP}[f(n)]$ against every polylogtime-uniform AC^0 algorithm.*
4) *(Constructive Lower Bound for a Different Hard Language) There is a $\mathrm{quasipoly}(N)$-size polylogtime-uniform-$\mathsf{AC}^0[\text{quasipoly}]$-list-refuter for Parity against every polylogtime-uniform AC^0 algorithm.*

Note that in item 3, the input size N to the problem is $N = 2^n$, hence $2^{\mathrm{poly}(f(n))}$ is only slightly super-quasipolynomial in N.

Comparison with Theorem I.2: It is very interesting to contrast Theorem I.2 with the various theorems of this subsection. On the one hand, Theorem I.2 tells us that many longstanding open problems in lower bounds would automatically imply constructive separations, when resolved. On the other hand, we see that extending simple and well-known lower bounds to become constructive would resolve other major lower bounds! Taken together, we view the problem of understanding which lower bounds can be made constructive as a significant key to understanding the future landscape of complexity lower bounds.

C. Certain Lower Bounds Cannot Be Made Constructive

Finally, we can give some negative answers to our Question 1. We show that for some hard functions, there are *no* constructive separations from *any* complexity classes. Specifically, we show (unconditionally or under plausible complexity conjectures) that there are no refuters for these problems against a trivial decision algorithm that *always returns the same answer* (zero, or one). Hence, there can be no constructive separations of these hard languages from any complexity class containing the constant zero or constant one function. (All complexity classes that we know of contain both the constant zero and one function.)

For a string $x \in \{0,1\}^*$, the t-time-bounded Kolmogorov complexity of x, denote by $\mathsf{K}^t(x)$, is defined as the length of the shortest program prints x in time $t(|x|)$. We use $\mathsf{R}_{\mathsf{K}^t}$ to denote the set of strings x such that $\mathsf{K}^t(x) \geq |x| - 1$. Hirahara [25] recently proved that for any super-polynomial $t(n) \geq n^{\omega(1)}$, $\mathsf{R}_{\mathsf{K}^t} \notin \mathsf{P}$. We observe that this separation cannot be made P-constructive.

Proposition I.8. *For any $t(n) \geq n^{\omega(1)}$, there is no P-refuter for $\mathsf{R}_{\mathsf{K}^t}$ against the constant zero function.*

Since $\mathsf{R}_{\mathsf{K}^t}$ is a function in EXP, it would be interesting to find functions in NP with no constructive separations.[9] We show that under plausible conjectures, such languages in NP exist.

Theorem I.9. *The following hold:*

- *If* NE $\neq$ E*, then there is a language in* NP $\setminus$ P *that does not have* P *refuters against the constant one function.*[10]
- *If* NE $\neq$ RE*, then there is a language in* NP $\setminus$ P *that does not have* BPP *refuters against the constant one function.*[11]

Thus, under natural conjectures about exponential-time classes, there are some problems in NP with no constructive separations at all, not even against the trivial algorithm that always accepts.

D. Intuition

Let us briefly discuss the intuition behind some of our results. We will first focus on the results showing that constructive separations of known lower bounds would imply complexity breakthroughs, as we believe these are the most interesting of our paper.

Constructive Separations of Known Lower Bounds Imply Breakthroughs: Suppose for example we want to show that a constructive separation of SAT from quick low-space streaming algorithms implies $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{BPP}$. The proof is by contradiction: assuming $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{BPP}$, we aim to construct a streaming algorithm running in $n(\log n)^{\omega(1)}$ time and $(\log n)^{\omega(1)}$ space which solves 3SAT correctly on all instances produced by P^{NP} algorithms. Given a P^{NP} algorithm R, $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{BPP}$ implies $\mathsf{EXP}^{\mathsf{NP}} \subset \mathsf{P}_{/\mathsf{poly}}$, which further implies that the output of $R(1^n)$ must have circuit complexity at most $\mathrm{polylog}(n)$ (construed as a truth table).

Extending work of McKay, Murray, and Williams [21], we show that $\mathsf{NP} \subset \mathsf{BPP}$ (implied by $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{BPP}$) implies there is an $n(\log n)^{\omega(1)}$ time and $(\log n)^{\omega(1)}$ space randomized algorithm with one-sided error for finding a $\mathrm{polylog}(n)$-size circuit encoding the given length-n input, if such a circuit exists. So given any input $R(1^n)$ from a potential refuter R, our streaming algorithm can first compute a $\mathrm{polylog}(n)$-size circuit C encoding $R(1^n)$, and it construes this circuit C as an instance of the Succinct-3SAT problem. Since Succinct-3SAT $\in$ NEXP $=$ BPP, our streaming algorithm can solve Succinct-3SAT(C) in $\mathrm{polylog}(n)$ randomized time, which completes the proof.

For our results on constructive query lower bounds, we use ideas from learning theory. Set $\varepsilon \ll 1/\mathrm{poly}(\log n)$. Assuming PSPACE $=$ P, we want to show that for every n-bit string printed by an uniform NC^1 circuit C on the input 1^n, we can decide the $\mathsf{PromiseMAJORITY}_{n,\varepsilon}$ problem with $o(1/\varepsilon^2)$ randomized queries in $\mathrm{poly}(1/\varepsilon)$ time. (Then, any sufficiently constructive lower bound that $\mathsf{PromiseMAJORITY}_{n,\varepsilon}$ requires $\Omega(1/\varepsilon^2)$ queries would imply P $\neq$ PSPACE.) PSPACE $=$ P implies that for every uniform NC^1 circuit C, its output can be encoded by some $\mathrm{polylog}(n)$-size circuit D. Now, also assuming PSPACE $=$ P, this circuit D can be PAC-learned with error $\varepsilon/2$ and failure probability $1/10$ using only $\mathrm{poly}\log(n)/\varepsilon$ queries (and randomness). Let D' be the circuit we learnt through this process; it approximates D well enough that we can make $O(1/\varepsilon^2)$ random queries *to the circuit* D'*, without querying* D in $\mathrm{poly}(1/\varepsilon, \log n)$ time, and return the majority answer as a good answer for the original n-bit answer. Such an algorithm only makes $\mathrm{polylog}(n)/\varepsilon \ll o(1/\varepsilon^2)$ queries to the original input and runs in $\mathrm{poly}(1/\varepsilon)$ time.

Constructive Separations for Uniform Complexity Separations: Next, we highlight some insights behind the proof of Theorem I.2, whose proof appears in the full version of this paper [26]. The proof is divided into several different cases; we will focus on the intuition behind one of them, which applies to all complexity classes with a downward self-reducible complete language (such as PSPACE or $\Sigma_k\mathsf{P}$).

We take the PSPACE vs. P problem as an example. Gutfreund, Shaltiel, and Ta-Shma [13] showed how to construct refuters for P $\neq$ NP, but their proof utilizes the search-to-decision reduction for NP-complete problems, and no such reduction exists for PSPACE. We show how a downward self-reduction can be used to engineer a situation similar to that of [13].

Let M be a downward self-reducible PSPACE-complete language and let A be a P algorithm. We also let D be a polynomial-time algorithm defining a downward-self reduction for M, so that for all but finitely many $n \in \mathbb{N}$ and $x \in \{0,1\}^n$,

$$D(x)^{M_{\le n-1}} = M(x). \tag{1}$$

That is, D can compute $M(x)$ given access to an M-oracle for all strings of length less than $|x|$. Our key idea is that (1) *also* defines M. Assuming the polynomial-time algorithm A cannot compute M, it follows that (1) does not always hold if M is replaced by A. In particular, the following NP statement is true for infinitely many n:

$$\exists x \in \{0,1\}^n \text{ such that } D(x)^{A_{\le n-1}} \neq A(x). \tag{2}$$

Now we use a similar approach as in [13]: we use A and a standard search-to-decision reduction to find the shortest string x^* so that (2) holds. If A fails to do so, we can construct a counterexample to the claim that A solves the PSPACE-complete language M similarly to [13]. If A finds such an x^*, then by definition $A(y) = M(y)$ for all y with $|y| \le |x^*| - 1$ and we have $A(x^*) \neq M(x^*)$ from (2), also

[9]Note that $\mathsf{R}_{\mathsf{K}^t}$ is in $\mathsf{coNTIME}[t(n)]$, but it is likely not in coNP.

[10]Here, $\mathsf{E} = \mathsf{TIME}[2^{O(n)}]$, the class of languages decidable in (deterministic) $2^{O(n)}$ time, and NE is the corresponding nondeterministic class.

[11]Here, $\mathsf{RE} = \mathsf{RTIME}[2^{O(n)}]$, the class of languages decidable in randomized $2^{O(n)}$ time with one-sided error.

a counterexample.[12]

E. Organization

In Section II we introduce the necessary definitions and technical tools for this paper, as well as review other related work. In Section III we show that making known streaming and query lower bounds constructive implies major complexity separations, proving Theorem I.3 and Theorem I.4.

The full version of the paper [26] contains proofs of our other results, such as Theorem I.7, Theorem I.2, Proposition I.8 and Theorem I.9.

In Section IV we conclude with some potential future work.

II. Preliminaries

A. Notation

We use $\widetilde{O}(f)$ as shorthand for $O(f \cdot \mathrm{polylog}(f))$ throughout the paper. All logarithms are base-2. We use n to denote the number of input bits. We say a language $L \subseteq \{0,1\}^\star$ is $f(n)$-sparse if $|L_n| \le f(n)$, where $L_n = L \cap \{0,1\}^n$. We assume knowledge of basic complexity theory (see [27], [28]).

B. Other Refuter Notions

For some of our results, it will be useful to generalize the notion of a refuter to allow the production of a *list* of strings, such that at least one of them is a counterexample.

Definition II.1 (List-Refuters)**.** For a function $s\colon \mathbb{N} \to \mathbb{N}$, a language L and an algorithm A that fails to solve L, an s-size *$\mathcal{D}$-list-refuter* (where $\mathcal{D} \in \{\mathsf{P}, \mathsf{BPP}, \mathsf{ZPP}\}$) for L against A is a $\mathcal{D}$-algorithm B that, given input 1^n, prints a list of $s(n)$ strings $x_n^{(1)}, x_n^{(2)}, \dots, x_n^{(s(n))}$ of lengths $n^{\Omega(1)}$, such that for infinitely many n, the following hold:

1) If $\mathcal{D} = \mathsf{P}$ there is an $i \in [s(n)]$ for which $A(x_n^{(i)}) \ne L(x_n^{(i)})$.
2) If $\mathcal{D} = \mathsf{BPP}$, with constant probability there exists $i \in [s(n)]$ for which $A(x_n^{(i)}) \ne L(x_n^{(i)})$.
3) If $\mathcal{D} = \mathsf{ZPP}$, then either the algorithm outputs "fail" or there exists $i \in [s(n)]$ for which $A(x_n^{(i)}) \ne L(x_n^{(i)})$, and the latter event happens with constant probability.

Refuters for Non-Uniform Models: We can also define refuters for circuit families. For a circuit class $\mathcal{C}$, we use $D_{L,\mathcal{C}}$ to denote the family $\{D_{L,A}\}_{A\in\mathcal{C}}$. We say a deterministic oracle algorithm R solves the search problem family $D_{L,\mathcal{C}}$ infinitely often, if for every $\{C_n\}_{n\in\mathbb{N}} \in \mathcal{C}$, there are infinitely many integers n such that $(1^n, R(1^n)) \in D_{L,\{C_n\}_{n\in\mathbb{N}}}$. We use $R^{\{\mathsf{desc}(C_n)\}_{n\in\mathbb{N}}}$ to denote that the oracle algorithm R gets access to the descriptions of the circuit family $\{C_n\}$, instead of only black box query access to it.

[12]Note the argument above only finds a single counterexample; using a paddable PSPACE-complete language, one can adapt the above argument to find infinitely many counter examples, see the full version [26] for details.

We can similarly generalize the above to randomized or zero-error randomized algorithms in the natural way.

Definition II.2 (Refuters and Constructive Separations for Language L against Nonuniform Class $\mathcal{C}$)**.** For a language L, a $\mathcal{D}$ refuter R for L against circuit class $\mathcal{C}$ is a $\mathcal{D}$ algorithm solving $D_{L,\mathcal{C}}$ infinitely often. We also say that R gives a $\mathcal{D}$-constructive separation $L \notin \mathcal{C}$.

We can extend the above definitions to list-refuters by allowing the corresponding algorithm to output a (polynomial-size) candidate list instead of a single counterexample. And we say a P list-refuter R solves $D_{L,A}$ infinite often if for infinitely many n, there exists $a \in R(1^n)$ such that $(1^n, a) \in D_{L,A}$. One can also similarly define BPP or ZPP list-refuters.

Finally, for a list-refuter according to Definition II.2, we say it is an oblivious list-refuter, if it does not need access to $\{\mathsf{desc}(C_n)\}_{n\in\mathbb{N}}$.

C. Definitions of MCSP and time-bounded Kolmogorov complexity

The Minimum Circuit Size Problem (MCSP) [29] and t-time-bounded Kolmogorov complexity (K^t) are studied in (the full version of) this paper [26]. We recall their definitions.

Definition II.3 (MCSP)**.** Let $s\colon \mathbb{N} \to \mathbb{N}$ satisfy $s(m) \ge m-1$ for all m.

Problem: $\mathsf{MCSP}[s(m)]$.

Input: A function $f\colon \{0,1\}^m \to \{0,1\}$, presented as a truth table of $n = 2^m$ bits.

Decide: Does f have a (fan-in two) Boolean circuit C of size at most $s(m)$?

We will also consider search-MCSP, the search version of MCSP, in which the small circuit C must be output when it exists.

For a time bound $t\colon \mathbb{N} \to \mathbb{N}$, recall that the K^t complexity (t-time-bounded Kolmogorov complexity) of string x is the length of the shortest program which outputs x in at most $t(|x|)$ time.

Definition II.4 ($\mathsf{R}_{\mathsf{K}^\mathsf{t}}$)**.** Let $t\colon \mathbb{N} \to \mathbb{N}$.

Problem: $\mathsf{R}_{\mathsf{K}^\mathsf{t}}$.

Input: A string $x \in \{0,1\}^n$.

Decide: Does x have $\mathsf{K}^\mathsf{t}(x)$ complexity at least $n-1$?

D. Implications of Circuit Complexity Assumptions on Refuters

The following technical lemma shows that, assuming uniform classes have non-trivially smaller circuits, the output of a refuter may be assumed to have low circuit complexity. This basic fact will be useful for several proofs in the paper.

Lemma II.5. *Let $s\colon \mathbb{N} \to \mathbb{N}$ be an increasing function. The following hold:*

1) *Assuming* $\mathsf{E}^{\mathsf{NP}} \subset \mathsf{SIZE}[s(n)]$*, then for every* P^{NP} *algorithm* A *such that* $A(1^n)$ *outputs* n *bits, it holds that* $A(1^n)$ *has circuit complexity at most* $s(O(\log n))$.
2) *Assuming* $\mathsf{E} \subset \mathsf{SIZE}[s(n)]$*, then for every* P *algorithm* A *such that* $A(1^n)$ *outputs* n *bits, it holds that* $A(1^n)$ *has circuit complexity at most* $s(O(\log n))$.
3) *Assuming* $\mathsf{SPACE}[O(n)] \subset \mathsf{SIZE}[s(n)]$*, then for every LOGSPACE algorithm* A *such that* $A(1^n)$ *outputs* n *bits, it holds that* $A(1^n)$ *has circuit complexity at most* $s(O(\log n))$.

Proof: In the following we only prove the first item, the generalization to the other two items are straightforward.

Consider the following function $f_A(n, i)$, which takes two binary integers n and $i \in [n]$ as inputs, and output the i-th bit of the output of $A(1^n)$. The inputs to f_A can be encoded in $O(\log n)$ bits in a way that all inputs (n, i) with the same n has the same length.

Since A is in P^{NP}, we have $f_A \in \mathsf{E}^{\mathsf{NP}}$. By our assumption and fix the first part of the input to f_A as n, it follows that $A(1^n)$ has circuit complexity at most $s(O(\log n))$. ■

The following simple corollary of Lemma II.5 will also be useful.

Corollary II.6. *If* $\mathsf{E}^{\mathsf{NP}} \subset \mathsf{P}_{/\mathsf{poly}}$ *(*$\mathsf{E} \subset \mathsf{P}_{/\mathsf{poly}}$ *or* $\mathsf{SPACE}[O(n)] \subseteq \mathsf{P}_{/\mathsf{poly}}$*), then for every* P^{NP} *(*P *or LOGSPACE) algorithm* A *such that* $A(1^n)$ *outputs* n *bits, it holds that* $A(1^n)$ *has circuit complexity at most* $\mathrm{polylog}(n)$.

We also observe that $\mathsf{P} = \mathsf{NP}$ has strong consequences for polylogtime-uniform AC^0 circuits.

Lemma II.7. *The following hold:*

1) *Assuming* $\mathsf{P} = \mathsf{NP}$*, then for every polylogtime-uniform* AC^0 *algorithm* A *such that* $A(1^n)$ *outputs* n *bits, it holds that* $A(1^n)$ *has circuit size complexity at most* $\mathrm{polylog}(n)$.
2) *Assuming* $\mathsf{P} = \mathsf{PSPACE}$*, then for every polylogtime-uniform* NC^1 *algorithm* A *such that* $A(1^n)$ *outputs* n *bits, it holds that* $A(1^n)$ *has circuit size complexity at most* $\mathrm{polylog}(n)$.

Proof: Let B be a polylogtime-uniform algorithm that, on the integer n (in binary) and $O(\log n)$-bit additional input, reports gate and wire information for an AC^0 circuit A_n. Consider the function $f(n, i)$ which determines the i-th output bit of the circuit A_n on the input 1^n, given n and i in binary. The function f is a problem in PH: given input of length $m = O(\log n)$, by existentially and universally guessing and checking gate/wire information (and using the $\mathrm{polylog}(n)$-time algorithm B to verify the information), the A_n of $n^{O(1)}$ size can be evaluated in $\Sigma_d\mathsf{TIME}[m^k]$ for a constant d depending on the depth of A_n, and a constant k depending on the algorithm B. Since $\mathsf{P} = \mathsf{NP}$, f is computable in P, i.e., f is in time at most αm^α for some constant α depending on k, d, and the polynomial-time SAT algorithm. Therefore f has a circuit family of size at most m^c for some fixed c, where $m = c \log n$. Thus the output of such a family always has small circuits.

The same argument applies if we replace AC^0 by NC^1 and replace PH by PSPACE. ■

E. Other Related Work

As mentioned in the introduction, Kabanets [9] defined and studied refuters in the context of derandomization. A primary result from that paper is that it is possible to simulate one-sided error polynomial time (RP) in zero-error subexponential time (ZPSUBEXP) on all inputs produced by refuters (efficient time algorithms that take 1^n and output strings of length n).[13] In other words, nontrivial derandomization is indeed possible when we only consider the outputs of refuters: there is **no** constructive separation of $\mathsf{RP} \not\subset \mathsf{ZPSUBEXP}$. This result contrasts nicely with some of our own, which show that if we could prove (for example) $\mathsf{EXP} = \mathsf{ZPP}$ holds with respect to refuters, then $\mathsf{EXP} = \mathsf{ZPP}$ holds unconditionally. (Of course this is a contrapositive way of stating our results; we don't believe that $\mathsf{EXP} = \mathsf{ZPP}$ holds!) Kabanets' work effectively shows that if $\mathsf{RP} \not\subseteq \mathsf{ZPSUBEXP}$ implied a *constructive separation* of $\mathsf{RP} \not\subseteq \mathsf{ZPSUBEXP}$, then $\mathsf{RP} \subseteq \mathsf{ZPSUBEXP}$ holds unconditionally (because there is no constructive separation of RP from ZPSUBEXP).

Chen, Jin, and Williams [6] studied a notion of constructive proof they called *explicit obstructions*. Roughly speaking, an explicit obstruction against a circuit class $\mathcal{C}$ is a (deterministic) polynomial-time algorithm A outputting a list L_n of input/output pairs $\{(x_i, y_i)\}$ with distinct x_i, such that all circuits in $\mathcal{C}$ fail to be consistent on at least one input/output pair. Chen, Jin, and Williams show several "sharp threshold" results for explicit obstructions, demonstrating (for example) that explicit obstructions unconditionally exist for $n^{2-\varepsilon}$-size DeMorgan formulas, but if they existed for $n^{2+\varepsilon}$-size formulas then one could prove the breakthrough lower bound $\mathsf{EXP} \not\subset \mathsf{NC}^1$. In this work, we are considering a "uniform" version of this concept: instead of outputting a list of bad input/output pairs (that do not depend on the algorithm), here we only have to output one bad instance that depends on the algorithm given.

Another motivation for studying constructive proofs comes from proof complexity and bounded arithmetic. A circuit lower bound for a language $L \in \mathsf{P}$ can naturally be expressed by a Π_2 statement S_n that says: "For all circuits C of a certain type, there exists x of length n such that $C(x) \neq L(x)$". In systems of bounded arithmetic such as Cook's theory PV_1 [31] (formalizing poly-time reasoning) or Jeřábek's theory APC_1 [32] (formalizing probabilistic

[13]The exact statement involves an "infinitely-often" qualifier, which we omit here for simplicity. A version of the simulation that removes the restriction to refuters, with the addition of a small amount of advice, was given in [30].

poly-time reasoning), a proof of S_n for infinitely many n immediately implies a constructive separation. The reason is that these theories have efficient witnessing: any proof of a Π_2 statement $\forall x \exists y R(x, y)$ (for R that can be expressed purely with bounded quantifiers and poly-time concepts) in these theories constructs an efficiently computable function f such that $R(x, f(x))$ holds. Here the function f plays the role of the refuter in a constructive separation. Therefore, situations in which constructive separations are unlikely to exist may provide clues about whether complexity lower bounds could be independent of feasible theories. Conversely, the constructiveness of a separation is a precondition for the provability of that separation in these feasible theories.[14]

Hardness Magnification: Another related line of work is hardness magnification [20], [21], [35], [23]. This line of work shows how very minor-looking lower bounds actually hide the whole difficulty of P vs NP and related problems. However, one might say that those results simply illuminate large holes in our intuition: those minor-looking lower bounds are far more difficult to prove than previously believed. One has to be skeptical in considering hardness magnification as a viable lower bounds approach, because we really don't understand how difficult the "minor-looking" lower bounds actually are.

In this paper, in contrast, we are mainly focused on situations where we already *know* the lower bound holds (and can prove that in multiple ways), but we are striving to prove the known lower bound in a more constructive, algorithmic way. This sort of situation comes up routinely in applications of the probabilistic method, where an object we want can be constructed with randomness, but it is a major open problem to construct it deterministically. Our results indicate that there is a deep technical gap between the major complexity class separation problems, versus many lower bounds we know how to prove. The former type of lower bound problem automatically has constructive aspects built into it, while the latter type of lower bound requires a breakthrough in derandomization in order to be made constructive.

III. Constructive Separations for Streaming and Query Algorithms Imply Breakthrough Lower Bounds

Streaming lower bounds and query complexity lower bounds are often regarded as well-understood, and certain lower bounds against one-tape Turing machines have been known for 50 years. In this section we show that surprisingly, making these separations constructive would imply breakthrough separations such as $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{BPP}$ or even $\mathsf{P} \neq \mathsf{NP}$.

A. Making Most Streaming Lower Bounds Constructive Implies Breakthrough Separations

We show that if randomized streaming lower bounds for *any* language L in NP can be made constructive, even with a P^{NP} refuter, then $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{BPP}$.

Reminder of Theorem I.3. *Let $f(n) \geq \omega(1)$. For every language $L \in \mathsf{NP}$, a P^{NP}-constructive separation of L from uniform randomized streaming algorithms with $O(n \cdot (\log n)^{f(n)})$ time and $O(\log n)^{f(n)}$ space implies $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{BPP}$.*

Remark III.1. *Let $V(x, y)$ be a verifier for L, and assume that the witness length $|y|$ is at most $|x|$.[15] Then the randomized streaming algorithms considered in Theorem I.3 can be further assumed to solve the search-version of L with one-sided error in the following sense: (1) A is also required to output a witness y when it decides $x \in L$ (2) whenever A outputs a witness y, we have $V(x, y) = 1$.*

We need the following lemma for solving search-MCSP, which adapts an oracle algorithm from [21]. The original algorithm of [21] has two-sided error: that is, when $x \notin \mathsf{MCSP}[s(n)]$, there is a small probability that the algorithm outputs an incorrect circuit. We modify their approach with a carefully designed checking approach so that the algorithm has only one-sided error.

Lemma III.2 ([21, Theorem 1.2], adapted)**.** *Assuming $\mathsf{NP} \subseteq \mathsf{BPP}$, for a time-constructive $s\colon \mathbb{N} \to \mathbb{N}$, there is a randomized streaming algorithm for search-$\mathsf{MCSP}[s(n)]$ on n-bit instances with $O(n \cdot s(n)^c)$ time and $O(s(n)^c)$ space for a constant c such that the following holds.*

- *If the input $x \in \mathsf{MCSP}[s(n)]$, the algorithm outputs a circuit C of size at most s computing x with probability at least $1 - 1/n$.*
- *If the input $x \notin \mathsf{MCSP}[s(n)]$, the algorithm always outputs NO.*

Alternatively, if we assume $\mathsf{NP} = \mathsf{P}$ instead, the above randomized streaming algorithm can be made deterministic.

Proof: We first recall the $\Sigma_3\mathsf{P}$ problem Circuit-Min-Merge introduced in [21]; here, we will only consider the version with two given input circuits. In the following we identify the integer i from $[2^m]$ with the i-th string from $\{0,1\}^m$ (ordered lexicographically).

Note that since $\mathsf{NP} \subseteq \mathsf{BPP}$, it follows that Circuit-Min-Merge is also in BPP. We can without loss of

[14]We note, however, that these connections depend on the complexity classes being separated. A circuit lower bound for an NP problem does not have an obvious Π_2 formulation, so the efficient witnessing results mentioned above do not directly apply. More complicated witnessing theorems might still be relevant; we refer to [33] and the recent book on Proof Complexity by Krajíček [34] for a more detailed discussion of these matters.

[15]That is, $x \in L$ if and only if there exists $y \in \{0,1\}^*$ such that $|y| \leq |x|$ and $V(x, y) = 1$.

Algorithm 1. Circuit-Min-Merge

Circuit-Min-Merge$[s(n)]$

Input: Given two circuits C_1, C_2 on $m = \log n$ input bits and three integers $\alpha < \beta \leq \gamma \in [2^m]$.
Output: The lexicographically first circuit C' such that for all $\alpha \leq z \leq \beta - 1$, $C'(z) = C_1(z)$, and for all $\beta \leq z \leq \gamma$, $C'(z) = C_2(z)$. If there are no such circuits, it outputs an all-zero string.

generality assume we have a BPP algorithm for it with error at most $1/n^3$.

After processing the first $p \in [2^m]$ bits of the input x, our streaming algorithm maintains a list of at most m circuits. Specifically, let $p = \sum_{k=0}^{m} a_k \cdot 2^k$ be the binary representation of p, for each $k \in [m]$. We maintain a circuit C_k that is intended to satisfy $C_k(z) = x_z$ for all $\sum_{\ell>k} a_\ell \cdot 2^\ell < z \leq \sum_{\ell \geq k} a_\ell \cdot 2^\ell$. Note that when $a_k = 0$, there is indeed no requirement on the circuit C_k and we can simply set it to a trivial circuit.

Now, suppose we get the $p+1$ bit of the input x. We update the circuit list via the following algorithm.

- We initialize D to be the linear-size circuit which outputs x_{p+1} on the input $p+1$, and outputs 0 on all other inputs.
- For k from 0 to m:
  - If $a_k = 1$, we set $D = \text{Circuit-Min-Merge}(C_k, D, \alpha, \beta, \gamma)$ with suitable α, β, γ, and set $a_k = 0$ and C_k to be a trivial circuit. We next check whether D is indeed the correct output of $\text{Circuit-Min-Merge}(C_k, D, \alpha, \beta, \gamma)$ by going through all inputs in $[\alpha, \gamma]$. We output NO and halt the algorithm immediately if we found D is not the correct output (if $\text{Circuit-Min-Merge}(C_k, D, \alpha, \beta, \gamma)$ outputs the all-zero string, we also output NO and halt the algorithm).
  - If $a_k = 0$, we set $C_k = D$, and halt the update procedure.

After we have processed the 2^m-bit of x, we simply output C_n. If $x \in \mathsf{MCSP}[s(n)]$, then by a simple union bound, with probability at least $1 - 1/n$, all calls to our BPP algorithm for Circuit-Min-Merge are answered correctly. In this case C_n is a correct algorithm computing the input x. If $x \notin \mathsf{MCSP}[s(n)]$, since we have indeed checked the output of all Circuit-Min-Merge calls, our algorithm will only output the circuit C_n if it is indeed of size at most $s(n)$ and computes x exactly. Since $x \notin \mathsf{MCSP}[s(n)]$ implies there is no such circuit C_n, our algorithm always outputs NO in this case.

For the running time, note that the above algorithm calls Circuit-Min-Merge at most $n \cdot \log n$ times on input of length $\widetilde{O}(s(n))$. Therefore calling Circuit-Min-Merge only takes $n \cdot \operatorname{poly}(s(n))$ time in total. Note that merging C_k and D takes $2^k \cdot \operatorname{poly}(s(n))$ time to verify the resulting circuit, but this only happens at most $n/2^k$ times. So the entire algorithm runs in $n \cdot \operatorname{poly}(s(n))$ time and $\operatorname{poly}(s(n))$ space as stated. ∎

Now we are ready to prove Theorem I.3.

Proof of Theorem I.3: The idea is to show that if $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{BPP}$ then we can construct a randomized streaming algorithm for $L \in \mathsf{NP}$ that "fools" all possible P^{NP} refuters. Interestingly, the assumption is used in three different ways: (1) to bound the circuit complexity of the outputs of P^{NP} algorithms, (2) to obtain a randomized streaming algorithm that finds a small circuit encoding the input, and (3) to get an efficient algorithm to find a small circuit encoding a correct witness when it exists.

Let $L \in \mathsf{NP}$, and $V(x, y)$ be a polynomial-time verifier for L. Assuming $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{BPP}$, we are going to construct a randomized streaming algorithm A, such that it solves L correctly on all possible instances which can be generated by a P^{NP} refuter.

Let B be an arbitrary P^{NP} refuter. First, by Corollary II.6, $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{BPP} \subset \mathsf{P}_{/\mathsf{poly}}$ implies that for all $n \in \mathbb{N}$, $B(1^n)$ has a circuit complexity of $w(n) = \operatorname{polylog}(n)$.

Second, note that $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{BPP}$ also implies that $\mathsf{NP} \subseteq \mathsf{BPP}$. Let $f(n) \geq \omega(1)$ and $s(n) = (\log n)^{f(n)/c_1}$ for a sufficiently large constant $c_1 > 1$. By Lemma III.2, we have a one-sided error randomized streaming algorithm A_{MCSP} for search-MCSP$[s(n)]$ with running time $n \cdot s(n)^{O(1)}$ and space $s(n)^{O(1)}$. Since $w(n) \leq s(n)$, we apply A_{MCSP} to find an $s(n)$-size circuit C encoding $B(1^n)$.

Now, we have an $s(n)$-size circuit encoding the n-bit input $B(1^n)$, and we wish to solve the Succinct-L problem[16] on this circuit. Note that Succinct-L is a problem in NEXP.

$\mathsf{EXP}^{\mathsf{NP}} = \mathsf{BPP}$ implies $\mathsf{NEXP} \subset \mathsf{P}_{/\mathsf{poly}}$, so every Succinct-$L$ instance has a succinct witness with respect to the verifier V: this follows from the easy witness lemma of [36]. Formally, there exists a universal constant $k \in \mathbb{N}$ such that, for every $s(n)$-size circuit D such that $\mathsf{tt}(D) \in L$, there exists an $s(n)^k$-size circuit E such that $V(\mathsf{tt}(D), \mathsf{tt}(E)) = 1$.

We consider the following problem:

> Given an $s(n)$-size circuit D with truth-table length n and an integer $i \in [\log(s(n)^k)]$, exhaustively try all circuits of size at most $s(n)^k$, find the first circuit E such that $V(\mathsf{tt}(D), \mathsf{tt}(E)) = 1$, and output the i-th bit of the description of E.

Note that the above algorithm runs in $2^{\operatorname{poly}(s(n))}$-time on $\operatorname{poly}(s(n))$-bit inputs, hence it is in EXP. Since $\mathsf{EXP} = \mathsf{BPP}$, this problem is also in BPP. Therefore there is a BPP algorithm which, given a Succinct-L instance D of size $s(n)$, outputs a description of a canonical circuit of size $s(n)^k$

[16]Here, we define "Succinct-L" to be: given a circuit C with ℓ input bits, decide whether $\mathsf{tt}(C) \in L$, where $\mathsf{tt}(C)$ is the truth table of C.

which encodes a witness for input $\mathsf{tt}(D)$ with respect to verifier V.

Thus we obtain a randomized algorithm for L on all instances with $s(n)$-size circuits. When the witness for x has length at most $|x| = n$, the algorithm can take $n \cdot \mathrm{poly}(s(n))$ time to output the found witness by outputting the truth-table of the circuit encoding the witness.

Setting c_1 to be large enough and putting everything together, we get the desired randomized streaming algorithm which solves all instances generated by P^{NP} refuters, which is a contradiction to our assumption. Therefore, it follows that $\mathsf{EXP}^{\mathsf{NP}} \neq \mathsf{BPP}$. ■

IV. Conclusion

Many interesting questions remain for future work. While we have given many examples of complexity separations that can automatically be made constructive, it is unclear how to extend our results to separations with complexity classes within P. For example, let L be a P-complete language. If L is not in uniform NC^1, does a P-constructive separation of L from uniform NC^1 follow? How about separations of P from $\mathsf{LOGSPACE}$? Would establishing constructive separations in these lower complexity classes have any interesting consequences?

Note that there is no P-constructive separation of $\mathsf{MCSP}[s] \notin \mathsf{P}$ for super-polynomially large s, unless EXP requires super-polynomial size Boolean circuits. (A polynomial-time refuter for the trivial algorithm that always accepts, must print a hard function!) But do any interesting consequences follow from a constructive separation of *search versions* of MCSP from P? The same proof strategy (of applying the conjectured refuter for the trivial algorithm that always accepts) does not make sense in this case, as the only hard instances for search problems are YES instances.

It would also be interesting to examine which proof methods for circuit lower bounds can be made constructive. We list a few examples which should be particularly interesting:

(1) the $\widetilde{\Omega}(n^3)$ size lower bound against DeMorgan formulas for Andreev's function [37], [38],
(2) the $\widetilde{\Omega}(n^2)$ size lower bound against formulas for Element-Distinctness [39],
(3) $\mathsf{AC}^0[p]$ size-depth lower bounds via the approximation method [40], [41].

Chen, Jin, and Williams [6] showed that constructing corresponding explicit obstructions for (1) and (2) above would imply $\mathsf{EXP} \not\subset \mathsf{NC}^1$, but it is unclear whether one can get a P-constructive separation without implying a major breakthrough lower bound.

We remark that as shown in [6], most lower bounds proved by random restrictions *can* be made constructive, by constructing an appropriate pseudorandom restriction generator. [6] explicitly constructed an oblivious list-refuter for parity against subquadratic-size formulas, and we remark that a similar oblivious list-refuter for parity against polynomial-size AC^0 circuits follows from the pseudorandom restriction generator for AC^0 of [42].

Finally, it would be interesting to consider constructive separations against *non-uniform* algorithms. Should we expect a proof of $\mathsf{NP} \not\subset \mathsf{P}/\mathrm{poly}$ or $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$ to imply a refuter of some kind? In such a setting, one would presumably need to feed the code of the non-uniform algorithm to the polynomial-time algorithm as part of its input (the algorithm should get the non-uniform code as advice, one way or another).

Acknowledgment

The authors from MIT were partially supported by NSF CCF-2127597 and CCF-1909429. L.C. was partially supported by an IBM PhD Fellowship. Part of this work was completed while R.W. was visiting the Simons Institute for the Theory of Computing, participating in the *Theoretical Foundations of Computer Systems* and *Satisfiability: Theory, Practice, and Beyond* programs.

References

[1] L. Levin, "Universal sequential search problems," *Problems of Information Transmission*, vol. 9, no. 3, pp. 265–266, 1973.

[2] K. Mulmuley, "Explicit proofs and the flip," *CoRR*, vol. abs/1009.0246, 2010. [Online]. Available: http://arxiv.org/abs/1009.0246

[3] ——, "Geometric complexity theory VI: the flip via saturated and positive integer programming in representation theory and algebraic geometry," *CoRR*, vol. abs/0704.0229, 2007. [Online]. Available: http://arxiv.org/abs/0704.0229

[4] ——, "The GCT program toward the *P* vs. *NP* problem," *Commun. ACM*, vol. 55, no. 6, pp. 98–107, 2012. [Online]. Available: https://doi.org/10.1145/2184319.2184341

[5] C. Ikenmeyer and U. Kandasamy, "Implementing geometric complexity theory: on the separation of orbit closures via symmetries," in *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*. ACM, 2020, pp. 713–726. [Online]. Available: https://doi.org/10.1145/3357713.3384257

[6] L. Chen, C. Jin, and R. R. Williams, "Sharp threshold results for computational complexity," in *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*. ACM, 2020, pp. 1335–1348.

[7] L. Chen, X. Lyu, and R. R. Williams, "Almost-everywhere circuit lower bounds from non-trivial derandomization," in *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*. IEEE, 2020, pp. 1–12. [Online]. Available: https://doi.org/10.1109/FOCS46700.2020.00009

[8] L. Fortnow and R. Santhanam, "New non-uniform lower bounds for uniform classes," in *31st Conference on Computational Complexity (CCC 2016)*, 2016.

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

[10] T. P. Baker, J. Gill, and R. Solovay, "Relativizations of the P =?NP question," *SIAM J. Comput.*, vol. 4, no. 4, pp. 431–442, 1975. [Online]. Available: https://doi.org/10.1137/0204037

[11] S. Aaronson and A. Wigderson, "Algebrization: A new barrier in complexity theory," *ACM Trans. Comput. Theory*, vol. 1, no. 1, pp. 2:1–2:54, 2009. [Online]. Available: https://doi.org/10.1145/1490270.1490272

[12] A. A. Razborov and S. Rudich, "Natural proofs," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 24–35, 1997. [Online]. Available: https://doi.org/10.1006/jcss.1997.1494

[13] D. Gutfreund, R. Shaltiel, and A. Ta-Shma, "If NP languages are hard on the worst-case, then it is easy to find their hard instances," *Computational Complexity*, vol. 16, no. 4, pp. 412–441, 2007. [Online]. Available: https://doi.org/10.1007/s00037-007-0235-8

[14] S. Dolev, N. Fandina, and D. Gutfreund, "Succinct permanent is *NEXP*-hard with many hard instances," in *Algorithms and Complexity, 8th International Conference, CIAC 2013. Proceedings*, ser. Lecture Notes in Computer Science, vol. 7878. Springer, 2013, pp. 183–196. [Online]. Available: https://doi.org/10.1007/978-3-642-38233-8_16

[15] B. Kalyanasundaram and G. Schnitger, "The probabilistic communication complexity of set intersection," *SIAM J. Discrete Math.*, vol. 5, no. 4, pp. 545–557, 1992. [Online]. Available: https://doi.org/10.1137/0405044

[16] A. A. Razborov, "On the distributional complexity of disjointness," *Theor. Comput. Sci.*, vol. 106, no. 2, pp. 385–390, 1992. [Online]. Available: https://doi.org/10.1016/0304-3975(92)90260-M

[17] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar, "An information statistics approach to data stream and communication complexity," *J. Comput. Syst. Sci.*, vol. 68, no. 4, pp. 702–732, 2004. [Online]. Available: https://doi.org/10.1016/j.jcss.2003.11.006

[18] W. Maass, "Quadratic lower bounds for deterministic and nondeterministic one-tape turing machines (extended abstract)," in *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*. ACM, 1984, pp. 401–408. [Online]. Available: https://doi.org/10.1145/800057.808706

[19] J. Brody and E. Verbin, "The coin problem and pseudorandomness for branching programs," in *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010*, 2010, pp. 30–39. [Online]. Available: https://doi.org/10.1109/FOCS.2010.10

[20] I. C. Oliveira and R. Santhanam, "Hardness magnification for natural problems," in *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, 2018, pp. 65–76. [Online]. Available: https://doi.org/10.1109/FOCS.2018.00016

[21] D. M. McKay, C. D. Murray, and R. R. Williams, "Weak lower bounds on resource-bounded compression imply strong separations of complexity classes," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*. ACM, 2019, pp. 1215–1225. [Online]. Available: https://doi.org/10.1145/3313276.3316396

[22] L. Chen, C. Jin, and R. R. Williams, "Hardness magnification for all sparse NP languages," in *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, 2019, pp. 1240–1255. [Online]. Available: https://doi.org/10.1109/FOCS.2019.00077

[23] L. Chen, S. Hirahara, I. C. Oliveira, J. Pich, N. Rajgopal, and R. Santhanam, "Beyond natural proofs: Hardness magnification and locality," in *11th Innovations in Theoretical Computer Science Conference, ITCS*, 2020, pp. 70:1–70:48. [Online]. Available: https://doi.org/10.4230/LIPIcs.ITCS.2020.70

[24] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, and D. Ronneburger, "Power from random strings," *SIAM J. Comput.*, vol. 35, no. 6, pp. 1467–1493, 2006. Preliminary version in FOCS'02.

[25] S. 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*. ACM, 2020, pp. 1038–1051. [Online]. Available: https://doi.org/10.1145/3357713.3384251

[26] L. Chen, C. Jin, R. Santhanam, and R. R. Williams, "Constructive separations and their consequences," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. TR21-159, 2021. [Online]. Available: https://eccc.weizmann.ac.il/report/2021/159/

[27] S. Arora and B. Barak, *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

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

[29] V. Kabanets and J. Cai, "Circuit minimization problem," in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, 2000, pp. 73–79. [Online]. Available: https://doi.org/10.1145/335305.335314

[30] R. R. Williams, "Natural proofs versus derandomization," *SIAM J. Comput.*, vol. 45, no. 2, pp. 497–529, 2016. [Online]. Available: https://doi.org/10.1137/130938219

[31] S. A. Cook, "Feasibly constructive proofs and the propositional calculus (preliminary version)," in *Proceedings of the 7th Annual ACM Symposium on Theory of Computing, 1975*. ACM, 1975, pp. 83–97.

[32] E. Jeřábek, "Approximate counting in bounded arithmetic," *J. Symb. Log.*, vol. 72, no. 3, pp. 959–993, 2007.

[33] M. Müller and J. Pich, "Feasibly constructive proofs of succinct weak circuit lower bounds," *Ann. Pure Appl. Log.*, vol. 171, no. 2, 2020. [Online]. Available: https://doi.org/10.1016/j.apal.2019.102735

[34] J. Krajíček, *Proof complexity*. Cambridge University Press, 2019, vol. 170.

[35] I. C. Oliveira, J. Pich, and R. Santhanam, "Hardness magnification near state-of-the-art lower bounds," in *34th Computational Complexity Conference, CCC 2019*, 2019, pp. 27:1–27:29. [Online]. Available: https://doi.org/10.4230/LIPIcs.CCC.2019.27

[36] R. Impagliazzo, V. Kabanets, and A. Wigderson, "In search of an easy witness: exponential time vs. probabilistic polynomial time," *J. Comput. Syst. Sci.*, vol. 65, no. 4, pp. 672–694, 2002. [Online]. Available: https://doi.org/10.1016/S0022-0000(02)00024-7

[37] J. Håstad, "The shrinkage exponent of de Morgan formulas is 2," *SIAM J. Comput.*, vol. 27, no. 1, pp. 48–64, 1998. [Online]. Available: https://doi.org/10.1137/S0097539794261556

[38] A. Tal, "Shrinkage of de morgan formulae by spectral techniques," in *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, 2014, pp. 551–560. [Online]. Available: https://doi.org/10.1109/FOCS.2014.65

[39] E. Neciporuk, "On a boolean function," *Doklady of the Academy of the USSR*, vol. 169, no. 4, pp. 765–766, 1966.

[40] A. A. Razborov, "Lower bounds on the size of bounded depth circuits over a complete basis with logical addition," *Mathematical Notes of the Academy of Sciences of the USSR*, vol. 41, no. 4, pp. 333–338, 1987.

[41] R. Smolensky, "Algebraic methods in the theory of lower bounds for boolean circuit complexity," in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987*, 1987, pp. 77–82. [Online]. Available: https://doi.org/10.1145/28395.28404

[42] O. Goldreich and A. Wigderson, "On derandomizing algorithms that err extremely rarely," in *Symposium on Theory of Computing, STOC 2014*. ACM, 2014, pp. 109–118. [Online]. Available: https://doi.org/10.1145/2591796.2591808