Complexity theoretic implications of pseudodeterministic algorithms for PPT-search problems

Oded Goldreich and Roei Tell

February 17, 2025

Abstract

Pseudodeterministic algorithms are probabilistic algorithms that solve search problems but do so by always providing the same ("canonical") solution to a given instance, except with small probability. While the complexity theoretic implications of pseudodeterministic algorithms were explored in the past, we suggest to conduct this exploration within the framework of the recently defined class of PPT-search problems. Specifically, we use this framework in order to re-formulate some known observations as well as present some new results.

In particular, as observed in the past, the difference between general probabilistic polynomial-time algorithms and pseudodeterministic ones is reflected in the difference between promise-$\mathcal{BPP}$ and $\mathcal{BPP}$. We make this connection stronger by showing that every PPT-search problem has a pseudodeterministic polynomial-time algorithm if and only if every decisional problem in promise-$\mathcal{BPP}$ can be extended to a problem in $\mathcal{BPP}$.

Our main focus is on the class of PPT-search problems with *unary* instances (a.k.a explicit construction problems). In particular, we prove the following results.

- Pseudodeterministic polynomial-time algorithms exist for all unary PPT-search problems if and only if there exist functions that are computable in probabilistic exponential-time but are hard to learn in significantly smaller exponential time.

  The underlying technique is used in order to identify the pseudodeterministic construction of a pseudorandom-set as "universal" for the class of unary PPT-search problems.
- The existence of pseudodeterministic polynomial-time algorithms for all unary PPT-search problems implies that every unary problem in promise-$\mathcal{BPP}$ can be extended to a (unary) problem in $\mathcal{BPP}$. As a corollary, we obtain an alternative proof for a result of Lu, Oliveira, and Santhanam (STOC21), asserting that such algorithms imply a BPtime hierarchy.
- The existence of pseudodeterministic polynomial-time algorithms for a subclass of the unary PPT-search problems (wherein solutions can be verified deterministically) implies results akin to a BPtime hierarchy. For example, it implies that RTime(p) is not contained in BPtime(q), for any polynomial p and a related polynomial q.

We discuss the gap between the former hypotheses and the result provided by Chen, Lu, Oliveira, Ren, and Santhanam (FOCS23).

Contents

1 Introduction

Loosely speaking, pseudodeterministic algorithms, introduced and initially studied by Gal and Goldwasser [9], are probabilistic algorithms that solve search problems but do so by almost always providing the same ("canonical") solution to each given instance. This paper explores the complexity theoretic implications of the existence of pseudodeterministic algorithms for certain search problems.

Our focus is on search problems that can be solved by probabilistic polynomial-time (PPT) algorithms and for which the quality of potential solutions can be evaluated in PPT. Two definitions of this class, hereafter called the class of *PPT-search problems*, were presented by us in [14, 26], and the results of this paper hold for both. (These two definitions are reviewed in Sections 2.2 and 2.3.)

We note that the complexity theoretic implications of pseudodeterministic algorithms were explored in the past (see, for example, [13, 8, 23]). Our main conceptual contribution consists of the suggestion to conduct this exploration within the framework of PPT-search problems. Indeed, we use this framework in order to re-formulate some known observations as well as present some new results.

1.1 On pseudodeterministic algorithm for all PPT-search problems

We first consider the possible existence of pseudodeterministic polynomial-time algorithms for all PPT-search problems. Our main result in this context (Theorem 3.5) asserts that the following two conditions are equivalent:

1. Every PPT-search problem can be solved in pseudodeterministic polynomial-time.
2. Every problem in promise-$\mathcal{BPP}$ can be extended to a problem (i.e., set) in $\mathcal{BPP}$.

We recall that Dixon, Pavan, and Vinodchandran [8] proved that the second statement is equivalent to the existence of a polynomial-time pseudodeterministic algorithm for estimating the acceptance probability of a given circuit, denoted APEP (standing for acceptance probability estimation problem).[1] Since APEP is a PPT-search problem, it follows that the first statement implies the second statement. Proving the opposite direction (i.e., that the second statement implies the first) amounts to showing that APEP is actually complete for PPT-search. Actually, we observe that every PPT-search problem is reducible (in deterministic polynomial-time) to promise-$\mathcal{BPP}$, and use the fact that every promise-$\mathcal{BPP}$ problem is reducible (in deterministic polynomial-time) to estimating the acceptance probability of a circuit (APEP).

Hence, the crucial step in being able to go beyond the aforementioned result of Dixon *et al.* [8] is using the actual definitions of PPT-search problems, and showing that solving PPT-search problems is reducible in deterministic polynomial-time to promise-$\mathcal{BPP}$ (as was done by us in [14, 26]).

1.2 On pseudodeterministic algorithm for all unary PPT-search problems

Our main focus is on PPT-search problems in which the instances are *unary* (i.e., the search algorithm is given 1^n as input, and outputs an n-bit solution); these problems are often referred

[1]The search problem APEP is closely related to the decisional promise problem CAPP (wherein circuits with high acceptance should be accepted, and circuits with low acceptance probability should be rejected).

to as "explicit construction" problems.[2] We show a few implications of the possible existence of pseudodeterministic polynomial-time algorithms for unary PPT-search problems.

Hardness of learning. Loosely speaking, we say that a Boolean function (on bit strings) is hard to learn if, for some constant $\epsilon > 0$, no algorithm that runs in time $2^{\epsilon \cdot n}$ and makes queries to the function (on n-bit inputs) can produce a circuit that computes the function on a 0.99-fraction of the n-bit inputs.

We show that the existence of pseudodeterministic polynomial-time algorithms for all unary PPT-search problems is equivalent to the existence of functions that are computable in probabilistic exponential-time but are hard to learn (see Theorem 4.2). In one direction, we use the observation that pseudodeterminism enables converting a probabilistic argument into an explicit construction, provided that the probabilistic argument asserting the existence of an object can be cast as a unary PPT-search problem. In the opposite direction, we use the observation that we can get a pseudodeterministic algorithm for a PPT-search problem by scanning all the possible outputs of an adequate pseudorandom generator (which may not suffice for full derandomization of the PPT-search solver) and take the first sequence that makes the original PPT algorithm produce a valid solution. The proof of this result relies on technical ideas that appeared in several recent works, such as [6, Sec. 2.1], [21, 22], and [2, Sec. 5.2.2].

The technique underlying the latter result is used to show that a pseudodeterministic polynomial-time construction of a pseudorandom-set, which is itself a unary PPT-search problem, implies pseudodeterministic polynomial-time algorithms for all unary PPT-search problems (see Theorem 4.5). We stress that the aforementioned pseudorandom-sets withstand (uniform) quadratic-time algorithms rather than non-uniform quadratic-size circuits (as constructed in [19] based on string non-uniform assumptions). We comment that a deterministic polynomial-time construction (of the same type) of a pseudorandom-set implies deterministic polynomial-time algorithms for all unary PPT-search problems; this fact is implicit in the proof of Theorem 4.5. We also mention that constructions of "targeted pseudorandom-sets" have analogous ramifications for solving all (binary) PPT-search problems (see last paragraph of Section 4.2).

BPP-extendability of unary promise-BPP and a BPtime hierarchy. We show that the existence of pseudodeterministic polynomial-time algorithms for all unary PPT-search problems implies that every unary problem in promise-$\mathcal{BPP}$ can be extended to a (unary) problem in $\mathcal{BPP}$ (see Theorem 4.6). As a corollary (see Theorem 4.7), we obtain an alternative proof for a result of Lu, Oliveira, and Santhanam [23] that implies that the existence of pseudodeterministic polynomial-time algorithms for all unary PPT-search problems implies a BPtime hierarchy.

Moreover, we show that even the existence of pseudodeterministic polynomial-time algorithms for all unary PPT-search problems that have a *deterministic* (rather than probabilistic) solution-verification procedure will have implications of similar flavour. In particular, the foregoing hypothesis implies a separation of RTime(t) from BPTime(poly(t)); see Theorem 4.9.

The gaps between the required algorithms and what is already known. We recall that, in a recent breakthrough, Chen, Lu, Oliveira, Ren, and Santhanam [5] present polynomial-time

[2]Actually, PPT-search problems with unary instances are only a subset of general explicit construction problems, since the definition of PPT-search (also) postulates that valid solutions must be efficiently *verifiable*, or that the quality of solutions is efficiently measurable; see Section 2 for precise details.

algorithms that pseudodeterministic solve a broad subclass of unary PPT-search problems *on infinitely many input lengths.* Specifically, in these problems, valid solutions are postulated to be verifiable in deterministic (rather than probabilistic) polynomial-time.

Indeed, this class of problems matches the aforementioned one (i.e., the one in the hypothesis that implies a separation of $\mathrm{RTime}(t)$ from $\mathrm{BPTime}(\mathrm{poly}(t))$), but the algorithm of [5] is pseudodeterministic only on infinitely many input lengths (rather than on all but finitely many input lengths). This issue is further discussed in Section 4.5, where we also explain why the feeling that the algorithm of [5] works also for probabilistic verification of solutions is mistaken.

1.3 Related work

As mentioned above, the notion of PPT-search problems is pivotal to this work. Such definitions were provided by us in [14, 26] along with a (deterministic polynomial-time) reduction of solving PPT-search problems to promise-$\mathcal{BPP}$. The foregoing reduction follows a reduction in [11] that was shown for a less appealing class of search problems (which also contains APEP).

In contrast, the work of Dixon, Pavan, and Vinodchandran [8] is pivoted at the Acceptance Probability Estimation Problem (APEP). As stated above, they proved that APEP can be solved by a pseudodeterministic polynomial-time algorithm if any only of every problem in promise-$\mathcal{BPP}$ can be extended to a problem in $\mathcal{BPP}$.

Lu, Oliveira, and Santhanam [23] also explored the consequences of pseudodeterministic algorithms for search problems solvable in PPT, and in particular for unary search problems solvable in PPT. They proved that a pseudodeterministic polynomial-time algorithm for a specific unary search problem, which happens to be a PPT-search problem, would imply a BPtime hierarchy. Specifically, their unary search problem calls for finding a string of logarithmic length that has relatively high *randomized time-bounded Kolmogorov complexity.* (We mention that [23] contains many other results, including a relation between a BPtime hierarchy and an average-case notion of pseudodeterminism.)

1.4 Organization

We start with a preliminaries section (Section 2), which contains the two aforementioned definitions of PPT-search problems. Next, in Section 3 we consider general PPT-search problems, whereas in Section 4 we consider unary PPT-search problems. The preamble of each of these section detailed its contents and internal organization.

We stress that all the results in the paper hold regardless of whether one defines solving PPT-search problem as in [14, Def. 2.4] and [26, Sec. 3] or as in [26, Sec. 2]. Indeed, we will show proofs that hold for both definitions.

2 Preliminaries

Section 2.1 presents definitions that are standard in the area, whereas Sections 2.2 and 2.3 review two definitions of PPT-search problems that were recently suggested in [14, 26].

2.1 Standard definitions

For the sake of good order, we recall the standard definitions of search problems, pseudodeterministic algorithms, and promise (decisional) $\mathcal{BPP}$-problems.

Search problems. Search problems are typically represented by binary relations. For such a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, the set of solutions for the instance x is denoted $R(x) \stackrel{\rm def}{=} \{y : (x,y) \in R\}$, and the set of instances having a solution is denoted $S_R \stackrel{\rm def}{=} \{x : R(x) \neq \emptyset\}$. The search problem associated with R is stated as "given x, find an element of $R(x)$" (or determine that no such solution exists). The candid search problem of R refers to inputs in S_R only (see [10, Def. 2.30]).

Following [11, Def. 3.1] (and [14, Def. 2.1]), we say that a search problem R is a simple PPT-search problem if the following two conditions hold: (1) the search problem associated with R can be solved by a probabilistic polynomial-time (PPT) algorithm, and (2) the problem of deciding membership in R is in $\mathcal{BPP}$. (In Sections 2.2 and 2.3, we consider two extensions of this definition.)

Pseudodeterministic algorithms. In contrast to general probabilistic algorithms that solve a search problem (i.e., F solves R if $\Pr[F(x) \in R(x)] \geq 2/3$), a pseudodeterministic algorithm is required to output the same ("canonical") solution with probability at least $2/3$; that is, F is a pseudodeterministic algorithm solving the search problem of R if *for every $x \in S_R$ there exists $v_x \in R(x)$ such that* $\Pr[F(x) = v_x] \geq 2/3$. The study of pseudodeterministic algorithms was initiated by Gal and Goldwasser [9], who also proved the following:

Theorem 2.1 (pseudodeterministic algorithms vs reductions to $\mathcal{BPP}$): *For any relation $R \subseteq \{0,1\}^*$, the search problem represented by R can be solved by a polynomial-time pseudodeterministic algorithm if and only if solving R is reducible in deterministic polynomial-time to $\mathcal{BPP}$.*

Promise problems and promise-$\mathcal{BPP}$ (i.e., pr$\mathcal{BPP}$). The (decisional) promise problem $\Pi = (\Pi_{\sf yes}, \Pi_{\sf no})$ consists of distinguishing between instances in $\Pi_{\sf yes}$ and instances in $\Pi_{\sf no}$, whereas $\Pi_{\sf yes} \cup \Pi_{\sf no}$ is called the promise.[3] Standard (i.e., pure) decision problems correspond to the special case in which the promise contains all bit strings (i.e., $\Pi_{\sf yes} \cup \Pi_{\sf no} = \{0,1\}^*$). The promise problem $\Pi' = (\Pi'_{\sf yes}, \Pi'_{\sf no})$ is an extension of the promise problem $\Pi = (\Pi_{\sf yes}, \Pi_{\sf no})$ if $\Pi'_{\sf yes} \supseteq \Pi_{\sf yes}$ and $\Pi'_{\sf no} \supseteq \Pi_{\sf no}$.

The class promise-$\mathcal{BPP}$ (i.e., pr$\mathcal{BPP}$) contains decisional promise problems that can be solved in probabilistic polynomial-time; that is, the problem $\Pi = (\Pi_{\sf yes}, \Pi_{\sf no})$ is in promise-BPP if there exists a probabilistic polynomial-time algorithm A such that $\Pr[A(x)=1] \geq 2/3$ if $x \in \Pi_{\sf yes}$ and $\Pr[A(x)=0] \geq 2/3$ if $x \in \Pi_{\sf no}$. We stress that for $x \notin \Pi_{\sf yes} \cup \Pi_{\sf no}$, there is no condition on $A(x)$.

The hierarchy theorem for prBPTime**.** Although a prBPTime hierarchy is considered a folklore result, it was recently indicated by He [18] that the commonly believed proof, via direct diagonalization, is flawed.[4] Fortunately, He [18] also provided an alternative proof, which uses

[3]For a general discussion of promise problems, both of the decisional and search types, see [10, Sec. 2.4.1].

[4]The flawed argument proceeds by considering an efficient enumeration of probabilistic machines. We let $1^n \in \Pi_{\sf yes}$ (resp., $1^n \in \Pi_{\sf no}$) if, on input 1^n, with probability at least $2/3$, the $n^{\rm th}$ machine, denoted M_n, halts within $t(n)$ steps with output 0 (resp., with output 1). Indeed, $(\Pi_{\sf yes}, \Pi_{\sf no})$ is in prBPTime$(\widetilde{O}(t))$, but the impression that it is not in prBPTime(t) is not supported. The point is that, in general, M_n may decide a promise problem (rather than

delayed diagonalization, and establishes the following results (where a unary promise problem is one in which the promise is a unary set).

Theorem 2.2 (a prBPTime hierarchy): *For every time constructible function $t:\mathbb{N}\to\mathbb{N}$ such that $t(n) \geq n$ there exists a unary promise problem in* $\mathrm{prBPTime}(\widetilde{O}(t)) \setminus \mathrm{prBPTime}(t)$.

2.2 PPT-search problems, the first definition

In this section we present a definition of PPT-search problems that was recently suggested in [14, Def. 2.4] and in [26, Sec 3]. In the latter work, the resulting class was called "PPT-optimization", whereas the name "PPT-search" was used for another class that will be presented in Section 2.3.

(Throughout the current paper, whenever we mention PPT-search, we will be clear about which definition we are referring to. Fortunately, this distinction will not be crucial, since all of our results apply equally to both classes.)

Here we extend the foregoing definition of simple PPT-search problems (cf., [11, Def. 3.1] and [14, Def. 2.1]) by replacing the dichotomy between valid and invalid solution (i.e., strings in $R(x)$ versus strings not in $R(x)$) with a quantitative view of the quality of possible solutions.[5] We do so by considering quality functions of the form $q : \{0,1\}^* \times \{0,1\}^* \to [0,1]$ and defining solving the instance x as finding y of almost maximal quality (i.e., $q(x,y) \approx \max_z\{q(x,z)\}$), where the function q is easy to approximate. Actually, for greater flexibility, we consider quality functions of the form $q : P \times \{0,1\}^* \to [0,1]$, where $P \subseteq \{0,1\}^*$ is a promise on the instances. (We stress that this promise on instances (only) is fundamentally different from the promise on instance-solution pairs, which underlies the definition presented in Section 2.3).[6]

Definition 2.3 (PPT-search problem (following [14, Def. 2.4] as modified in [14, Sec. 4.3] (equiv., PPT-optimization problem [26, Def 5])): *For any $P \subseteq \{0,1\}^*$, the function $q : P \times \{0,1\}^* \to [0,1]$ constitutes a* PPT-search problem (called PPT-optimization in [26]) *if the following two conditions hold.*

1. *A relaxed solving condition: There exists a PPT algorithm F such that, for every $x \in P$ and $t \in \mathbb{N}$, it holds that $\Pr[q(x, F(x,1^t)) \geq q'(x) - (1/t)] \;\geq\; 2/3$, where $q'(x) \stackrel{\text{def}}{=} \max_y\{q(x,y)\}$.*
2. *Efficient approximation of q: There exists a PPT algorithm Q such that, for every $x \in P$, $y \in \{0,1\}^*$ and $t \in N$, it holds that $\Pr[|Q(x,y,1^t) - q(x,y)| \leq 1/t] \;\geq\; 2/3$.*

We say that a probabilistic process, M, solves *the search problem q if M satisfies the first condition; that is, $\Pr[q(x, M(x,1^t)) \geq q'(x) - (1/t)] \;\geq\; 2/3$ for every $x \in P$ and $t \in \mathbb{N}$.*

membership in a set) and that 1^n may violate that promise (equiv., M_n machine may accept 1^n with probability that is larger than 1/3 but smaller than 2/3).

[5]The first author considers this quantitative perspective as more natural than the trichotomy that underlies Definition 2.5.

[6]As noted in [14, Sec. 2.1], this promise on the instances falls within the framework of [10, Def. 2.29], and its conceptual meaning is clear (i.e., we totally discard instances that violate the promise and do not care what the algorithm does with them). In contrast, allowing (as in Definition 2.5) a promise on the set of instance-solution pairs leads to a tri-partition of the set of possible solutions into valid, invalid, and in-between solutions. This relaxes the distinguishing task, which only refers to valid and invalid solutions, but raises the question of how to treat the in-between solutions in the solving task. In Definition 2.5 in-between solutions are not allowed in an algorithm that witnesses membership of the search problem in the class, but are allowed in the more relaxed definition of solving.

Note that Definition 2.3 allows for error reduction. Furthermore, simple PPT-search problems (cf. [11, Def. 3.1] and [14, Def. 2.1]) can be cast as a special case of Definition 2.3 by considering $q(x,y) \stackrel{\text{def}}{=} 1$ if $(x,y) \in R$ and $q(x,y) \stackrel{\text{def}}{=} 0$ otherwise.

We shall use the following result, which is proved in [14, 26] by relying on the ideas that underlie the proof of [11, Thm. 3.5].

Theorem 2.4 (solving PPT-search problems is deterministically reducible to pr$\mathcal{BPP}$): *Let $q : P \times \{0,1\}^* \to [0,1]$ be a PPT-search problem as in Definition 2.3. Then, solving q is reducible in deterministic polynomial-time to a promise problem in* pr$\mathcal{BPP}$*; that is, there exists $\Pi \in$* pr$\mathcal{BPP}$ *and a deterministic polynomial-time oracle machine M such that $q(x, M^{\Pi}(x, 1^t)) \geq q'(x) - (1/t)$ for every $x \in P$ and $t \in \mathbb{N}$.*

2.3 PPT-search problems, the second definition

Another definition for PPT-search problems was suggested in [26]. Jumping ahead, the class of problems outlined next is computationally equivalent to the class from Definition 2.3, in the sense that every problem from the class in Definition 2.3 is reducible in deterministic polynomial-time to a problem from the class outlined next, and vice versa (see [26, Sec 3.2]).

Our goal again is to extend the definition of simple search problems. Following [11], instead of formulating search problems as sets of instance-solution pairs $R \subseteq \{0,1\}^* \times \{0,1\}^*$, we formulate search problems as promise problems of pairs, and relax the verifiability condition so that it only applies to pairs in the promise. Specifically, a search problem is now specified by two sets of pairs $\overline{R} = (R_{\mathsf{yes}}, R_{\mathsf{no}})$ where $R_{\mathsf{yes}}, R_{\mathsf{no}} \subseteq \{0,1\}^* \times \{0,1\}^*$ and $R_{\mathsf{yes}} \cap R_{\mathsf{no}} = \emptyset$.

The point is to impose a relaxed verifiability condition that only requires $\overline{R}$ to be efficiently decidable; that is, instead of requiring an efficient verification algorithm that computes a total function, we only require the algorithm to compute a partial function, allowing it to behave arbitrarily on instance-solution pairs outside the promise. For every $x \in \{0,1\}^*$, this yields a tripartition of the possible solutions, into good solutions in $R_{\mathsf{yes}}(x)$ that must be accepted, and bad solutions in $R_{\mathsf{no}}(x)$ that must be rejected, and solutions that are outside the promise $R_{\mathsf{yes}}(x) \cup R_{\mathsf{no}}(x)$ on which the verification algorithm may behave arbitrarily.[7]

The non-trivial part in the definition is the notion of solving a search problem. For a problem $\overline{R}$ to be a PPT-search problem (i.e., to belong to this class of search problems), we require a PPT algorithm that gets input x and outputs a solution in $R_{\mathsf{yes}}(x)$, whp. Now, recall that we are interested in the complexity of solving PPT-search problems (e.g., whether we can solve them deterministically), but what does it mean to *solve* $\overline{R}$? The naive notion, which requires an algorithm that gets x and outputs a solution in $R_{\mathsf{yes}}(x)$, actually fails: some PPT-search problems cannot be solved deterministically in this manner (see [26, Apdx A], in which some other alternative definitions are also critiqued). Thus, we define the notion of solving a PPT-search problem in a more relaxed way, allowing the solving algorithm to output a solution outside the promise.

Definition 2.5 (PPT-search problem [26, Sec 2]): *We say that $\overline{R} = (R_{\mathsf{yes}}, R_{\mathsf{no}})$ is a* PPT-search problem *if the following two conditions hold:*

1. *There is a PPT algorithm V that distinguishes between pairs in R_{yes} and pairs in R_{no}; that is, for every $(x,y) \in R_{\mathsf{yes}}$ it holds that $\Pr[V(x,y) = 1] \geq 2/3$, whereas for every $(x,y) \in R_{\mathsf{no}}$ it holds that $\Pr[V(x,y) = 0] \geq 2/3$,*

[7]Here we use the standard notation $R_{\mathsf{yes}}(x) = \{y : (x,y) \in R_{\mathsf{yes}}\}$ and $R_{\mathsf{no}}(x) = \{y : (x,y) \in R_{\mathsf{no}}\}$.

2. *There is a PPT algorithm F that, when given x such that $R_{\mathsf{yes}}(x) \neq \emptyset$, satisfies* $\Pr[F(x) \in R_{\mathsf{yes}}(x)] \geq 2/3$.

We say that an algorithm solves the search problem $\overline{R}$ *if, when given input x such that $R_{\mathsf{yes}}(x) \neq \emptyset$, it outputs a string that is* not *in $R_{\mathsf{no}}(x)$. In particular, we say that a probabilistic process M solves the search problem $\overline{R}$ if* $\Pr[M(x) \notin R_{\mathsf{no}}(x)] \geq 2/3$ *for any x such that $R_{\mathsf{yes}}(x) \neq \emptyset$.*

One may think of solutions outside the promise (i.e., $y \notin R_{\mathsf{yes}}(x) \cup R_{\mathsf{no}}(x)$) as being adequate and admissible, although not as good as solutions in $R_{\mathsf{yes}}(x)$.[8] It turns out that, like Definition 2.3, the class of PPT-search problems along with the notion of solving that is captured by Definition 2.5 has a useful structure: it has a complete problem and a time hierarchy, it is reducible to pr$\mathcal{BPP}$, and it supports error-reduction (see [26, Sec 2.3]).

The first author finds this hybrid of [14, Def. 2.2] and [14, Def. 2.3] somewhat unnatural: The notion of solving that is implicit in the definition of the class is different from the notion of solving that is defined explicitly later.

Theorem 2.6 (relating Definition 2.5 to Definition 2.3):

1. *For any problem $\overline{R}$ as in Definition 2.5 there is a problem q as in Definition 2.3 such that solving $\overline{R}$ reduces in deterministic polynomial-time to solving q.*

2. *For any problem q as in Definition 2.3 there is a problem $\overline{R}$ as in Definition 2.5 such that solving q reduces in deterministic polynomial-time to solving $\overline{R}$.*

Throughout the paper, for completeness, we will include proofs that refer to PPT-search as in Definition 2.3 as well as to PPT-search as in Definition 2.5. Indeed, an alternative approach would be to prove our results only for one definition, and then rely on Theorem 2.6.

3 Pseudodeterministic algorithms for PPT-search problems

As observed in [13, Sec. 4], the difference between general probabilistic polynomial-time algorithms and pseudodeterministic ones is reflected in the difference between promise-$\mathcal{BPP}$ and $\mathcal{BPP}$, in the sense that a *simple* PPT-search problem is solvable in PPT (resp., in pseudodeterministic polynomial-time) if and only if it is reducible in deterministic polynomial-time to promise-$\mathcal{BPP}$ (resp., to $\mathcal{BPP}$).

The results in this section make the foregoing connection stronger by formalizing it in the context of PPT-search problems as defined in Section 2. We start (in Section 3.1) by presenting the result of Dixon *et al.* [8] in terms of solving a specific PPT-search problem (i.e., APEP is viewed as a PPT-search problem and it is shown to have a pseudordeterministic algorithm if and only if every problem in pr$\mathcal{BPP}$ can be extended to $\mathcal{BPP}$). This result is used as a basis for presenting (in Section 3.2) the main result of this section, which asserts that every PPT-search problem has a pseudordeterministic algorithm if and only if every problem in pr$\mathcal{BPP}$ can be extended to $\mathcal{BPP}$. We conclude this section by stepping outside the scope of PPT-search problems (see Section 3.3).

[8]Indeed, this yields a quality function analogous to the one in Definition 2.3 that has only three possible values.

3.1 An equivalence concerning a specific problem

We begin by recalling the result of Dixon, Pavan, and Vinodchandran [8] that asserts that *the problem of estimating the acceptance probability of a given circuit*, hereafter denoted APEP, *can be solved by pseudodeterministic polynomial-time algorithm if and only if every problem in promise-$\mathcal{BPP}$ has an extension in $\mathcal{BPP}$.* Here we present two expositions of their result: The first in terms of Definition 2.3, and the second in terms of Definition 2.5.

Presentation in terms of Definition 2.3. The following definition formalizes APEP in terms of Definition 2.3, whereas the result that follows establishes the foregoing equivalence (of [8]) in these terms.

Definition 3.1 (the Acceptance Probability Estimation Problem in terms of Definition 2.3 ($\texttt{APE}^{2.3}$)): *Let $\texttt{APE}^{2.3}$ be a PPT-search problem that captures the problem of estimating the acceptance probability of a given Boolean circuit; that is, letting $\texttt{ap}(C) \stackrel{\rm def}{=} \Pr_x[C(x)=1]$, we consider the problem of solving the PPT-search problem q such that $q(C,v) \stackrel{\rm def}{=} 1 - |\texttt{ap}(C) - v|$.*

Note that $q(C,v) = 1$ if and only if $v = \texttt{ap}(C)$ (and that $q(C,v) = 0$ if and only if $\{v, \texttt{ap}(C)\} = \{0,1\}$). Hence, solving the search problem q corresponds to approximating $\texttt{ap}$ (i.e., finding v such that $v \approx \texttt{ap}(C)$, when given a circuit C). Thus, $\texttt{APE}^{2.3}$ (or rather q) is indeed a PPT-search problem, because estimating $\Pr[C(x)=1]$ up to an additive deviation of $1/t$ (and error probability $1/3$) can be done by evaluating C on $O(t^2)$ random inputs.

Theorem 3.2 (pseudodeterministically solving $\texttt{APE}^{2.3}$ vs extending pr$\mathcal{BPP}$ to $\mathcal{BPP}$, following [8]): *The search problem $\texttt{APE}^{2.3}$ has a pseudodeterministic polynomial-time algorithm if and only if every problem in promise-$\mathcal{BPP}$ has an extension in $\mathcal{BPP}$.*

Proof: Recall that $\texttt{APE}^{2.3}$ is a search problem that captures the problem of estimating the acceptance probability of a given Boolean circuit. In particular, the generic decisional promise-BPP problem is easily reducible to $\texttt{APE}^{2.3}$, and so a pseudodeterministic polynomial-time algorithm for $\texttt{APE}^{2.3}$ implies that the former promise problem has an extension in $\mathcal{BPP}$ (see details next). On the other hand, distinguishing $\texttt{APE}_{\texttt{yes}} \stackrel{\rm def}{=} \{(C,v,1^t) : |\texttt{ap}(C) - v| \leq 1/t\}$ from $\texttt{APE}_{\texttt{no}} \stackrel{\rm def}{=} \{(C,v,1^t) : |\texttt{ap}(C) - v| > 3/t\}$ is a promise-BPP problem, and (as we will show later) a $\mathcal{BPP}$ extension of it yields a pseudodeterministic polynomial-time algorithm for $\texttt{APE}^{2.3}$.

To see the "pseudodeterminism to extension" direction, consider any decisional problem $\Pi = (\Pi_{\texttt{yes}}, \Pi_{\texttt{no}})$ in promise-BPP, and let A be the guaranteed algorithm solving it (i.e., $\Pr[A(x) = 1] \geq 2/3$ if $x \in \Pi_{\texttt{yes}}$ and $\Pr[A(x) = 1] \leq 1/3$ if $x \in \Pi_{\texttt{no}}$). Then, for every x, construct the circuit C_x such that $C_x(r)$ represents the decision of A on input x and coins r, and observe that $q'(C_x) = q(C_x, \texttt{ap}(C_x)) = 1$, because $\texttt{ap}(C_x) = \Pr[A(x) = 1]$. Recall that there exist a PPT algorithm F that solves $\texttt{APE}^{2.3}$ (equiv., solves q); that is, $\Pr[q(x, F(C_x, 1^t)) \geq 1 - (1/t)] \geq 2/3$ (equiv., $\Pr[|F(C_x, 1^t) - \texttt{ap}(C_x)| \leq (1/t)] \geq 2/3$). Fixing $t = 7$, and using the ("pseudodeterminism") hypothesis, we consider the corresponding pseudodeterministic polynomial-time algorithm F', and infer that, for every $x \in \{0,1\}^*$, there exists a unique value p_x such that $\Pr[F'(C_x) = p_x] \geq 2/3$, whereas $p_x \in [\texttt{ap}(C_x) \pm 1/7]$ holds. Hence, $x \in \Pi_{\texttt{yes}}$ implies $p_x \geq \frac{2}{3} - \frac{1}{7} > 0.5$ and $x \in \Pi_{\texttt{no}}$ implies $p_x \leq \frac{1}{3} + \frac{1}{7} < 0.5$. The point is that p_x is well-defined for every x, and F' (probabilistically) computes the mapping $x \mapsto p_x$. Letting $S \overset{\text{def}}{=} \{x \in \{0,1\}^* : p_x > 0.5\}$, is follows that $S \in \mathcal{BPP}$ (by using F'), whereas $(S, \overline{S})$ is an extension of Π.

To see the opposite direction (i.e., "extension to pseudodeterminism"), consider the aforementioned decisional promise problem $(\texttt{APE}_{\texttt{yes}}, \texttt{APE}_{\texttt{no}})$, and recall that it is in promise-BPP. Using the ("extension") hypothesis, this decisional problem has an extension in $\mathcal{BPP}$, and we consider a probabilistic polynomial-time algorithm A that solves this an extension. Thus, for every circuit C, every $v \in [0,1]$ and $t \in \mathbb{N}$, it holds that $\Pr[A(C,v,1^t) = 1]$ is either at least $2/3$ or at most $1/3$. Furthermore, it holds that $\Pr[A(C,v,1^t) = 1] \geq 2/3$ if $|v - \texttt{ap}(C)| \leq 1/t$ and $\Pr[A(C,v,1^t) = 1] \leq 1/3$ if $|v - \texttt{ap}(C)| > 3/t$. By invoking A for t times and taking the average value, we obtain an algorithm A' such that, with probability at least $1 - \exp(-\Omega(t))$, it holds that $A'(C,v,1^t) = \Pr[A(C,v,1^t) = 1] \pm 1/9$. Hence, with probability at least $1 - \exp(-\Omega(t))$, it holds that $A'(C,v,1^t) \geq 5/9$ if $|v - \texttt{ap}(C)| \leq 1/t$ and $A'(C,v,1^t) \leq 4/9$ if $|v - \texttt{ap}(C)| > 3/t$. Now, consider an algorithm that, on input a circuit C and 1^t, invokes $A'(C,v,1^{3t})$ for each $v \in \{(i-0.5)/3t : i \in [3t]\}$, and outputs the smallest value v such that $A'(C,v,1^{3t}) > 1/2$.[9] Then, with probability greater than $1 - 3t \cdot \exp(-\Omega(t))$, this output (i.e., v) is uniquely defined and is within an additive term of $3/3t$ away from $\texttt{ap}(C)$. Hence, we obtained a PPT pseudodeterministic algorithm for solving $\texttt{APE}^{2.3}$. ■

Presentation in terms of Definition 2.5. The following definition formalizes APEP in terms of Definition 2.5, whereas the result that follows establishes the foregoing equivalence (of [8]) in these terms. (The following definition appeared in [26], where it was also proved that this problem is complete for the class of PPT-search problems.)

Definition 3.3 (the Acceptance Probability Estimation Problem in terms of Definition 2.5 ($\texttt{APE}^{2.5}$)): *Let $\texttt{APE}^{2.5} = (\texttt{APE}^{2.5}_{\texttt{yes}}, \texttt{APE}^{2.5}_{\texttt{no}})$ be the search problem in which inputs are pairs $(C, 1^t)$ such that $C : \{0,1\}^n \to \{0,1\}$ is a Boolean circuit and $t \in \mathbb{N}$, and solutions are real values v such that $v \in \texttt{APE}^{2.5}_{\texttt{yes}}(C, 1^t)$ if and only if $|\texttt{ap}(C) - v| \leq 1/t$, and $v \in \texttt{APE}^{2.5}_{\texttt{no}}(C, 1^t)$ if and only if $|\texttt{ap}(C) - v| \geq 2/t$, where $\texttt{ap}(C) = \Pr_{x \in \{0,1\}^n}[C(x) = 1]$.*

Theorem 3.4 (pseudodeterministically solving $\texttt{APE}^{2.5}$ vs extending pr$\mathcal{BPP}$ to $\mathcal{BPP}$, following [8]): *The problem $\texttt{APE}^{2.5}$ can be solved in pseudodeterministic polynomial-time if and only if every problem in pr$\mathcal{BPP}$ has an extension in $\mathcal{BPP}$.*

Proof: Assume that every problem in pr$\mathcal{BPP}$ has an extension in $\mathcal{BPP}$. Since $\texttt{APE}^{2.5}$ is a PPT-search problem (see [26, Prop. 2]), it is reducible to pr$\mathcal{BPP}$ in deterministic polynomial-time (see [26, Thm 3]), and hence to $\mathcal{BPP}$. By Theorem 2.1, $\texttt{APE}^{2.5}$ can be solved by a polynomial-time pseudodeterministic algorithm.[10]

For the other direction, assume that $\texttt{APE}^{2.5}$ can be solved in pseudodeterministic polynomial-time, and let $\Pi = (\Pi_{\texttt{yes}}, \Pi_{\texttt{no}})$ be a problem in pr$\mathcal{BPP}$. Observe that Π is reducible to $\texttt{APE}^{2.5}$,

[9] In general, with very high probability, there may be a constant number of v's that satisfy $A'(C,v,1^{3t}) > 1/2$, and $|v - \texttt{ap}(C)| \leq 1/t$ holds for each of them. Selecting the smallest of these is an arbitrary deterministic choice, and any other deterministic choice would do.

[10] Specifically, to invoke Theorem 2.1 we define $R = \{(x,v) : v \notin \Pi_{\texttt{no}}(x)\}$, where $\Pi_{\texttt{no}}(x)$ is the set of invalid solutions for $x = (C, 1^t)$ in $\texttt{APE}^{2.5}$ (i.e., $v \in \Pi_{\texttt{no}}(C, 1^t)$ if $|\texttt{ap}(C) - v| \geq 2/t$). The reduction to $\mathcal{BPP}$ finds solutions that are not in $\Pi_{\texttt{no}}(x)$, and hence in R. By Theorem 2.1, there is a pseudodeterministic algorithm that finds solutions in R, hence not in $\Pi_{\texttt{no}}(x)$.

by mapping an input x to $(C_x, 1^{20})$.[11] Since $\texttt{APE}^{2.5}$ is solvable pseudodeterministically, there is a PPT algorithm A that on input $(C_x, 1^{20})$ outputs, with probability at least 2/3, a canonical value $v(C_x)$ such that $|v(C_x) - \texttt{ap}(C_x)| \leq 1/10$. Consider a probabilistic algorithm A' that accepts x (i.e., outputs 1) if and only if the output of $A(C_x, 1^{10})$ is larger than 1/2. Then, for every input x, there exists a bit $b(x)$ such that $\Pr[A'(x) = b(x)] \geq 2/3$ and $b(x) = 1$ (resp., $b(x) = 0$) if $x \in \Pi_{\texttt{yes}}$ (resp., $x \in \Pi_{\texttt{no}}$). Thus, A' defines a $\mathcal{BPP}$ extension of Π. ■

3.2 The general equivalence

We now prove the equivalence outlined in Section 1.1. The gist of the proof is that the problem of estimating the acceptance probability of a given circuit is complete for PPT-search. Alternatively, the claim follows by recalling that (i) every PPT-search problem is reducible (in deterministic polynomial-time) to promise-$\mathcal{BPP}$ (see Sections 2.2 and 2.3), and that (ii) search problems that are reducible (in deterministic polynomial-time) to $\mathcal{BPP}$ have a polynomial-time pseudodeterministic algorithm [9].

Theorem 3.5 (the main equivalence): *The following three statements are equivalent.*

1. *Every problem in promise-$\mathcal{BPP}$ can be extended to a problem in $\mathcal{BPP}$.*
2. *There exists a polynomial-time pseudodeterministic algorithm for estimating the acceptance probability of a given circuit* (equiv., solving either the search problem $\texttt{APE}^{2.3}$ or $\texttt{APE}^{2.5}$, in the adequate sense).
3. *Every PPT-search problem can be solved by a polynomial-time pseudodeterministic algorithm, regardless if we use Definition 2.3 or Definition 2.5.*

Proof: By Theorems 3.2 and 3.4, the first two statements are equivalent. We present two alternative proofs showing the equivalence of these two statements to the third. Actually, since the second statement follows immediately from the third, it suffices to show that one of the first two statements implies the third.

In the first alternative we show that the second statement implies the third. This demonstration is based on the fact that every PPT-search problem is reducible in deterministic polynomial-time to the problem of estimating the acceptance probability of a given circuit. (Specifically, for PPT-search as in Definition 2.3, combine the reduction of Theorem 2.4 with the straightforward reduction of pr$\mathcal{BPP}$ to $\texttt{APE}^{2.3}$; whereas for PPT-search as in Definition 2.5 apply [26, Thm. 3].)

In the second alternative we show that the first statement implies the third. Here we recall that every PPT-search problem is reducible to pr$\mathcal{BPP}$ in deterministic polynomial-time (either using Theorem 2.4 or using [26, Thm. 3]). Using the first statement, it follows that every PPT-search problem is reducible to $\mathcal{BPP}$ in deterministic polynomial-time. We are done by applying Theorem 2.1, which asserts that search problems that are reducible in deterministic polynomial-time to $\mathcal{BPP}$ can be solved by a polynomial-time pseudodeterministic algorithm.[12] ■

[11] See the proof of Theorem 3.2.

[12] Specifically, when using Definition 2.5, invoking Theorem 2.1 requires defining a relation R that consists of all valid solutions (as in the proof of Theorem 3.4).

3.3 An equivalence for a special case of non-PPT-search

While most of this paper refers to PPT-search problems, in this subsection we consider search problems that are solvable by PPT algorithms but do not necessarily have efficiently recognizable instance-solution pairs. We ask when can such search problems be solvable by polynomial-time pseudodeterministic algorithms.

An obvious negative example is provided by the search problem that consists of finding strings of high Kolmogorov complexity; that is, the search problem that corresponds to the binary relation $R_K = \{(x,y) : |y| = |x| \ \&\ K(y) \geq |x|/2\}$, where $K(y)$ is the Kolmogorov complexity of y. Indeed, the search problem of R_K can be solved by a linear-time algorithm that, on input x, generates a random $|x|$-bit string; but R_K cannot be solved by a pseudodeterministic algorithm, regardless of its running-time.

Seeking a meaningful positive example, we confine ourselves to search problems that are not evidently solvable by deterministic polynomial-time algorithms. The best evidence for such non-evidence is provided by an equivalence the existence of polynomial-time pseudodeterministic algorithms for all PPT-search problems (equiv., extendability of pr$\mathcal{BPP}$ to $\mathcal{BPP}$). Such a result is provided next, indicating that the efficient verification (of instance-solution pairs) postulate can be replaced by syntactic postulate; specifically, an extremely strong upper bound on the length of possible solutions.

Theorem 3.6 (on search problems with single-bit solutions): *The following two statements are equivalent*

1. *Every total search problem R with single-bit solutions* (i.e., $|R(x)| \geq 1$ and $R(x) \subseteq \{0,1\}$ for every x) *that can be solved by a PPT algorithm can also be solved by a polynomial-time pseudodeterministic algorithm.*
2. *Every decisional promise problem in promise-BPP has an extension in $\mathcal{BPP}$.*

A more appealing formulation of the syntactic condition in Statement 1 refers to search problem R such that there exists an easy to compute set of at most two possible solutions for R; that is, there exists a polynomial-time computable function $F : \{0,1\}^* \to \binom{\{0,1\}^*}{2}$ such that $R(x) \subseteq F(x)$ for every x. Interestingly, the foregoing result cannot be extended beyond two possible solutions (see Proposition 3.7).

Proof: Intuitively, we show that search problems as in the hypothesis of Statement 1 (i.e., total search problems with single-bit solutions that can be solved in probabilistic polynomial-time) are closely related to decisional problems in promise-BPP. Furthermore, polynomial-time pseudodeterministic algorithms for the former correspond to sets in $\mathcal{BPP}$ that extend the latter.

We first show that Statement 2 implies Statement 1. Intuitively, we show that search problems as in the hypothesis of Statement 1 can be cast as decisional problems in promise-BPP. Let A be a probabilistic polynomial-time algorithm that solves a total search problem $R \subseteq \{0,1\}^* \times \{0,1\}$. Define $\Pi = (\Pi_{\tt yes}, \Pi_{\tt no})$ such that

$$\begin{aligned} \Pi_{\tt yes} &\stackrel{\rm def}{=} \{x \in \{0,1\}^* : \Pr[A(x)=1] \geq 2/3\}, \\ \Pi_{\tt no} &\stackrel{\rm def}{=} \{x \in \{0,1\}^* : \Pr[A(x)=0] \geq 2/3\}. \end{aligned}$$

Note that A actually solves the decisional promise problem Π, and that for every $x \in \{0,1\}^* \setminus (\Pi_{\tt yes} \cup \Pi_{\tt no})$ it holds that $R(x) = \{0,1\}$, because if $R(x) = \{b\}$ then $\Pr[A(x)=b] \geq 2/3$ must hold (whereas $R(x) \neq \emptyset$ by the hypothesis). By Statement 2, the promise problem Π has an extension $(S, \overline{S})$ that is in $\mathcal{BPP}$; that is, there exists a probabilistic polynomial-time algorithm A' such that $\Pr[A'(x)=1] \geq 2/3$ if $x \in S$ and $\Pr[A'(x)=0] \geq 2/3$ otherwise. It follows that A' constitutes a polynomial-time pseudodeterministic algorithm that solves the search problem R.

We now show that Statement 1 implies Statement 2. Intuitively, we show that decisional problems in promise-BPP can be cast as search problems as in the hypothesis of Statement 1. Given a decisional promise problem $\Pi = (\Pi_{\tt yes}, \Pi_{\tt no})$ in promise-BPP, define R such that

$$R(x) \overset{\rm def}{=} \begin{cases} \{1\} & \text{if } x \in \Pi_{\tt yes} \\ \{0\} & \text{if } x \in \Pi_{\tt no} \\ \{0,1\} & \text{otherwise (i.e., } x \notin \Pi_{\tt yes} \cup \Pi_{\tt no}) \end{cases}$$

Note that the search problem R is total and can be solved by a probabilistic polynomial-time algorithm (by invoking the algorithm that is guaranteed for the decisional promise problem Π, where we assume that the latter algorithm always outputs a bit on any input (including when the input violates the promise)). Then, by Statement 1, there exists a polynomial-time pseudodeterministic algorithm A that solves the search problem R (i.e., for every x there exists $b \in R(x)$ such that $\Pr[A(x) = b] \geq 2/3$). Now, define $S \overset{\rm def}{=} \{x \in \{0,1\}^* : \Pr[A(x) = 1] > 1/2\}$, and note that $(S, \overline{S})$ extends Π (because for $x \in \Pi_{\tt yes} \cup \Pi_{\tt no}$ it holds that $\Pr[A(x)=1] \geq 2/3$ if $x \in \Pi_{\tt yes}$ and $\Pr[A(x)=0] \geq 2/3$ otherwise). Last, observe that $S \in \mathcal{BPP}$ (by using algorithm A itself). ■

Proposition 3.7 (on search problems with at most three solutions): *There exists a total search problem $R \subseteq \{0,1\}^* \times \{1,2,3\}$ that can be solved in probabilistic polynomial-time but cannot be solved by a pseudodeterministic algorithm. Furthermore, $|R(x)| = 2$ for every x.*

The constant 3 is an artifact of the standard definition of solving a search problem, which requires outputting a valid solution with probability at least 2/3 (see Section 2.1); Proposition 3.7 holds also if we replace 2/3 with any threshold in (0.5, 2/3), but replacing 2/3 with $(t-1)/t$ would require a set of t values (see Footnote 14).[13] Recall that error reduction is not available to us here, since we lack a mechanism for recognizing valid solutions.

Proof: We first observe that any relation $R \subseteq \{0,1\}^* \times \{1,2,3\}$ such that $|R(x)| = 2$ for every x can be solved by a PPT algorithm that just selects $r \in \{1,2,3\}$ uniformly at random and outputs it. We prove the existence of such a relation R that cannot be solved by a pseudodeterministic algorithm by a diagonalization argument. (The crucial point here is that the probability of outputting a valid solution is $|R(x)|/|\{1,2,3\}|$, which equals 2/3.)[14]

Specifically, we consider an enumeration of probabilistic machines and associate with each machine infinitely many inputs. For each machine M and input x associated with it, we let $b_x \overset{\rm def}{=} y$ if

[13]On the other hand, Proposition 3.7 can be extended to assert that for every time-constructible $k : \mathbb{N} \to \mathbb{N} \setminus \{1,2\}$ there exists a total search problem $R \subseteq \bigcup_{n\in\mathbb{N}} \{0,1\}^* \times [k(n)]$ such that (i) $|R(x)| = k(|x|) - 1$ for every x, and (ii) R can be solved in probabilistic polynomial-time but cannot be solved by a pseudodeterministic algorithm.

[14]We stress that the argument relies on the fact that the threshold probability is not larger than 2/3. If the definition of solving had required outputting a valid solution with probability at least 0.99, then we could have modified the argument using $R \subseteq \{0,1\}^* \times \{1,2,..,100\}$ (and $|R(x)| = 99$ for every x). In contrast, the threshold probability for pseudodeterministic solving is immaterial (as long as it is noticeably larger than 1/2), since error reduction is available in this case.

$\Pr[M(x)=y] \geq 2/3$, and $b_x \stackrel{\text{def}}{=} \perp$ if no such y exists. Next, for simplicity, we redefine b_x arbitrarily such that $b_x \stackrel{\text{def}}{=} 1$ if $b_x \notin \{1,2,3\}$. Lastly, we define $R(x) \stackrel{\text{def}}{=} \{1,2,3\} \setminus \{b_x\}$.

Now, suppose towards the contradiction that M is a pseudodeterministic machine solving R and let x be an input x associated with M. Then, for some $c_x \in R(x)$, it holds that $\Pr[M(x)=c_x] \geq 2/3$. But this implies that $b_x = c_x$ and $c_x \notin R(x)$ follows, in contradiction to the hypothesis that M solves R. ■

Conclusion. Recall that in this subsection we asked which search problems that are solvable by PPT algorithms but do not have efficiently recognizable instance-solution pairs can be solvable by polynomial-time pseudodeterministic algorithms. Proposition 3.7 indicates that the direction taken by Theorem 3.6 cannot be extended, but this does not rule out the possibility of other directions.

4 Pseudodeterministic algorithms for unary PPT-search problems

In this section, we consider unary PPT-search problems, in which we only care about unary instances (i.e., of the form 1^n). Formally, in both Definitions 2.3 and 2.5, we consider a promise that the instance is unary. (Alternatively, in Definition 2.3 we may impose the additional condition that the quality function q is identical on all instances x of the same length (i.e., $q(x,y) = q(1^{|x|}, y)$ for all $x, y \in \{0,1\}^*$).)

We ask *what are the implications of the existence of pseudodeterministic algorithms for all unary PPT-search problems.* We show a few such implications. The first implication, presented in Section 4.1, is that the existence of such algorithms implies the hardness of learning a set in $\mathcal{BPE}$. In fact, we show that these two conditions are equivalent; that is, we also show that the hardness of learning a set in $\mathcal{BPE}$ implies the existence of pseudodeterministic algorithms for all unary PPT-search problems. In Section 4.2 we use the techniques underlying the second direction in order to show that a pseudodeterministic construction of a pseudorandom-set implies pseudodeterministic algorithms for all unary PPT-search problems.

Another implication, presented in Section 4.3, is that the existence of pseudodeterministic algorithms for all unary PPT-search problems implies that every unary problem in pr$\mathcal{BPP}$ can be extended to a unary set in $\mathcal{BPP}$. This is analogous to one direction of Theorem 3.5, and it implies a BPtime hierarchy (by combining it with Theorem 2.2).

In Section 4.4 we show that even a more modest hypothesis yield a time hierarchy. Specifically, the hypothesis postulates the existence of pseudodeterministic algorithms for all unary PPT-search problems in which solutions are deterministically (rather than probabilistically) verifiable. We show that this hypothesis implies that a separation between RTime(p) and BPTime(poly(p)) for any polynomial p.

In Section 4.5, we complement all these results by recalling the recent result of [5] and explaining the gaps between this result and the hypotheses in the foregoing results.

4.1 A connection to hardness of learning with queries

We start by defining the notion of learning that we use. This notion is in line with the standard definition of learning with membership queries (with respect to the uniform distribution), but it deviates from the standard by considering a single Boolean function defined on the set of all bit strings. This makes sense because we model potential learners as uniform machines, and so the fact that the function is fixed does not trivialize the learning task, which refers to restrictions of this function to ℓ-bit strings (where ℓ is a varying parameter).

Definition 4.1 (learning from membership queries): *For a function $f : \{0,1\}^* \to \{0,1\}$ and $\ell \in \mathbb{N}$, we denote the restriction of f to $\{0,1\}^\ell$ by f_ℓ. For a constant $\delta > 0$, we say that a probabilistic algorithm A* learns f from membership queries with accuracy $1-\delta$ *if for infinitely many $\ell \in \mathbb{N}$, on input 1^ℓ and oracle access to f_ℓ, with probability at least $2/3$, the algorithm outputs a circuit C such that $\Pr_{x\in\{0,1\}^\ell}[C(x)=f_\ell(x)] \geq 1-\delta$.*

Indeed, postulating that learning succeeds only on infinitely many input-lengths deviates from the standard conventions. But, our focus is actually on the hardness of learning. Hence, saying that A does not learn f (from membership queries with accuracy $1-\delta$) means that for all but finitely many $\ell \in \mathbb{N}$ it holds that $\Pr_{x\in\{0,1\}^\ell}[C(x)\neq f_\ell(x)] > \delta$, where C is the circuit output by A (on input 1^ℓ and oracle access to f_ℓ). Lastly, we associate the function f with the set $f^{-1}(1)$.

Theorem 4.2 (pseudodeterministic polynomial-time algorithms for unary PPT-search problems versus a set in $\mathcal{BPE}$ that is hard to learn): *The following two statement are equivalent.*

1. *Every unary PPT-search problem can be solved by a pseudodeterministic polynomial-time algorithm.*
2. *There exist $f \in \mathcal{BPE}$ and a constant $\epsilon > 0$ such that for every probabilistic algorithm A that makes at most $2^{\epsilon\cdot\ell}$ steps on input 1^ℓ it holds that A does not learn f from membership queries with accuracy 0.99.*

Note that Statement 2 is weaker than the hypothesis that $\mathcal{BPE}$ does not have relatively small circuits (i.e., circuits of size at most $2^{\epsilon\cdot\ell}$ for ℓ-bit inputs).

Proof: We prove the claimed equivalence while referring to the notion of PPT-search problems as in Definition 2.5. The same strategy can be employed when referring to Definition 2.3. Alternatively, we can use the reductions asserted in Theorem 2.6 in order to derive the claimed equivalence under Definition 2.3.

Proving that Statement 1 implies Statement 2: From pseudodeterminism to hardness of learning. The key observation is that finding (the restriction to ℓ-bit strings of) a Boolean function that is hard to learn (in the sense of Definition 4.1) is a unary PPT-search problem (when the input is 1^{2^ℓ}). Hence, a polynomial-time pseudodeterministic algorithm that solves this search problem implies that computing such a Boolean function is in $\mathcal{BPE}$. Details follow.

First, let us elaborate the foregoing idea. To show that finding a hard function is a PPT-search problem, we notice (i) that a random function is hard (even wrt small-exponential resources), and (ii) it is feasible to check whether a (restriction of) a function is hard. The validity of (ii) relies on the fact that hardness is against a small number of machines (since we use a uniform model of computation) and that the success probability of a machine can be estimated in PPT. The latter approximation problem translates to a gap (search) problem, which explains why we use a non-simple PPT-search problem. Now, the main observation is that a pseudodeterministic algorithm that solves this search problem yields a single hard function rather than a distribution on hard functions. This single function will yield a set in $\mathcal{BPE}$ (i.e., set of the 1-preimages of the function).

We now turn to the actual proof. For a machine M and a Boolean function f, we denote by $s_\ell(M,f)$ the probability, over M's random coins, that $M^{f_\ell}(1^\ell)$ is successful, where an execution of $M^{f_\ell}(1^\ell)$ is called successful if it outputs a circuit C such that $\Pr_{x\in\{0,1\}^\ell}[C(x)=f_\ell(x)] \geq 0.99$.

Claim 4.2.1 (finding a hard function is a PPT-search problem): *Let $M_1, M_2, ...$ be an efficient enumeration of probabilistic oracle machines, modified so that on input 1^ℓ they run in time at most $2^{\ell/2}$. For $n = 2^\ell$, we view $g : \{0,1\}^\ell \to \{0,1\}$ as an n-bit string and define*

$$\begin{aligned} R_{\tt yes}(1^n) &= \{g \in \{0,1\}^n : \forall i \in [\ell]\ s(M_i, g) < 1/2\} \\ R_{\tt no}(1^n) &= \{g \in \{0,1\}^n : \exists i \in [\ell]\ s(M_i, g) \geq 2/3\}. \end{aligned}$$

Then, $\overline{R} = (R_{\tt yes}, R_{\tt no})$ is a unary PPT-search problem (according to Definition 2.5) *and $R_{\tt yes}(1^n) \neq \emptyset$ for every $n \in \{2^\ell : \ell \in \mathbb{N}\}$.*

Note that $g \notin R_{\tt no}(1^n)$ means that for every $i \in [\ell]$ it holds that $s(M_i, g) < 2/3$, which means that none of the relevant machines satisfies the learning requirement for length ℓ. Recall that solving a PPT-search problem according to Definition 2.5 means that if $R_{\tt yes}(1^n) \neq \emptyset$, then the solver outputs an element of $\{0,1\}^n \setminus R_{\tt no}(1^n)$.

Proof: Deciding whether a given g is in $R_{\tt yes}(1^n)$ or in $R_{\tt no}(1^n)$ reduces to estimating the success probability of each of the first $\log_2 n$ machine (up to accuracy of 0.01). Showing that $\overline{R}$ can be efficiently solved (and that $R_{\tt yes}(1^n) \neq \emptyset$) reduces to establishing that a random g is in $R_{\tt yes}(1^n)$, with high probability.[15] $\square$

Combining Claim 4.2.1 with our hypothesis (i.e., Statement 1), we obtain a polynomial-time pseudodeterministic algorithm A that, on input 1^{2^ℓ}, outputs (w.v.h.p) a canonical $f_\ell \notin R_{\tt no}(1^{2^\ell})$. Combining the various f_ℓ's, we obtain a function $f : \{0,1\}^* \to \{0,1\}$ that is not learnable in the sense of Statement 2 (i.e., the relevant algorithms do not learn f from membership queries with accuracy 0.99). Lastly, we note that the problem of computing f is in $\mathcal{BPE}$; specifically, on input x, we find the truth-table of $f_{|x|}$, and return the relevant bit, where finding $f_{|x|}$ is done in probabilistic poly$(2^{|x|})$-time. This completes the first part of the proof (i.e., proving that Statement 1 implies Statement 2).

Proving that Statement 2 implies Statement 1: From hardness of learning to pseudodeterminism. As a warm-up, we shall first prove that Statement 2 implies a special case of Statement 2, which asserts a pseudodeterministic polynomial-time algorithm for every *simple* unary PPT-search problem. Recall that a simple PPT-search problem refers to a single relation R and solving it means finding an element of $R(x)$ when given $x \in S_R$. (In contrast, a general PPT-search problem (under Definition 2.5) refers to a pair of non-intersecting relations.)

We again start with an overview. This proof follows the hardness-to-randomness paradigm, but our starting hardness hypothesis is weaker than usual (i.e., the hard problem is in $\mathcal{BPE}$ rather than in $\mathcal{E}$)[16] and so is our conclusion (i.e., pseudodeterministic algorithms for search problems rather

[15] Actually, what one shows is that any device D that makes at most $2^{\ell/2}$ queries to a random $g : \{0,1\}^\ell \to \{0,1\}$ function cannot predict the value of the function on other points. Indeed, the success probability of a randomized device, when querying a random function, is upper-bounded by its success probability on the best choice of its coins. We stress that the only restriction made on the device is an upper bound on the number of queries to the random function.

[16] The fact that hardness refers to learning relatively small circuits rather than to their non-existence is also weaker than the typical hardness hypothesis.

than deterministic algorithms for them). Once we obtain an adequate pseudorandom generator, we invoke the original PPT solver F for R on all possible outcomes of the generator (which are used as random coins for F), test which of these invocations yields a valid solution (using the ppt verification algorithm for R) and output the lexicographically-first valid solution. Hence, we obtain a pseudodeterministic solver, whose error probability is due to the negligible error probability of the probabilistic computation of the hard function and of the probabilistic verification procedure. (We stress that since we are dealing with simple PPT-search problem in this warm-up, we can easily distinguish valid from invalid solution (w.v.h.p.).) Details follow.

Our starting point is the following version of the hardness-to-randomness paradigm. It is obtained by combining the NW-generator [24] with the derandomized direct-product encoding [19] (for a standard proof see, e.g., [6, Thm. A.6]).[17]

Claim 4.2.2 (a hardness to pseudorandomness result): *For every $\epsilon > 0$ there exist a constant $\alpha \in (0, \epsilon)$ and an exponential-time oracle machine G that constitutes a pseudorandom generator in the following sense.*

1. *On input an $O(\ell)$-bit long seed, G outputs an $2^{\alpha \cdot \ell}$-bit sequence.*
2. *If f is hard in the sense of Statement 2* (i.e., f_ℓ is hard to learn with $2^{\epsilon \cdot \ell}$ queries), *then no probabilistic quadratic-time algorithm can distinguish between random output sequences of G^{f_ℓ} and uniformly distributed strings of the same length* (with gap at least $2^{-\alpha \cdot \ell}$).

   Moreover, the indistinguishability claim holds even if the algorithm is given an $\epsilon \cdot \ell$-bit long advice.

Furthermore, G only makes queries of length ℓ; that is, it only queries f_ℓ.

Now, let $R \subseteq \{1\}^* \times \{0,1\}^*$ be a simple unary PPT-search problem, and let F and V denote its solver and its solution-verification algorithm, respectively. By using error reduction, we may assume that the error probability of both F and V is negligible (in n)[18], and let $p(n)$ be a polynomial upper-bounding the running time of F and V in terms of n (where 1^n is the input to F and $(1^n, \cdot)$ is the input to V). Let F_r denote the residual deterministic algorithm obtained when fixing the randomness of F to r. Using these algorithms along with G^{f_ℓ} such that G is as in Claim 4.2.2 (with $\alpha \in (0, \epsilon)$) and f is as guaranteed by Statement 2, we consider the following algorithm.

Algorithm 4.2.3 (pseudodeterministic algorithms for simple unary PPT-search problems): *Let F, V, G and $f \in \mathcal{BPE}$ be as above. Then, on input 1^n, letting $\ell = O(\log n)$ and $k = O(\ell)$ such that $2^{\alpha \ell} = p(n)$, proceed as follows:*

1. *Compute the value of f on all ℓ-bit long strings, thus obtaining the truth-table of f_ℓ.*

   This is done by employing error-reduction to the probabilistic exponential-time algorithm that computes f. (In the analysis we may assume for simplicity that f_ℓ was correctly reconstructed.)

[17] The "robustness to short advice" is not standard, but it can be proved by observing that in the reconstruction procedure, which underlies the standard proof-by-contradiction, we can try all possible advice and test the reconstructed circuits that we obtain.

[18] Recall that $\mu : \mathbb{N} \to \mathbb{N}$ is negligible if for every positive polynomial p and all sufficiently large n it holds that $\mu(n) < 1/p(n)$.

2. *Invoke G^{f_ℓ} on all k-bit long seeds, and let $S \leftarrow \{G^{f_\ell}(s) : s \in \{0,1\}^k\}$.*

3. *For every $r \in S$, compute $y_r \leftarrow F_r(1^n)$. Next, invoking V once on* (1^n and) *each y_r, let S' denote the set of solutions accepted by V; that is, w.v.h.p., $S' = \{y_r : r \in S\} \cap R(1^n)$.*

If $S' \neq \emptyset$, then output the lexicographically-first string in S'. Otherwise, output $\perp$.

Each of the foregoing steps runs in $\exp(O(\ell))$-time, where $\ell = O(\log n)$, and it follows that Algorithm 4.2.3 runs in polynomial-time. We stress that the only probabilistic steps taken by Algorithm 4.2.3 are the computation of f_ℓ (in Step 1) and the invocations of V, which is probabilistic, in Step 3. Recall that f_ℓ is correctly computed (w.v.h.p), whereas the number of invocations of V is polynomial in n and its error probability is negligible (in terms of n). It follows that Algorithm 4.2.3 (almost) always yields the same answer, which means that it is pseudodeterministic.

It is left to show that if $R(1^n) \neq \emptyset$, then (w.v.h.p.) $S' \neq \emptyset$, which means that Algorithm 4.2.3 solves the search problem R. Recall that $\Pr_{r \in \{0,1\}^{p(n)}}[F_r(1^n) \in R(1^n)] = 1 - o(1)$, and we will show that $\Pr_{s \in \{0,1\}^k}[F_{G^{f_\ell}(s)}(1^n) \in R(1^n)] \geq 0.7$. The latter fact is proved by observing that otherwise we obtain a quadratic-time algorithm that distinguishes random outputs of G from uniformly distributed strings. Specifically, for $m = p(n) \geq n$, given a tested sequence $r \in \{0,1\}^m$, the distinguisher consists of invoking the residual F_r on input 1^n such that $n \leq m = p(n)$ is efficiently determined by m, and outputting $V(1^n, F_r(1^n))$. Note that the running time of this distinguisher is $\widetilde{O}(p(n)) < m^2$.

The pseudodeterministic feature of Algorithm 4.2.3 relies on the fact that we are dealing with a simple (unary) PPT-search problem. Specifically, in this case it is easy to distinguish valid solutions from invalid ones. In contrast, when dealing with a general (unary) PPT-search problem $\overline{R} = (R_{\tt yes}, R_{\tt no})$, we need to take decisions also regarding in-between solutions (i.e., strings that are neither in $R_{\tt yes}(1^n)$ nor in $R_{\tt no}(1^n)$). Consequently, a verification algorithm V with error probability η may accept such limbo solutions with arbitrary probability that may be either slightly below $1 - \eta$ or slightly above η. Hence, using V may not yield a decisive answer about limbo solutions. Instead, we run a de-randomized version of V, which allows us to decide deterministically.

An issue at hand is the complexity of the distinguishers that threaten the soundness of this derandomization (or rather the corresponding pseudorandom generator). Such a distinguisher applies V to one of the solutions found by Step 2 of Algorithm 4.2.3. Our issue here is not the complexity of V (which was dealt with above), but rather the complexity of invoking G^{f_ℓ} on k-bit strings, which may exceed the hardness bound for f, because the complexity of G is (only) upper-bounded by $\exp(O(k)) = \exp(O(\ell))$ whereas f_ℓ is hard to learn with $2^{\epsilon\ell}$ queries. Hence, when derandomizing the verification of the foregoing solutions, we invoke G with (a constant times) longer seed (i.e., a seed of length ℓ' such that $2^{\epsilon\ell'} \gg \exp(O(\ell))$ and yet $\ell' = O(\ell)$).

We make a few comments before we present the actual algorithm. Firstly, here, it is not useful to reduce the error probability of verification algorithm V to a negligible level; an error probability of 1/3 is fine. Second, in the current context, we cannot reduce the error probability of the solver F. Consequently, the following pseudodeterministic algorithm will solve the unary PPT-search problem $\overline{R}$ (only) with probability $2/3 - o(1)$, but this is a non-issue because error reduction is available for pseudodeterministic algorithms.

Algorithm 4.2.4 (pseudodeterministic algorithms for general unary PPT-search problems): *Let F, V, G and $f \in \mathcal{BPE}$ be as in Algorithm 4.2.3, except that here F and V refer to algorithms as* *postulated in Definition 2.5 w.r.t PPT-search problem $\overline{R} = (R_{\texttt{yes}}, R_{\texttt{no}})$. Then, on input 1^n, proceed as follows.*

- *Let S be the set of m-bit long outcomes of G^{f_ℓ} as computed in Step 2 of Algorithm 4.2.3. As in Step 3, for every $r \in S$, compute $y_r \leftarrow F_r(1^n)$.*
- *Letting $\ell' = O(k)$ and $k' = O(\ell')$, where $|S| \leq 2^k$, let $R \leftarrow \{G^{f_{\ell'}}(s) : s \in \{0,1\}^{k'}\}$, where $f_{\ell'}$ is obtained as in Step 1 of Algorithm 4.2.3.*
- *For every $r \in S$, include $y_r \in S$ in S' if and only if $|\{r' \in R : V_{r'}(1^n, y_r) = 1\}| > 2^{k'-1}$, where $V_{r'}$ denotes the residual deterministic algorithm obtained when fixing the randomness of V to r'* (or rather the m-bit long prefix of r').[19]

If $S' \neq \emptyset$, then output the lexicographically-first string in S'. Otherwise, output $\perp$.

Again, it is readily verified that Algorithm 4.2.4 runs in polynomial-time and is pseudodeterministic, where the only randomized step here is the constructions of f_ℓ and $f_{\ell'}$, which has negligible error probability. Recall that since $\overline{R}$ is a PPT-search problem, on input 1^n such that $R_{\texttt{yes}}(1^n) \neq \emptyset$, it holds that $\Pr_{r\in\{0,1\}^m}[F_r(1^n) \in R_{\texttt{yes}}(1^n)] \geq 2/3$. Letting $R'_{\texttt{yes}}(1^n) \overset{\text{def}}{=} \{y : \Pr[V(1^n, y) = 1] \geq 0.65\}$, it follows that $\Pr_{r\in S}[F_r(1^n) \in R'_{\texttt{yes}}(1^n)] \geq (2/3) - o(1)$. (Specifically, analogously to the previous analysis, note that otherwise the output of G is distinguishable from the uniform distribution.)[20]

It is left to show that in case $F_r(1^n) \in R'_{\texttt{yes}}(1^n)$ for some $r \in S$, w.v.h.p., it holds that $S' \neq \emptyset$, which means that Algorithm 4.2.4 solves the PPT-search problem $\overline{R}$ (in the sense of Definition 2.5, which means that it outputs a solution that is not in $R_{\texttt{no}}(1^n)$). This amounts to showing that for every $r \in S$ it holds that if $y_r \in R'_{\texttt{yes}}(1^n)$ then $\Pr_{s\in\{0,1\}^{k'}}[V_{G^{f_{\ell'}}(s)}(1^n, y_r) = 1] > 1/2$, and if $y_r \in R_{\texttt{no}}(1^n)$ then $\Pr_{s\in\{0,1\}^{k'}}[V_{G^{f_{\ell'}}(s)}(1^n, y_r) = 1] \leq 1/2$. In other words, the claim is that every $r \in S$ such that $y_r \in (R'_{\texttt{yes}}(1^n) \cup R_{\texttt{no}}(1^n))$ it holds that

$$\left|\Pr_{s\in\{0,1\}^{k'}}[V_{G^{f_{\ell'}}(s)}(1^n, y_r) = 1] - \Pr_{r'\in\{0,1\}^m}[V_{r'}(1^n, y_r) = 1]\right| = o(1). \tag{1}$$

The latter fact is proved by observing that otherwise we obtain a quadratic-time algorithm that distinguishes random outputs of G from uniformly distributed strings. Actually, we use a slightly non-uniform distinguisher that utilizes an advice of length k that specifies an $r \in S$ (via the corresponding $s \in \{0,1\}^k$ (i.e., $r = G^{f_{\ell'}}(s)$)) such that $y_r \in (R'_{\texttt{yes}}(1^n) \cup R_{\texttt{no}}(1^n))$ violates Eq. (1). That is, given a tested sequence $\rho \in \{0,1\}^m$ and the advice $s \in \{0,1\}^k$, the distinguisher first computes $r = G^{f_\ell}(s)$, then computes $y_r = F_r(1^n)$, and finally outputs $V_\rho(1^n, y_r)$. Note that the running time of this distinguisher is dominated by the running time of $G^{f_\ell}(s)$, which is $\exp(O(k)) < 2^{\epsilon\ell'}$, whereas the advice string has length $k < \epsilon \cdot \ell'$. This contradicts the moreover clause of Claim 4.2.2. ■

[19] Recall that $m = p(n)$ denotes the running-time of V, which is used as the length of its random-tape. However, the length of G's output on a k'-bit long seed is $2^{\epsilon\ell'} \gg 2^{\epsilon\ell} = m$. From this point on, for simplicity, we ignore this inaccuracy.

[20] Given a tested sequence $r \in \{0,1\}^m$, the distinguisher consists of invoking the residual F_r on input 1^n, obtaining $y \leftarrow F_r(1^n)$, estimating $\Pr[V(1^n, y) = 1]$ by a few invocations of V, and outputting 1 if and only if the estimate exceeds 0.66. Suppose that the estimation is correct up to a deviation of 0.005, with error probability $\mu = o(1)$. Recalling that $y \in R_{\texttt{yes}}(1^n)$ implies that $\Pr[V(1^n, y) = 1] \geq 2/3$, it follows that, on a uniformly distributed $r \in \{0,1\}^m$, the distinguisher will output 1 with probability at least $(2/3) \cdot (1 - \mu) > (2/3) - \mu$. On the other hand, if $\Pr[F_{G^{f_\ell}(1^n)} \in R'_{\texttt{yes}}(1^n)] < (2/3) - 3\mu$, then the distinguisher will output 1 on a random output of G^{f_ℓ} with probability smaller than $(2/3) - 3\mu + \mu = (2/3) - 2\mu$.

An alternative proof of the second part of Theorem 4.2. This comment refers to the actual proof that Statement 2 (i.e., hardness of learning) implies Statement 1 (i.e., pseudodeterminism), while leaving the warm-up (which is confined to *simple* PPT-search problems) intact. Recall that the foregoing proof relied on the hypothesis that the pseudorandom generation is robust against distinguishers that obtain a short non-uniform advice string, and relies on the moreover clause of Claim 4.2.2. The alternative proof, presented in the appendix, only relies on a pseudorandom generation that is robust against uniform distinguishers. Unfortunately, this proof is slightly more complicated (when arguing that using pseudorandom sequences in the verification process maintains the behavior of truly random sequences).

4.2 A connection to the generation of pseudorandom sets

We start by defining the notion of pseudorandomness that we use. Recall that the most popular approach, which is also followed in Section 4.1, views pseudorandom generators as stretching a short random seed into a longer "pseudorandom" sequence (cf., e.g., [10, Chap. 8]). In contrast, we consider algorithms that generate a set of strings such that the uniform distribution over this set is pseudorandom. (We mention that this alternative perspective is most popular when discussing hitting-set generators (see, e.g., [1, 16]) and when studying superfast and "free lunch" derandomization (see, e.g., [7]).) The notion of pseudorandomness we use is a uniform version of the standard notion used in the context of derandomization. Specifically, we shall use the following definition.

Definition 4.3 (pseudorandom sets): *Let $M_1, M_2, ...$ be an enumeration of Turing machines modified to run in quadratic-time. A set $S \subseteq \{0,1\}^*$ is called* pseudorandom *if, for all sufficiently large $n \in \mathbb{N}$, the set $S_n \stackrel{\rm def}{=} S \cap \{0,1\}^n$ satisfies the following condition for every $i \in [n]$.*

$$\left|\Pr_{s\in S_n}[M_i(s)=1] - \Pr_{u\in\{0,1\}^n}[M_i(u)=1]\right| < \frac{1}{n} \qquad (2)$$

While a uniform notion of pseudorandomness requires considering an unbounded number of machines, our definition "sneaks in" a bounded amount of non-uniformity; specifically, $\log_2 n$ bits. Needless to say, the fact that we used n machines and an indistinguishability gap of $1/n$ is an arbitrary choice (i.e., any fixed polynomial would have had a similar utility for us). Likewise, our use of quadratic-time rather than any fixed polynomial time is immaterial.[21]

Intuitively, we say that a PPT algorithm constructs a pseudorandom set if, on input 1^n, with probability at least $2/3$, it outputs S_n as in Definition 4.3. Such an algorithm corresponds to a pseudodeterministic polynomial-time algorithm that solves the following PPT-search problem (which is phrased in terms of Definition 2.3).[22]

Definition 4.4 (finding a pseudorandom set as a PPT-search problem): *Let $M_1, M_2, ...$ be as in Definition 4.3 and $m(n) = \widetilde{O}(n^2)$. Then, consider the quality function $q_{\tt pr} : \{1\}^* \times \{0,1\}^* \to [0,1]$ such that*

$$q_{\tt pr}(1^n, (r_1, ..., r_{m(n)})) \stackrel{\rm def}{=} 1 - \max_{i\in[n]}\left\{\max\left(\frac{1}{n}\ ,\ \left|\Pr_{j\in[m(n)]}[M_i(r_j)=1] - \Pr_{u\in\{0,1\}^n}[M_i(u)=1]\right|\right)\right\}$$

[21]The reason we use quadratic-time instead of linear-time is that various manipulations that we wish to apply (e.g., emulating the execution of a given algorithm) require almost linear-time on the most popular models of computation.

[22]Indeed, the following definition as well as Theorem 4.5 can be easily formulated and analyzed also in terms of Definition 2.5.

where the r_j's are n-bit strings.

Note that $q_{\tt pr}(1^n,(r_1,...,r_{m(n)})) = 1-(1/n)$ if and only if the sequence $(r_1,...,r_{m(n)})$ forms a pseudorandom set.[23] It can be easily verified that $q_{\tt pr}$ can be approximated in PPT, and that a uniformly distributed sequence $\overline{r} \in \{0,1\}^{m(n)\cdot n}$ satisfies $q_{\tt pr}(1^n,\overline{r}) = 1-(1/n)$, with probability $1-n^{-\omega(1)}$. Thus, $q_{\tt pr}$ constitutes a unary PPT-search problem.

As stated upfront, a polynomial-time pseudodeterministic algorithm that solves the PPT-search problem of Definition 4.4 corresponds to a PPT construction of a pseudorandom-set. We show that such an algorithm yields polynomial-time pseudodeterministic algorithms for all unary PPT-search problems. Intuitively, this means that the search problem formulated in Definition 4.4 is "universal" for the class of unary PPT-search problems.

Theorem 4.5 (constructing pseudorandom-sets implies pseudodeterministic solvers for all unary PPT-search problems): *Suppose that there is a pseudodeterministic polynomial-time algorithm that solves the problem of Definition 4.4. Then, every unary PPT-search problem can be solved by a pseudodeterministic polynomial-time algorithm.*

Proof: The proof follows the strategy of the relevant direction of the proof of Theorem 4.2, except that here our starting point is already a pseudorandom generator (or rather a pseudorandom-set). Hence, we proceed directly to presenting a version of Algorithm 4.2.4.

Consider an arbitrary unary PPT-search problem captured by the quality function q, and let F and Q be the corresponding finding and quality-evaluation algorithms. Recall that F (resp., Q) is invoked on input $(1^n,1^t)$ (resp., $(1^n,y,1^t)$), where $1/t$ is the desired approximation level, and let $p(n,t) = \text{poly}(n,t) = \omega(n+t)$ denote its randomness and time complexity. Analogously to Algorithm 4.2.4, we let F_r (resp., $Q_{r'}$) denote the residual deterministic algorithm obtained from F (resp., Q) when fixing its randomness to r (resp., r').

Algorithm 4.5.1 (pseudodeterministic algorithms for unary PPT-search problems captured by q): *Let F and Q be as above, and S be a pseudorandom-set that can be constructed by a pseudodeterministic algorithm. Specifically, suppose* (w.l.o.g.) *that S_n is generated in time $O(|S_n|^2)$. Then, on input $(1^n,1^t)$, proceed as follows.*

1. *Setting $p = p(n,3t)$, construct $S_p = S \cap \{0,1\}^p$.*

   Specifically, the solver of the problem formulated in Definition 4.4 is invoked with deviation parameter $1/p$; hence, w.v.h.p., the solution S_p found has quality at least $1-(2/p)$.

2. *For every $r \in S_p$, compute $y_r \leftarrow F_r(1^n,1^{3t})$.*

3. *Setting $t' = |S_p|^2$ and $p' = p(n,t')$, construct $S_{p'} = S \cap \{0,1\}^{p'}$.*

   Specifically, the solver of the problem formulated in Definition 4.4 is invoked with deviation parameter $1/p'$; hence, w.v.h.p., the solution $S_{p'}$ found has quality at least $1-(2/p')$.

[23]Due to the internal maximization, we have $q'_{\rm pr}(1^n) \stackrel{\rm def}{=} \max_{\overline{r}}\{q_{\rm pr}(1^n,\overline{r})\} = 1-(1/n)$ (rather than possibly $q'_{\rm pr}(1^n)=1$, which could create technical difficulties). We mention that there exists a distribution Y with support of size at most $n+1$ such that $\Pr[M_i(Y)=1] = \Pr_{u\in\{0,1\}^n}[M_i(u)=1]$ for every $i \in [n]$ (see [15]), but it is unclear how close these distributions can be to uniform over a set of size $\widetilde{O}(n^2)$.

4. For every $r \in S_p$, declare y_r as a potential solution *if and only if*

$$|\{r' \in S_{p'} : Q_{r'}(1^n, y_r, 1^{8t}) > 1 - (1/n) - (1/2t)\}| > |S_{p'}|/2.$$

If the set of potential solutions is not empty, then output the lexicographically-first potential solution. Otherwise, output $\perp$.

It is readily verified that Algorithm 4.5.1 runs in polynomial-time and is pseudodeterministic, where the only randomized steps in Algorithm 4.5.1 are the constructions of the sets S_p and $S_{p'}$, which have negligible error probability. Recall that (by error reduction) we may assume that

$$\Pr_{r\in\{0,1\}^p}[q(1^n, F_r(1^n, 1^{3t})) > 1 - (1/n) - (1/3t)] = 1 - o(1),$$

whereas the pseudorandomness of the set S_p (w.r.t to a potential distinguisher that combines F and Q) implies that

$$\Pr_{r\in S_n}[q(1^n, F_r(1^n, 1^{3t})) > 1 - (1/n) - (1/3t) - (2/p)] = 1 - o(1).$$

Recalling that for every y it holds that

$$\Pr_{r'\in\{0,1\}^{p'}}[Q_{r'}(1^n, y, 1^{8t}) = q(1^n, y_r) \pm (1/8t)] = 1 - o(1/|S_p|),$$

we wish to show that for every $r \in S_p$ it holds that

$$\Pr_{r'\in S_{p'}}[Q_{r'}(1^n, y_r, 1^{8t}) = q(1^n, y_r) \pm (1/7t)] = 1 - o(1/|S_p|).$$

The proof, which relies on the pseudorandomness of the set $S_{p'}$, is analogous to the one given at the end of the proof of Theorem 4.2. Again, the crucial point is that p' is determined to be large enough so to dominate the running time of the algorithm constructing S_p, which in turn dominates the running times of F and Q. (When using the main argument, which is slightly non-uniform, note that Definition 4.4, just as Definition 4.3, accounts for a logarithmic amount of non-uniformity that the main argument uses.)[24] Lastly, using $(1/3t) + (2/p) + (1/7t) < 1/2t$ and $(1/t) - (1/7t) > 1/2t$, we conclude that the set of declared potential solutions is non-empty and that it contains only y_r's that satisfy $q(1^n, y_r) > 1 - (1/n) - (1/t)$. ■

A binary version of Theorem 4.5. The proof of Theorem 4.5 can be easily extended so that it yields pseudodeterministic polynomial-time algorithms for all PPT-search problems based on the construction of *targeted* pseudorandom-sets. Loosely speaking, in analogy to targeted pseudorandom generators, defined in [12] (following [11, Sec. 4.4]), here the construction is given a circuit $C : \{0,1\}^n \to \{0,1\}$ and outputs a set S_C such that

$$\left|\Pr_{s\in S_C}[C(s)=1] - \Pr_{u\in\{0,1\}^n}[C(u)=1]\right| < \frac{1}{n}$$

The corresponding PPT-search problem is formulated analogously to Definition 4.4. We consider the quality function $q_{\tt pr} : \{0,1\}^* \times \{0,1\}^* \to [0,1]$ such that

$$q_{\tt pr}(C, (r_1, ..., r_{m(n)})) \stackrel{\rm def}{=} 1 - \max\left(\frac{1}{n}\ ,\ \left|\Pr_{j\in[m(n)]}[C(r_j)=1] - \Pr_{u\in\{0,1\}^n}[C(u)=1]\right|\right) \qquad (3)$$

[24]The relevant algorithms as well as an index in the set S_p can be encoded in less that $\log_2 p'$ bits. In any case, note that we can set p' to be an arbitrary polynomial in $|S_p| = \text{poly}(p)$.

where n is the input length of C and the y_j's are n-bit strings. The analogue of Theorem 4.5 asserts that *if there is a pseudodeterministic polynomial-time algorithm that solves the search problem of* Eq. (3)*, then every PPT-search problem can be solved by a pseudodeterministic polynomial-time algorithm.*

4.3 Implications to $\mathcal{BPP}$-extensions of pr$\mathcal{BPP}$ and BPTime hierarchies

The key observation in this subsection is that the existence of pseudodeterministic algorithms for all unary PPT-search problems implies that every unary problem in pr$\mathcal{BPP}$ can be extended to a unary set in $\mathcal{BPP}$. This is a unary problem analogous of one direction of Theorem 3.5.

Theorem 4.6 (from pseudodeterminism to $\mathcal{BPP}$-extensions of pr$\mathcal{BPP}$, the unary version): *Suppose that every unary PPT-search problem can be solved by a polynomial-time pseudodeterministic algorithm. Then, every unary problem in* pr$\mathcal{BPP}$ *has a unary extension in* $\mathcal{BPP}$.

Proof: We shall present two proofs, one per each of the two definitions of PPT-search problems presented in Section 2. In both cases, for any decisional problem $\Pi = (\Pi_{\tt yes}, \Pi_{\tt no})$ in pr$\mathcal{BPP}$ such that $\Pi_{\tt yes} \cup \Pi_{\tt no} \subseteq \{1\}^*$, we shall show that Π has an extension $(S, \overline{S})$ such that $S \in \mathcal{BPP}$. (We can make S unary by redefining $S \leftarrow S \cap \{1\}^*$.)

We first prove this conclusion when the hypothesis (i.e., the existence of polynomial-time pseudodeterministic algorithm for every unary PPT-search problem) is formulated with respect to Definition 2.3. Consider a ("$\mathcal{BPP}$"-type) algorithm A that solves Π, and let $p(n) \stackrel{\rm def}{=} \Pr[A(1^n)=1]$, while noting that $p(n) \geq 2/3$ if $1^n \in \Pi_{\tt yes}$ and $p(n) \leq 1/3$ if $1^n \in \Pi_{\tt no}$. Now, define $q : \{0,1\}^* \times \{0,1\}^* \to [0,1]$ such that $q(x,v) \stackrel{\rm def}{=} 1 - |p(|x|) - v|$, while noting that $q(x,v) \approx 1$ if and only if $v \approx p(|x|)$. Let F be a PPT algorithm that solves q; that is, $\Pr[q(x, F(x,1^t)) \geq 1 - (1/t)] \geq 2/3$ (equiv., $\Pr[|F(x,1^t) - p(|x|)| \leq (1/t)] \geq 2/3$). Fixing $t = 7$, and using the ("pseudodeterminism") hypothesis, we consider the corresponding pseudodeterministic polynomial-time algorithm F', and infer that, for every $x \in \{0,1\}^*$, there exists a unique value v_x such that $\Pr[F'(x) = v_x] \geq 2/3$, whereas $v_x \in [p(|x|) \pm 1/7]$ holds. Hence, $x \in \Pi_{\tt yes}$ implies $v_x \geq \frac{2}{3} - \frac{1}{7} > 0.5$ and $x \in \Pi_{\tt no}$ implies $v_x \leq \frac{1}{3} + \frac{1}{7} < 0.5$. Letting $S \stackrel{\rm def}{=} \{x \in \{0,1\}^* : v_x > 0.5\}$, it follows that $S \in \mathcal{BPP}$ (by using F'), whereas $(S, \overline{S})$ is an extension of Π.

The proof when the hypothesis refers to PPT-search problems as in Definition 2.5 is very similar. We define a search problem $\overline{R} = (R_{\tt yes}, R_{\tt no})$ wherein instances are unary and solutions are real values, and $v \in R_{\tt yes}(1^n)$ if $|v - p(1^n)| \leq 1/10$ and $v \in R_{\tt no}(1^n)$ if $|v - p(1^n)| > 1/6$. The problem $\overline{R}$ is in PPT-search, since we can use A both for verifying solutions and for finding solutions in $R_{\tt yes}(1^n)$. Hence, the hypothesized pseudodeterministic algorithm F' finds, for every input 1^n, a value $v(n)$ such that $v(n) \notin R_{\tt no}(1^n)$, meaning that $|v(n) - p(1^n)| \leq 1/6$. The probabilistic algorithm solving an extension of Π accepts 1^n if and only if the output of $F'(1^n)$ is larger than $1/2$ (and its analysis is identical to the one above). ■

Theorem 4.7 (corollary of Theorem 4.6): *Suppose that every unary PPT-search problem can be solved by a polynomial-time pseudodeterministic algorithm. Then, for every polynomial p there exists a polynomial q such that* BPTime$(p) \subset$ BPTime(q).

We note that Theorem 4.7 follows from [23, Sec. 5].

Proof: Our starting point is a hierarchy theorem for pr$\mathcal{BPP}$. Specifically, by Theorem 2.2, for every polynomial p, there exists a unary decisional promise problem $\Pi = (\Pi_{\texttt{yes}}, \Pi_{\texttt{no}})$ in prBPTime$(\widetilde{O}(p)) \setminus$ prBPTime(p). Applying Theorem 4.6, it follows that Π has an extension in $\mathcal{BPP}$ (i.e., it is in BPTime(q) for some polynomial q that depends on $\widetilde{O}(p)$). Since this extension cannot be in BPTime(p), the claim follows. ■

4.4 On unary PPT-search problems with deterministic verification

Next, we show that even an algorithm meeting more relaxed requirements than the one hypothesized in Theorem 4.7 would have new and interesting implications. Specifically, in the results presented next, the hypothesized pseudodeterministic algorithm only needs to solve search problems in which solutions are verifiable deterministically. The main technical observation is captured by the following statement:

Claim 4.8 (using pseudodeterminism for extending promise RTime-problems to sets): *A binary relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is called* adequate *if it is recognizable in deterministic polynomial-time and there exists a time-constructible function $\ell : \mathbb{N} \to \mathbb{N}$ such that $(x, y) \in R$ implies that $|y| = \ell(|x|) \geq |x|$. For an adequate relation R, we define $\Pi_R = (\Pi_{\texttt{yes}}, \Pi_{\texttt{no}})$ such that $\Pi_{\texttt{yes}} = \{x : |R(x)| \geq 2^{\ell(x)-1}\}$ and $\Pi_{\texttt{no}} = \{x : R(x) = \emptyset\}$.*

***hypothesis*:** *Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be an adequate relation and suppose that there exists a* (poly$\circ\ell$)*-time pseudodeterministic algorithm A such that if $x \in \Pi_{\texttt{yes}}$, then* $\Pr[A(x) \in R(x)] \geq 2/3$.

***conclusion*:** *The problem Π_R has an extension in* BPTime(poly $\circ\, \ell$)*. That is, there exists a set $S \in$* BPTime(poly $\circ\, \ell$) *such that $S \supseteq \{x : |R(x)| \geq 2^{\ell(|x|)-1}\}$ and $S \subseteq \{x : R(x) \neq \emptyset\}$.*

We stress that saying that A is *pseudodeterministic* means that for every x there exists a string c_x such that $\Pr[A(x) = c_x] \geq 2/3$. We shall actually use this condition for instances that violate the promise of Π (i.e., instances x such that $0 < |R(x)| < 2^{\ell(|x|)-1}$).

Proof: We consider a decision procedure D that on input x, invokes the pseudodeterministic algorithm A on x, obtaining $y \leftarrow A(x)$, and outputs 1 if and only if $(x, y) \in R$. Then, the following three facts hold for every $x \in \{0,1\}^*$.

Fact 1: If $|R(x)| \geq 2^{\ell(|x|)-1}$, then $\Pr[D(x)=1] \geq 2/3$.

This holds because in this case $\Pr[A(x) \in R(x)] \geq 2/3$.

Fact 2: If $R(x) = \emptyset$, then $\Pr[D(x)=0] = 1$.

This holds because in this case $\Pr[A(x) \in R(x)] = 0$.

Fact 3: Either $\Pr[D(x)=1] \geq 2/3$ or $\Pr[D(x)=0] \geq 2/3$.

This holds because A is pseudodeterministic; that is, $\Pr[A(x)=c_x] \geq 2/3$ for c_x that depends on x only. Furthermore, letting $\chi(x) = 1$ if $c_x \in R(x)$ and $\chi(x) = 0$ otherwise, it holds that $\Pr[D(x)=\chi(x)] \geq \Pr[A(x)=c_x]$.

Consider the set $S \overset{\text{def}}{=} \{x : \Pr[D(x)=1] \geq 2/3\}$, and note that $S \in \mathcal{BPP}$ (by Fact 3). The claim follows by observing that Facts 1 and 2 imply that S is sandwiched between the sets $\{x : |R(x)| \geq 2^{\ell(|x|)-1}\}$ and $\{x : R(x) \neq \emptyset\}$. ■

Implications for Rtime. Applying Claim 4.8 to the witness relation of a unary promise problem in pr$\mathcal{RP}$, we obtain a unary set in $\mathcal{BPP}$ that extends the original promise problem. In particular, we obtain the following result.

Theorem 4.9 (an RTime vs BPTime separation): *Suppose that for every adequate unary relation $R \subseteq \{(1^n, y) : y \in \{0,1\}^{\ell(n)}\}$ there exists a* (poly $\circ\, \ell$)*-time pseudodeterministic algorithm A such that if $R(1^n) \geq 2^{\ell(n)-1}$, then* $\Pr[A(1^n) \in R(1^n)] \geq 2/3$. *Then, for every constant $k \in \mathbb{N}$ there exists $K > k$ such that* BPTime(n^K) *is not contained in* RTime(n^k).

The conclusion of Theorem 4.9 implies that either RTime$(n^K) \neq$ BPTime(n^K) or RTime(n^K) is not contained in RTime(n^k). More generally, *for every $K' \geq K$, either* RTime$(n^{K'})$ *does not contain* BPTime(n^K) *or* RTime$(n^{K'})$ *is not contained in* RTime(n^k).

We note that the search problems solved in the hypothesis of Theorem 4.9 can be cast as unary PPT-search problems as in Definition 2.5, by considering $\overline{R} = (R_{\mathtt{yes}}, R_{\mathtt{no}})$ such that $R_{\mathtt{yes}} = \{(1^n, y) \in R : |R(x)| \geq 2^{\ell(n)-1}\}$ and $R_{\mathtt{no}}$ equals the complement of R (i.e., $(1^n, y) \in R_{\mathtt{no}}$ iff $(1^n, y) \notin R$). The key point is that such problems are a subclass of unary PPT-search problems, since we require that R will be recognizable in *deterministic* polynomial time.

Proof: Using the delayed diagonalization, which underlies the proof of Theorem 2.2, for every constant $k \in \mathbb{N}$, there exists a unary promise problem $\Pi = (\Pi_{\mathtt{yes}}, \Pi_{\mathtt{no}})$ in prRTime$(n^{k+1}) \setminus$ co-prRTime(n^k). Letting $m = n^{k+1}$, there exists a unary relation R that is recognizable in almost-linear (deterministic) time such that

1. For every $1^n \in \Pi_{\mathtt{yes}}$, it holds that $|\{r \in \{0,1\}^m : (1^n, r) \in R\}| \geq 2^{m-1}$.
2. For every $1^n \in \Pi_{\mathtt{no}}$ and every $r \in \{0,1\}^m$, it holds that $(1^n, r) \notin R$.
3. For every $x \in \{0,1\}^* \setminus \{1\}^*$ and every $r \in \{0,1\}^m$, it holds that $(x, r) \notin R$.

Applying Claim 4.8 to $R \subseteq \{1\}^* \times \{0,1\}^*$, we obtain a unary set $S \in \mathcal{BPP}$ that extends Π; that is, S is sandwiched between $\Pi_{\mathtt{yes}}$ and $\{1\}^* \setminus \Pi_{\mathtt{no}}$. One the other hand, since S extends Π, it must hold that $S \notin$ co-RTime(n^k). Using $\{1\}^* \setminus S$, the theorem follows. ■

Implications for BPtime. Recall that the hypothesis that pr$\mathcal{RP} \subseteq$ prDTime(T) implies that $\mathcal{BPP} \subseteq$ DTime$(T(\text{poly}(T)))$. Using the hypothesis of Claim 4.8, we improve the conclusion to $\mathcal{BPP} \subseteq$ DTime(poly(T)). We note that the stronger conclusion is known under the hypothesis that hitting sets can be constructed in time T (which implies pr$\mathcal{RP} \subseteq$ prDTime(T)).

Theorem 4.10 (a BPTime hierarchy): *Suppose that* prRTime$(n^2) \subseteq$ prDTime(T).[25] *Then, under the hypothesis of Theorem 4.9, for every constant $k \in \mathbb{N}$ there exists $K > k$ such that* BPTime(poly$(T(n^K))$) *is not contained in* BPTime(n^k).

[25]Recall that prRTime$(n^2) \subseteq$ prDTime(T) implies that for every nice function $f(n) \geq n$ it holds that prRTime$(f(n)^2) \subseteq$ prDTime$(T(n^2))$; in particular, when $T(n) = \text{poly}(n)$ it implies $\mathcal{RP} = \mathcal{P}$. The reason we use quadratic-time (i.e., n^2) instead of linear-time (i.e., $O(n)$) is that various manipulations that we wish to apply (e.g., emulating the execution of a given algorithm) require almost linear-time on the most popular models of computation.

Proof: Using Theorem 2.2, for every constant $k \in \mathbb{N}$, there exists a unary promise problem $\Pi = (\Pi_{\texttt{yes}}, \Pi_{\texttt{no}})$ in prBPTime$(n^{k+1}) \setminus$ prBPTime(n^k). Recall that several of the standard proofs establishing $\mathcal{BPP} \subseteq \mathcal{PH}$ (e.g., [25, 20, 17]), when applied to Π, imply the existence of a polynomial $m = m(n) > n$ and a (deterministic) polynomial-time recognizable relation R such that

1. For every $1^n \in \Pi_{\texttt{yes}}$, it holds that
$$\Pr_{r\in\{0,1\}^m}[\forall s\in\{0,1\}^m \;\; (1^n, rs) \in R] \;\geq\; 1/2.$$

2. For every $1^n \in \Pi_{\texttt{no}}$ and every $r \in \{0,1\}^m$, it holds that
$$\Pr_{s\in\{0,1\}^m}[(1^n, rs) \in R] \;\leq\; 1/2.$$

Let $\Pi' = (\Pi'_{\texttt{yes}}, \Pi'_{\texttt{no}})$ be such that
$$\begin{aligned}
\Pi'_{\texttt{yes}} &\stackrel{\rm def}{=} \Big\{(1^n, r) : \forall s\in\{0,1\}^{m(n)} \;\; (1^n, rs) \in R\Big\} \\
\Pi'_{\texttt{no}} &\stackrel{\rm def}{=} \Big\{(1^n, r) : \Pr_{s\in\{0,1\}^{m(n)}}[(1^n, rs) \in R] \;\leq\; 1/2\Big\}
\end{aligned}$$
where in both cases $r \in \{0,1\}^{m(n)}$. (Indeed, if $1^n \in \Pi_{\texttt{yes}}$ (resp., $1^n \in \Pi_{\texttt{no}}$), then $\Pr_{r\in\{0,1\}^m}[(1^n, r)\in \Pi'_{\texttt{yes}}] \geq 1/2$ (resp., $(1^n, r)\in\Pi'_{\texttt{no}}$ for every $r \in \{0,1\}^m$).) Then, Π' is in promise-co$\mathcal{RP}$. Furthermore, without loss of generality, the corresponding algorithm runs in quadratic time. Using the hypothesis that prRTime$(q) \subseteq$ prDTime(T), where q is a quadratic function and $T > q$, we infer that Π' is in prDTime(T). Let us denote the corresponding decision procedure by D.

Now, let $R' \stackrel{\rm def}{=} \{(1^n, r) : D(1^n, r) = 1\}$. Noting that $D(1^n, r) = 1$ for every $(1^n, r) \in \Pi'_{\texttt{yes}}$ and $D(1^n, r) = 0$ for every $(1^n, r) \in \Pi'_{\texttt{no}}$, it follows that R' is sandwiched between $\Pi'_{\texttt{yes}}$ and $(\{1\}^* \times \{0,1\}^*) \setminus \Pi'_{\texttt{no}}$. We now consider a padded version of R'; specifically,
$$R'' \stackrel{\rm def}{=} \{(1^n, rp) : (1^n, r)\in R' \;\&\; p\in\{0,1\}^{T(n+m)-(n+m)}\}.$$

Then, R'' is recognizable in almost-linear (deterministic) time, and

1. For every $1^n \in \Pi_{\texttt{yes}}$, it holds that $|R''(1^n)| \geq 2^{T(n+m)-n-1}$, whereas $R''(1^n) \subseteq \{0,1\}^{T(n+m)-n}$.
(This holds because in this case $\Pr_{r\in\{0,1\}^m}[(1^n, r)\in\Pi'_{\texttt{yes}}] \geq 1/2$.)

2. For every $1^n \in \Pi_{\texttt{no}}$, it holds that $R''(1^n) = \emptyset$.
(This holds because in this case $(1^n, r)\in\Pi'_{\texttt{no}}$ for every $r \in \{0,1\}^m$.)

Applying Claim 4.8 to R'' we obtain a unary set $S \in$ BPTime(poly$(T(m(n)^{2)}))$) such that S is sandwiched between $\{1^n : |R''(1^n)| \geq 2^{T(n+m(n))-n-1}\}$ and $\{1^n : R''(1^n) \neq \emptyset\}$, which implies that S is sandwiched between $\Pi_{\texttt{yes}}$ and $\{1\}^* \setminus \Pi_{\texttt{no}}$. One the other hand, since S extends Π, it must hold that $S \notin$ BPTime(n^k). The theorem follows. ■

4.5 On the gaps between the algorithm of [5] and the foregoing hypotheses

Recently, Chen, Lu, Oliveira, Ren, and Santhanam [5] proved the following result, which asserts a polynomial-time pseudodeterministic algorithm for a subclass of unary PPT-search problems:

Theorem 4.11 (infinitely-often polynomial-time pseudodeterministic explicit constructions): *Let $R \subseteq \{1\}^* \times \{0,1\}^*$ be a search problem such that*

1. $|R(1^n) \cap \{0,1\}^n| \geq 2^n/\text{poly}(n)$ *for every* $n \in \mathbb{N}$.
2. *There is a deterministic polynomial-time algorithm V that decides membership in R.*

Then, there exists probabilistic polynomial-time algorithm F and a sequence $\{y_n \in R(1^n)\}_{n\in\mathbb{N}}$ such that the following two conditions hold:

1. *For every $n \in \mathbb{N}$ it holds that* $\Pr[F(1^n) \in \{y_n, \perp\}] \geq 1 - 2^{-n}$, *where $\perp \notin \{0,1\}^*$ is a special symbol.*
2. *For infinitely many $n \in \mathbb{N}$ it holds that* $\Pr[F(1^n) = y_n] \geq 1 - 2^{-n}$.

Recall that search problems as in Theorem 4.11 can be cast as PPT-search problems (see discussion after Theorem 4.9).

The subclass of unary PPT-search problems for which the algorithm of Theorem 4.11 works is indeed broad, but there are gaps between Theorem 4.11 and an ideal result. In particular, there are gaps between the algorithm in Theorem 4.11 and the required hypotheses for Theorems 4.2–4.7 and 4.9–4.10. We stress that pointing out these gaps is intended to *highlight them as interesting open problems*, whose resolution would imply the consequences spelled out in the aforementioned theorems.

First, the algorithm in Theorem 4.11 requires that valid solutions be recognized in *deterministic* polynomial-time, rather allowing a *probabilistic* algorithm (as required in Definitions 2.3 and 2.5). We warn that a superficial impression that Chen *et al.* [5]'s proof of Theorem 4.11 can be easily extended to the probabilistic case is unsound (see discussion below). While this falls short of the hypothesis of Theorems 4.2–4.7 (i.e., an algorithm that solves all unary PPT-search problems), it is not an issue in Theorems 4.9–4.10 (since their hypotheses also refer to deterministic polynomial-time verification of valid solutions).

Second, the algorithm in Theorem 4.11 finds solutions only for infinitely many input lengths. This shortcoming is actually less severe than it might seem: When the input lengths on which the algorithm finds solutions are easily recognizable (as is, essentially, the case in [5] – see Remark 4.12), such an algorithm may still suffice for Theorems 4.9 and 4.10, barring the last gap.

Third, the algorithm in Theorem 4.11 is *pseudodeterministic only on infinitely many input lengths*, rather than having a pseudodeterministic guarantee on all input lengths. We warn that the fact that the algorithm in Theorem 4.11 is 2-pseudodeterministic[26] on all input lengths (i.e., almost always outputs one of two canonical solutions) and that these two solutions are easily distinguishable in not good enough for the hypotheses of Theorems 4.9 and 4.10.

Remark 4.12 (the second shortcoming is less severe than it might seem): *We can obtain a* BPTime *hierarchy from an algorithm that is pseudodeterministic on all input lengths but only*

[26] The notion of multi-pseudodeterministic algorithms was introduced in [13].

solves the search problem R infinitely often, provided that the infinite set $S \subseteq \mathbb{N}$ of input lengths on which the algorithm solves R is efficiently recognizable. This is because such an algorithm allows using input lengths in S for diagonalization. In fact, even more relaxed conditions suffice, such as partitioning $\mathbb{N}$ into efficiently recognizable intervals and guaranteeing that in each interval there is an input length on which the algorithm solves R. This relaxed condition is satisfied by the algorithm in [5], and thus the crucial gap is that their algorithm may not be pseudodeterministic on input lengths outside S.

Can the proof of Theorem 4.11 be extended to the probabilistic case? Recall that Theorem 4.11 refers to a unary search problem R that can be decided in deterministic polynomial-time and that we believe that the impression that the proof in [5] can be easily extended to the probabilistic case is unsound.

On a very high level, the issue is that the algorithm A presented in [5] uses the decision procedure D_R for R in a non-black-box manner. Specifically, A uses the intermediate results of the computation of D_R, and in case D_R is probabilistic these intermediate results depend also on the coins used by D_R. Hence, the output of A may vary with its coins, which are fed into D_R. In contrast, if A was using D_R as a black-box, then indeed we could treat the probabilistic decisions of D_R as if they were deterministic (by employing error-reduction and accounting for the tiny error of D_R within the error probability allowed also for pseudodeterministic algorithms).[27]

Acknowledgments

We thank Hanlin Ren for a useful conversation, in which he told the second author that a strengthening of the algorithm of [5] so that it applies to search problems with efficient probabilistic verification would imply a BPtime hierarchy (as in Theorem 4.7). We also thank Zeyong Li for useful conversations with the second author about Theorem 3.2.

References

[1] Alexander E. Andreev, Andrea E.F. Clementi, Jose D.P. Rolim, and Luca Trevisan. Weak Random Sources, Hitting Sets, and BPP Simulations. *SIAM Journal on Computing*, Viol. 28 (6), pages 2103–2116, 1999.

[2] Marshall Ball, Lijie Chen, and Roei Tell. Towards Free Lunch Derandomization from Necessary Assumptions (and OWFs). *ECCC*, TR25-010, 2025.

[3] Boaz Barak. A Probabilistic-Time Hierarchy Theorem for “Slightly Non-uniform” Algorithms. In *6th RANDOM*, pages 194–208, LNCS 2483, Springer, 2002.

[27]For readers familiar with the proof of [5], let us point out the specific place in the proof in which this issue arises. Recall that the algorithm in [5] is based on a partition of all input lengths into intervals, and defining, in each interval, a sequence of candidate algorithms, where each algorithm handles a different input length. It is the last algorithm in the sequence that is the problematic one. Specifically, this algorithm outputs the lexicographically first element that resides both in R and in a hitting-set, where the latter set is generated with intention to fool D_R. The problem is that the latter generator (which follows the framework of [6]) depends on D_R in a non-black-box manner; in particular, the generator's output depends on intermediate results of the computation of D_R, whereas these may depend significantly on D_R's coins (despite the fact that the output of D_R is the same for almost all coins).

[4] Zvika Brakerski and Oded Goldreich, From absolute distinguishability to positive distinguishability. In *Studies in Complexity and Cryptography*, pages 141-155, 2011.

[5] Lijie Chen, Zhenjian Lu, Igor Oliveira, Hanlin Ren, and Rahul Santhanam. Polynomial-Time Pseudodeterministic Construction of Primes. In the *64th FOCS*, pages 1261–1270, 2023.

[6] Lijie Chen and Roei Tell. Hardness vs randomness, revised: uniform, non-black-box, and instance-wise. In *62nd FOCS*, pages 125–136, 2021.

[7] Lijie Chen and Roei Tell. Simple and fast derandomization from very hard functions: eliminating randomness at almost no cost. In *53rd STOC*, pages 283–291, 2021.

[8] Peter Dixon, A. Pavan, and N.V. Vinodchandran. Promise Problems Meet Pseudodeterminism. *ECCC*, TR21-043, 2021.

[9] Eran Gat and Shafi Goldwasser. Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications. *ECCC*, TR11-136, 2011.

[10] Oded Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, 2008.

[11] Oded Goldreich. In a World of P=BPP. In *Studies in Complexity and Cryptography*, LNCS Vol. 6650, Springer, pages 191–232, 2011.

[12] Oded Goldreich. Two Comments on Targeted Canonical Derandomizers. In *Computational Complexity and Property Testing*, LNCS Vol. 12050, pages 24–35, 2020.

[13] Oded Goldreich. Multi-pseudodeterministic algorithms. *ECCC*, TR19-012, 2019.

[14] Oded Goldreich. On defining PPT-search problems. *ECCC*, TR24-161, 2024.

[15] Oded Goldreich and Bernd Meyer. Computational Indistinguishability: Algorithms vs. Circuits. *Theor. Comput. Sci.*, Vol. 191 (1-2), pages 215–218, 1998.

[16] Oded Goldreich and Avi Wigderson. Improved Derandomization of BPP Using a Hitting Set Generator. In *3rd RANDOM*, Lecture Notes in Computer Science (Vol. 1671), pages 131–137, Springer, 1999.

[17] Oded Goldreich and David Zuckerman. Another Proof $\mathcal{BPP} \subseteq \mathcal{PH}$ (and More). In *Studies in Complexity and Cryptography*, Lecture Notes in Computer Science (Vol. 6650), pages 40–53, Springer, 2011.

[18] Songhua He. A note on a hierarchy theorem for promise-BPTIME. *ECCC*, TR25-004, 2025.

[19] Russell Impagliazzo and Avi Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *29th STOC*, pages 220–229, 1999.

[20] Clemens Lautemann. BPP and the Polynomial Hierarchy. *Information Processing Letters*, Vol. 17, pages 215–217, 1983.

[21] Yanyi Liu and Rafael Pass. Characterizing derandomization through hardness of Levin-Kolmogorov complexity. In *37th CCC*, Art. 35, 2022.

[22] Yanyi Liu and Rafael Pass. Leakage-resilient hardness vs randomness. In *38th CCC*, Art. 32, 2023.

[23] Zhenjian Lu, Igor Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In the *53rd STOC*, pages 303–316, 2021.

[24] Noam Nisan and Avi Wigderson. Hardness vs Randomness. *Journal of Computer and System Science*, Vol. 49, No. 2, pages 149–167, 1994. Preliminary version in *29th FOCS*, 1988.

[25] Michael Sipser. A Complexity Theoretic Approach to Randomness. In *15th ACM Symposium on the Theory of Computing*, pages 330–335, 1983.

[26] Roei Tell. On defining PPT-search problems and PPT-optimization problems. *ECCC*, TR24-163, 2024.

Appendix: An alternative proof of the second part of Theorem 4.2

Recall that we refer to the proof that a hardness result (i.e., Statement 2) implies efficient pseudodeterministic algorithms for unary PPT-search problems (i.e., Statement 1), where the hardness result is used in the construction of a specific pseudorandom generator. The analysis of these pseudodeterministic algorithms amounts to showing that their failure implies a violation of the pseudorandomness of the generator. In other words, violating pseudorandomness is reduced to failure of the pseudodeterministic algorithm.

The reduction in the main text is slightly non-uniform, and so the aforementioned pseudodeterminism result relies on the fact that the hardness assumption yields pseudorandomness with respect to slightly non-uniform PPT algorithms. The reduction presented here is totally uniform, and so the pseudodeterminism result only uses pseudorandomness with respect to PPT algorithms (rather than with respect to slightly non-uniform PPT algorithms). Indeed, a totally uniform reduction is formally stronger and conceptually more pleasing than a slightly non-uniform one, but (as is often the case) this comes at the cost of a more complicated analysis.

The algorithmic modification. We are going to use Algorithm 4.2.4 as is, but replace the verification procedure. Instead of using a verification algorithm that has error probability 1/3, we shall use a specific verification algorithm with negligible (in n) error probability. Denoting the original verification algorithm by V', recall that $\Pr[V'(1^n, y) = 1] \geq 2/3$ for every $y \in R_{\tt yes}(1^n)$ whereas $\Pr[V'(1^n, y) = 1] \leq 1/3$ for every $y \in R_{\tt no}(1^n)$. We shall use an algorithm, denoted V, that, on input $(1^n, y)$, estimates $\Pr[V'(1^n, y) = 1]$ upto 0.005 with negligible (in n) error probability. Specifically, $V(1^n, y)$ invokes $V'(1^n, y)$ for $\widetilde{O}(\log n)$ times and outputs 1 if and only if the fraction of invocations that return 1 exceeds 0.66. Hence, letting $\mu(n)$ denote a negligible function, it holds that

- If $\Pr[V'(1^n, y) = 1] \geq 2/3$, then $\Pr[V(1^n, y) = 1] \geq 1 - \mu(n)$;
- If $\Pr[V'(1^n, y) = 1] < 0.65$, then $\Pr[V(1^n, y) = 1] \leq \mu(n)$.

We shall use this V in Algorithm 4.2.4. In the current analysis, we redefine $R'_{\tt yes}$ such that $R'_{\tt yes}(1^n) \stackrel{\rm def}{=} \{y : \Pr[V'(1^n, y) = 1] \geq 0.65\}$, and observe that $\Pr_{r \in S}[F_r(1^n) \in R'_{\tt yes}(1^n)] \geq (2/3) - o(1)$. This is shown as in the main proof, except that here the (contradicting) distinguisher uses V' rather than V. (That is, it computes $y \leftarrow F_r(1^n)$, estimates $\Pr[V(1^n, y) = 1]$ by a few invocations of V', and outputs 1 if and only if the estimate exceeds 0.66.)

The main analysis and the issue that arises. As in the main proof, it is left to show that in case $F_r(1^n) \in R'_{\tt yes}(1^n)$, w.v.h.p., it holds that $S' \neq \emptyset$, which means that Algorithm 4.2.4 solves the PPT-search problem $\overline{R}$ (in the sense of Definition 2.5, which means that it outputs a solution that is not in $R_{\tt no}(1^n)$). This amounts to showing that for every $r \in S$ it holds that if $y_r \in R'_{\tt yes}(1^n)$ then $\Pr_{s \in \{0,1\}^{k'}}[V_{G^{f_{\ell'}}(s)}(1^n, y_r) = 1] > 1/2$, and if $y_r \in R_{\tt no}(1^n)$ then $\Pr_{s \in \{0,1\}^{k'}}[V_{G^{f_{\ell'}}(s)}(1^n, y_r) = 1] \leq 1/2$. In other words, the claim is that every $r \in S$ such that $y_r \in (R'_{\tt yes}(1^n) \cup R_{\tt no}(1^n))$ it holds that

$$\left| \Pr_{s \in \{0,1\}^{k'}}[V_{G^{f_{\ell'}}(s)}(1^n, y_r) = 1] - \Pr_{r' \in \{0,1\}^m}[V_{r'}(1^n, y_r) = 1] \right| = o(1). \tag{4}$$

The latter fact is proved by observing that otherwise we obtain a quadratic-time algorithm that distinguishes random outputs of G from uniformly distributed strings. Unlike in the main text, where a violating $r \in S$ (or rather the corresponding s) was given as advice to the distinguisher, here we seek a totally uniform distinguisher. The natural idea is to consider a distinguisher that, on input a tested sequence $\rho \in \{0,1\}^m$, selects at random $s \in \{0,1\}^k$, computes $y \leftarrow F_{G^{f_\ell(s)}}(1^n)$, and outputs $V_{r'}(1^n, y)$. This does not quite work, because "absolute distinguishability" should be converted to "positive distinguishability" (see next). In fact, bridging this gap is the source of the complication in the current proof, and it is towards this goal that we redefined V rather than used V' as in the main proof.

Let us spell out the issue. The point is that absolute differences on different y_r's may cancel out if we pick a random $r \in S$ and proceed as above. We warn that the standard solutions that are used in the context of distinguishers that are more complex than the generator (see [4]) cannot be used here. Instead, we use the fact that we care only about $y_r \in (R'_{\texttt{yes}}(1^n) \cup R_{\texttt{no}}(1^n))$, and that for these y's it holds that $p_y \stackrel{\text{def}}{=} \Pr_{r' \in \{0,1\}^m}[V_{r'}(1^n, y) = 1]$ is very close to either 1 or 0 (i.e., p_y is negligibly close to 1 if $y \in R'_{\texttt{yes}}(1^n)$ and is negligibly close to 0 otherwise (i.e., when $y \in R_{\texttt{no}}(1^n)$)). Loosely speaking, after obtaining y (from $F_{G^{f_\ell(s)}}(1^n)$ on a random s), we estimate p_y and use this estimation towards converting the absolute gap to a positive one. Details follow.

An abstraction. We first abstract the problem that we face at this point. The setting consists of two distributions – a uniform distribution over $\{0,1\}^m$, denoted U_m, and a pseudorandom distribution over m-bit strings, denoted Z_m – and a potential (probabilistic) test, denoted T, which consists of two stages, denoted T_1 and T_2. Specifically, on input $\rho \in \{0,1\}^m$, the tester obtains $y \leftarrow T_1(1^m)$, and outputs $T_2(y, \rho) \in \{0,1\}$. We stress that both T_1 and T_2 are probabilistic, and that $T(\rho) = T_2(T_1(1^{|\rho|}), \rho)$.

In our application $T_1(1^m) \leftarrow F_{G^{f_\ell(s)}}(1^n)$, where n is easily determined by m and s is uniformly selected in $\{0,1\}^k$, and $T_2(y, \rho) = V_\rho(1^n, y)$; hence, in our application T_2 is actually deterministic. The distribution Z_m equals $G^{f_{\ell'}}(s')$, where s' is uniformly selected in $\{0,1\}^{k'}$.

The hypothesis in our application implies that $T = (T_1, T_2)$ cannot distinguish between Z_m and U_m; specifically, $|\Pr[T(Z_m) = 1] - \Pr[T(U_m) = 1]| = o(2^{-k})$ (equiv., $|\Pr[T_2(T_1(1^m), Z_m) = 1] - \Pr[T_2(T_1(1^m), U_m) = 1]| = o(2^{-k})$). Unfortunately, this does not suffice for our purposes. Letting $P_m = (R'_{\texttt{yes}}(1^n) \cup R_{\texttt{no}}(1^n)) \cap \{F_{G^{f_\ell(s)}}(1^n) : s \in \{0,1\}^k\}$, we wish to prove that, for every $y \in P_m$, it holds that $|\Pr[T_2(y, Z_m) = 1] - \Pr[T_2(y, U_m) = 1]| = o(1)$. We shall essentially achieve this goal by relying on an addition condition, which holds in our application: This condition is that $\Pr[T_2(y, U_m) = 1]$ is negligibly close to either 1 or 0. (Recall that if $y \in R'_{\texttt{yes}}(1^n)$ then $\Pr[V(1^n, y) = 1] \approx 1$, whereas if $y \in R_{\texttt{no}}(1^n)$ then $\Pr[V(1^n, y) = 1] \approx 0$.) Actually, we will need a stronger hypothesis that asserts that also a tester of complexity related to T cannot distinguish between Z_m and U_m. It will be more convenient to state the contrapositive.

Claim A.1 (the contrapositive claim): *Let $T = (T_1, T_2)$ and Z_m be as defined above* (i.e., not necessarily as in our application)*, and suppose that the following conditions hold.*

1. *For every $y \in P_m$, it holds that $\Pr[T_2(y, U_m) = 1]$ is either smaller than ϵ or larger than $1 - \epsilon$.*

2. *There exists $y \in P_m$ such that*

$$|\Pr[T_2(y, Z_m) = 1] - \Pr[T_2(y, U_m) = 1]| \geq \delta.$$

Then, there exists an algorithm T'_2 that invokes T'_2 for $\widetilde{O}(1/\epsilon^2)$ times and satisfies

$$|\Pr[T'_2(T_1(1^m), Z_m) = 1] - \Pr[T'_2(T_1(1^m), U_m) = 1]| \geq \Pr[T_1(1^m) = y] \cdot \delta - 4\epsilon.$$

In our application this means that if some element of $\{r \in S : y_r \in (R'_{\tt yes}(1^n) \cup R_{\tt no}(1^n))\}$ violates Eq. (4) (per the foregoing Item 2 with $\delta = o(1)$), then this contradicts the pseudorandomness of $G^{f_{\ell'}}$, because $\Pr[T_1(1^m)=y_r] \geq 2^{-k}$ and $\epsilon = o(2^{-k} \cdot \delta$, which implies $2^{-k} \cdot \delta - 4\epsilon > 2^{-k-1} \cdot \delta > 2^{-\alpha\ell'}$. (Recall that the pseudorandomness hypothesis refers to a distinguishing gap of $2^{-\alpha\ell'}$, whereas we can use any $\delta = o(1)$ (in fact even $\delta = 0.1$ suffices).)

Proof: Recall that the issue is that, for a given y, we don't know whether or not $\Pr[T_2(y, Z_m)=1] > \Pr[T_2(y, U_m)=1]|$. If we knew the answer, then things would have been easy: We would just let $T'_2(y, Z_m) = T_2(y, Z_m)$ if $\Pr[T_2(y, Z_m)=1] > \Pr[T_2(y, U_m)=1]|$ and $T'_2(y, Z_m) = 1 - T_2(y, Z_m)$ otherwise. In this case, letting $G = \{y : \Pr[T_2(y, Z_m)=1] > \Pr[T_2(y, U_m)=1]|\}$, we would have

$$\begin{aligned}
&|\Pr[T'_2(T_1(1^m), Z_m)=1] - \Pr[T'_2(T_1(1^m), U_m)=1]| \\
&= \sum_{y \in G} \Pr[T_1(1^m)=y] \cdot (\Pr[T_2(y, Z_m)=1] - \Pr[T_2(y, U_m)=1]) \\
&\quad + \sum_{y \notin G} \Pr[T_1(1^m)=y] \cdot ((1 - \Pr[T_2(y, Z_m)=1]) - (1 - \Pr[T_2(y, U_m)=1])) \\
&= \sum_{y} \Pr[T_1(1^m)=y] \cdot |\Pr[T_2(y, Z_m)=1] - \Pr[T_2(y, U_m)=1]| \\
&\geq \min_{y \in P_m} \{\Pr[T_1(1^m)=y]\} \cdot \delta.
\end{aligned}$$

In settings in which Z_m is efficiently sampleable, we can approximately determine whether or not $y \in G$ by estimating the relevant probability. Note that this estimation has to be good enough so that the gain from the contribution of y is not cancelled by the wrong decisions regarding membership in G. However, the real problem is that it is not guaranteed that Z_m is efficiently sampleable, Nevertheless, the fact that U_m is efficiently sampleable and the guarantee that $\Pr[T_2(y, U_m)=1]$ is very close to either 0 or 1 (when $y \in P_m$) will do.

The key observation is that if $p_y \stackrel{\rm def}{=} \Pr[T_2(y, U_m) = 1]$ is very close to either 0 or 1, then we lose little when making a wrong decision about whether or not y is in G. Since we only aim to gain from $y \in P_m$ and each of these y's satisfies $p_y \notin [\epsilon, 1-\epsilon]$, we can just avoid judgement in case $p_y \in [\epsilon, 1-\epsilon]$ (or approximately so). Hence, we propose the following algorithm T'_2, which proceeds as follows on input y (generated by T_1) and (tested sequence) ρ.

1. Estimate p_y up to ϵ (with error probability $o(\epsilon)$), denoting the result by $\widetilde{p}_y$.
2. If $\widetilde{p}_y \in [2\epsilon, 1-2\epsilon]$, then halt with an output that is oblivious of ρ (e.g., output 1).
3. Otherwise, if $\widetilde{p}_y < 2\epsilon$, then output $T_2(y, Z_m)$, and otherwise (i.e., $\widetilde{p}_y > 1-2\epsilon$) output $1 - T_2(y, Z_m)$.

In the rest of the analysis, we assume that $|\widetilde{p}_y - p_y| \leq \epsilon$, let $q_y \stackrel{\rm def}{=} \Pr[T_2(y, Z_m)=1]$, and focus on the case that $\delta > \epsilon$. We rely on the following observations:

- If $y \in P_m$ and $|q_y - p_y| \geq \delta$, then $\Pr[T'_2(y, Z_m) = 1] - \Pr[T'_2(y, U_m) = 1] \geq \delta$ regardless of whether or not $q_y > p_y$, because in that case $p_y \notin [\epsilon, 1-\epsilon]$, which implies $\widetilde{p}_y \notin [2\epsilon, 1-2\epsilon]$, whereas $q_y > \epsilon$ if $p_y < \epsilon$ and $q_y < 1-\epsilon$ if $p_y > 1-\epsilon$.

- If $\widetilde{p}_y \notin [2\epsilon, 1-2\epsilon]$, then $p_y \notin [3\epsilon, 1-3\epsilon]$, and so $\Pr[T_2'(y, Z_m)=1] - \Pr[T_2'(y, U_m)=1] \geq -3\epsilon$, because we get a negative contribution only if $q_y \in [0, p_y]$ when $p_y < 3\epsilon$ and $q_y \in [p_y, 1]$ when $p_y > 1-3\epsilon$.

Hence, assuming that $|\widetilde{p}_{T_1(1^m)} - p_{T_1(1^m)}| \leq \epsilon$ and defining $U \stackrel{\rm def}{=} \{y : p_y \notin [\epsilon, 1-\epsilon] \;\&\; |p_y - q_y| \geq \delta\}$, while noting that $P_m \subseteq U$, it holds that

$$\begin{aligned}
&\Pr[T_2'(T_1(1^m), Z_m)=1] - \Pr[T_2'(T_1(1^m), U_m)=1] \\
&\quad = \sum_{y \in U} \Pr[T_1(1^m)=y] \cdot \big(\Pr[T_2'(y, Z_m)=1] - \Pr[T_2'(y, U_m)=1]\big) \\
&\qquad + \sum_{y \notin U} \Pr[T_1(1^m)=y] \cdot \big(\Pr[T_2'(y, Z_m)=1] - \Pr[T_2'(y, U_m)=1]\big) \\
&\quad \geq \min_{y \in P_m} \Pr[T_1(1^m)=y] \cdot \delta - 3\epsilon,
\end{aligned}$$

since $y \notin U$ makes a (possibly negative) contribution to the sum only if $p_y \notin [3\epsilon, 1-3\epsilon]$ (and its contribution is lower-bounded by -3ϵ). Accounting for the probability of an estimation error, the claim follows. ■