Electronic Colloquium on Computational Complexity, Report No. 49 (2024)

Oblivious Classes Revisited: Lower Bounds and Hierarchies

Karthik Gajulapalli* Zeyong Li† Ilya Volkovich‡

Abstract

In this work we study oblivious complexity classes. Among our results:

- For each $k \in \mathbb{N}$, we construct an explicit language $L_k \in \mathsf{O}_2\mathsf{P}$ that cannot be computed by circuits of size n^k.
- We prove a hierarchy theorem for $\mathsf{O}_2\mathsf{TIME}$. In particular, for any function $t : \mathbb{N} \to \mathbb{N}$ we show that: $\mathsf{O}_2\mathsf{TIME}[t(n)] \subsetneq \mathsf{O}_2\mathsf{TIME}[t(n)^4 \log^9(t(n))]$.
- We prove new structural results connecting $\mathsf{O}_2\mathsf{P}$ and $\mathsf{S}_2\mathsf{P}$.
- We make partial progress towards the resolution of an open question posed by Goldreich and Meir (TOCT 2015).
- We identify the first natural problem in $\mathsf{O}_2\mathsf{P}$, that is not expected to be in either P or even BPP.

To the best of our knowledge, these results constitute the first explicit fixed-polynomial lower bound, hierarchy theorem and hard natural problem for $\mathsf{O}_2\mathsf{P}$. The smallest uniform complexity class for which such lower bounds were previously known was $\mathsf{S}_2\mathsf{P}$ due to Cai (JCSS 2007). In addition, this is the first uniform hierarchy theorem for a semantic class. All previous results required some non-uniformity.

In order to obtain some of the results in the paper, we introduce the notion of *uniformly-sparse extensions* which could be of independent interest.

*Georgetown University. Email: kg816@georgetown.edu.

†National University of Singapore. Email: li.zeyong@u.nus.edu.

‡Boston College. Email: ilya.volkovich@bc.edu.

ISSN 1433-8092

1 Introduction

Proving circuit lower bounds has been one of the holy grails of theory of computation with strong connections to many fundamental questions in complexity theory. For instance, proving that there exists a function in E[1] that requires exponential-size circuits would entail a strong derandomization: $\mathsf{BPP} = \mathsf{P}$ and $\mathsf{MA} = \mathsf{NP}$ [NW94, IW97]. And yet, while by counting arguments (i.e. [Sha49]) the vast majority of Boolean functions/languages do require exponential-size circuits, the best 'explicit' lower bounds are still linear! (in fact the best known lower bound for any language in E^{NP} is just linear [LY22]). Indeed, although it is widely *believed* that NP requires super-polynomial-size circuits (i.e. $\mathsf{NP} \not\subseteq \mathsf{P}/\mathsf{poly}$) establishing the statement even for NEXP (i.e. $\mathsf{NEXP} \not\subseteq \mathsf{P}/\mathsf{poly}$), the exponential version of NP, has remained elusive for many years. The best known explicit lower bound is due to a seminal work of Williams [Wil14], where it was shown that NEXP requires super-polynomial-size circuits in a 'very' restricted model ($\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$).

In the high-end regime, Kannan [Kan82] has shown that the exponential hierarchy requires exponential-size circuits, via diagonalization[2]. More precisely, it was shown that the class $\Sigma_3\mathsf{E} \cap \Pi_3\mathsf{E}$ contains a language that cannot be computed by a circuit family of size $2^n/n$. This result was later improved to $\Delta_3\mathsf{E} = \mathsf{E}^{\Sigma_2\mathsf{P}}$ by Miltersen, Vinodchandran and Watanabe [MVW99]. Moreover, it was shown that $\Delta_3\mathsf{E}$ actually requires circuits of 'maximum possible' size. Subsequently, the status of the problem remained stagnant for more than two decades until very recently, Chen, Hirahara and Ren [CHR24] and a follow-up work by Li [Li24] improved the result to $\mathsf{S}_2\mathsf{E}$ [3]. In particular, this result was obtained via solving the Range Avoidance (Avoid) problem with 'single-valued, symmetric polynomial-time' algorithm. Indeed, the focus of our work is on 'oblivious' symmetric polynomial time and related complexity classes.

1.1 Background

1.1.1 Symmetric Time

Symmetric polynomial time, denoted by $\mathsf{S}_2\mathsf{P}$, was introduced independently by Canetti [Can96], and Russell and Sundaram [RS98]. Intuitively speaking, this class captures the interaction between an efficient (polynomial-time) verifier V and two all-powerful provers: the 'YES'-prover Y and the 'NO'-prover Z, exhibiting the following behaviour:

- If x is a yes-instance, then the 'YES'-prover Y can send an irrefutable proof/certificate y to V that will make V *accept*, **regardless** of the communication from Z.
- Likewise, if x is a no-instance, then the 'NO'-prover can send an irrefutable proof/certificate proof z to V that will make V *reject*, **regardless** of the communication from Y.

We stress that in both cases the irrefutable certificates can depend on x itself. One can also define $\mathsf{S}_2\mathsf{E}$ - the exponential version of $\mathsf{S}_2\mathsf{P}$, by allowing the verifier to run in linear-exponential time. For a formal definition see Definition 2.2. A seminal result of [Cai07] provides the best known upper bound $\mathsf{S}_2\mathsf{P} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$. At the same time, $\mathsf{S}_2\mathsf{P}$ appears to be a very powerful class as it contains MA and $\Delta_2\mathsf{P} = \mathsf{P}^{\mathsf{NP}}$.

[1] Deterministic time $2^{O(n)}$.
[2] In fact, this argument could be viewed as solving an instance of the *Range Avoidance* problem. See below.
[3] Symmetric exponential time. Indeed, $\mathsf{S}_2\mathsf{E} \subseteq \Sigma_2\mathsf{E} \cap \Pi_2\mathsf{E} \subseteq \Delta_3\mathsf{E}$. For a formal definition see Definition 2.2.

1.1.2 Oblivious Complexity Classes

The study of oblivious complexity classes was initiated in [CR06] and has subsequently received more attention [Aar07, FSW09, CR11, GM15]. Roughly speaking, let Λ be a complexity class such that in addition to the input x, the machines $M(x, w)$ of Λ also take a witness w (and possibly other inputs). Examples of such classes include: $\mathsf{NP}, \mathsf{MA}, \mathsf{S_2P}$, etc. The corresponding *oblivious* version of Λ is obtained by stipulating that the for every $n \in \mathbb{N}$ there exists a 'common' witness w_n for **all** the 'respective' inputs of length n. For instance, a language L belongs to ONP – the oblivious version of NP, if there exists a polynomial-time machine $M(x, w)$ such that:

1. $\forall n \in \mathbb{N}$ there exists w_n such that $\forall x \in \{0,1\}^n : x \in L \implies M(x, w_n) = 1$.
2. $x \notin L \implies \forall w : M(x, w) = 0$.

Thus, in a similar manner, one can define the class $\mathsf{O_2P}$ — the oblivious version of $\mathsf{S_2P}$, that is referred to as 'oblivious symmetric polynomial time' in the literature. $\mathsf{O_2P}$ has the additional requirement that for every $n \in \mathbb{N}$ there exist an irrefutable yes-certificate y^* and an irrefutable no-certificate z^* for all the yes-instances and the no-instances of length n, respectively. For a formal definition, see Definition 2.4.

It is immediate from the definitions that $\mathsf{ONP} \subseteq \mathsf{NP}$, $\mathsf{O_2P} \subseteq \mathsf{S_2P}$ and $\mathsf{ONP} \subseteq \mathsf{O_2P}$. On the other hand, by hard-codding the witnesses/certificates we get that $\mathsf{ONP} \subseteq \mathsf{O_2P} \subseteq \mathsf{P/poly}$. While the oblivious classes seem to be more restricted than their non-oblivious counterparts, proving any non-trivial upper bounds could still be challenging. In terms of lower bounds, the best known non-oblivious containment is $\mathsf{BPP} \subseteq \mathsf{O_2P}$. For more details and discussion see [CR06, GM15]. Nonetheless, to the best of our knowledge, no "natural" problem believed to lie outside BPP, but in either ONP or $\mathsf{O_2P}$ has been previously identified in the literature.

1.1.3 Sparsity

A language L is *sparse*, if for every input length $n \in \mathbb{N}$ the number of yes-instances of size n is at most $\text{poly}(n)$. We will denote the class of all sparse languages by SPARSE. Sparse languages have seen many applications in complexity theory. Perhaps, the most fundamental one is known as "Mahaney's theorem" [Mah82] that implies that a sparse language cannot be NP-hard, unless $\mathsf{P} = \mathsf{NP}$. In [FSW09] and [GM15], sparse languages were also studied in the context of oblivious complexly classes. In particular, it was shown that $\mathsf{NP} \cap \mathsf{SPARSE} \subseteq \mathsf{ONP}$. That is, every sparse NP language is also in ONP. The same argument also implies that $\mathsf{NE} = \mathsf{ONE}$, that is, **equality** between the exponential versions of NP and ONP, respectively. Given the former claim we observe that the *Grid Coloring* problem, defined in [AGL23], constitutes a natural ONP (and hence $\mathsf{O_2P}$) problem. For a formal statement, see Observation 1.

Subsequently, Goldreich and Meir [GM15] posed an open question whether a similar relation holds true for coNP and coONP. That is, whether every sparse coNP language is also in coONP[4]. Motivated by this question, we observe that essentially the same issues arise when one attempts to show that every sparse $\mathsf{S_2P}$ language is also in $\mathsf{O_2P}$. While we do not accomplish this task, we make a partial progress by introducing *uniformly-sparse extensions*. The intuition behind this definition is to have a uniform 'cover' of the segments of the yes-instances for **all** input lengths. For a formal definition see Definition 2.19. This is our main conceptual contribution. As a corollary, we obtain

[4] The original (equivalent) formulation of the question in [GM15] was w.r.t to NP and co-sparse languages.

that $\mathsf{S_2E} = \mathsf{O_2E}$. Although this might not be a new result, to the best of our knowledge, this result has not appeared in the literature previously.

1.1.4 Range Avoidance

The study of the Range Avoidance problem (Avoid) was initiated in [KKMP21]. The problem itself takes an input-expanding Boolean Circuit $\mathsf{C} : \{0,1\}^n \to \{0,1\}^{n+1}$ as input and asks to find an element y, outside the range of C. Since its introduction, there has been a steady line of exciting work studying the complexity and applications of Avoid [Kor22, GLW22, RSW22, CHLR23, GGNS23, ILW23, CHR24, Li24, CGL+23, CL23].

Informally, Avoid algorithmically captures the probabilistic method where the existence of an object with some property follows from a union bound. In particular, Korten [Kor22] showed that solving Avoid would result in finding optimal explicit constructions of many important combinatorial objects, including but not limited to Ramsey graphs [Rad21], rigid matrices [GLW22, GGNS23], pseudorandom generators [CT22], two-source extractors [CZ19, Li23], linear codes [GLW22], strings with maximum time-bounded Kolmogorov complexity (K^{poly}-random strings) [RSW22] and truth tables of high circuit complexity [Kor22].

The connection between Avoid and hard truth table makes it relevant to the study of circuit lower bounds. It has been observed and pointed out in many prior works (see, e.g. [CHR24]) that proving explicit circuit lower bounds is effectively finding single-valued[5] constructions of hard truth tables. Indeed, this is the framework adopted for proving circuit lower bound in [CHLR23, CHR24, Li24]: Designing a single-valued algorithm for solving Avoid.

1.1.5 Time Hierarchy Theorem

Time Hierarchy theorems are among the most fundamental results in computational complexity theory, which (loosely speaking) assert that computation with more time is strictly more powerful. Time hierarchy theorems are known for deterministic computation (DTIME) [HS65, HS66] and non-deterministic computation (NTIME) [Coo72, SFM78, Žák83] which are syntactic classes. The situation for semantic classes such as BPTIME is much more elusive as it is unclear how to enumerate and simulate all BPTIME machines while ensuring that the simulating machine itself remains a proper BPTIME machine. In fact, even verifying that a machine is a BPTIME machine is itself an undecidable problem. For BPTIME, a time hierarchy theorem is only known for its promise version, or when given one bit of advice [Bar02, FS04, FST05]. This was further generalized in [MP07], where they show most semantic classes (e.g. MA, $\mathsf{S_2TIME}$) admit a time hierarchy theorem with one bit of advice.

Along a different line of research, it was shown in [LOS21, DPWV22] that coming up with a pseudo-deterministic algorithm (single-valued randomized algorithms) for estimating the acceptance probability of a circuit would imply a uniform hierarchy theorem for BPTIME.

1.2 Previous Results

A parallel line of work focused on the 'low-end' regime by proving the so-called 'fixed-polynomial' circuit lower bounds. That is, the goal is to show that for every $k \in \mathbb{N}$ there is a language L_k (that

[5] Roughly speaking, a single-valued algorithm on successful executions should output a fixed (canonical) solution given the same input.

may depend on k) which cannot be computed by circuits of size n^k. The first result in this sequel — fixed-polynomial lower bounds for the polynomial hierarchy, was obtained by Kannan [Kan82] via diagonalization. In particular, it was shown that for every $k \in \mathbb{N}$ there exists a language $L_k \in \Sigma_4\mathsf{P}$ that cannot be computed by circuits of size n^k. This result was then improved to $\Sigma_2\mathsf{P}$. The key idea behind this and, in fact, the vast majority of subsequent improvements is a 'win-win' argument that relies on the *Karp-Lipton* collapse theorem [KL80]: if NP has polynomial-size circuits (i.e $\mathsf{NP} \subseteq \mathsf{P/poly}$) then the (whole) polynomial hierarchy collapses to $\Sigma_2\mathsf{P}$. More specifically, the argument proceeds by a two-pronged approach:

- Suppose $\mathsf{NP} \nsubseteq \mathsf{P/poly}$. Then the claim follows as $\mathsf{NP} \subseteq \Sigma_2\mathsf{P}$.
- On the other hand, suppose $\mathsf{NP} \subseteq \mathsf{P/poly}$. Then by Karp-Lipton: $\Sigma_4\mathsf{P} = \Sigma_2\mathsf{P}$ and in particular for all $k \in \mathbb{N} : L_k \in \Sigma_2\mathsf{P}$.

Indeed, by deepening the collapse, the result was further improved to $\mathsf{ZPP}^{\mathsf{NP}}$ [KW98, BCG+96], $\mathsf{P}^{\mathsf{prMA}}$ [CR11] and $\mathsf{S_2P}$ [Cai07]. By using different versions of the Karp-Lipton theorem, the result has also been extended to PP [Vin05, Aar06] and $\mathsf{MA}/1$ [San09].

Yet, despite the success of the 'win-win' argument, the obtained lower bounds are often non-explicit due to the non-constructiveness nature of the argument. Different results [CW04, San09] were required to exhibit explicit 'hard' languages in Σ_2, PP and $\mathsf{MA}/1$. Nonetheless, the last word has been said yet about $\mathsf{S_2P}$. For instance, we know that there is a language in $\mathsf{S_2P}$ that requires circuits of size, say, n^2. However, prior to our to result to the best of our knowledge, we could not prove any super-linear lower bound for **any** particular language in $\mathsf{S_2P}$. Another limitation of the 'win-win' argument stems from the fact that it only applies to complexity classes which (provably) contain NP. In particular, in [CR06] it was actually shown that if $\mathsf{NP} \subseteq \mathsf{P/poly}$ then the polynomial hierarchy collapses all the way to $\mathsf{O_2P}$! Unfortunately, this result does not immediately imply fixed-polynomial lower bounds for $\mathsf{O_2P}$[6] as it is unknown and, in fact, *unlikely* that $\mathsf{O_2P}$ contains NP. Furthermore, such a containment will be 'self-defeating'. Recall that $\mathsf{O_2P} \subseteq \mathsf{P/poly}$. Hence, if $\mathsf{NP} \subseteq \mathsf{O_2P}$ then $\mathsf{NP} \subseteq \mathsf{P/poly}$ which in and of itself already implies the collapse of the whole polynomial hierarchy!

Finally, it is important to mention a result of [FSW09] that for any $k \in \mathbb{N}$, NP has circuits of size n^k iff $\mathsf{ONP}/1$ does. In that sense ONP already nearly captures the hardness of showing fixed-polynomial lower bounds for NP.

1.3 Our Results

In our first result we extend the lower bounds for $\mathsf{S_2P}$ and $\mathsf{S_2E}$, to their weaker oblivious counterparts $\mathsf{O_2P}$ and $\mathsf{O_2E}$, respectively. This result follows the recent line of research that obtains circuit lower bounds by means of solving instances of the Range Avoidance problem [CHLR23, CHR24, Li24].

Theorem 1. *For all $k \in \mathbb{N}$, $\mathsf{O_2P} \nsubseteq \mathsf{SIZE}[n^k]$. Moreover, for each k there exists an explicit language $L_k \in \mathsf{O_2P}$ such that $L_k \notin \mathsf{SIZE}[n^k]$.*

In fact we prove a stronger parameterized version of this statement (see Theorem 3.2, Corollary 4.2, and Corollary 3.3). We now highlight three main reasons why such a result is fascinating:

[6] Indeed, the authors in [CR06] could only obtain fixed-polynomial lower bounds for $\mathsf{NP}^{\mathsf{O_2P}}$ which was later subsumed by the results of [San09].

1. Our lower bound does not follow the framework of "win-win" style Karp-Lipton collapses. As was mentioned above, since already $\mathsf{O_2P} \subseteq \mathsf{P/poly}$ the pre-requisite for proving the bound via the "win-win" argument will be self-defeating.

2. Our proof is constructive and for every $k \in \mathbb{N}$ we define an explicit language $L_k \in \mathsf{O_2P}$ for which $L_k \not\subseteq \mathsf{SIZE}[n^k]$.

3. $\mathsf{O_2P}$ becomes the smallest uniform complexity class known for which fixed-polynomial lower bounds are known. Moreover, after more than 15 years, this class coincides again with the deepest known collapse result of the Karp-Lipton Theorem[7].

Our second result gives a hierarchy theorem for $\mathsf{O_2TIME}$.

Theorem 2. *For any function $t : \mathbb{N} \to \mathbb{N}$ it holds that:* $\mathsf{O_2TIME}[t(n)] \subsetneq \mathsf{O_2TIME}[t(n)^4 \log^9(t(n))]$.

We remark, that to the best of our knowledge, this is the first known hierarchy theorem for a uniform semantic class (that contains BPTIME). At the same time, we observe that the proof of the non-deterministic time hierarchy theorem (NTIME) (see e.g. [Žák83]) actually extends to the *oblivious* non-deterministic time (ONTIME) since the hard language constructed in their proof is unary and hence is contained in ONTIME. On the other hand, that same language also diagonalizes against **all** NTIME machines which is a superset of all ONTIME machines.

In our time hierarchy theorem for $\mathsf{O_2TIME}$, which goes through a reduction to Avoid, one can view Avoid as a tool for diagonalization against all circuits of fixed size, which in turn contains all $\mathsf{O_2TIME}$ machines with bounded time complexity. This (together with the time hierarchy theorem for ONTIME) might suggest an approach for proving time hierarchy theorem for semantic classes in general: diagonalize against a syntactic class that encompasses the semantic class in consideration.

Finally, we introduce the notion of *uniformly-sparse extensions* (for a formal definition, see Definition 2.19) to get structural complexity results relating $\mathsf{O_2TIME}$ and $\mathsf{S_2TIME}$. This relation provides an alternate method of proving Theorem 1.

Theorem 3. *Let $L \in \mathsf{S_2P}$. If L has a* uniformly-sparse extension *then $L \in \mathsf{O_2P}$.*

While not much was known between the classes $\mathsf{O_2TIME}$ and $\mathsf{S_2TIME}$, except that $\mathsf{O_2TIME} \subseteq \mathsf{S_2TIME}$, we show new connections between the two classes. In fact, we prove a stronger parameterized version of Theorem 3 that yields as a corollary a proof of the equivalence $\mathsf{S_2E} = \mathsf{O_2E}$ (see Corollary 4.3). Going back to the original motivation, by repeating the same argument, we make a partial progress towards the resolution of the open question posed by Goldreich and Meir in [GM15]. See Lemma 4.1 for more details.

Theorem 4. *Let $L \in \mathsf{coNP}$. If L has a* uniformly-sparse extension *then $L \in \mathsf{coONP}$.*

Finally, we observe that the *Grid Coloring* problem, defined in [AGL23], constitutes a natural ONP (and hence $\mathsf{O_2P}$) problem. The problem emerges from the area of computational Ramsey Theory. To the best of our knowledge, this is the first natural problem identified in the literature.

Definition 1.1 (Grid Coloring[AGL23])**.**
$\mathsf{GC} = \{(1^n 0 1^m 0 1^c) \mid$ the $n \times m$ grid can be c-colored and not have any monochromatic squares.$\}$

[7] Indeed, in the universe of [Cai07] and [CR06] prior to our work, the smallest class has been $\mathsf{S_2P}$, while the deepest known collapse was to $\mathsf{O_2P}$.

Observation 1. $\mathsf{GC} \in \mathsf{ONP} \subseteq \mathsf{O_2P}$.

Below we make a few remarks. For a further discussion see [Gas10].

- $\mathsf{GC} \in \mathsf{NP}$ since the coloring itself is a witness.
- GC is not known to be in P or even BPP.
- $\mathsf{GC} \in \mathsf{SPARSE}$. In fact, GC has a uniformly-sparse extension.
- Therefore, by the results of [FSW09, GM15], $\mathsf{GC} \in \mathsf{ONP}$.
- On the other hand, by Mahaney's theorem GC is *unlikely* to be NP-complete.

1.4 Proof Overview

Our work builds on the recent line of work on Range Avoidance. [Kor22] provides a reduction of generating hard truth tables from Avoid, and [CHR24, Li24] give a single-valued $\mathsf{S_2P}$ time algorithm for Avoid.

Avoid Framework for Circuit Lower bounds Let $\mathsf{TT_{n,s}} : \{0,1\}^{s \log s} \to \{0,1\}^{2^n}$ be the truth table generator circuit (see Definition 2.13), i.e. $\mathsf{TT_{n,s}}$ take as input an encoding of a n-input s-size circuit and outputs the truth table of the circuit. By construction, $\mathsf{TT_{n,s}}$ maps all circuits of size s (encoded using $s \log s$ bits) to their corresponding truth tables. Then, $\mathsf{Avoid}(\mathsf{TT_{n,s}})$ will output a truth-table not in the range of $\mathsf{TT_{n,s}}$ and hence not decided by any s-sized circuit (a circuit lower bound!!). For correctness we only need to ensure that $s \log s < 2^n$, so the $\mathsf{TT_{n,s}}$ is input-expanding, and hence a valid instance of Avoid.

While the above construction gives us a way of getting explicit exponential lower bounds against even $\mathsf{SIZE}[2^n/n]$, the input to Avoid is also exponential in input length n. As a result, the lower bounds we get are for the exponential class $\mathsf{S_2E}$ and not $\mathsf{S_2P}$. Note that one can scale down this lower bound in a black-box manner to get fixed-polynomial lower bounds for $\mathsf{S_2P}$, but will lose explicitness in the process.

To fix this we modify the above reduction to take as input the prefix truth table generator circuit, $\mathsf{PTT_{n,s}} : \{0,1\}^{s \log s} \to \{0,1\}^{s \log s + 1}$, where instead of evaluating the input circuit on the whole truth table, $\mathsf{PTT_{n,s}}$ evaluates on the lexicographically first $(s \log s + 1)$ inputs (see Definition 2.14). Let $f_{n,s} = \mathsf{Avoid}(\mathsf{PTT_{n,s}})$, and define the truth table of the hard language to be $L := f_{n,s} || 0^{2^n - s \log s - 1}$. By construction, L cannot be decided by any n-input s-size circuit. Moreover, when s is polynomial, the size of $\mathsf{PTT_{n,s}}$ is also polynomial[8] (Lemma 2.15). Hence the single-valued[9] algorithm computing $f_{n,s}$ is in $\mathsf{S_2P}$ and the explicit fixed-polynomial bounds follow.

To see that the language $L \in \mathsf{O_2P}$, observe that the $\mathsf{S_2P}$ time algorithm is oblivious to x, since for any x of length n, $f_{n,s}$ is the same. One important observation here is that, for the purpose of obtaining circuit lower bound, it suffices to solve Range Avoidance on *one* specific family of circuits (the truth table generating circuit that maps another circuit to its truth table). Hence, while it is unclear whether Range Avoidance can be solved in $\mathsf{FO_2P}$, we could still obtain circuit lower bound for $\mathsf{O_2P}$.

[8] In literature the complexity of computing $\mathsf{PTT_{n,s}}$ (Circuit-Eval) is often left as poly, however for our application of getting explicit lower bounds it is crucial to get its fine-grained complexity (see Lemma 2.10 and Lemma 2.15).

[9] For the language to be well defined it is essential for the output of our algorithm to be single-valued.

Hierarchy Theorems for $\mathsf{O_2TIME}$ To get a hierarchy theorem for $\mathsf{O_2TIME}$, we first get an upper bound on $\mathsf{O_2TIME}$ computation via a standard Cook-Levin argument that converts the $\mathsf{O_2TIME}$ verifier into a circuit (SAT-formula) for which we can hard code the "YES" and "NO" irrefutable certificates at every input length (Lemma 3.4). A lower bound follows via the Avoid framework discussed above (Theorem 3.2). We can now lift the hierarchy theorem on circuit size (Theorem 2.8) to get a hierarchy on $\mathsf{O_2TIME}$ (see Theorem 3.5).

Sparsity and Lower Bounds We begin by introducing the notion of *uniformly-sparse extensions*. Roughly speaking a sparse language L has a *uniformly-sparse extension* if there is a language $L' \in \mathsf{P}$, such that $L \subseteq L'$ and L' is also sparse (for formal definitions see Section 2.4).

We show that if a language $L \in \mathsf{S_2P}$ has a *uniformly-sparse extension*, then $L \in \mathsf{O_2P}$. Let L' be the *uniformly-sparse extension* of a language $L \in \mathsf{S_2P}$ and let $X = \{x \in L'\}$. Since $L' \in \mathsf{P}$, we first apply the polynomial time algorithm for L' which let us filter out most inputs, i.e. $x \notin L'$, and hence $x \notin L$. We are now left with deciding membership in L over the set X, where $|X| \leq \text{poly}$.

Let V^* be the polynomial time $\mathsf{S_2}$-verifier for L, then for every $x \in X$ there exists either an irrefutable YES certificate (y_x) s.t. $V^*(x, y_x, \cdot) = 1$, or an irrefutable NO certificate (z_x) s.t. $V^*(x, \cdot, z_x) = 0$. Let Y be the set of all such y_x's and Z be the set of all such z_x's. Now for any $x \in X$, it suffices to find the correct y_x from Y (or z_x from Z) and apply $V^*(x, y_x, z_x)$ to decide x.

In Lemma 4.1 we prove a more efficient parameterized version of this argument. In addition, we are able to apply this in the exponential regime to show the equivalence $\mathsf{O_2E} = \mathsf{S_2E}$ (see Corollary 4.3).

2 Preliminaries

Let $L \subseteq \{0,1\}^*$ be a language. For $n \geq 1$ we define the *n-th slice of* L, $L|_n := L \cap \{0,1\}^n$ as all the strings in L of length n. The characteristic string of $L|_n$, denoted by $\mathcal{X}_{L|_n}$, is the binary string of length 2^n which represents the truth table defined by $L|_n$.

2.1 Complexity Classes

We assume familiarity with complexity theory and notion of non-uniform circuit families (see for e.g. [AB09, Gol08]).

Definition 2.1 (Deterministic Time)**.** Let $t : \mathbb{N} \to \mathbb{N}$. We say that a language $L \in \mathsf{TIME}[t(n)]$, if there exists a deterministic time multi-tape Turing machine that decides L, in at most $O(t(n))$ steps.

Definition 2.2 (Symmetric Time)**.** Let $t : \mathbb{N} \to \mathbb{N}$. We say that a language $L \in \mathsf{S_2TIME}[t(n)]$, if there exists a $O(t(n))$-time predicate $P(x, y, z)$ that takes $x \in \{0,1\}^n$ and $y, z \in \{0,1\}^{t(n)}$ as input, satisfying that:

1. If $x \in L$, then there exists a y such that for all z, $P(x, y, z) = 1$.
2. If $x \notin L$, then there exists a z such that for all y, $P(x, y, z) = 0$.

Moreover, we say $L \in \mathsf{S_2P}$, if $L \in \mathsf{S_2TIME}[p(n)]$ for some polynomial $p(n)$, and $L \in \mathsf{S_2E}$, if $L \in \mathsf{S_2TIME}[t(n)]$ for $t(n) \leq 2^{O(n)}$.

Definition 2.3 (Single-valued $\mathsf{FS_2P}$ algorithm)**.** A single-valued $\mathsf{FS_2P}$ algorithm A is specified by a polynomial $\ell(\cdot)$ together with a polynomial-time algorithm $V_A(x, \pi_1, \pi_2)$. On an input $x \in \{0,1\}^*$, we say that A outputs $y_x \in \{0,1\}^*$, if the following hold:

1. There exists a $\pi_1 \in \{0,1\}^{\ell(|x|)}$ such that for every $\pi_2 \in \{0,1\}^{\ell(|x|)}$, $V_A(x, \pi_1, \pi_2)$ outputs y_x.
2. For every $\pi_1 \in \{0,1\}^{\ell(|x|)}$ there exists a $\pi_2 \in \{0,1\}^{\ell(|x|)}$, such that $V_A(x, \pi_1, \pi_2)$ outputs either y_x or $\perp$.

And we say that A solves a search problem Π if on any input x it outputs a string y_x and $y_x \in \Pi_x$, where a search problem Π maps every input $x \in \{0,1\}^*$ into a solution set $\Pi_x \subseteq \{0,1\}^*$.

We now formally define $\mathsf{O_2TIME}$ - the oblivious version of the class $\mathsf{S_2TIME}$. The key difference is that unlike $\mathsf{S_2TIME}$, where each irrefutable yes/no certificate can depend on the input x itself, in $\mathsf{O_2TIME}$ the yes/no certificates can **only** depend on $|x|$, the length of x. In other words, for every input length n, there exist a common YES-certificate $\mathbf{y}^*$ and a common NO-certificate $\mathbf{z}^*$ for checking membership of $x \in L|_n$.

Definition 2.4 (Oblivious Symmetric Time)**.** Let $t : \mathbb{N} \to \mathbb{N}$. We say that a language $L \in \mathsf{O_2TIME}[t(n)]$, if there exists a $O(t(n))$-time predicate $P(x, y, z)$ such that for every $n \in \mathbb{N}$ there exist $\mathbf{y}^*$ and $\mathbf{z}^*$ of length $O(t(n))$ satisfying the following for every input $x \in \{0,1\}^n$:

1. If $x \in L$, then for all z, $P(x, \mathbf{y}^*, z) = 1$.
2. If $x \notin L$, then for all y, $P(x, y, \mathbf{z}^*) = 0$.

Moreover, we say $L \in \mathsf{O_2P}$, if $L \in \mathsf{O_2TIME}[p(n)]$ for some polynomial $p(n)$, and $L \in \mathsf{O_2E}$, if $L \in \mathsf{O_2TIME}[t(n)]$ for $t(n) \leq 2^{O(n)}$.

2.2 Nonuniformity

We recall certain circuit properties:

Definition 2.5. A boolean circuit C with n inputs and size s, is a Directed Acyclic Graph (DAG) with $(s+n)$ nodes. There are n source nodes corresponding to the inputs labelled $1, \ldots, n$ and one sink node labelled $(n+s)$ corresponding to the output. Each node, labelled $(n+i)$, for $1 \leq i \leq s$ has an in-degree of 2 and corresponds to a gate computing a binary operation over its two incoming edges.

Definition 2.6. Let $s : \mathbb{N} \to \mathbb{N}$. We say that a language $L \in \mathsf{SIZE}[s(n)]$ if L can be computed by circuit families of size $O(s(n))$ for all sufficiently large input size n.

Definition 2.7. Let $s : \mathbb{N} \to \mathbb{N}$. We say that a language $L \in i.o.\text{-}\mathsf{SIZE}[s(n)]$ if L can be computed by circuit families of size $O(s(n))$ for infinitely many input size n.

By definition, we have $\mathsf{SIZE}[s(n)] \subseteq i.o.\text{-}\mathsf{SIZE}[s(n)]$. Hence, circuit lower bounds against $i.o.\text{-}\mathsf{SIZE}[s(n)]$ are stronger and sometimes denoted as *almost-everywhere* circuit lower bound in the literature.

We now state the hierarchy theorem for circuit size. The standard proof of this result is existential and goes through a counting argument (see e.g. [AB09]). However, we comment that using the framework of Avoid, we can now actually get a constructive size hierarchy theorem, albeit with worse parameters.

Theorem 2.8 (Circuit Size Hierarchy Theorem[AB09])**.** *For all functions $s : \mathbb{N} \to \mathbb{N}$ with $n \leq s(n) < o(2^n/n)$:*

$$\mathsf{SIZE}[s(n)] \subsetneq \mathsf{SIZE}[10s(n)] \ .$$

For our applications, it will be essential to have a tight encoding scheme for circuits. In fact, we will also need the fine-grained complexity of the Turing machine computing Circuit-Eval (i.e. given as input a description of a circuit C and a point x, computes $C(x)$).

Lemma 2.9. *For $n, s \in \mathbb{N}$, and $s \geq n \geq 12$, any n-input, s-size circuit C, there exists an encoding scheme $E_{n,s}$ which encodes C using $5s \log s$ bits.*

Proof. Let C be an n-input, s-size circuit, we now define $E_{n,s}$. Each gate label from $1, \ldots, n+s$ can be encoded using $\log(n+s)$ bits. First encode the n inputs using $n\log(n+s)$ bits. Next fix a topological ordering of the remaining gates. For each gate we can encode its two inputs (two previous gates) with $2\log(n+s)$ bits and the binary operation which requires 4 bits (since there 16 possible binary operations). So the length of our encoding is $n\log(n+s) + s(2\log(n+s)+4) \leq 3s\log(2s) + 4s \leq 5s\log s$ for all $n \geq 12$. $\square$

Lemma 2.10. *For $n, s \in \mathbb{N}$, and $s \geq n$, let $E_{n,s}$ be an encoding of an n-input, s-size circuit C using Lemma 2.9. Then there exists a multi-tape Turing machine M such that $M(E_{n,s}, x) = C(x)$ and it runs in $O(s^2 \log s)$ time.*

Proof. We utilize one tape (memory tape) to store all the intermediate values computed at each gate g_i using $n+s$ cells, and a second tape (evaluation tape) using 6 cells to compute the value at each g_i. We process each gate sequentially as it appears in the encoding scheme, and let g_{i_l} and g_{i_r} be the two gates feeding into g_i. Since Lemma 2.9 encodes the gates in a topological order, we can assume that when computing g_i, both g_{i_l} and g_{i_r} have already been computed. First copy the value of input bits of x onto the memory tape, and move the head of the input tape to the right by $n\log(n+s)$ steps in $O(s\log s)$ time. Now to compute a gate g_i we write the values of g_{i_l} and g_{i_r} along with the binary operation onto the evaluation tape. We can compute any binary operation with just constant overhead and write its value onto the ith cell of the memory tape. To output the evaluation of the circuit we output the value on the $(n+s)$th cell of the memory tape. The cost of evaluating each gate is dominated by the 2 read and 1 write operations on the memory tape that take $O(s)$ time. Since the size of the input upper bounds the number of gates we have that the simulation takes $O(s|E_{n,s}|) = O(s^2 \log s)$ time. $\square$

Finally, we recall the famous Cook-Levin theorem that lets us convert a machine $M \in \mathsf{TIME}[t(n)]$ into a circuit $C \in \mathsf{SIZE}[t(n)\log t(n)]$.

Theorem 2.11 (Cook-Levin Theorem [AB09])**.** *Let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function. Then any multi-tape Turing machine running in $\mathsf{TIME}[t(n)]$ time can be simulated by a circuit-family of $\mathsf{SIZE}[t(n)\log t(n)]$.*

2.3 Range Avoidance

Definition 2.12. The Range Avoidance (Avoid) problem is defined as follows: given as input the description of a Boolean circuit $C : \{0,1\}^n \to \{0,1\}^m$, for $m > n$, find a $y \in \{0,1\}^m$ such that $\forall x \in \{0,1\}^n : C(x) \neq y$.

An important object that connects Avoid and circuit lower bound is the truth table generator circuit.

Definition 2.13. [CHR24, Section 2.3] For $n, s \in \mathbb{N}$ where $n \leq s \leq 2^n$, the truth table generator circuit $\mathsf{TT}_{n,s} : \{0,1\}^{L_{n,s}} \to \{0,1\}^{2^n}$ maps a n-input size s circuit using $L_{n,s} = (s+1)(7+\log(n+s))$ bits of description[10] into its truth table. Moreover, such circuit can be uniformly constructed in time $\mathrm{poly}(2^n)$.

For the purpose of obtaining fixed polynomial circuit lower bound, we generalise the truth table generator circuit above into one that outputs the prefix of the truth table. We also use a different encoding scheme (with constant factor loss in the parameter) for the convenience of presentation.

Definition 2.14. For $n, s \in \mathbb{N}$ where $12 \leq n \leq s \leq 2^n$ and $|E_{n,s}| = 5s \log s < 2^n$, the prefix truth table generator circuit $\mathsf{PTT}_{n,s} : \{0,1\}^{|E_{n,s}|} \to \{0,1\}^{|E_{n,s}|+1}$ maps a n-input circuit of size s described with $|E_{n,s}|$ bits into the lexicographically first $|E_{n,s}|+1$ entries of its truth table.

Since we want to prove lower bounds not just in the exponential regime, but also in the polynomial regime for any fixed polynomial, we need a more fine-grained analysis for the running time of uniformly generating $\mathsf{PTT}_{n,s}$

Lemma 2.15. *The prefix truth table generator circuit* $\mathsf{PTT}_{n,s} : \{0,1\}^{|E_{n,s}|} \to \{0,1\}^{|E_{n,s}|+1}$ *has size* $O(|E_{n,s}|^3)$ *and can be uniformly constructed in time* $O(|E_{n,s}|^3)$.

Proof. Let M be the multi-tape Turing machine from Lemma 2.10 that takes as input an encoding of a circuit and a bitstring, and evaluates the circuit on that bitstring. Let C be the circuit generated from Theorem 2.11 that simulates M. Then $\mathsf{SIZE}(C) = O(s^2 \log^2 s) = O(|E_{n,s}|^2)$. Making $|E_{n,s}|+1$ copies of C for each output gate gives a circuit of size $O(|E_{n,s}|^3)$. ∎

Theorem 2.16 ([Li24, CHR24])**.** *There exists a single-valued* $\mathsf{FS_2P}$ *algorithm for* Avoid*. Moreover, on input circuit* $C : \{0,1\}^n \to \{0,1\}^{n+1}$*, the algorithm runs in time* $O(n|C|)$[11].

Theorem 2.17 ([Li24, CHR24])**.** *There exists an explicit language* $L \in \mathsf{S_2E} \setminus i.o.\text{-}\mathsf{SIZE}[2^n/n]$.

Proof. For any $n \in \mathbb{Z}$, let $\mathsf{TT}_n : \{0,1\}^{2^n-1} \to \{0,1\}^{2^n}$ be the truth table generator circuit. Let $f_n \in \{0,1\}^{2^n}$ be the canonical solution output by the single-valued algorithm from Theorem 2.16 on input TT_n.

The hard language L is defined as follows: for any $x \in \{0,1\}^*$, $x \in L$ if and only if the $(x+1)$-th bit of $f_{|x|} = 1$, treating x as an integer from 0 to $2^n - 1$. ∎

2.4 Sparse Languages

We define some notions of sparsity below, we first introduce natural definitions of sparsity and *sparse extensions* in the polynomial regime, and then give their generalizations in the fine-grained setting.

[10] in fact, it maps a stack program of description size $L_{n,s}$ and it is known that every n-input size s circuit has an equivalent stack program of size $L_{n,s}$ [FM05].

[11] the running time was implicit in the proof of [Li24], but easy to verify.

Definition 2.18. A language $L \in \mathsf{SPARSE}$ if for all n, $|L \cap \{0,1\}^n| \leq \mathsf{poly}(n)$. Moreover, L is called *uniformly-sparse* if $L \in \mathsf{P} \cap \mathsf{SPARSE}$.

It is easy to see that $\mathsf{SPARSE} \subseteq \mathsf{P/poly}$. That is, one can identify the yes-instances efficiently, albeit in non-uniform fashion. The purpose of introducing the *uniform-sparsity* is to be able to identify these inputs efficiently in a uniform fashion. Unfortunately, we cannot expect any such language L to lie even in a modestly hard class as, by definition, $L \in \mathsf{P}$. The purpose of the *uniformly-sparse extensions*, on the other hand, is to bridge this gap. One can observe that unlike the *uniformly-sparse* languages, which are contained in P, languages with uniformly-sparse extension can even be undecidable! In particular, any unary language has uniformly-sparse extension in form of 1^*.

Definition 2.19. A language L has a *uniformly-sparse extension*, if there exists a L' s.t. :

1. $L \subseteq L'$
2. L' is *uniformly-sparse*

Generalizing the above definitions in the fine-grained setting, we get:

Definition 2.20. Let $t : \mathbb{N} \to \mathbb{N}$ be a computable function, then a language L is $t(n)$-SPARSE if for all $n, |L \cap \{0,1\}^n| = O(t(n))$. Moreover we say that L is $t(n)$-*uniformly-sparse* if $L \in \mathsf{TIME}[t(n)] \cap t(n)$-$\mathsf{SPARSE}$.

Definition 2.21. L has a $t(n)$-*uniformly-sparse extension*, if there exists a L' s.t.:

1. $L \subseteq L'$
2. L' is $t(n)$-*uniformly-sparse*.

Observe that **every** binary language L is 2^n-SPARSE. Furthermore, every such L has a trivial 2^n-uniformly-sparse extension: $\{0,1\}^*$.

3 Lower Bounds & Hierarchy Theorem

In this section, we first present (Theorem 3.1) a fine-grained, parameterised version of Theorem 2.17. This allows us to use the Avoid framework and get circuit lower bounds in $\mathsf{S_2TIME}[t(n)]$ instead of $\mathsf{S_2E}$. We then observe that our $\mathsf{S_2TIME}[t(n)]$ witness is oblivious of the input, and hence the lower bounds we get are actually in $\mathsf{O_2TIME}[t(n)]$ as highlighted in Theorem 3.2.

In Theorem 3.5 we present the first time hierarchy theorem for $\mathsf{O_2P}$. In fact, we note to the best of our knowledge that this is the first known time hierarchy theorem for a semantic class.

Theorem 3.1. *For $n \in \mathbb{N}$, let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible function, s.t. $t(n) > n \geq 12$ then*

$$\mathsf{S_2TIME}[t(n)] \not\subseteq i.o.\text{-}\mathsf{SIZE}\left[\frac{t(n)^{1/4}}{\log(t(n))}\right].$$

Proof. We construct a language $L_t \in \mathsf{S_2TIME}[t(n)]$ and $L_t \not\subseteq i.o.\text{-}\mathsf{SIZE}\left[\frac{t(n)^{1/4}}{\log(t(n))}\right]$.

For any $n \in \mathbb{N}$, let $s = \lfloor \frac{t(n)^{1/4}}{\log(t(n))} \rfloor$ and $|E_{n,s}| = \lceil 5s \log s \rceil$. Let $\mathsf{PTT}_{n,s} : \{0,1\}^{|E_{n,s}|} \to \{0,1\}^{|E_{n,s}|+1}$ be the prefix truth table generator circuit as in Definition 2.14. Let $f_n \in \{0,1\}^{|E_{n,s}|+1}$ be the canonical solution to $\mathsf{Avoid}(\mathsf{PTT_{n,s}})$ as outputted by the single-valued algorithm from Theorem 2.16.

The hard language L_t is defined as follows: for any $n \in \mathbb{Z}$, the characteristic string of $L_t|_n$ is set to be $\mathcal{X}_{L_t|_n} := f_n || 0^{2^n - |E_{n,s}|-1}$.

By definition of $\mathsf{PTT}_{n,s}$ and the fact that $f_n \notin \mathsf{Image}(\mathsf{PTT}_{n,s})$, we have that $L_t \notin i.o.\text{-}\mathsf{SIZE}[s]$. On the other hand, the single-valued algorithm for finding f_n runs in time $O(|E_{n,s}| \cdot |\mathsf{PTT}_{n,s}|) = O(t(n))$. Hence, $L_t \in \mathsf{S_2TIME}[t(n)]$. □

We make the observation that the witness in the $\mathsf{S_2TIME}$ machine above is oblivious to the actual input x.

Theorem 3.2. *For $n \in \mathbb{N}$, let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible function, s.t. $t(n) > n \geq 12$ then*

$$\mathsf{O_2TIME}[t(n)] \not\subseteq i.o.\text{-}\mathsf{SIZE}\left[\frac{t(n)^{1/4}}{\log(t(n))}\right].$$

Proof. Consider the same language L_t in the proof of Theorem 3.1. Notice that for any input x of the same length n, the $\mathsf{FS_2P}$ algorithm is run on the same circuit $\mathsf{PTT}_{n,s}$ and hence the witness is the same for inputs of the same length. Thus, it follows that $L_t \in \mathsf{O_2TIME}[t(n)]$. □

We now get as a corollary a proof of Theorem 1.

Corollary 3.3. *For all $k \in \mathbb{N}$, there exists an explicit language $L_k \in \mathsf{O_2P}$ s.t. $L_k \not\subseteq \mathsf{SIZE}[n^k]$.*

Proof. Fix $t(n) = n^{5k}$. Then there is an explicit hard language L_t as defined in the proof of Theorem 3.1, such that $L_t \not\subseteq \mathsf{SIZE}[n^k]$. Moreover, by Theorem 3.2 we have that $L_t \in \mathsf{O_2P}$. □

Before proving our hierarchy theorem for $\mathsf{O_2TIME}$, we prove a simple lemma that bounds from above the size of a circuit family computing languages in $\mathsf{O_2TIME}$.

Lemma 3.4. $\mathsf{O_2TIME}[t(n)] \subseteq \mathsf{SIZE}[t(n)\log(t(n))]$.

Proof. Consider any language $L \in \mathsf{O_2TIME}[t(n)]$, and let $V(\cdot,\cdot,\cdot)$ be its $t(n)$-time verifier. For any integer $n \in \mathbb{Z}$, let $y_n, z_n \in \{0,1\}^{t(n)}$ be the irrefutable proofs for input size n. By Theorem 2.11 we can convert $V(\cdot,\cdot,\cdot)$ into a circuit family $\{C_n\} \subseteq \mathsf{SIZE}[t(n)\log(t(n)]$. The values y_n and z_n can be hard-coded into C_n, and hence this circuit will decide L on all inputs of size n. □

Having both an upper bound on the size of circuits simulating an $\mathsf{O_2TIME}$ computation, and also a lower bound for $\mathsf{O_2TIME}$ against circuits, we can use the circuit size hierarchy (Theorem 2.8) to define a time hierarchy on $\mathsf{O_2TIME}$.

Theorem 3.5. *For $n \in \mathbb{N}$, let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function, s.t. $t(n) > n \geq 12$ then:* $\mathsf{O_2TIME}[t(n)] \subsetneq \mathsf{O_2TIME}[t(n)^4 \log^9(t(n))]$.

Proof. Combining Theorem 3.2, Lemma 3.4, and Circuit Size Hierarchy (Theorem 2.8) we have:

$$\mathsf{O_2TIME}[t(n)] \subseteq \mathsf{SIZE}[t(n)\log t(n)] \subsetneq \mathsf{SIZE}[t(n)\log^{\frac{5}{4}} t(n)] \ ,$$

and

$$\mathsf{O_2TIME}[t(n)^4 \log^9(t(n))] \not\subseteq \mathsf{SIZE}[t(n)\log^{\frac{5}{4}} t(n)] \ .$$

□

4 Sparsity

In this section, we use *sparse extensions* to get various structural complexity results. We prove a more fine-grained statement of Theorem 3 which states that any language in $\mathsf{S_2TIME}[t(n)]$ with a *uniformly-sparse extension* is actually in $\mathsf{O_2TIME}[t(n)^2]$. This lets us extract as a corollary another proof of $\mathsf{S_2E} = \mathsf{O_2E}$. As another application of *sparse extensions*, we are able to recover the fixed polynomial lowerbounds for $\mathsf{O_2P}$ from the previous section as stated in Theorem 1. Finally we show connections between *sparse extensions* and open problems posed by [GM15].

Lemma 4.1. *Let $L \in \mathsf{S_2TIME}[t(n)]$. If L has a $t(n)$-uniformly-sparse extension then $L \in \mathsf{O_2TIME}[t(n)^2]$.*

Proof. For any n, let L' be the $t(n)$-*uniformly-sparse extension* of L, and let $\mathcal{F}$ be the $\mathsf{TIME}[t(n)]$ predicate that decides membership in L'. We will now design an $\mathsf{O_2TIME}[t(n)^2]$ verifier V for $L|_n$. Since both L and L' are $t(n)$-SPARSE, we have that for most $x \in \{0,1\}^n$: $L|_n(x) = L'|_n(x) = 0$. V will first use $\mathcal{F}$ to efficiently filter out most non-membership in $L'|_n$, and hence $L|_n$ in $\mathsf{TIME}[t(n)]$. Now V only has to decide membership in L over $t(n)$ many inputs $X = \{x \in \{0,1\}^n : \mathcal{F}(x) = 1\}$. We will use the fact that since $L \in \mathsf{S_2TIME}[t(n)]$, for all $x \in X$, if $x \in L$ there is an irrefutable YES certificate y_x and if $x \notin L$ there is an irrefutable NO certificate z_x and a verifier V^*, running in $\mathsf{TIME}[t(n)]$ s.t.

- if $x \in L$, $\exists y_x$, $\forall z$ s.t. $V^*(x, y_x, z) = 1$
- if $x \notin L$, $\exists z_x$, $\forall y$ s.t. $V^*(x, y, z_x) = 0$

Consider the string Y^* which encodes a table of YES witnesses y_x^* for every input $x \in X$. When $x \in L$ we set $y_x^* = y_x$, and when $x \notin L$ we will set $y_x^* = 0^{t(n)}$. The size of Y^* is $O(t(n)^2)$, since there are at most $t(n)$ entries in the table each of length $t(n) + n$. For every $x \in X \cap \overline{L}$, let z_x be the irrefutable NO-certificate corresponding to x for V^*. We set Z^* to be the concatenation of all such z_x. The size of Z^* is also at most $t(n)^2$.

We now show that Y^* and Z^* will serve as oblivious irrefutable "YES" and "NO" certificates respectively for V. On input (x, Y^*, Z^*), V first parses Y^* to find the corresponding y_x^* in time $\mathsf{TIME}[t(n)^2]$. Then for each $z_i \in Z^*$ we run $V^*(x, y_x^*, z_i)$. If for all z_i, $V^*(x, y_x^*, z_i) = 1$ then V outputs 1, otherwise V will output 0. Since we are making at most $t(n)$ calls that each cost $\mathsf{TIME}[t(n)]$, V runs in $\mathsf{TIME}[t(n)^2]$.

To see correctness, we first analyze the case when $x \in L$, then by construction Y^* includes $y_x^* = y_x$ and V will output 1. On the other hand if $x \notin L$ then there is an irrefutable no-certificate z_x in Z^* so there is no y_i such that $V(x, y_i, z_x) = 1$. Hence V outputs 0.

□

$V(x, Y^*, Z^*)$:

(1) Set output $= 1$.

(2) If $\mathcal{F}(x) = 0$, return 0.

(3) Parse Y^* to get y_x^*.

(4) For $z_i \in Z^*$, do:

   (a) output $=$ output $\wedge V^*(x, y_x^*, z_i)$.

(5) Return output.

Figure 1: $\mathsf{O_2TIME}[t(n)^2]$ Verifier for Language in $\mathsf{S_2TIME}[t(n)]$ with $t(n)$-*uniformly-sparse extension*

By taking $t(n)$ to be a polynomial in Lemma 4.1 we directly get Corollary 4.2 (also Theorem 3) relating $\mathsf{O_2P}$ and $\mathsf{S_2P}$.

Corollary 4.2. *If $L \in \mathsf{S_2P}$ and L has an uniformly-sparse extension, then $L \in \mathsf{O_2P}$*

Similarly, one can prove Theorem 4 by showing the same consequence for coNP vs coONP, thus making a partial progress towards the open questions posed by Goldreich and Meir in [GM15]. In the exponential regime, since all languages have the trivial 2^n-*uniformly-sparse extension* we get the equivalence between $\mathsf{O_2E}$ and $\mathsf{S_2E}$ as seen in Corollary 4.3.

Corollary 4.3. $\mathsf{S_2E} = \mathsf{O_2E}$

Proof. As noted in Section 2.4, every language is 2^n-SPARSE, and has the trivial 2^n-*uniformly-sparse extension*: $\{0,1\}^*$. When $t(n) = 2^n$, by Lemma 4.1 we get that $\mathsf{S_2TIME}[2^n] \subseteq \mathsf{O_2TIME}[2^{2n}]$. $\square$

In particular, the following lemma shows that the hard language in $\mathsf{S_2TIME}[t(n)]$ defined in Theorem 3.1 admits a $t(n)$-*uniformly-sparse extension*, giving another proof of Corollary 3.3.

Lemma 4.4. *For $n \in \mathbb{N}$, let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible function, s.t. $t(n) > n \geq 12$ then, there is an explicit language $L_t \in \mathsf{S_2TIME}[t(n)]$ s.t. $L_t \notin \mathsf{SIZE}\left[\frac{t(n)^{1/4}}{\log(t(n))}\right]$. Moreover, L_t has a $t(n)$-uniformly-sparse extension L'_t.*

Proof. Let L_t be the $\mathsf{S_2TIME}[t(n)]$ language defined in the proof Theorem 3.1 with the characteristic string $\mathcal{X}_{L_t|_n} := f_n || 0^{2^n - |E_{n,s}| - 1}$. We now define the language L'_t whose characteristic string $\mathcal{X}_{L'_t|_n} := 1^{|E_{n,s}|+1} || 0^{2^n - |E_{n,s}| - 1}$. To see that this L'_t is a $t(n)$-*uniformly-sparse extension* of L_t, clearly $L_t \subseteq L'_t$. Moreover membership of $x \in L'_t$ can be decided by checking if the binary value of x is less than or equal to $|E_{n,s}| + 1$ which can be done in $\mathsf{TIME}[n] \subseteq \mathsf{TIME}[t(n)]$. $\square$

Equipped with this lemma we have an alternative proof of fixed polynomial lower bounds for $\mathsf{O_2P}$ as stated in Theorem 1.

Corollary 4.5. (Theorem 1) *For every $k \in \mathbb{N}$, $\mathsf{O_2P} \not\subseteq \mathsf{SIZE}[n^k]$. Moreover, for every k there is an explicit language L_k in $\mathsf{O_2P}$ s.t. $L_k \notin \mathsf{SIZE}[n^k]$.*

Proof. Fix $t(n) = n^{5k}$. Then by Lemma 4.4 there is an explicit language L_k such that $L_k \not\subseteq \mathsf{SIZE}[n^k]$, and L_k has an *uniformly-sparse extension.* Applying Lemma 4.1 we have that $L_k \in \mathsf{O_2TIME}[n^{10k}] \subseteq \mathsf{O_2P}$. $\square$

5 Open Problems

We conclude with a few interesting open problems:

- Can we show that every sparse $\mathsf{S_2P}$ language is also in $\mathsf{O_2P}$?
- Can we tighten the gap in the $\mathsf{O_2TIME}$ hierarchy theorem (Theorem 2)?
- Can we show a non-trivial upper bound for $\mathsf{O_2P}$, for example $\mathsf{P^{NP}}, \mathsf{MA}, \mathsf{PP}$? This would imply explicit fixed-polynomial lower bounds for such classes. On the other hand, we do note that under reasonable derandomization assumptions, $\mathsf{O_2P} \subseteq \mathsf{S_2P} = \mathsf{P^{NP}}$.
- Can we arrive at something interesting about time hierarchy theorem for semantic classes where fixed-polynomial lower bounds are known e.g. $\mathsf{S_2P}$, $\mathsf{ZPP^{NP}}$, assuming $\mathsf{NP} \not\subseteq \mathsf{P/poly}$? For instance, if $\mathsf{NP} \subseteq \mathsf{P/poly}$, then it follows that $\mathsf{S_2P} \subseteq \mathsf{P/poly}$. One could then invoke the circuit size hierarchy theorem (Theorem 2.8) to establish a hierarchy theorem for $\mathsf{S_2TIME}$, similar to how we obtain the hierarchy theorem for $\mathsf{O_2TIME}$.

6 Acknowledgements

We thank Alexander Golovnev for many helpful discussions and feedback on an earlier draft of this manuscript.

References

[Aar06] S. Aaronson. Oracles are subtle but not malicious. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 340–354. IEEE Computer Society, 2006. 4

[Aar07] S. Aaronson. The learnability of quantum states. In *Proceedings of the Royal Society A*, volume 463, page 3089–3114, 2007. 2

[AB09] S. Arora and B. Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009. 7, 8, 9

[AGL23] D. Apon, W. Gasarch, and K. Lawler. The complexity of grid coloring. *Theory Comput. Syst.*, 67(3):521–547, 2023. 2, 5

[Bar02] B. Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In *RANDOM*, pages 194–208, 2002. 3

[BCG+96] N. H. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *J. Comput. Syst. Sci.*, 52(3):421–433, 1996. 4

[Cai07] J.-Y. Cai. $S_2P \subseteq ZPP^{NP}$. *Journal of Computer and System Sciences*, 73(1):25–35, 2007. 1, 4, 5

[Can96] R. Canetti. More on BPP and the polynomial-time hierarchy. *Inf. Process. Lett.*, 57(5):237–241, 1996. 1

[CGL+23] E. Chung, A. Golovnev, Z. Li, M. Obremski, S. Saraogi, and N. Stephens-Davidowitz. On the randomized complexity of range avoidance, with applications to cryptography and metacomplexity. *ECCC preprint https://eccc.weizmann.ac.il/report/2023/193/*, 2023. 3

[CHLR23] Y. Chen, Y. Huang, J. Li, and H. Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 1058–1066, New York, NY, USA, 2023. Association for Computing Machinery. 3, 4

[CHR24] L. Chen, S. Hirahara, and H. Ren. Symmetric exponential time requires near-maximum circuit size. In *Proceedings of the 56th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2024, to appear. Association for Computing Machinery, 2024. 1, 3, 4, 6, 10

[CL23] Y. Chen and J. Li. Hardness of range avoidance and remote point for restricted circuits via cryptography. *ECCC preprint https://eccc.weizmann.ac.il/report/2023/206/*, 2023. 3

[Coo72] S. A. Cook. A hierarchy for nondeterministic time complexity. In *Proceedings of the fourth annual ACM symposium on Theory of computing - STOC '72*, STOC '72. ACM Press, 1972. 3

[CR06] V. T. Chakaravarthy and S. Roy. Oblivious symmetric alternation. In *STACS*, pages 230–241, 2006. 2, 4, 5

[CR11] V. T. Chakaravarthy and S. Roy. Arthur and merlin as oracles. *Comput. Complex.*, 20(3):505–558, 2011. 2, 4

[CT22] L. Chen and R. Tell. Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 125–136, 2022. 3

[CW04] J.-Y. Cai and O. Watanabe. On proving circuit lower bounds against the polynomial-time hierarchy. *SIAM J. Comput.*, 33(4):984–1009, 2004. 4

[CZ19] E. Chattopadhyay and D. Zuckerman. Explicit two-source extractors and resilient functions. *Annals of Mathematics*, 189(3):653 – 705, 2019. 3

[DPWV22] P. Dixon, A. Pavan, J. Vander Woude, and N. V. Vinodchandran. Pseudodeterminism: promises and lowerbounds. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 1552–1565, New York, NY, USA, 2022. Association for Computing Machinery. 3

[FM05] G. S. Frandsen and P. B. Miltersen. Reviewing bounds on the circuit size of the hardest functions. *Information processing letters*, 95(2):354–357, 2005. 10

[FS04] L. Fortnow and R. Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 316–324, 2004. 3

[FST05] L. Fortnow, R. Santhanam, and L. Trevisan. Hierarchies for semantic classes. In *Proceedings of the 37th Annual ACM SIGACT Symposium on Theory of Computing*, pages 348–355. ACM, New York, 2005. 3

[FSW09] L. Fortnow, R. Santhanam, and R. Williams. Fixed-polynomial size circuit bounds. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 19–26. IEEE Computer Society, 2009. 2, 4, 6

[Gas10] W. Gasarch. https://blog.computationalcomplexity.org/2010/07/spares-problems-in-np-thought-to-not-be.html, 2010. 6

[GGNS23] K. Gajulapalli, A. Golovnev, S. Nagargoje, and S. Saraogi. Range avoidance for constant depth circuits: Hardness and algorithms. In Nicole Megow and Adam D. Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, September 11-13, 2023, Atlanta, Georgia, USA*, volume 275 of *LIPIcs*, pages 65:1–65:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. 3

[GLW22] V. Guruswami, X. Lyu, and X. Wang. Range avoidance for low-depth circuits and connections to pseudorandomness. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 20:1–20:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. 3

[GM15] O. Goldreich and O. Meir. Input-oblivious proof systems and a uniform complexity perspective on p/poly. *ACM Transactions on Computation Theory (TOCT)*, 7(4):1–13, 2015. 2, 5, 6, 13, 14

[Gol08] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. 7

[HS65] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117(0):285–306, 1965. 3

[HS66] F. C. Hennie and R. E. Stearns. Two-tape simulation of multitape turing machines. *J. ACM*, 13(4):533–546, oct 1966. 3

[ILW23] R. Ilango, J. Li, and R. Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 1076–1089, New York, NY, USA, 2023. Association for Computing Machinery. 3

[IW97] R. Impagliazzo and A. Wigderson. P=BPP unless E has subexponential circuits: derandomizing the xor lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 220–229, 1997. 1

[Kan82] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1-3):40–56, 1982. 1, 4

[KKMP21] R. Kleinberg, O. Korten, D. Mitropolsky, and C. Papadimitriou. Total Functions in the Polynomial Hierarchy. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 44:1–44:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 3

[KL80] R. M. Karp and R. J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 302–309, 1980. 4

[Kor22] O. Korten. The hardest explicit construction. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 433–444. IEEE, 2022. 3, 6

[KW98] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. *SIAM J. Comput.*, 28(1):311–324, 1998. 4

[Li23] X. Li. Two source extractors for asymptotically optimal entropy, and (many) more. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1271–1281, Los Alamitos, CA, USA, nov 2023. IEEE Computer Society. 3

[Li24] Z. Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In *Proceedings of the 56th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2024, to appear. Association for Computing Machinery, 2024. 1, 3, 4, 6, 10

[LOS21] Z. Lu, I. C. Oliveira, and R. Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 303–316, New York, NY, USA, 2021. Association for Computing Machinery. 3

[LY22] J. Li and T. Yang. 3.1n- o (n) circuit lower bounds for explicit functions. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1180–1193, 2022. 1

[Mah82] S. R. Mahaney. Sparse complete sets of NP: solution of a conjecture of berman and hartmanis. *J. Comput. Syst. Sci.*, 25(2):130–143, 1982. 2

[MP07] D. van Melkebeek and K. Pervyshev. A generic time hierarchy with one bit of advice. *Computational Complexity*, 16(2):139–179, 2007. 3

[MVW99] P. B. Miltersen, N. V. Vinodchandran, and O. Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *COCOON*, pages 210–220, 1999. 1

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. 1

[Rad21] S. P. Radziszowski. Small ramsey numbers. *The Electronic Journal of Combinatorics [electronic only]*, DS01, 2021. 3

[RS98] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Comput. Complex.*, 7(2):152–162, 1998. 1

[RSW22] H. Ren, R. Santhanam, and Z. Wang. On the range avoidance problem for circuits. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 640–650, Los Alamitos, CA, USA, nov 2022. IEEE Computer Society. 3

[San09] R. Santhanam. Circuit lower bounds for merlin–arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009. 4

[SFM78] J. I. Seiferas, M. J. Fischer, and A. R. Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM*, 25(1):146–167, January 1978. 3

[Sha49] C. E. Shannon. The synthesis of two-terminal switching circuits. *Bell Syst. Tech. J.*, 28(1):59–98, 1949. 1

[Vin05] N. V. Vinodchandran. A note on the circuit complexity of PP. *Theor. Comput. Sci.*, 347(1-2):415–418, 2005. 4

[Wil14] R. Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. 1

[Žák83] S. Žák. A turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, October 1983. 3, 5

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il