Consequences of Contructive Separations

A presentation on work by Chen, Jin, Santhanam, and Williams

2025-12-18

William He, Mustafa Motiwala

Contents

1 Introduction 3
1.a What is a constructive lower bound? 4
1.b When do constructive lower bounds exist? 5
2 Many constructive separations imply breakthrough lowerbounds 6
2.a Minimum Circuit Size Problem (MCSP) 7
2.b One-Tape Turing Machines 11
2.c Streaming Algorithms 15
2.d Query Algorithms 16
3 Some separations are impossible to constructivize 17
3.a Time bounded Kolmogorov complexity 18
3.b $\mathsf{R}_{\mathsf{K}^t}$ has no P-refuter 19
4 Some separations automatically constructivize 20
4.a Any proof of $\mathsf{P} \neq \mathsf{NP}$ constructivizes 21
5 Conclusion 25
5.a Thank you! 26
5.b References 27

1 Introduction

What is a constructive lower bound?

Let $f : \{0,1\}^\star \to \{0,1\}$ and $\mathcal{A}$ a class of algorithms.

A lower bound "$f \notin \mathcal{A}$" is a claim of the form

$$(\forall A \in \mathcal{A})(\exists\infty\ n)(\exists x_A \in \{0,1\}^n)(A(x_A) \neq f(x_A))$$

We are interested in *constructivizing* lower bounds; i.e, finding algorithms that can compute the "bad inputs" x_A given access to the algorithm A.

P-constructive separations

Let $f : \{0,1\}^\star \to \{0,1\}$ and $\mathcal{C}$ some complexity class. We say there is a P-constructive separation of $f \notin \mathcal{C}$ if for every algorithm A computable in $\mathcal{C}$, there is a "refuter" algorithm $R \in \mathsf{P}$ which on input 1^n outputs a string in $\{0,1\}^n$, such that for infinitely many n, we have $A(R(1^n)) \neq f(R(1^n))$.

By replacing P to other classes $\mathcal{D}$, we obtain the notion of $\mathcal{D}$-constructive separations.

When do constructive lower bounds exist?

One might expect that constructivizing lower bounds is possible only when the claimed lower bound is "easy to prove". It turns out that this intuition is misinformed:

1. **There are (many) known separations whose constructivizations would imply breakthrough lower bounds.** We will present examples of problems for which certain lower bounds are known but constructivizations of said lower bounds would imply results like $\mathsf{P} \neq \mathsf{NP}$.

2. **Some known separations cannot be made constructive.** For superpolynomial t, the set of t-time incompressible strings $\mathsf{R}_{\mathsf{K}^t}$ is known to not be in P, but we show that there is no P-constructive separation that witnesses this.

3. **Many conjectured separations automatically constructivize.** We show that, for example, any proof that $\mathsf{P} \neq \mathsf{NP}$ automatically yields a P-constructive separation. Results of this type are not original to [1] indeed, the result that proofs of $\mathsf{P} \neq \mathsf{NP}$ automatically constructivize is due to Gutfreund, Shaltiel, and Ta Shma in [3].

2 Many constructive separations imply breakthrough lowerbounds

Minimum Circuit Size Problem (MCSP)

The Minimum Circuit Size Problem

Let $s : \mathbb{N} \to \mathbb{N}$ satisfy $s(n) \geq n - 1$ for all n. Then $\mathsf{MCSP}[s(n)]$ is the following problem:

Input: A function $f : \{0,1\}^n \to \{0,1\}$, represented as a truth table with $N = 2^n$ bits.

Output:

- **Decision version:** whether f has a (fan-in two) Boolean circuit C of size at most $s(n)$.
- **Search version:** also output the circuit C when it exists.

Definition (polylogtime-uniform-$\mathsf{AC}^0[f(n)]$-refuter). A *polylogtime-uniform-*$\mathsf{AC}^0[f(n)]$*-refuter* is a refuter that is represented by a circuit family that is

- Polylogtime-uniform (uniformly constructible in polylogtime)
- AC^0 (constant depth)
- In size $f(n)$

Minimum Circuit Size Problem (MCSP) (ii)

Theorem. ([1], 1.7) *Let $s(n) \geq n^{\log(n)^{\omega(1)}}$ be any time-constructive super-quasipolynomial function. If there exists a polylogtime-uniform* $\mathsf{AC}^0[\text{quasipoly}]$ *refuter for* $\mathsf{MCSP}[s(n)]$ *against every polylogtime-uniform-*AC^0 *algorithm, then* $\mathsf{P} \neq \mathsf{NP}$*.*

It will be more convenient to state the bounds in terms of $N = 2^n$, so let $f(N) = s(n)$. As the constant-one function is a polylogtime-uniform-AC^0 algorithm, it will suffice to prove the following.

Theorem. *Let $f(N) \geq 2^{\log(\log(N))^{\omega(1)}}$ be be some* $\text{poly}(f(N))$*-function. If there exists a polylogtime-uniform* $\mathsf{AC}^0[\text{quasipoly}]$ *refuter for* $\mathsf{MCSP}[f(N)]$ *against the constant* 1 *function, then* $\mathsf{P} \neq \mathsf{NP}$*.*

Minimum Circuit Size Problem (MCSP) (iii)

Theorem.

Let $f(N) \geq 2^{\log(\log(N))^{\omega(1)}}$ *be some* $\mathrm{poly}(f(N))$*-function. If there exists a polylogtime-uniform* $\mathsf{AC}^0[\mathrm{quasipoly}]$ *refuter for* $\mathsf{MCSP}[f(N)]$ *against the constant* 1 *function, then* $\mathsf{P} \neq \mathsf{NP}$.

Proof. Suppose $\mathsf{P} = \mathsf{NP}$ and that there is a polylogtime-uniform-AC^0 refuter R for $\mathsf{MCSP}[f(N)]$ against the constant 1 function. The refuter R must infinitely often output a string x such that $x \notin \mathsf{MCSP}[f(N)]$.

We claim that the output of R must have circuit complexity $\mathrm{poly}\log(N)$. Indeed, for each N, the behaviour of the circuit $R_N : \{0,1\}^N \to \{0,1\}^N$ can be encoded by a formula with finitely many quantifiers, so the function $h(N, i)$ which receives N, i in binary and outputs the ith output bit of R_N on 1^N is in $\mathsf{PH} = \mathsf{P}$ (as the family is polylogtime-uniform).

We claim that the output of R must have circuit complexity $\mathrm{poly}\log(N)$. Indeed, consider the function $h(N, i)$ which receives N, i in binary and outputs the ith output bit of R_N on 1^N can be encoded by a formula with finitely many quantifiers, so it is in $\mathsf{PH} = \mathsf{P}$ (assuming $\mathsf{P} = \mathsf{NP}$) with respect to the input length (which is in $O(\log(N))$).

Minimum Circuit Size Problem (MCSP) (iv)

So, the refuter R outputs a string x of circuit complexity $\text{poly}\log(N) \in 2^{\log(\log(N))^{O(1)}} \leq 2^{\log(\log(N))^{\omega(1)}} \leq f(N)$, showing that $x \in \mathsf{MCSP}[f(N)]$, a contradiction. ∎

One-Tape Turing Machines

Definition. A 3-SAT formula family $\{C_n\}_{n\in\mathbb{N}}$, each with $S(n)$ number of clauses, is strongly explicit if there is an algorithm A such that $A(n, \cdot\, i)$ outputs the i-th clause of C_n in $\operatorname{poly}\log(S(n))$ time.

Lemma 3.5. ([2], [5]) Let M be a $T(n)$-time nondeterministic RAM. There exists a strongly explicit family of 3-SAT formulas $\{C_n\}_{n\in\mathbb{N}}$ of $T \cdot \operatorname{poly}\log(T)$ size, such that for every $x \in \{0,1\}^n$, $M(x) = 1$ if and only if there exists y such that $C_n(x, y) = 1$.

Theorem. ([1], 1.5) For every language L computable by a nondeterministic $n^{1+o(1)}$-time RAM, if there is a P^{NP}-constructive separation of L from nondeterministic $O(n^{1.1})$-time one-tape Turing machines, then $\mathsf{E}^{\mathsf{NP}} \not\subset \mathsf{SIZE}[2^{\delta n}]$ for some constant $\delta > 0$.

Here, E is the class of languages decidable in deterministic $2^{O(n)}$ time.

One-Tape Turing Machines (ii)

Theorem.

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

Proof. Suppose for a contradiction that $E^{\mathsf{NP}} \subset \mathsf{SIZE}\left[2^{\delta n}\right]$ for all $\delta > 0$. Consider an language L computable by a nondeterministic $n^{1+o(1)}$-time RAM denoted as M_{RAM}.

By Lemma 3.5, we can obtain a strongly explicit family of 3-SAT formulas $\{C_n\}_{\{n \in \mathbb{N}\}}$ with $n^{1+o(1)} \cdot \text{poly}\log\left(n^{1+o(1)}\right) = n^{1+o(1)}$ size and $s = n^{1+o(1)}$ variables.

One-Tape Turing Machines (iii)

We will construct a nondeterministic $O(n^{1.1})$-time one-tape Turing machine for L. Consider a nondeterministic one-tape Turing machine M_{δ_1}, for some $\delta_1 > 0$. On input x:

- M_{δ_1} guesses a circuit D of size n^{δ_1}
  - M_{δ_1} checks that $D(i) = x_i$ for all $i = 1, \ldots, |x|$ (this checks if D describes x)
- M_{δ_1} guesses a circuit E of size n^{δ_1}, and accepts if and only if

  $$D(1), \ldots, D(n), E(1), \ldots, E(s-n)$$

  satisfies C_n.

Time Complexity: Both operations can be done in $n^{1+O(\delta_1)}$ time. The first operation can be done by storing the n^{δ_1} size circuit close to the tapehead when moving from x_1 to x_n. The second operation is done by evaluating D, E on given values, and using these vaoues to evaluate C_n by enumerating all clauses in C_n. Hence, we can take δ_1 to be small enough so that M_{δ_1} is in $O(n^{1.1})$ time.

By assumption there is a P^{NP} refuter B for L against M_{δ_1}. We will show M solves $B(1^n)$ correctly.

One-Tape Turing Machines (iv)

$B(1^n)$ has circuit complexity n^{δ_1}. Indeed, we assumed that $\mathsf{E}^{\mathsf{NP}} \subset \mathsf{SIZE}\left[2^{\frac{\delta_1}{2}n}\right]$. Denote the function $f_R(n, i)$ which outputs the i-th bit of the n-bit string $B(1^n)$. Since $B \in \mathsf{P}^{\mathsf{NP}}$, we have $f_R \in \mathsf{E}^{\mathsf{NP}}$ as its input can be written in $2\log(n)$ bits. Hence, $f_R(n, i)$ has circuit complexity $2^{2^{\frac{\delta_1}{2}\log(n)}} = n^{\delta_1}$.

Since $B(1^n) \in L$, the lexicographically first string $y_n \in \{0,1\}^{s-n}$ such that $C_n(B(1^n), y_n) = 1$ has circuit complexity n^{δ_1}. By Lemma 3.5, M solves $B(1^n)$ correctly, a contradiction.

Streaming Algorithms

The Set-Disjointness problem (DISJ)

The DISJ problem is the problem of determining whether two subsets of $[n]$ are disjoint.

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

Output: $(x \cdot y) \bmod 2$

Theorem. ([1], 1.4) *Let $f(n) \geq \omega(1)$. A polylogtime uniform-AC^0-constructive separation of* DISJ *from randomized streaming algorithms with $O\big(n \cdot (\log(n))^{f(n)}\big)$ time and $O(\log(n))^{f(n)}$ space implies* $\mathsf{P} \neq \mathsf{NP}$.

Query Algorithms

The coin problem (PromiseMAJORITY)

For $0 < \varepsilon < \frac{1}{2}$, the $\mathsf{PromiseMAJORITY}_{n,\varepsilon}$ problem (also known as the coin problem) is the problem of determining in which direction a coin is biased given a sequence of n coin flips.

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

Output: Letting $p = \frac{1}{n} \sum x_i$, whether $p < 1/2 - \varepsilon$ or $p > 1/2 + \varepsilon$.

Theorem. ([1], 1.6) *Let ε be a function of n satisfying $\varepsilon \leq 1/(\log(n)^{\omega(1)}$, and $1/\varepsilon$ is a positive integer computable in $\mathrm{poly}(1/\varepsilon)$ time given n in binary. 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 $\mathsf{P} \neq \mathsf{NP}$.*

3 Some separations are impossible to constructivize

Time bounded Kolmogorov complexity

Fix $t : \mathbb{N} \to \mathbb{N}$ and assume $t(n) \geq n^{\omega(1)}$.

Kolmogorov complexity

Define $\mathsf{K}^t : \{0,1\}^\star \to \mathbb{N}$ by $\mathsf{K}^t(x)$ is the length of the shortest program which prints x in time $t(|x|)$.

Kolmogorov incompressible strings

Let $\mathsf{R}_{\mathsf{K}^t}$ denote the set of strings $x \in \{0,1\}^*$ such that $\mathsf{K}^t(x) \geq |x| - 1$.

Hirahara [4] showed in 2020 that $\mathsf{R}_{\mathsf{K}^t} \notin \mathsf{P}$. We show that there is no P-constructive separation of $\mathsf{R}_{\mathsf{K}^t}$ from any class which contains the constant-zero function. In other words, there is no polynomial time refuter which can fool even the function which always rejects.

$\mathsf{R}_{\mathsf{K}^t}$ has no P-refuter

Theorem. *There is no* P*-refuter for* $\mathsf{R}_{\mathsf{K}^t}$ *against the constant-zero algorithm.*

Proof. Suppose there were such a refuter R. Consider the machine M which takes n in binary and outputs $R(1^n)$. For infinitely many n, $R(1^n)$ outputs a string $y_n \in \mathsf{R}_{\mathsf{K}^t}$ of length n. For these n, we have $\langle M, n \rangle$ a program of size $O(\log n)$ which runs in time $\text{poly}(n) < t(|y_n|)$ and outputs y_n, contradicting $\mathsf{K}^t(y_n) \geq n - 1$. ■

This is an unconditionally hard problem with no constructive separations. In [1], it is shown that we can also find problems like these in $\mathsf{NP} \setminus \mathsf{P}$ under reasonable assumptions.

Theorem ([1], 1.9). *If* $\mathsf{NE} \neq \mathsf{E}$ $(\mathsf{NE} \neq \mathsf{RE})$, *then there is a language in* $\mathsf{NP} \setminus \mathsf{P}$ *with no* P*-refuter (*BPP*-refuter) against the constant one function.*

Here, E is the class of languages decidable in deterministic $2^{O(n)}$ time, NE is the corresponding non-deterministic class, and RE is the corresponding randomized class with one-sided error.

4 Some separations automatically constructivize

Any proof of $\mathsf{P} \neq \mathsf{NP}$ constructivizes

In this section, we show the following:

Theorem. *Suppose* $\mathsf{P} \neq \mathsf{NP}$. *Then, for any paddable* NP*-complete language* L*, there is a* P*-constructive separation of* $L \notin \mathsf{P}$.

Here, *paddable* means that any string x can be extended to any longer length in polynomial time while preserving membership in the language. Note, for example, that SAT is paddable.

The proof requires the existence of a *downwards self-reducible* NP-complete language.

Downwards self-reducible

We say a language $L \in \mathsf{NP}$ is downwards self-reducible if there is a polynomial time oracle algorithm D such that for all $x \in \{0,1\}^m$, we have $L(x) = D^{L_{\leq m-1}}(x)$.

Every NP-complete language is downwards self-reducible.

Any proof of $\mathsf{P} \neq \mathsf{NP}$ constructivizes (ii)

Fix $M \in \mathsf{NP}$ to be NP-complete and downwards self-reducible and let D be a polynomial time oracle algorithm such that $M(x) = D^{M_{\leq |x|-1}}(x)$ for all strings x.

Let L be NP-complete and paddable and A a polytime algorithm. By NP-completeness of L, fix a reduction p of M to L, so that $M(x) = L(p(x))$ for all x.

The polytime refuter for L against A

On input 1^n, we make a sequence of queries to A to construct the shortest string x^* with $|x^*| \leq n$ satisfying the property

$$A(p(x)) \neq D^{O_{|x|-1}}(x), \text{where } O_{|x|-1} = \{x' : |x'| < |x|, A(p(x')) = 1\} \qquad (\star)$$

Either A answers correctly on all queries, in which case we output $p(x^*)$, or A answers incorrectly in which case we output the queries on which A made a mistake.

We show that for sufficiently large n, our refuter outputs at least one string y for which $A(y) \neq L(y)$.

Any proof of $\mathsf{P} \neq \mathsf{NP}$ constructivizes (iii)

We first show that for sufficiently large n, there does exist a string x satisfying $(\star)$. Since M is NP-complete, $\mathsf{P} \neq \mathsf{NP}$, and $A \in \mathsf{P}$, we cannot have $A(p(x)) = M(x)$ for all x and so for large enough n, there is x' with $|x'| \leq n$ satisfying $A(p(x')) \neq M(x')$. If no x satisfies $(\star)$, then an induction on $m \leq n$ and the defining property of D shows $A(p(x)) = M(x)$ for all $|x| \leq n$, contradicting $A(p(x')) \neq M(x')$.

So, there is a string satisfying $(\star)$; we let x^* be the shortest such string. Minimality of x^* guarantees $D^{O_{|x^*|-1}}(x^*) = M(x^*) = L(p(x^*))$ so we have $A(p(x^*)) \neq L(p(x^*))$. Thus, if we can construct x^*, then $p(x^*)$ is the desired counterexample.

To construct x^*, notice that the decision problem $f(1^m, 1^n) =$ "$\exists x.\ |x| \leq m.n, x, m$ satisfy $(\star)$" is in NP. Thus, we can reduce the computation of f to an instance of L and then use A to answer that instance. The least m such that A says YES on $f(1^m, 1^n)$ but NO on $f(1^{m-1}, 1^n)$ is the length of x^*.

Similarly, the problem $f(y, 1^m, 1^n) =$ "$\exists x.y$ is a prefix of x and n, x, m satisfy $(\star)$" is in NP so by reducing to L, we can use A to answer this question and iteratively build up y one character at a time.

If A never answers incorrectly, then we can find x^* and return $p(x^*)$, otherwise we can return the instance on which A got the wrong answer. This completes the proof. ■

Any proof of $\mathsf{P} \neq \mathsf{NP}$ constructivizes (iv)

The theorem just presented is a specialization of

Theorem 5.5 (from [1]). Let $\mathcal{C} \in \{\mathsf{P}, \mathsf{BPP}, \mathsf{ZPP}\}$ and $\mathcal{D}$ a complexity class such that $\mathsf{NP} \subseteq \mathcal{D}$ and there is a $\mathcal{D}$-complete, downwards self-reducible M. If $\mathcal{D} \subseteq \mathcal{C}$, then, for any paddable $\mathcal{D}$-complete language L, there is a $\mathcal{C}$-constructive separation of $L \notin \mathcal{C}$.

The authors of [1] present various other theorems that demonstrate analogous results for $\mathcal{D} \in \{\mathsf{PSPACE}, \mathsf{EXP}, \mathsf{NEXP}\}$ and more.

5 Conclusion

Thank you!

A lower bound on a decision problem is *constructive* when for any algorithm claiming to solve the decision problem there is a refuter algorithm which can efficiently construct counterexamples. We have shown that the question of when lower bounds constructivize has some counterintuitive answers and that constructivization is a desirable property of lower bounds.

Thank you to Professor Roei Tell for an illuminating semester and thank you to our classmates for many enlightening discussions.

References

[1] Lijie Chen, Ce Jin, Rahul Santhanam, and Ryan Williams. 2024. Constructive Separations and Their Consequences. *TheoretiCS* (February 2024). https://doi.org/10.46298/theoretics.24.3

[2] Lance Fortnow, Richard Lipton, Dieter van Melkebeek, and Anastasios Viglas. 2005. Time-space lower bounds for satisfiability. *J. ACM* 52, 6 (November 2005), 835–865. https://doi.org/10.1145/1101821.1101822

[3] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. 2007. If NP Languages are Hard on the Worst-Case, Then it is Easy to Find Their Hard Instances. *computational complexity* 16, 4 (December 2007), 412–441. https://doi.org/10.1007/s00037-007-0235-8

[4] Shuichi Hirahara. 2020. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2020)*, 2020. Association for Computing Machinery, Chicago, IL, USA, 1038–1051. https://doi.org/10.1145/3357713.3384251

[5] Iannis Tourlakis. 2001. Time–Space Tradeoffs for SAT on Nonuniform Machines. *Journal of Computer and System Sciences* 63, 2 (2001), 268–287. https://doi.org/https://doi.org/10.1006/jcss.2001.1767