Promise Problems Meet Pseudodeterminism *

Peter Dixon[†] A. Pavan[‡] N. V. Vinodchandran[§]

Abstract

The Acceptance Probability Estimation Problem (APEP) is to additively approximate the acceptance probability of a Boolean circuit. This problem admits a probabilistic approximation scheme. A central question is whether we can design a *pseudodeterministic* approximation algorithm for this problem: a probabilistic polynomial-time algorithm that outputs a canonical approximation with high probability. Recently, it was shown that such an algorithm would imply that *every approximation algorithm can be made pseudodeterministic* (Dixon, Pavan, Vinodchandran; *ITCS 2021*).

The main conceptual contribution of this work is to establish that the existence of a pseudodeterministic algorithm for APEP is fundamentally connected to the relationship between probabilistic promise classes and the corresponding standard complexity classes. In particular, we show the following equivalence: *every promise problem in* PromiseBPP *has a solution in* BPP *if and only if* APEP *has a pseudodeterministic algorithm.* Based on this intuition, we show that pseudodeterministic algorithms for APEP can shed light on a few central topics in complexity theory such as circuit lowerbounds, probabilistic hierarchy theorems, and multi-pseudodeterminism.

1 Introduction

Promise Problems: A promise problem Π is a pair of disjoint sets (Π_y, Π_n) of instances. Introduced by Even, Selman and Yacobi [ESY84], promise problems arise naturally in several settings such as hardness of approximations, public-key cryptography, derandomization, and completeness. While much of complexity theory is based on language recognition problems (where every problem instance is either in Π_y or in Π_n), the study of promise problems turned out be an indispensable tool that led to new insights in the area. Many interesting open questions regarding probabilistic complexity classes can be answered when we consider their promise versions. For example significant questions such as whether derandomization of BPP implies derandomization of MA, whether derandomization of BPP implies Boolean circuit lower bounds, whether derandomization of the one-sided-error class RP implies derandomization of BPP, or whether probabilistic complexity classes have complete problems remain open in the traditional classes. All these questions have an affirmative answer if we consider their promise analogues. For example, it is known that derandomizing PromiseBPP implies a derandomization of MA [GZ], and also implies Boolean circuit lower bounds [IKW02]. Similarly, there exist promise problems that are complete for classes such as PromiseBPP, PromiseRP, and SZK [SV03]. We refer the reader to the comprehensive survey article by Goldreich [Gol06] for a treatment on the wide-ranging applicability of promise problems.

The role of promise problems in circumventing certain deficiencies of language recognition problems is intriguing. A way to understand the gap between promise problems and languages is by considering *solutions* to promise problems. A set S is a solution to a promise problem $\Pi = (\Pi_y, \Pi_n)$ if $\Pi_y \subseteq S$ and $S \cap \Pi_n = \emptyset$. A natural question is to investigate the complexity of solutions to a promise problem. Informally, we say that for a complexity class $\mathcal{C}$ (for example BPP), Promise$\mathcal{C} = \mathcal{C}$, if every promise problem in Promise$\mathcal{C}$ has a

*Research supported in part by NSF grant 1934884

[†]Department of Computer Science, Iowa State University. tooplark@iastate.edu

[‡]Department of Computer Science, Iowa State University. pavan@cs.iastate.edu

[§]Department of Computer Science and Engineering, University of Nebraska, Lincoln. vinod@cse.unl.edu

solution in $\mathcal{C}$. Intuitively, when Promise$\mathcal{C}$ equals $\mathcal{C}$, then there is no gap between the class $\mathcal{C}$ and its promise counterpart.

In this paper we establish a close connection between promise problems and the seemingly unrelated notion of *pseudodeterminism*. More concretely, we establish that PromiseBPP = BPP if and only if all probabilistic approximation algorithms can be made pseudodeterministic.

Pseudodeterminism. The notion of a *pseudodeterministic* algorithm was introduced by Gat and Goldwasser [GG11][1]. Informally, a probabilistic algorithm M is pseudodeterministic if for every x, there exists a *canonical value* v such that $\Pr[M(x) = v]$ is high. Pseudodeterministic algorithms are appealing in several contexts, such as distributed computing and cryptography, where it is desirable that different invocations of a probabilistic algorithm by different parties should produce the same output. In complexity theory, the notion of pseudodeterminism clarifies the relationship between search and decision problems in the context of randomized computations. It is not known whether derandomizing BPP to P implies derandomization of probabilistic search algorithms. However, BPP = P implies derandomization of *pseudodeterministic* search algorithms [GGR13]. Since its introduction, the notion of pseudodeterminism has received considerable attention. Section 1.1 details prior and related work on pseudodeterminism.

Our Results

The main conceptual contribution of this paper is that the gap between PromiseBPP and BPP can be completely explained by the existence of pseudodeterministic algorithms for APEP: the problem of approximating the acceptance probability of Boolean circuits additively. While it is easy to design a probabilistic approximation algorithm for this problem, we do not know whether there exists a pseudodeterministic algorithm for this problem. Very recently the authors proved this problem *complete* for problems that admit approximation algorithms (more generally multi-pseudodeterministic algorithms as defined by Goldreich [Gol19]) in the context of pseudodeterminism [DPV21]. In particular, they showed that if APEP admits a pseudodeterministic algorithm, then every probabilistic approximation algorithm can be made pseudodeterministic. Our connection between pseudodeterminism and promise problems is established via APEP and is stated below.

Result 1. PromiseBPP *has a solution in* BPP *if and only if* APEP *has a pseudodeterministic approximation algorithm.*

Based on the above result, we obtain results that connect pseudodeterminism to circuit lower bounds, probabilistic hierarchy theorems, and multi-pseudodeterminism.

Circuit lower bounds: Establishing lower bounds against fixed polynomial-size circuits has a long history in complexity theory. In this line of work, the focus is on establishing upper bounds on the complexity of languages that can not be solved by any Boolean circuit of a fixed polynomial size. One of the central open questions in this area is to show that NP has languages that cannot be solved by linear-size Boolean circuits. Over the years researchers have made steady progress on this question. Kannan [Kan82] showed that there are problems in Σ_2^{P} that do not have linear-size circuits (more generally, size $O(n^k)$ for any constant k). Later, using techniques from learning theory, this upper bound was improved to $\mathrm{ZPP}^{\mathrm{NP}}$ [BCG+96, KW98] and later to S_2^{P} [Cai01]. Vinodchandran showed that the class PP does not have fixed polynomial-size circuits [Vin05]. Santhanam [San09] showed that further progress can be made if we relax the complexity classes to also include *promise classes*. In particular, he showed that PromiseMA does not have fixed polynomial-size circuits. It is not known whether this result can be improved to the traditional class MA. We show that if APEP has pseudodeterministic algorithms then MA has languages that can not be solved by $O(n^k)$ size circuits for any k.

Result 2. *If* APEP *admits pseudodeterministic approximation algorithms, then for any k, there are languages in* MA *that do not have $O(n^k)$ size Boolean circuits.*

[1] Originally termed Bellagio algorithms

In fact we show that under the assumption, MA = $\exists$.BPP and thus $\exists$.BPP does not have fixed polynomial-size circuits. The above result improves the connection between pseudodeterministic algorithms and circuit lower bounds established in [DPV18], where it was shown that designing a $\mathrm{BPP}^{\mathrm{NP}}_{tt}$ pseudodeterministic algorithm for problems in #NP would yield super-linear circuit lower bounds for languages in $\mathrm{ZPP}^{\mathrm{NP}}_{tt}$.

Hierarchy theorem for probabilistic classes: Some of the most fundamental results in complexity theory are hierarchy theorems – given more resources, more languages can be recognized. The time hierarchy theorem states that if $T_1(n) \log T_1(n) \in o(T_2(n))$, then there exist languages that can be decided in deterministic time $O(T_2(n))$, but not in deterministic time $O(T_1(n))$ [HS66, SHI65]. Similar hierarchy results hold for deterministic space and nondeterministic time [Coo73, SFM78, Žák83]. Proving hierarchy theorems for probabilistic time is a lot more challenging. There has been significant work in this direction [Bar02, FS04, FST05, vMP06]. All these results use an "advice bit", i.e. the results established are of the form "there is a language in $\mathrm{BPTIME}(T_2(n))/1$ that is not in $\mathrm{BPTIME}(T_1(n))/1$. Removing the advice bit has been a vexing open problem. We show that a pseudodeterministic algorithm for APEP leads to hierarchy theorems for bounded-error probabilistic time.

Result 3. *If* APEP *admits pseudodeterministic approximation algorithms, then hierarchy theorems for* BPTIME *hold. In particular* $\mathrm{BPTIME}(n^\alpha) \subsetneq \mathrm{BPTIME}(n^\beta)$ *for constant* $1 \leq \alpha < \beta$*.*

Multi-pseudodeterminism: Goldreich observed that the problem of estimating the average value of a function over a large universe admits a *2-pseudodeterministic algorithm*: a probabilistic polynomial-time algorithm that outputs *two canonical* values with high probability [Gol19]. Motivated by this, Goldreich introduced the notion of *multi-pseudodeterminism* [Gol19]. A *k-pseudodeterministic* algorithm is a probabilistic-polynomial time algorithm that, for every input x, outputs a value from a set S_x of size at most k with high probability (the exact probability bound has to be carefully defined, see Section 2 for a formal definition and [Gol19] for justification for the definition).

In [DPV21], the authors show that APEP is a complete problem for functions that admit k-pseudodeterministic algorithms for any constant k, in the sense that such functions admit pseudodeterministic algorithms if APEP admits a pseudodeterministic algorithm. Here we improve this result to functions that admit k-pseudodeterministic algorithms for any polynomial k.

Result 4. *If* APEP *admits a pseudodeterministic approximation algorithm, then every multi-valued function* f *that admits a* $k(n)$*-pseudodeterministic algorithm, for a polynomial* $k(n)$*, is in* Search BPP*. Moreover under the assumption, every multi-valued function* f *that admits a* $k(n)$*-pseudodeterministic algorithm also admits a pseudodeterministic algorithm.*

Concurrent Work: In an independent and recent work, Lu, Oliveria, Santhanam [LOS21] also explored the consequences of pseudodeterministic algorithms for APEP (they use CAPP to denote APEP). There is some intersection between their work and ours. In particular, they also establish results on probabilistic hierarchy. They showed that if there is a pseudodeterministic algorithm for APEP that is correct on average at infinitely many input lengths, then the hierarchy theorems for BPTIME follow. Note that our work considers existence of pseudodeterminitic algorithms for APEP in the worst-case. The rest of the work is different. Their work has results that include designing pseudodeterministic pseudorandom generators, and an equivalence between probabilistic hierarchy theorems and pseudodeterministic algorithms for constructing strings with large rKt complexity, which we do not have. Their work did not explore the relationships of pseudodeterministic algorithms with promise problems, circuit lowerbounds, and multi-pseudodeterminism which we establish.

1.1 Prior and Related Work on Pseudodeterminism

One line of research on pseudodeterminism has focused on designing pseudodeterministic algorithms for concrete problems. Gat and Goldwasser designed polynomial-time pseudodeterministic algorithms for various algebraic problems such as finding quadratic non-residues and finding non-roots of multivariate polynomials [GG11]. Goldwasser and Grossman exhibited a pseudodeterministic NC algorithm for computing matchings in bipartite graphs [GG17]. Recently, Anari and Vazirani [AV20] improved this result general graphs. Grossman designed a pseudodeterministic algorithm for computing primitive roots whose runtime matches the best known Las Vegas algorithm [Gro15]. Oliveira and Santhanam [OS17] designed a subexponential time pseudodeterministic algorithm for generating primes that works at infinitely many input lengths. Subsequently, Oliveira and Santhanam also showed that APEP admits a subexponential-time pseudodeterministic algorithm that is correct on average at infinitely many input lengths [OS18]. Goldreich, Goldwasser and Ron [GGR13], and later Holden [Hol17], investigated the possibility of obtaining pseudodeterministic algorithms for BPP search problems.

Other lines of work extended the notion of pseudodeterminism to several other scenarios including interactive proofs, streaming and sublinear algorithms, and learning algorithms [GGH17, GGH19, GGMW20, GGR13, OS18]. The works of Grossman and Liu, and Goldreich introduced generalizations of pseudodeterminism such as *reproducible algorithms*, *influential bit algorithms*, and *multi-pseudodeterministic algorithms* [GL19, Gol19]. Very recently the authors exhibited *complete problems* for functions that admit approximation algorithms, more generally multi-pseudodeterministic algorithms as defined by Goldreich [Gol19], in the context of pseudodeterminism [DPV21].

2 Preliminaries

In this paper, we are concerned with additive error approximations. A probabilistic algorithm A is an (ε, δ)-additive approximation algorithm for a function $f : \{0,1\}^* \to \mathbb{R}$ if the probability that $A(x) \in [f(x) - \varepsilon, f(x) + \varepsilon]$ is at least $1 - \delta$.

2.1 Pseudodeterminism

Definition 2.1. Acceptance Probability Estimation Problem: $\mathrm{APEP}_{(\varepsilon,\delta)}$: Given a Boolean circuit $C : \{0,1\}^n \to \{0,1\}$, give an (ε, δ)-additive approximation for $\Pr_{x \in U_n}[C(x) = 1]$.

Definition 2.2 ([GG11],[Gol19])**.** Let f be a multivalued function, i.e. $f(x)$ is a non-empty set. We say that f admits pseudodeterministic algorithms if there is a probabilistic polynomial-time algorithm A such that for every x, there exists a $v \in f(x)$ such that $A(x) = v$ with probability at least $2/3$. f admits k-pseudodeterministic algorithms if there is a probabilistic polynomial-time algorithm A such that for every x, there exists a set $S_x \subseteq f(x)$ of size at most k and the probability that $A(x) \in S(x)$ is at least $\frac{k+1}{k+2}$.

Note that the above definition captures pseudodeterminism for approximation algorithms, as approximation algorithms can be viewed as multivalued functions. It is known that any function that admits an (ε, δ) approximation algorithm admits a $(2\varepsilon, \delta)$ 2-pseudodeterministic algorithm (see [Gol19, DPV21] for a proof).

Proposition 1. *For every* $0 < \varepsilon, \delta < 1$*, there is a 2-pseudodeterministic algorithm for* $\mathrm{APEP}_{(\varepsilon,\delta)}$.

Gat and Goldwasser proved the following characterization [GG11].

Theorem 2.3. *A function admits a pseudodeterministic algorithm if and only if it is computable in* $\mathrm{PF}^{\mathrm{BPP}}$.

Definition 2.4 (SearchBPP [Gol11])**.** A search problem is a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$. For every x, the witness set W_x of x with respect to R is $W_x = \{y \mid (x, y) \in R\}$. A search problem R is in SearchBPP if

1. For every x, there is an efficient probabilistic algorithm to output an element of W_x: i.e. there exists a probabilistic polynomial-time algorithm A such that for every x for which $W_x \neq \phi$, $A(x) \in W_x$ with probability $\geq 2/3$, and

2. $R \in$ BPP: i.e. there exists a probabilistic polynomial-time algorithm B such that if $(x, y) \in R$, then $B(x, y)$ accepts with probability $> 2/3$, and if $(x, y) \notin R$ then $B(x, y)$ accepts with probability $< 1/3$.

Definition 2.5. For a multivalued function f, we say that f is in SearchBPP if there is a relation R in SearchBPP so that $\forall x$, the witness set $W_x \neq \phi$ and $W_x \subseteq f(x)$.

Dixon, Pavan and Vinodchandran [DPV21] proved that APEP is a complete problem for pseudodeterministic approximation algorithms and pseudodeterministic SearchBPP in the following sense.

Theorem 2.6. *If* $\mathrm{APEP}_{(1/100,1/8)}$ *admits a pseudodeterministic algorithm then*

1. *every function f that has an (ε, δ)-approximation algorithm has a pseudodeterministic $(3\varepsilon, \delta)$-approximation algorithm.*

2. *every problem in* SearchBPP *has a pseudodeterministic algorithm.*

It is well known that for every $0 < \varepsilon, \delta < 1$, there is a probabilistic algorithm for $\mathrm{APEP}_{(\varepsilon,\delta)}$ that runs in time $\texttt{poly}(n, 1/\varepsilon, \log 1/\delta)$ where n is the input length. Thus by the above result, we obtain the following proposition.

Proposition 2. *If* $\mathrm{APEP}_{(1/100,1/8)}$ *has a pseudodeterministic algorithm then for every* $0 < \varepsilon, \delta < 1$, $\mathrm{APEP}_{(\varepsilon,\delta)}$ *has a pseudodeterministic algorithm.*

Remark. In the rest of the paper, we use the phrase "APEP has a pseudodeterministic algorithm" in place of "$\mathrm{APEP}_{(1/100,1/8)}$ admits a pseudodeterministic algorithm", and denote the presumed pseudodeterministic algorithm with A_{ape}.

2.2 Promise Problems

Definition 2.7. A promise problem $\Pi = (\Pi_y, \Pi_n) \in$ PromiseBPP if there exists a probabilistic polynomial-time machine M such that $\forall x$

$$x \in \Pi_y \Leftrightarrow \Pr[M(x) = \text{ accepts}] \geq 2/3,$$

$$x \in \Pi_n \Leftrightarrow \Pr[M(x) = \text{ accepts}] < 1/3,$$

We can similarly define promise classes such as PromiseMA.

Definition 2.8. Let $\mathcal{C}$ be a complexity class. We say that a promise (Π_y, Π_n) has a solution in $\mathcal{C}$ if there exists a language L in $\mathcal{C}$ such that $\Pi_y \subseteq L$ and $L \cap \Pi_n = \emptyset$.

Definition 2.9. Let $\Pi = (\Pi_y, \Pi_n)$ be a promise problem. $\Pi' = \exists \cdot \Pi$ is a promise problem (Π'_y, Π'_n) defined as follows. There is a polynomial p such that $\forall x$

$$x \in \Pi'_y \Leftrightarrow \exists w \in \{0,1\}^{p(|x|)}, \langle x, w \rangle \in \Pi_y$$

$$x \in \Pi'_n \Leftrightarrow \forall w \in \{0,1\}^{p(|x|)}, \langle x, w \rangle \in \Pi_n$$

Definition 2.10. We say that a promise problem $\Pi = (\Pi_y, \Pi_n) \in \exists \cdot$ PromiseBPP if there is a promise problem $\Pi' \in$ PromiseBPP such that $\Pi = \exists \cdot \Pi'$.

Definition 2.11. A probabilistic polynomial-time machine M has BPP-type behaviour if on every input x, $\Pr[M(x)$ accepts$]$ is either $\geq 2/3$ or $< 1/3$.

3 Consequences of Pseudodeterministic Algorithm for APEP

3.1 Promise Problems

Theorem 3.1. PromiseBPP *has a solution in* BPP *if and only if* APEP *has a pseudodeterministic approximation algorithm.*

Proof. ($\Leftarrow$) : We will first prove that if APEP has a pseudodeterministic algorithm, then PromiseBPP has a solution in BPP. Let Π be a promise problem in PromiseBPP and let M be a probabilistic polynomial-time machine that witnesses this. Given x, let C_x be the following Boolean circuit:

$$C_x(r) = 1 \text{ if and only if } M(x) \text{ on random string } r \text{ accepts.}$$

Note that given x, we can construct C_x in time $\texttt{poly}(|x|)$. Consider the following probabilistic algorithm:

Algorithm B: On input x, construct C_x and run $A_{\text{ape}}(C_x)$. If $A_{\text{ape}}(C_x) \geq 1/2$, accept; else reject.

Claim 3.1.1. *B has a* BPP*-type behavior.*

Proof. Let x be an input to B. Recall that A_{ape} is a pseudodeterministic approximation algorithm that outputs a canonical value v on input C_x with probability at least 7/8. So either with probability at least 7/8, v is $\geq 1/2$, in which case B accepts x, or with probability at least 7/8, v is $< 1/2$ and B rejects. Thus for every input x, B either accepts with probability $\geq 7/8$ or rejects with probability $\geq 7/8$, and thus B has BPP-type behaviour. ☐

Let L be the language accepted by the above machine. Then by the above claim $L \in$ BPP.

Claim 3.1.2. *L is a solution to the promise problem* Π*.*

Proof. Let x be a string in Π_y. Thus $\Pr[C_x(r) = 1] \geq 2/3$. Thus $A_{\text{ape}}(C_x)$ outputs a canonical value $v \geq 2/3 - 1/100 > 1/2$ with probability at least 7/8, and thus B accepts with probability at least 7/8, and thus $x \in L$.

Suppose $x \in \Pi_n$. Thus Thus $\Pr[C_x(r) = 1] < 1/3$. Thus $A_{\text{ape}}(C_x)$ outputs a canonical value $v \leq 1/3 + 1/100 < 1/2$ with probability at least 7/8, and thus B rejects with probability at least 7/8, and thus $x \notin L$. ☐

By the above two claims we obtain that if APEP has a pseudodeterministic approximation algorithm, PromiseBPP has a solution in BPP.

($\Rightarrow$): Now suppose that PromiseBPP has a solution in BPP. By Proposition 1, there is a 2-pseudodeterministic (ε, δ) approximation algorithm M for APEP where $\delta = 1/4$ and $\epsilon = 1/200$. We slightly modify M as follows: whenever M outputs a value v, then output a value v' that is the closest integer multiple of ε to v. Note that the modified machine M is a $(2\varepsilon, \delta)$ approximation algorithm for APEP. The machine M has the property that every output is of the form $k\varepsilon$, $0 \leq k \leq 1/\varepsilon$.

For a Boolean circuit C, let p_C denote the acceptance probability of C. Thus for every C, we have

$$\Pr[M(C) \in (p_C - 2\varepsilon, p_C + 2\varepsilon)] \geq 3/4 \tag{1}$$

We associate a promise problem $\Pi = (\Pi_y, \Pi_n)$ with M. This definition of promise problem is inspired by the work of Goldreich [Gol11].

$$\Pi_y = \{\langle C, v\rangle \mid M(C) \text{ outputs } v \text{ with probability at least } 3/8\}$$

$$\Pi_n = \{\langle C, v\rangle \mid M(C) \text{ outputs } v \text{ with probability at most } 1/4\}$$

We make the following two critical observations.

Observation 3.2. *If $\langle C, v\rangle \notin \Pi_n$, then $v \in (p_C - 2\varepsilon, p_C + 2\varepsilon)$.*

This observation follows from equation 1.

Observation 3.3. *For every Boolean circuit C, there exists a v such that $\langle C, v\rangle \in \Pi_y$ and $v = k\epsilon$ for some $k > 0$,* *Proof.* Since M is 2-pseudodeterministic, there is a set S of size at most 2 such that every element in S lies between $p_C - 2\epsilon$ and $p_C + 2\epsilon$ and $\Pr[M(C) \in S] \geq 3/4$. Thus there must exist an element v from S such that $M(C)$ outputs v with probability at least $3/8$. Finally note that the modification of M described earlier ensures that M always outputs a multiple of ϵ. □

Claim 3.3.1. $\Pi \in$ PromiseBPP.

Proof. Consider the algorithm M_Π: On input $\langle C, v\rangle$ run $M(C)$. If it outputs v, then accept, else reject. This algorithm accepts all instances from Π_y with probability at least $3/8$ and accepts all instances from Π_n with probability at most $1/4$. Since there is a gap between $3/8$ and $1/4$, this gap can be amplified with standard amplification techniques. This implies that Π is in PromiseBPP. □

Now we will complete the proof by designing a pseudodeterministic algorithm for APEP. By our assumption there is a language $L_\Pi \in$ BPP that is a solution to Π. Consider the following deterministic algorithm for APEP with oracle access to L_Π. On input C, check if $\langle C, k\varepsilon\rangle \in L_\Pi$ for integer values of k, $0 \leq k \leq 1/\varepsilon$. Let ℓ be the first value such that $\langle C, \ell\varepsilon\rangle \in L_\Pi$, then output $\ell\varepsilon$. By Observation 3.3, such an ℓ must exist. Moreover, if $\langle C, \ell\varepsilon\rangle \in L_\Pi$, then it must be the case that $\langle C, \ell\varepsilon\rangle \notin \Pi_n$. By Observation 3.2, we have that $\ell\varepsilon \in (p_c + 2\varepsilon, p_c - 2\varepsilon)$. Thus APEP has a $(2\varepsilon, \delta)$, $\mathrm{PF}^{\mathrm{BPP}}$ approximation algorithm. This implies that APEP has a $(2\varepsilon, \delta)$ pseudodeterministic algorithm by Theorem 2.3.

□

We obtain the following corollary by using the completeness result of APEP .

Corollary 3.4. *If* PromiseBPP *has a solution in* BPP*, then* SearchBPP *admits pseudodeterministic algorithms.*

Proof. From the above theorem, if PromiseBPP has a solution in BPP, then APEP has pseudodeterministic algorithms. The proof follows from Theorem 2.6. □

3.2 Circuit Lower Bounds

Theorem 3.5. *If* APEP *admits pseudodeterministic approximation algorithms, then*

1. *Every promise problems* $\Pi = (\Pi_Y, \Pi_N)$ *in* PromiseMA *has a solution in* MA.
2. MA $= \exists \cdot$ BPP.
3. MA *does not have fixed polynomial-size circuits.*

Proof.

1. We first show that if Π is a promise problem in PromiseMA, then $\Pi \in \exists \cdot$ PromiseBPP. Let M be a probabilistic polynomial-time verifier. Consider the following promise problem Π': A tuple $\langle x, w\rangle$ is a positive instance if M accepts $\langle x, w\rangle$ with probability at least $2/3$ and is a negative instance if M accepts $\langle x, w\rangle$ with probability at most $1/3$. It is easy to see that $\Pi = \exists.\Pi'$. By Theorem 3.1, Π' has a solution L' in BPP if APEP admits pseudodeterministic algorithms. Note that the language $L = \exists \cdot L'$ is a solution to Π, and $\exists \cdot L'$ is in $\exists \cdot$ BPP. Since $\exists \cdot$ BPP is a subset of MA, the claim follows.
2. The above proof showed that every promise problem in PromiseMA has a solution in $\exists \cdot$ BPP. Thus it follows that MA $= \exists \cdot$ BPP.
3. Santhanam [San09] showed that for every k, there is a problem Π_k in PromiseMA that does not have any solution that admits $O(n^k)$ size circuits. Since by Item 1 Π_k has a solution $L_k \in$ MA, we get that L_k does not have $O(n^k)$ size circuits. Combining this with 2, it follows that $\exists \cdot$ BPP does not have $O(n^k)$ size circuits.

□

The above result reveals an interesting connection between pseudodeterminism, derandomization of BPP, and circuit complexity. If APEP has pseudodeterministic algorithms, then derandomizing BPP to P implies that NP does not have fixed polynomial-size circuits.

3.3 Hierarchy Theorems

Theorem 3.6. *If* APEP *admits pseudodeterministic approximation algorithms, then hierarchy theorems for* BPTIME *hold. In particular,* $\mathrm{BPTIME}(n^\alpha) \subsetneq \mathrm{BPTIME}(n^\beta)$ *for constant* $1 \leq \alpha < \beta$.

Proof. We will first show that there is a constant c so that $\mathrm{BPTIME}(n) \subsetneq \mathrm{BPTIME}(n^c)$. A similar arguments will show that $\mathrm{BPTIME}(n^a) \subsetneq \mathrm{BPTIME}(n^{ca})$ for every $a > 0$. Then the theorem will follow from padding arguments.

Let $\{M_i\}_{i \geq 1}$ be an enumeration of probabilistic linear-time Turing machines. Suppose that A_{ape} runs in time m^a in circuits of size m for some $a > 0$. For every M_i, consider a probabilistic machine M_i' defined as follows. M_i' on input x constructs a circuit $C_{i,x}$ as follows. The circuit $C_{i,x}$ on input r simulates $M_i(x)$ with r as random bits and accepts if and only if M_i accepts. Now M_i' runs $A_{\mathrm{ape}}(C_{i,x})$, and accepts if and only if the output of $A_{\mathrm{ape}}(C_{i,x}) \geq 1/2$.

Claim 3.6.1. *There exists a constant* $c > 0$ *such that for every* i*, the machine* M_i' *runs in time* $O(n^c)$*.*

Proof. Let n be the length of input x to M_i'. The machine $M_i'(x)$ first constructs the circuit $C_{i,x}$. Since $M_i(x)$ runs in $O(n)$ time, the size of the circuit $C_{i,x}$ is bounded by $O(n^2)$, it can be constructed in $O(n^2)$ time. Next M_i' runs A_{ape} on $C_{i,x}$, this steps takes $O(n^{2a})$ time. Since a is a constant, there is a universal constant c such that the runtime of M_i' is $O(n^c)$.

□

Claim 3.6.2. *For every* i*,* M_i' *has* BPP*-type behaviour*

Proof. This follows because the pseudodeterministic algorithm A_{ape}, on every input, outputs a canonical value v with probability at least $2/3$. If the canonical value $v \geq 1/2$, then M_i' accepts with probability at least $2/3$, else M_i' accepts with probability at most $1/3$. Thus M_i has BPP-type behaviour. □

Claim 3.6.3. *For every* $L \in \mathrm{BPTIME}(n)$*, there is* $i > 0$ *such that* M_i' *accepts* L*.*

Proof. Since $L \in \mathrm{BPTIME}(n)$, there exists an $i > 0$ such that M_i accepts L and M_i has BPP-type behaviour. Let $x \in L$ be an input to M_i. The probability that M_i accepts is $\geq 2/3$. Thus the acceptance probability of the circuit $C_{i,x}$ is at least $2/3$. Thus $A_{\mathrm{ape}}(C_{i,x})$ outputs a canonical $v \geq 2/3 - 1/100 \geq 1/2$ with probability at least $2/3$. Thus M_i' accepts x with probability $\geq 2/3$. Similar arguments show that if $x \notin L$, M_i' rejects x with probability $\geq 2/3$.

□

Now using the standard diagonalization argument, we construct a language L_D in $\mathrm{BPTIME}(n^{c+1})$. The language L_D is a tally language and we describe it via a $\mathrm{BPTIME}(n^{c+1})$ machine N that accepts it. The machine N on input 0^i simulates $M_i'(0^i)$ and accepts if and only if $M_i'(0^i)$ rejects. Since M_i' has BPP-type behaviour, N also has BPP-type behaviour. Since M_i' runs in time $O(n^c)$, N can simulate it in time $O(n^{c+1})$. Thus $L_D \in \mathrm{BPTIME}(n^{c+1})$. Suppose that $L_D \in \mathrm{BPTIME}(n)$. By Claim 3.6.3, there exists $i > 0$ such that M_i' accepts L_D. Now consider input 0^i. Observe that M_i' accepts 0^i if and only if N rejects 0^i. Thus $0^i \in L_D$ if and only if M_i' rejects 0^i. This is a contradiction, and thus $L_D \notin \mathrm{BPTIME}(n)$. □

3.4 Multivalued Functions

Theorem 3.7. *If* APEP *admits pseudodeterministic approximation algorithms, then every multivalued function* f *that admits a* $k(n)$*-pseudodeterministic algorithm for a polynomial* $k(n)$ *is in* SearchBPP*.*

Proof. Let f be a multi-valued function and let M_f be a $k(n)$-pseudodeterministic algorithm for f. Without loss of generality we can assume that f maps strings of length n to strings of length $p(n)$ for some polynomial p. For input x of length n, let S_x be the set of size $\leq k(n)$ such that $S_x \subseteq f(x)$ and $M_f(x) \in S_x$ with probability $\geq \frac{k(n)+1}{k(n)+2}$. From the definition of $k(n)$-pseudodeterminism, we have the following claim.

Claim 3.7.1. *$\exists v* \in S_x$ such that $\Pr[M_f(x) = v*] \geq \frac{1+1/k(n)}{k(n)+2}$. Moreover, $\forall v \notin S_x$ $\Pr[M_f(x) = v] < \frac{1}{k(n)+2}$.*

Let $\tau = \frac{1+1/2k(n)}{k(n)+2}$ be a threshold that is the middle point of $\frac{1+1/k(n)}{k(n)+2}$ and $\frac{1}{k(n)+2}$. For a pair of strings $\langle x, v\rangle$, where $|x| = n$ and $|v| = p(n)$, let $C_{x,v}$ be the following Boolean circuit. $C_{x,v}$ on input r, outputs 1 if $M_f(x)$ on random string r outputs v, 0 otherwise. We will show that there is a relation R so that (1) $\forall x : W_x \neq \phi$ and $W_x \subseteq f(x)$, and (2) $R \in$ SearchBPP. We define the relation R as follows.

$$R = \{\langle x, v\rangle \mid \text{the canonical output of } A_{\text{ape}}(C_{x,v}) \geq \tau\}$$

Here A_{ape} is the (ε, δ) pseudodeterministic algorithm for APEP, where $\varepsilon = 1/2k(n)(k(n) + 2)$ and $\delta = 2^{-n}$. Note that such an algorithm exists under the assumption by Proposition 2 and standard error reduction techniques.

Claim 3.7.2. *$\forall x, W_x \subseteq f(x)$ and W_x is not empty.*

Proof. For this we show that $W_x \subseteq S_x$. If $v \notin S_x$, then $M_f(x)$ outputs v with probability at most $1/(k(n)+2)$, thus the canonical output of $A(C_{x,v})$ is $< \frac{1}{k(n)+2} + \varepsilon = \tau$ and by definition $v \notin W_x$. On the other hand, Since $v^* \in W_x$, the canonical output of $A_{\text{ape}}(C_{x,v^*})$ is $\geq \frac{1+1/k(n)}{k(n)+2} - \varepsilon = \tau$. Thus $v^* \in W_x$. Thus $W_x \neq \phi$ $\square$

Claim 3.7.3. *$R \in$ BPP.*

Proof. Consider the algorithm that on input $\langle x, v\rangle$, runs $A_{\text{ape}}(C_{x,v})$ and accepts if and only if the output of A_{ape} is $\geq \tau$. Since A_{ape} is a pseudodeterministic algorithm for APEP it outputs a canonical value with probability at least $1 - 1/2^n$. This shows that R is in BPP. $\square$

Claim 3.7.4. *There is a probabilistic algorithm B that on input x outputs $v \in W_x$ with probability $> 2/3$.*

Proof. We first design an algorithm B' with a nontrivial success probability and boost it to get algorithm B.

Algorithm B': On input x, run $M_f(x)$. Let v be an output. Construct circuit $C_{x,v}$ and run A_{ape} on $C_{x,v}$. If the output of A_{ape} is $\geq \tau$, output v. Otherwise output $\perp$.

Consider a $v \notin W_x$. Then by definition of R, we have that the canonical output of $A_{\text{ape}}(C_{x,v})$ is less than τ. Thus $A_{\text{ape}}(C_{x,v})$ outputs a value larger than τ with probability at most $1/2^n$. Thus we have that for every $v \notin W_x$

$$\Pr[B' \text{ outputs } v | M_f(x) \text{ outputs } v] \leq 1/2^n$$

$$\begin{aligned}
\Pr[B' \text{ outputs a } v \notin W_x] &= \sum_{v \notin W_x} \Pr[B' \text{ outputs } v | M_f(x) \text{ outputs } v] \times \Pr[M_f(x) \text{outputs } v] \\
&\leq 1/2^n \sum_{v} \Pr[M_f(x) \text{outputs } v] \\
&\leq 1/2^n
\end{aligned}$$

By Claim 3.7.1, probability that $M_f(x)$ outputs v^* is at least $\frac{1+1/k(n)}{k(n)+2}$, it must be the case that the canonical output of $A(C_{x,v^*})$ is at least $\frac{1+1/k(n)}{k(n)+2} - \varepsilon = \tau$. Thus $v^* \in W_x$. Thus $A_{\text{ape}}(C_{x,v^*})$ outputs a value $\geq \tau$ with probability at least $1-1/2^n$. Thus the probability that B' outputs v^* is at least $\frac{1+1/k(n)}{k(n)+2} \times (1-1/2^n)$.

Thus B' outputs a value that is not in W_x with probability at most $1/2^n$, it outputs a value in W_x with probability at least $\frac{1+1/k(n)}{k(n)+2} \times (1 - 1/2^n)$, and outputs $\perp$ with the remaining probability. We obtain B by repeated invocations ($O(k(n)^3)$ many) of B' and outputting the most frequent output.

$\square$

This completes the proof that f is in SearchBPP.

□

Using the above result, we obtain the following corollary, which improves a result from [DPV21].

Theorem 3.8. *If* APEP *admits pseudodeterministic algorithm, then any multivalued function that admits a* $k(n)$*-pseudodeterministic algorithm also admits a pseudodeterministic algorithms, where* $k(n)$ *is a polynomial.*

Proof. By the above theorem, if APEP admits pseudodeterministic algorithm, then any problem that admits a $k(n)$-pseudodeterministic algorithm is in SearchBPP. By Theorem 2.6, if APEP admits pseudodeterministic algorithms, every problem in SearchBPP has a pseudodeterministic algorithm. □

Acknowledgements. We thank Zhenjian Lu, Igor Oliveira, and Rahul Santhanam for sharing a draft of their work.

References

[AV20] Nima Anari and Vijay V. Vazirani. Matching is as easy as the decision problem, in the NC model. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 54:1–54:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[Bar02] Boaz Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings*, volume 2483 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2002.

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

[Cai01] J-Y. Cai. $s_2^P \subseteq zpp^{NP}$. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 620–629, 2001.

[Coo73] Stephen A. Cook. A hierarchy for nondeterministic time complexity. *J. Comput. Syst. Sci.*, 7(4):343–353, 1973.

[DPV18] Peter Dixon, A. Pavan, and N. V. Vinodchandran. On pseudodeterministic approximation algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 61:1–61:11, 2018.

[DPV21] Peter Dixon, A. Pavan, and N. V. Vinodchandran. Complete problems for multi-pseudodeterministic computations. In *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 66:1–66:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[ESY84] Shimon Even, Alan L. Selman, and Yacov Yacobi. The complexity of promise problems with applications to public-key cryptography. *Inf. Control.*, 61(2):159–173, 1984.

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 316–324. IEEE Computer Society, 2004.

[FST05] Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Hierarchies for semantic classes. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 348–355. ACM, 2005.

[GG11] E. Gat and S. Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:136, 2011.

[GG17] S. Goldwasser and O. Grossman. Bipartite perfect matching in pseudo-deterministic NC. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 87:1–87:13, 2017.

[GGH17] S. Goldwasser, O. Grossman, and D. Holden. Pseudo-deterministic proofs. *CoRR*, abs/1706.04641, 2017.

[GGH19] Michel Goemans, Shafi Goldwasser, and Dhiraj Holden. Doubly-efficient pseudo-deterministic proofs. *arXiv*, 2019.

[GGMW20] Shafi Goldwasser, Ofer Grossman, Sidhanth Mohanty, and David P. Woodruff. Pseudo-deterministic streaming. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS*, volume 151 of *LIPIcs*, pages 79:1–79:25, 2020.

[GGR13] O. Goldreich, S. Goldwasser, and D. Ron. On the possibilities and limitations of pseudodeterministic algorithms. In *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 127–138, 2013.

[GL19] Ofer Grossman and Yang P. Liu. Reproducibility and pseudo-determinism in log-space. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 606–620. SIAM, 2019.

[Gol06] Oded Goldreich. On promise problems: A survey. In Oded Goldreich, Arnold L. Rosenberg, and Alan L. Selman, editors, *Theoretical Computer Science, Essays in Memory of Shimon Even*, volume 3895 of *Lecture Notes in Computer Science*, pages 254–290. Springer, 2006.

[Gol11] Oded Goldreich. In a world of P=BPP. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *Lecture Notes in Computer Science*, pages 191–232. Springer, 2011.

[Gol19] Oded Goldreich. Multi-pseudodeterministic algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:12, 2019.

[Gro15] O. Grossman. Finding primitive roots pseudo-deterministically. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:207, 2015.

[GZ] Oded Goldreich and David Zuckerman. Another proof that BPP $\subseteq$ PH (and more). In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *Lecture Notes in Computer Science*, pages 40–53. Springer.

[Hol17] Dhiraj Holden. A note on unconditional subexponential-time pseudo-deterministic algorithms for BPP search problems. *CoRR*, abs/1707.05808, 2017.

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

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

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

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

[LOS21] Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *STOC*, 2021. To Appear. ECCC Tech Report 21-039.

[OS17] I. Oliveira and R. Santhanam. Pseudodeterministic constructions in subexponential time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 665–677, 2017.

[OS18] Igor Carboni Oliveira and Rahul Santhanam. Pseudo-derandomizing learning and approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018*, volume 116 of *LIPIcs*, pages 55:1–55:19, 2018.

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

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

[SHI65] Richard Edwin Stearns, Juris Hartmanis, and Philip M. Lewis II. Hierarchies of memory limited computations. In *6th Annual Symposium on Switching Circuit Theory and Logical Design, Ann Arbor, Michigan, USA, October 6-8, 1965*, pages 179–190. IEEE Computer Society, 1965.

[SV03] Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003.

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

[vMP06] Dieter van Melkebeek and Konstantin Pervyshev. A generic time hierarchy for semantic models with one bit of advice. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 129–144. IEEE Computer Society, 2006.

[Zák83] Stanislav Zák. A turing machine time hierarchy. *Theor. Comput. Sci.*, 26:327–333, 1983.