Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 4 (2025)

A note on a hierarchy theorem for promise-BPTIME

Songhua He[*]

December 5, 2025

Abstract

We prove a time hierarchy theorem for the pr-BPTIME. This is considered to be a folklore problem and was thought to follow from the existence of complete problems or through direct diagonalization. We observe that neither argument carries through in some immediate way in the promise version. However, the hierarchy theorem can be proved by the standard delayed diagonalization for the nondeterministic time hierarchy theorem [Coo72, SFM78, Žák83], or, as it was observed by Rahul Santhanam, from the established BPTIME hierarchies with advice [Bar02, FS04, GST11, FST05, Per05, vMP07, San25].

1 Introduction

Hierarchy theorems are fundamental results in complexity theory. They state that with increased computational resources, one can solve strictly more problems. The time hierarchy theorem for BPTIME remains an infamously elusive topic. To date, unconditional hierarchy theorems for BPTIME are only known to hold when logarithmic or constant advice bits are provided [Bar02, FS04, GST11, FST05, Per05, vMP07]. Additionally, hierarchy theorems are known to hold conditioned on the existence of complete problems for BPP [Bar02]. Unlike deterministic [HS65, HS66] or nondeterministic time hierarchies [Coo72, SFM78, Žák83], the hierarchy theorem for BPTIME remains open since, intuitively, it seems infeasible to efficiently determine whether a randomized Turing machine given an input accepts or rejects with bounded error or not. Hence, the standard diagonalization fails at the step enumerating all the randomized Turing machines with bounded two-sided error. In fact, determining whether a randomized Turing machine has bounded error on every input or not is undecidable.

The situation is believed to be different in its promise version. The time hierarchy for pr-BPTIME (promise probabilistic time classes) is a folklore statement that has appeared in talks, courses, and popular textbooks such as [AB09]. We observed that there is no source sketching its proof, and its validity is potentially assumed to follow from the direct diagonalization or follow by the existence of complete problems for pr-BPTIME; see e.g., [Gaj22].

However, we observed that the direct diagonalization or proofs that are based on direct diagonalization (e.g., the reduction to BPTIME complete problems [Bar02]) does not easily carry through the pr-BPTIME hierarchy theorem. At a high level, a critical step in diagonalization involves negating the output of the enumerated Turing machine. By negating the output, the constructed language

[*]Department of Computer Science, Rutgers University, New Jersey, United States.
Email: sh1511@scarletmail.rutgers.edu

ISSN 1433-8092

will be computed incorrectly by the enumerated Turing machine on that input. However, if the input does not satisfy the promise, negating the output does not yield a valid promise-satisfying result, and no contradiction is achieved. This means that the standard resource-bounded diagonalization arguments fail for promise classes. Note that for non-deterministic time hierarchies, the reason that direct diagonalization does not work is very different. For NTIME, direct diagonalization does not work because negating the output is computationally hard. For promise problems, negating the output is easy, but it does not lead to a contradiction.

To that end, this note shows that the hierarchy theorem follows by adapting the non-deterministic time hierarchy theorem of [Žák83]. Santhanam [San25] also pointed out that a coarser hierarchy theorem for $\mathsf{pr\text{-}BPTIME}$ can also be obtained directly from the well-established BPTIME hierarchy theorems with advice. Both proofs are presented in this note. I have added an appendix where I explain where the argument [Bar02] that shows the connection between complete problems and the hierarchy for BPTIME does not carry through for $\mathsf{pr\text{-}BPTIME}$.

Previous works. The technique of delayed diagonalization was firstly developed in establishing non-deterministic time hierarchies. Direct diagonalization fails in this setting because non-deterministic time classes are not closed under complementation. This challenge was initially addressed by [Coo72], who introduced a padding technique to create sufficient collapses for diagonalization. Subsequently, [SFM78, Žák83] improved the time hierarchy and introduced the delayed diagonalization method.

The time hierarchy theorem for BPTIME was widely open until [Bar02] had the novel idea of proving hierarchy theorems using optimal algorithms, where he showed a time hierarchy of the form $\mathsf{BPP}/\log\log n \not\subseteq \mathsf{BPTIME}[n^d]/\log n$ for every constant $d \geq 1$. The obstacle of testing whether an input satisfies the promise or not was bypassed by using the instance checker for EXP complete problems. [FS04, GST11] improved the usage of advice bits from $\log\log n$ to only 1 bit (i.e., $\mathsf{BPP}/1 \not\subseteq \mathsf{BPTIME}[n^d]/1$) by a more careful padding argument.

Semantic classes with one-sided error posed additional challenges, as the instance checker for EXP complete problems inherently relies on two-sided error. To that end, [FST05] obtained a strong hierarchy theorem for $\mathsf{RP}/1$ by using Levin's optimal algorithm [Lev73] instead. [Per05, vMP07] settled down this line of works by obtaining a strong time hierarchy theorem for every reasonable semantic class with one bit of advice. The use of indirect diagonalization arguments [Coo72, SFM78, Žák83] was also introduced in establishing time hierarchies for semantic classes in the aforementioned previous works [FST05, vMP07]. Readers interested in a comprehensive discussion of these results and techniques are encouraged to [vMP07].

All the above hierarchies with advice imply hierarchies for promise problems [San25]. This is achieved by incorporating the advice into the input of the promise problem. We will present the proof in the next section.

2 The time hierarchy theorem for $\mathsf{pr\text{-}BPTIME}$

We will show two different proofs to the hierarchy theorem for $\mathsf{promise\text{-}BPTIME}$. In addition, both approaches extend naturally to the *promise* versions of several other semantic complexity classes, including ZPTIME, RTIME, MATIME, AMTIME, and $\mathsf{NTIME} \cap \mathsf{coNTIME}$. The first proof is based on the delayed diagonalization argument [Žák83]. The second proof, credit to Santhanam [San25], is obtained by a reduction to the BPTIME hierarchies with advice, which are well established in the literature. We note that the hierarchy theorem obtained in the first way is tighter.

We start with a formal definition of the promise classes we use in this note. Our definition extends the standard definition of promise classes (see, e.g., [Gol08]) to BPTIME.

Definition 1. *A promise problem L consists of a pair of nonintersecting sets of strings, denoted $L = (L_{YES}, L_{NO})$.*

For every function $t : \mathbb{N} \to \mathbb{N}$, the class $\mathsf{pr\text{-}BPTIME}[t(n)]$ contains all the promise problems $L = (L_{YES}, L_{NO})$ such that there exists a constant $c > 0$ and a randomized Turing machine M for which the following happens

1. *For every $x \in L_{YES}$, $M(x)$ halts in $c \cdot t(n)$ steps regardless of its random choices, and $\Pr[M(x) = 1] \geq 2/3$.*
2. *For every $x \in L_{NO}$, $M(x)$ halts in $c \cdot t(n)$ steps regardless of its random choices, and $\Pr[M(x) = 0] \geq 2/3$.*
3. *For other x, $M(x)$ is unrestricted.*

For every constant $k \geq 2$, $\mathsf{pr\text{-}BPTIME}_k[t(n)]$ is defined analogously with the number of tapes of M restricted to be k.

We also define $\mathsf{pr\text{-}BPP} = \cup_{c \geq 1} \mathsf{pr\text{-}BPTIME}[n^c]$.

2.1 Hierarchy by delayed diagonalization

The key observation in the first proof is that even though efficiently determining whether an input satisfies the promise is infeasible, the delayed diagonalization technique can still be used to construct a contradiction and establish the desired separations.

Theorem 2. *For every time-constructible $t(n) \geq n$ and $g(n) = \omega(t(n+1) \log t(n+1))$,*

$$\mathsf{pr\text{-}BPTIME}[t(n)] \subsetneq \mathsf{pr\text{-}BPTIME}[g(n)].$$

Note that the monotonicity of $t(n)$ is not required. This is handled by a careful discussion in the proof.

Proof. We will construct a unary problem in $\mathsf{pr\text{-}BPTIME}[g(n)]$ that is not in $\mathsf{pr\text{-}BPTIME}[t(n)]$. Since the language is unary, its length determines the input.

We want to construct a promise problem $L = (L_{YES}, L_{NO})$ that does not agree with the output of any randomized Turing machine running in $O(t(n))$ steps. To achieve that, we enumerate the Turing machines. The enumeration is constructed inductively. Denote by $h : \mathbb{N}_+ \mapsto \mathbb{N}_+$ the enumeration function, which is a non-decreasing function. We let $h(1) = h(2) = \cdots = h(t(1) \cdot 2^{t(1)}) = 1$. For every $i \geq 2$, let l_i be the smallest number (this number is the Turing machine description) such that $h(l_i)$ is yet to be defined, and $r_i = t(l_i)^2 \cdot 2^{t(l_i)^2}$ to be a large enough number. We set $h(l_i) = h(l_i + 1) = \cdots = h(r_i) = i$. One can see that the function h increases slowly, which is needed for its time constructibility.

For each $i \geq 1$, denote by M_i the i-th randomized Turing machine. Viewing L as a boolean problem, we want to construct L such that at least one among $L(1^{l_i}), \ldots, L(1^{r_i})$ disagrees with the corresponding $M_i(1^{l_i}), \ldots, M_i(1^{r_i})$; i.e., for each i, the Turing machine M_i computes the problem L incorrectly on at least one input length between l_i and r_i. More specifically, we want to show that there exists $l_i \leq n \leq r_i$, such that one of the following happens

1. The input 1^n cannot be computed in $O(t(n))$ steps by M_i with accept or reject probability $\geq 2/3$, but $1^n \in L_{YES} \cup L_{NO}$.

2. $M_i(1^n)$ rejects in $O(t(n))$ steps with $\geq 2/3$ probability while $1^n \in L_{YES}$.

3. $M_i(1^n)$ accepts in $O(t(n))$ steps with $\geq 2/3$ probability while $1^n \in L_{NO}$.

Construction of the adversarial L. The construction is adapted from [Žák83].

The promise is critically used to deal with the case that the acceptance probability is close to 1/2.

For every i, we construct part of the promise problem L. For every $n = l_i, \ldots, r_i - 1$, we let $1^n \in L_{YES}$ if $M_i(1^{n+1})$ accepts in bounded time with $\geq 2/3$ probability, and $1^n \in L_{NO}$ if $M_i(1^{n+1})$ rejects in bounded time with $\geq 2/3$ probability. Additionally, $1^{r_i} \in L_{YES}$ if $M_i(1^{l_i})$ rejects in bounded time with $\geq 2/3$ probability, $1^{r_i} \in L_{NO}$ otherwise. Here, by bounded time we mean $M_i(1^{n+1})$ (resp., $M_i(1^{l_i})$) halts in $p(n)$ (resp., $t(l_i)\log t(l_i)$) steps, where $p(n) := \omega(t(n+1))$ is the maximum number such that $p(n)\log p(n) \leq g(n)$. $p(n)$ can be computed efficiently because $g(n) \geq n$ is time-constructible. So does $t(l_i)$ when $n = r_i \gg t(l_i)$.

We show that computing L can be done in $O(g(n))$ time. The algorithm will compute i, l_i and determine if $n = r_i$, then simulate M_i on either 1^{n+1} or 1^{l_i}. The simulation can be done efficiently. When $n = r_i$, simulating M_i on all the possible randomness will cost at most $p(l_i-1)\cdot 2^{p(l_i-1)} = o(n)$ time. Computing i and l_i given n can be done efficiently by simulating the construction of the function h until $h(l_i-1)$. The construction will stop either when $r_i = t(l_i)^2 2^{t(l_i)^2} \geq n$ or $t(l_i)$ cannot be computed in less than n steps. By the time-constructivity of $t(l_i)$, we also have $r_i > t(l_i) > n$. In both cases, the cost of time is $O(g(n))$.

Next, we are going to show that for every $i \geq 1$, M_i does not compute L. For simplicity, we say $L(1^n) = 1$ if $1^n \in L_{YES}$, $L(1^n) = 0$ if $1^n \in L_{NO}$, and $L(1^n) = \perp$ otherwise. Similarly, we say $M_i(1^n) = \perp$ if its accept and reject probability in $p(n-1)$ steps are $< 2/3$. There are two cases.

1. If for every $l_i \leq n \leq r_i$, $M_i(1^n) \neq \perp$, by our construction, it means that $L(1^{l_i}) = L(1^{l_i+1}) = \cdots = L(1^{r_i})$. However, since $L(1^{r_i}) \neq M_i(1^{l_i}) = L(1^{l_i})$, we get a contradiction. This part is the same as in [Žák83].

2. Otherwise, we let $n \leq r_i$ to be the largest number such that $M_i(1^n) = \perp$. When $n = r_i$, $L(1^n) \neq \perp$ by the definition. When $n < r_i$, $L(1^n) = M_i(1^{n+1}) \neq \perp$ by our assumption. Therefore, $M_i(1^n) = \perp \neq L(1^n)$.

This shows that none of the enumerated randomized Turing machines M_i can compute L correctly. Since $L \in \textsf{pr-BPTIME}[g(n)]$, we obtain the desired hierarchy theorem. □

Corollary 3. *For every $k \geq 2$ and every k-tape time-constructible functions $t_1(n), t_2(n) \geq n$ such that $t_1(n+1) = o(t_2(n))$,*

$$\textsf{pr-BPTIME}_k[t_1(n)] \subsetneq \textsf{pr-BPTIME}_k[t_2(n)]$$

The proof of Corollary 3 is almost identical to Theorem 2. We achieve a tighter time hierarchy theorem by using the linear-time simulation to k-tape Turing machines using the same number of tapes [Für82].

2.2 Hierarchy from non-uniform BPTIME hierarchies

Alternatively, the hierarchy theorem also directly follows from the non-uniform BPTIME hierarchy theorems [FS04, vMP07]. This proof is credited to Rahul Santhanam [San25].

Theorem 4 ([San25])**.** *For every non-decreasing functions $0 \le a(n) \le b(n) \le n$ and every constant $c \ge 1$, if*

$$\mathsf{BPP}/a(n) \not\subseteq \mathsf{BPTIME}[n^c]/b(n)$$

then

$$\mathsf{pr\text{-}BPP} \not\subseteq \mathsf{pr\text{-}BPTIME}[n^c]$$

Proof. Let L be a problem in $\mathsf{BPP}/a(n) - \mathsf{BPTIME}[n^c]/b(n)$. $L \in \mathsf{BPP}/a(n)$ implies that there exists a polynomial-time randomized Turing machine M and advice $s(n) \in \{0,1\}^{a(n)}$ such that M with $s(n)$ decides L within bounded error.

We construct a promise problem $L' = (L'_{YES}, L'_{NO})$ such that $L' \in \mathsf{pr\text{-}BPP} - \mathsf{pr\text{-}BPTIME}[n^c]$. L'_{YES} consists of all the pairs $(x, s(|x|))$ where $x \in L$ and L'_{NO} consists of all the pairs $(x, s(|x|))$ where $x \notin L$. Notice that the input length only increases by at most a factor of 2. Therefore, the asymptotic running time remains the same.

$L' \in \mathsf{pr\text{-}BPP}$ since the advice-taking Turing machine M computes L' if we let M to take the advice from the input.

Suppose for the sake of contradiction that there exists a randomized Turing machine M' that computes L' within time n^c. It means that the same Turing machine given as advice $s(n)$ also computes L, which is a contradiction to $L \notin \mathsf{BPTIME}[n^c]/a(n) \subseteq \mathsf{BPTIME}[n^c]/b(n)$. □

Combined with the BPTIME hierarchy theorem with advice $\mathsf{BPP}/1 \not\subseteq \mathsf{BPTIME}[n^c]/1$ for every $c \ge 1$ [FS04, vMP07], we obtain the following.

Corollary 5. *For every constant $c \ge 1$,*

$$\mathsf{pr\text{-}BPP} \not\subseteq \mathsf{pr\text{-}BPTIME}[n^c]$$

By the standard padding argument, we are able to strengthen it to the following.

Corollary 6. *For every constants $d > c \ge 1$,*

$$\mathsf{pr\text{-}BPTIME}[n^d] \not\subseteq \mathsf{pr\text{-}BPTIME}[n^c]$$

Proof. Suppose for the sake of contradiction that there exists $d > c \ge 1$ such that $\mathsf{pr\text{-}BPTIME}[n^d] = \mathsf{pr\text{-}BPTIME}[n^c]$.

This combined with the standard padding argument imply that for every constant $r > d$,

$$\mathsf{pr\text{-}BPTIME}[n^r] = \mathsf{pr\text{-}BPTIME}[n^{rc/d}] = \mathsf{pr\text{-}BPTIME}[n^{rc^2/d^2}] = \cdots = \mathsf{pr\text{-}BPTIME}[n^c]$$

and hence $\mathsf{pr\text{-}BPP} = \mathsf{pr\text{-}BPTIME}[n^c]$, a contradiction to Corollary 5.

More specifically, for every problem $L \in \mathsf{pr\text{-}BPTIME}[n^r]$, we construct a problem $L' = \{(x, 1^{|x|^{r/d}}) : x \in L\} \in \mathsf{pr\text{-}BPTIME}[n^d]$. By our assumption, $L' \in \mathsf{pr\text{-}BPTIME}[n^d] = \mathsf{pr\text{-}BPTIME}[n^c]$. Therefore, L has an $n^{rc/d}$-time algorithm. By repeating this argument finitely many times, we get $L \in \mathsf{pr\text{-}BPTIME}[n^c]$. □

3 Acknowledgement

The author thanks Eric Allender for encouraging him to write this note, and Oded Goldreich, Periklis Papakonstantinou, and Emanuele Viola for the valuable discussions and suggestions in writing this note and improving the presentation. The author would also like to thank Rahul Santhanam for bringing the previous works to his attention and for sharing an alternative proof of the promise BPTIME hierarchy theorems.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009.

[Bar02] Boaz Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In *International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM/APPROX)*, pages 194–208, 2002.

[Coo72] Stephen A Cook. A hierarchy for nondeterministic time complexity. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing (STOC)*, pages 187–192, 1972.

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

[FST05] Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Hierarchies for semantic classes. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing (STOC)*, pages 348–355, 2005.

[Für82] Martin Fürer. The tight deterministic time hierarchy. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 8–16, 1982.

[Gaj22] Karthik Gajulapalli. In pursuit of a randomized time hierarchy theorem. `https://kgajulapalli.org/expositions/randomized_time_hierarchy.pdf`, 2022.

[Gol08] Oded Goldreich. Computational complexity: a conceptual perspective. *ACM Sigact News*, 39(3):35–39, 2008.

[GST11] Oded Goldreich, Madhu Sudan, and Luca Trevisan. From logarithmic advice to single-bit advice. *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 109–113, 2011.

[HS65] Juris Hartmanis and Richard E Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[HS66] Fred C Hennie and Richard Edwin Stearns. Two-tape simulation of multitape turing machines. *Journal of the ACM (JACM)*, 13(4):533–546, 1966.

[Lev73] Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.

[Per05] Konstantin Pervyshev. Time hierarchies for computations with a bit of advice. In *Electronic Colloquium on Computatioinal Complexity (ECCC), TR05-054*, 2005.

[San25] Rahul Santhanam. Personal communication. 2025.

[SFM78] Joel I Seiferas, Michael J Fischer, and Albert R Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM (JACM)*, 25(1):146–167, 1978.

[vMP07] Dieter van Melkebeek and Konstantin Pervyshev. A generic time hierarchy with one bit of advice. *Computational Complexity*, 16:139–179, 2007.

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

Appendix

A The "gap" in adapting the argument "from complete problem to hierarchy for $\mathsf{pr\text{-}BPTIME}$"

It is well-known that the existence of $\mathsf{BPP\text{-}complete}$ problems implies time hierarchy theorems for BPTIME [Bar02].

Theorem 7 (Theorem 3.6 of [Bar02])**.** *Suppose that* BPP *has a complete problem.*[1] *Then there exists a constant c such that for every time constructible $t : \mathbb{N} \mapsto \mathbb{N}$ it holds that* $\mathsf{BPTIME}[t(n)] \subsetneq \mathsf{BPTIME}[(t(n))^c]$.

Together with a padding argument, the above theorem also implies that under this assumption $\mathsf{BPTIME}[n^d] \subsetneq \mathsf{BPTIME}[n^{d+1}]$ for every constant $d \geq 1$.

To prove a hierarchy theorem for $\mathsf{pr\text{-}BPTIME}$, a natural approach is to combine Theorem 7 with the fact that there is a complete problem for $\mathsf{pr\text{-}BPP}$. However, there is a gap in adapting the proof when we replace BPP by $\mathsf{pr\text{-}BPP}$ and BPTIME by $\mathsf{pr\text{-}BPTIME}$ in the statement.

We will first go through the proof to the above theorem, and then point out the missing part in its promise version.

Proof. Let L be the $\mathsf{BPP\text{-}complete}$ problem, and M_L be a probabilistic machine that computes L in time n^a for some constant a. As we specified in the footnote, there exists a constant $b \geq 1$ such that for every $L' \in \mathsf{BPTIME}[t(n)]$, the deterministic reduction from L' to L runs in time $O(t(n)^b)$. To apply the diagonalization argument, instead of enumerating randomized Turing machines, we

[1]We require a more restricted form of completeness. We say a problem L is $\mathsf{BPP\text{-}complete}$ if for every problem $L' \in \mathsf{BPP}$, there is a deterministic polynomial-time reduction from L' to L. Moreover, we require that for every time-constructible $t(n) \geq n$, there exists $b \geq 1$ such that the reduction from any $L' \in \mathsf{BPTIME}[t(n)]$ runs in time $O(t(n)^b)$. This is not a strict requirement. Please refer to [Bar02] for more discussions.

enumerate deterministic reductions to L. Specifically, denote by M_i^t the i-th deterministic machine restricted to run for at most t steps. The following problem K is computable in $\mathsf{BPTIME}[t(n)^{O(ab)}]$ but not in $\mathsf{BPTIME}[t(n)]$.

$$K : x \in K \text{ iff } M_x^{t(|x|)^b}(x) \notin L$$

The algorithm for K is simply $1 - M_L(M_x^{t(|x|)^b}(x))$, which runs in $t(|x|)^{O(ab)}$ steps.

Suppose for the sake of contradiction that $K \in \mathsf{BPTIME}[t(n)]$. By the completeness of L, there exists an integer i such that for every $x \in \{0,1\}^*$, $x \in K \iff M_i^{t(|x|)^b}(x) \in L$. In particular for $x = i$ it holds that $i \in K \iff M_i^{t(|i|)^b}(i) \in L$. Yet by the definition of K this happens if and only if $i \notin K$, and so we get a contradiction. $\square$

The last step of the above proof can fail in the promise setting (i.e. we do not know whether it is true). Denote by $K = (K_{YES}, K_{NO})$ and $L = (L_{YES}, L_{NO})$ the promise problems defined in the a similar way as above. We still have $i \in K_{YES} \iff M_i^{t(|i|)^b}(i) \in L_{NO}$ and $i \in K_{NO} \iff M_i^{t(|i|)^b}(i) \in L_{YES}$, in which case the above proof works. However, when $i \notin K_{YES} \cup K_{NO}$, $M_i^{t(|i|)^b}(i) \notin L_{YES} \cup L_{NO}$. The negation of an output that lies outside the promise remains outside the promise, and no contradiction appears to arise.

We note that the approach described above represents a straightforward adaptation of the proof of Theorem 7 to the promise setting. However, there may exist a smarter way that circumvents this gap that the author is not aware of.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il