Pseudodeterminism: Promises and Lowerbounds

Peter Dixon*
tooplark@gmail.com
Ben-Gurion University of the Negev
Be'er Sheva, Israel

A. Pavan
pavan@cs.iastate.edu
Iowa State University
Ames, USA

Jason Vander Woude
jasonvw@huskers.unl.edu
University of Nebraska-Lincoln
Lincoln, USA

N. V. Vinodchandran
vinod@cse.unl.edu
University of Nebraska-Lincoln
Lincoln, USA

ABSTRACT

A probabilistic algorithm A is *pseudodeterministic* if, on every input, there exists a canonical value that is output with high probability. If the algorithm outputs one of k canonical values with high probability, then it is called a k-pseudodeterministic algorithm. In the study of pseudodeterminism, the Acceptance Probability Estimation Problem (APEP), which is to additively approximate the acceptance probability of a Boolean circuit, is emerging as a central computational problem. This problem admits a 2-pseudodeterministic algorithm. Recently, it was shown that a pseudodeterministic algorithm for this problem would imply that any multi-valued function that admits a k-pseudodeterministic algorithm for a constant k (including approximation algorithms) also admits a pseudodeterministic algorithm (Dixon, Pavan, Vinodchandran; *ITCS 2021*).

The contribution of the present work is two-fold. First, as our main conceptual contribution, we establish that the existence of a pseudodeterministic algorithm for APEP is fundamentally related to the gap between probabilistic promise classes and the corresponding standard complexity classes. In particular, we show the following equivalence: APEP *has a pseudodeterministic approximation algorithm if and only if every promise problem in* PromiseBPP *has a solution in* BPP. A conceptual interpretation of this equivalence is that the algorithmic gap between 2-pseudodeterminism and pseudodeterminism is equivalent to the gap between PromiseBPP and BPP. Based on this connection, we show that designing pseudodeterministic algorithms for APEP leads to the solution of some open problems in complexity theory, including new Boolean circuit lower bounds. This equivalence also explains how multi-pseudodeterminism is connected to problems in SearchBPP. In particular, we show that if APEP has a pseudodeterministic algorithm, then every problem that admits a $k(n)$-pseudodeterministic algorithm (for any polynomial k) is in SearchBPP and admits a pseudodeterministic algorithm. Motivated by this connection, we also explore its connection to probabilistic search problems and establish that APEP is complete for certain notions of search problems in the context of pseudodeterminism.

Our second contribution is establishing query complexity lower bounds for multi-pseudodeterministic computations. We prove that for every $k \geq 1$, there exists a problem whose $(k+1)$-pseudodeterministic query complexity, in the uniform query model, is $O(1)$ but has a k-pseudodeterministic query complexity of $\Omega(n)$, even in the more general nonadaptive query model. A key contribution of this part of the work is the utilization of Sperner's lemma in establishing query complexity lower bounds.

CCS CONCEPTS

• **Theory of computation → Pseudorandomness and derandomization**; **Complexity classes**; **Circuit complexity**; *Oracles and decision trees.*

KEYWORDS

probabilistic computations, pseudodeterminism, promise problems, circuit lower bounds, completeness, hierarchy theorems, query complexity

ACM Reference Format:
Peter Dixon, A. Pavan, Jason Vander Woude, and N. V. Vinodchandran. 2022. Pseudodeterminism: Promises and Lowerbounds. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519935.3520043

1 INTRODUCTION

Probabilistic algorithms lack *reproducibilty* compared to their deterministic counterparts. Two different runs of a probabilistic algorithm can produce two different outputs. For example, consider the problem of generating an n-bit prime number. A straightforward probabilistic algorithm for this problem randomly picks an n-bit positive integer and outputs it if it is a prime number. However, two different simulations of this algorithm will most likely produce two different prime numbers. Can we design a probabilistic algorithm that consistently outputs the *same prime number* on different runs of the algorithm? Although there is a polynomial-time deterministic algorithm for testing primality [1], there are no deterministic algorithms known for the prime generation problem. There are many other computational problems for which the only efficient algorithms known are non-reproducible probabilistic algorithms. This deficiency led Gat and Goldwasser to introduce the notion

*Part of the work done while the author was at Iowa State University

STOC '22, June 20–24, 2022, Rome, Italy

ACM ISBN 978-1-4503-9264-8/22/06.
https://doi.org/10.1145/3519935.3520043

of *pseudodeterministic algorithms* [14]. (originally termed Bellagio algorithms in their paper). A probabilistic algorithm A is pseudodeterministic if for every x, there exists a *canonical value* v_x such that $\Pr[A(x) = v_x]$ is high. Unless otherwise stated, a pseudodeterministic algorithm is assumed to run in polynomial time.

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 produce the same output. For example, in cryptography, it is important to share a common key among multiple parties. It is desirable to have a mechanism to share a common key without communication and shared randomness. Reproducibility guaranteed by pseudodeterminism is a sought-after feature in software engineering. In enterprise software products, the reproducibility of computational results has long been considered the gold standard. Applications that cannot reproduce their results on multiple runs may not be trusted by users and may be viewed as an error in the code. The need for reproducibility in software development and the associated challenges involved has been highlighted in the literature, for example [3].

Since its introduction, the notion of pseudodeterminism has received considerable attention from the theory community. One line of research focused on designing pseudodeterministic algorithms for various natural computational problems. Gat and Goldwasser designed polynomial-time pseudodeterministic algorithms for algebraic problems such as finding quadratic non-residues and finding non-roots of multivariate polynomials [14]. Goldwasser and Grossman exhibited a pseudodeterministic NC algorithm for computing matchings in bipartite graphs [22]. Anari and Vazirani [2] improved this result to general graphs. Grossman designed a pseudodeterministic algorithm for computing primitive roots whose runtime matches the best known Las Vegas algorithm [26]. Oliveira and Santhanam [37] designed a sub-exponential time pseudodeterministic algorithm for generating primes that works at infinitely many input lengths. Goldreich, Goldwasser and Ron [19], and Holden [29], 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, learning algorithms, influential bit algorithms, and multi-pseudodeterministic algorithms [15, 18, 19, 23, 24, 27, 38]. The works of Goldreich, Goldwasser and Ron [19], and Goldwasser, Grossman, Mohanty, and Woodruff [24] exhibited impossibility results on the existence of pseudodeterministic algorithms in sub-linear and streaming computation models.

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 every search algorithm that is *pseudodeterministic* can be derandomized [19].

1.1 Circuit Acceptance Probability Estimation Problem (APEP)

APEP is the following computational problem[1]. Given a Boolean circuit $C : \{0,1\}^n \to \{0,1\}$, additively estimate the acceptance probability of C. This problem has a simple and efficient probabilistic algorithm. Indeed, the algorithm that randomly samples $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ strings from $\{0,1\}^n$ and outputs the fraction of strings on which circuit C evaluates to 1 is an ε-additive approximation of the acceptance probability of C with probability $\geq 1 - \delta$. However, this standard algorithm is not pseudodeterministic: different runs of this algorithm will output different correct approximations. *Does there exist a pseudodeterministic algorithm for* APEP*?* This computational question is emerging as a central question in the study of pseudodeterminism [10, 35, 38]. In [10], it was shown that APEP is a *complete problem* in the context of pseudodeterminism [10]: if there is a pseudodeterministic algorithm for APEP, then every problem in SearchBPP, including the prime generation problem mentioned earlier, admits pseudodeterministic algorithms, and APEP admits a pseudodeterministic polynomial algorithm if and only every efficient approximation algorithm can be made pseudodeterministic. Thus APEP captures the challenge of making a large class of randomized algorithms pseudodeterministic. While designing a pseudodeterministic algorithm for APEP is an open problem, Oliveira and Santhanam have designed a subexponential-time pseudodeterministic algorithm for APEP that is correct on average at infinitely many input lengths [38]. The significance of the above question in computational complexity theory is emerging in very recent works. Lu, Oliviera, and Santhanam [35] explored the relationship between pseudodeterministic algorithms for APEP and the structure of probabilistic polynomial time classes. In particular, they showed that a pseudodeterministic algorithm for APEP (even on average) will result in hierarchy theorems for bounded probabilistic polynomial-time classes (BPTIME).

Our first set of results establishes that the question of designing pseudodeterministic algorithms for APEP *is intricately connected to several well-studied notions and questions in complexity theory.* In particular, one of our main conceptual contributions is to establish a crisp connection between the question of designing pseudodeterministic algorithms for APEP and the complexity of *probabilistic promise problems*. Based on this connection, we relate pseudodeterminism to circuit lower bounds, derandomization, hierarchy theorems, completeness, and probabilistic search problems.

1.2 Multi-Pseudodeterminism

It is easy to observe that the above-mentioned probabilistic algorithm for APEP can be modified to output *two canonical values* with very high probability. This is done by rounding the value computed to the nearest multiple of ε. The resulting algorithm has an accuracy guarantee of 2ε. This rounding trick can be used for converting any additive approximation algorithm to a *2-pseudodeterministic algorithm*, a probabilistic polynomial-time algorithm that outputs *two canonical* values with high probability. To capture this, Goldreich introduced the notion of *multi-pseudodeterminism* [18]. A *k-pseudodeterministic* algorithm is a probabilistic-polynomial time

[1] In the literature, APEP is also referred to as CAPP.

algorithm that, for every input x, outputs a value from a set S_x of size at most k with probability at least $\frac{k+1}{k+2}$. The probability bound $\frac{k+1}{k+2}$ is crucial in the definition. See [18] for justification. Given that APEP admits a straightforward 2-pseudodeterministic algorithm, the difficulty in designing a pseudodeterministic algorithm for APEP is intriguing. In addition, the results we establish in the first part of the paper show that designing a pseudodeterministic algorithm for APEP will lead to the resolution of some long-standing open questions in complexity theory. Thus, it is important to understand the gap between multi-pseudodeterminism and pseudodeterminism. Is 2-pseudodeterminism more powerful than pseudodeterminism? More generally, are there multivalued functions that admit $(k+1)$-pseudodeterministic algorithms but do not admit k-pseudodeterministic algorithms?

Our second set of results establishes an exponential separation between $(k+1)$-pseudodeterminism and k-pseudodeterminism in certain query complexity models. In particular, as one of our main results, we establish that for every constant $k \geq 1$, there exists a function whose $(k+1)$-pseudodeterministic uniform query complexity is constant, but its k-pseudodeterministic non-adaptive query complexity is $\Omega(n)$. We utilize *Sperner's lemma* to establish these lower bounds.

Remark. Pseudodeterminism can be seen as a finer notion of derandomization. The study of pseudodeterminism is only relevant in the scenario where we are unable to construct pseudorandom generators. Indeed, if pseudorandom generators that fool linear-size Boolean circuits exist, various classes of probabilistic algorithms, including probabilistic search algorithms and multi-pseudodeterministic algorithms can be made deterministic (and hence trivially pseudodeterministic). Thus it is only meaningful to study the notion of pseudodeterminism without assumptions about the existence of pseudorandom generators.

1.3 Organization

The rest of the paper is organized as follows. In the next section, we give an overview of the results we establish in this paper. In Section 3, we introduce the necessary notation and definitions. In Section 4, we show our main conceptual contribution that equates the existence of pseudodeterministic algorithms for APEP to probabilistic promise problems. In Section 5, we show consequences of the equivalence theorems established in Section 4. In Section 6, we show equivalence between probabilistic search problems and the existence of pseudodeterministic algorithms for APEP. The equivalence and implication results that we establish in this paper are depicted in Figure 1. In Section 7 we establish lower bounds in the query complexity model. Finally, in Section 8 we discuss some open problems that are raised by this work.

2 OUR RESULTS

2.1 Pseudodeterminism and Promise Problems

Our main conceptual contribution is a new connection between the relatively new notion of pseudodeterminism and the well established notion of promise problems. The notion of promise problems was introduced in the work of Even, Selman, and Yacobi in the 1980's [12]. A promise problem Π is a pair of disjoint sets (Π_y, Π_n) of instances. An algorithm solving Π is only required to distinguish instances in Π_y from instances in Π_n. 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 many areas in theoretical computer science. Promise problems arise naturally in several settings such as hardness of approximations, public-key cryptography, derandomization, and completeness. For example, typically, hardness of approximation results are obtained by reducing NP to an appropriate promise problem (often called a gap problem). We refer the reader to the comprehensive survey article by Goldreich [16] for a treatment on the necessity and wide-ranging applicability of promise problems.

Many significant open questions regarding probabilistic complexity classes can be answered when we consider their promise versions. Does derandomization of BPP imply a derandomization of MA?; does derandomization of BPP imply Boolean circuit lower bounds?; does derandomization of the one-sided-error class RP imply derandomization of BPP?; do probabilistic complexity classes have complete problems? As of now, we do not know the answers to any of these questions. However, all these questions have an affirmative answer if we consider the promise-version of the corresponding complexity classes. For example, it is known that derandomizing PromiseBPP (refer to Section 3 for definitions of the promise classes) implies a derandomization of MA [21], and also implies Boolean circuit lower bounds [30]. It is known that derandomizing PromiseRP derandomizes BPP. Similarly, there exist promise problems that are complete for classes such as PromiseBPP, PromiseRP, and SZK [39].

The role of promise problems in circumventing certain deficiencies of language recognition problems is intriguing. A way to formalize the gap between promise problems and languages is by considering *solutions* to promise problems. A language $L \subseteq \{0,1\}^n$ is a solution to a promise problem $\Pi = (\Pi_y, \Pi_n)$ if $\Pi_y \subseteq L$ and $L \cap \Pi_n = \emptyset$. For a complexity class C, we say that Promise$C = C$ if every promise problem in Promise$C$ has a solution in C. Intuitively, when Promise$C$ equals C, then there is no gap between the class C and its promise counterpart.

A contribution of this paper is the discovery that the gap between PromiseBPP and BPP can be exactly characterized by the existence of pseudodeterministic algorithms for APEP.

Theorem 2.1. APEP *has a pseudodeterministic approximation algorithm if and only if* PromiseBPP = BPP.

APEP has an efficient 2-pseudodeterministic algorithm, so the algorithmic gap between 2-pseudodeterminism and pseudodeterminism in the context of APEP (and more generally in the context of approximation algorithms) precisely captures the gap between PromiseBPP and BPP.

We also show that a similar equivalence between pseudodeterminism and promise problems happens in other settings including zero-error probabilistic classes and randomized space bounded classes. In particular, we show that the notion of *safe pseudodeterminism* can be used to characterize the gap between PromiseBPP and the zero-error complexity class ZPP. These equivalence results have implications in the derandomization of MA which is discussed in the next subsection.

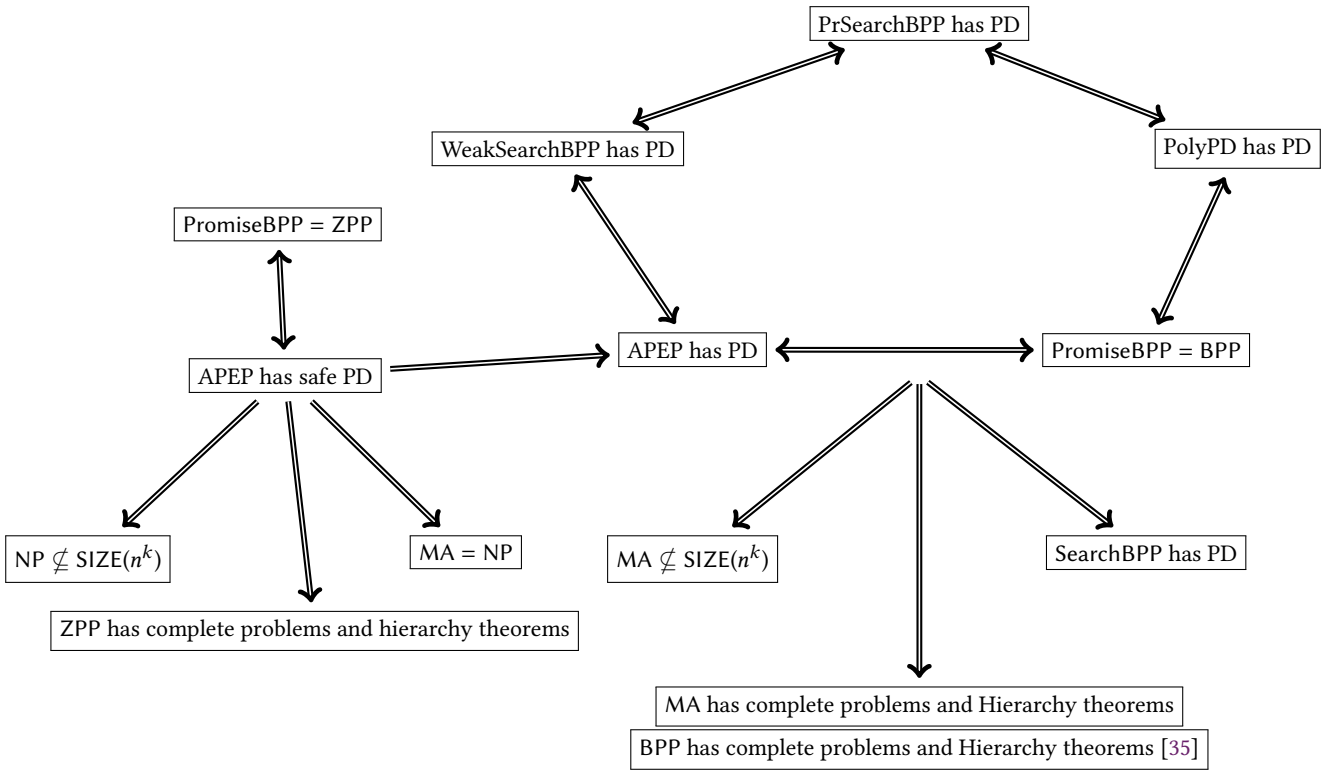

Figure 1: Equivalences and Implications established.

2.2 Consequences of the Equivalences

The equivalences we establish give rise to new results that connect pseudodeterminism to circuit lower bounds, probabilistic hierarchy theorems, SearchBPP, 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 cannot be solved by any Boolean circuit of a fixed polynomial size. For a constant k, let $\mathsf{SIZE}(n^k)$ denote the set of languages that can be solved using Boolean circuits of size $O(n^k)$. One of the central open questions in this area is to show that $\mathsf{NP} \not\subseteq \mathsf{SIZE}(n^k)$ for any k. That is, to show that for any k there is a language in NP that cannot be solved by Boolean circuits of size $O(n^k)$. Over the years, researchers have made steady progress on this question. Kannan [32] showed that $\Sigma_2^{\mathsf{P}} \not\subseteq \mathsf{SIZE}(n^k)$ for any k. Later, using techniques from learning theory, the Σ_2^{P} upper bound was improved to $\mathsf{ZPP}^{\mathsf{NP}}$ [6, 33] and later to S_2^{P} [7]. Vinodchandran showed that the class PP $\not\subseteq \mathsf{SIZE}(n^k)$ [41]. Santhanam [40] 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. Showing that $\mathsf{MA} \not\subseteq \mathsf{SIZE}(n^k)$ for any k is a significant open question in complexity theory. We show that this problem can in fact be solved by designing a pseudodeterministic algorithm for APEP. In particular, as a corollary to our main equivalence theorem, we show that if APEP has a pseudodeterministic algorithm then $\mathsf{MA} \not\subseteq \mathsf{SIZE}(n^k)$.

Theorem 2.2. *If* APEP *admits a pseudodeterministic approximation algorithm, then for any k,* $\mathsf{MA} \not\subseteq \mathsf{SIZE}(n^k)$.

In fact, we show that under the assumption, we get a slightly better result that $\mathsf{MA} = \exists \cdot \mathsf{BPP}$ (please refer to Section 3 for a definition of $\exists \cdot \mathsf{BPP}$) and thus $\exists \cdot \mathsf{BPP} \not\subseteq \mathsf{SIZE}(n^k)$. The above result strengthens the connection between pseudodeterministic algorithms and circuit lower bounds established in [9], where it was shown that designing a $\mathsf{BPP}_{tt}^{\mathsf{NP}}$ pseudodeterministic algorithm for problems in #NP would yield super-linear circuit lower bounds for languages in $\mathsf{ZPP}_{tt}^{\mathsf{NP}}$.

Derandomization. A significant open problem in complexity theory is whether MA can be derandomized to NP. It is known that any pseudorandom generator that derandomizes PromiseBPP can also derandomize MA to NP [20] and such pseudorandom generators are known to exist if E has problems with $2^{\epsilon n}$-circuit complexity [31]. We show that designing a certain type of pseudodeterministic algorithm for APEP will lead to a derandomization of MA to NP. A pseudodeterministic algorithm for APEP is called *safe* if the algorithm is allowed to output $\perp$, never outputs a wrong approximation, and outputs a canonical value with a probability at least 2/3. Safe pseudodeterministic algorithms are considered explicitly in the works of Goldreich, Goldwasser and Ron [19] and implicitly in [13].

Theorem 2.3. *If* APEP *has a safe pseudodeterministic approximation algorithm, then* MA = NP *and* NP $\not\subseteq$ SIZE(n^k).

The above theorem is a consequence of the equivalence that we show: APEP has a safe pseudodeterministic approximation algorithm if and only if PromiseBPP = ZPP. Combining this with Theorem 2.2, we obtain that if APEP has safe pseudodeterministic algorithm, then NP $\not\subseteq$ SIZE(n^k).

This theorem presents an alternate hypothesis that can derandomize MA to NP. We observe that the hypothesis that "APEP has a safe pseudodeterministic algorithm" is potentially weaker than the hypothesis "E does not have $2^{\epsilon n}$-size circuits". If E does not have $2^{\epsilon n}$-size circuits, then APEP has deterministic algorithms (and hence trivially have safe pseudodeterministic algorithms). Moreover, in the relativized world where EXP equals ZPP, EXP has polynomial-size circuits and APEP has safe-pseudodeterministic algorithms.

2.3 Completeness for Poly-Pseudodeterminism

In [10], it is shown that, in the context of pseudodeterminism, APEP is complete for functions that admit k-pseudodeterministic algorithms for any constant k. That is, multi-valued functions that admit k-pseudodeterministic algorithms for any constant k have pseudodeterministic algorithms if and only if APEP admits a pseudodeterministic algorithm. It was not clear how the techniques used in [10] can be extended to show similar results for functions that admit k-pseudodeterministic for a non-constant k. Here we improve this result to functions that admit $k(n)$-pseudodeterministic algorithms for any polynomial $k(n)$.

Theorem 2.4. APEP *admits a pseudodeterministic approximation algorithm if and only if every multi-valued function f that admits a $k(n)$-pseudodeterministic algorithm for a polynomial $k(n)$ has pseudodeterministic algorithms.*

We show this by showing that under the assumption APEP admits a pseudodeterministic approximation algorithm, every multi-valued function f that admits a $k(n)$-pseudodeterministic algorithm for a polynomial $k(n)$ is in SearchBPP. Earlier, in [10], it was shown that this assumption implies every problem in SearchBPP has pseudodeterministic algorithms.

2.4 Equivalence of Probabilistic Search Problems

A search problem is a relation $R \subset \Sigma^* \times \Sigma^*$. Given x, a string y is a witness for x, if $\langle x, y\rangle \in R$. In standard definition, a relation R is in SearchBPP if $R \in$ BPP and there is a probabilistic polynomial-time machine M that on an input x, outputs a y such that $\langle x, y\rangle \in R$ with high probability (if such y exists). However, earlier works have considered generalized versions of SearchBPP as they appear to be more useful in certain contexts. In particular, these works have studied variants that are obtained by weakening the requirement that that $R \in$ BPP (which we call WeakSearchBPP) [35] or considering promise versions (which we call PrSearchBPP) [17].

A significant open question in pseudodeterminism is whether SearchBPP has pseudodeterministic algorithms [14, 19]. One of our contributions is to clarify the relations among various notions of SearchBPP in regard to pseudodeterminism and show that APEP has pseudodeterministic algorithms if and only if the above-mentioned variants of SearchBPP have pseudodeterministic algorithms.

Theorem 2.5. *The following statements are equivalent.*

(1) APEP *admits a pseudodeterministic approximation algorithm.*
(2) PrSearchBPP *admits pseudodeterministic algorithms.*
(3) WeakSearchBPP *admits pseudodeterministic algorithms.*

2.5 Query Complexity Lower Bounds for Multi-Pseudodeterminism

The above set of results indicate that there is a significant gap between 2-pseudodeterministic algorithms and pseudodeterministic algorithms: on the one hand, there is a simple 2-pseudodeterministic algorithm for APEP. However, designing a pseudodeterministic algorithm for APEP will lead to many significant results in complexity theory. Thus, it is important to investigate the gap between multi-pseudodeterminism and pseudodeterminism. In general, we ask the following: Are there multivalued functions that admit $(k+1)$-pseudodeterministic algorithms but do not admit k-pseudodeterministic algorithms in some computational setting? We investigate this question in the *query complexity model*.

Goldreich, Goldwasser and Ron [19] studied the capabilities and limitations of pseudodeterministic algorithms in the query complexity model. Here, the underlying algorithm has oracle access to the bits of the input string x. The complexity of the algorithm is measured in terms of the number of queries made to the input. They showed that there exists a search problem R that can be solved using a constant number of queries by a probabilistic algorithm, but every pseudodeterministic algorithm has query complexity $\Omega(n)$. It turns out that there is a 2-pseudodeterministic algorithm that can solve R by making a constant number of queries. Recent work of Goldwasser, Impagliazzo, Pitassi and Santhanam [25] exhibited search problems that have constant query complexity for probabilistic algorithms but require $\Omega(\sqrt{n})$ queries for any pseudodeterministic algorithm.

Our next result is a fine separation on multi-pseudodeterminism in the *uniform query model* and *non-adaptive query model*. Details of the models are given in Section 3. Here we briefly introduce them for discussing the results and the proof outline. In the uniform query model, the algorithm accesses the input by making uniformly random queries. Non-adaptive query model is a generalization of the uniform query model where the algorithm can choose the queries in advance, although non-adaptively.

Theorem 2.6. *For every $k > 0$, there exists a function f whose $(k+1)$-pseudodeterministic query complexity, in the uniform query model, is $O(1)$, but its k-pseudodeterminstic query complexity, in the (more general) non-adaptive query models, is $\Omega(n)$.*

For establishing the lower bounds, we employ a technique that uses *Sperner's lemma*. While the original version of Sperner's lemma is concerned with the subdivision of an n-dimensional simplex into smaller simplices, we use a *cubical Sperner's lemma* [42]. We give a proof sketch in the uniform query model. We note that Sperner's lemma has been shown to be useful in establishing lower bounds. For example, see [8] in the context of communication complexity lower bounds.

Proof Sketch. For a binary string x, let $h(x)$ be its *hamming weight*—the number of 1's in x divided by the length of x. Consider the problem of approximating the hamming weight of a string x. It is well known that there is a probabilistic algorithm that outputs an approximation of $h(x)$ by making constant queries to x. Randomly query $O(1/\epsilon^2)$ indices of x and output the fraction of indices that are 1. The output will be ϵ-additive approximation of $h(x)$ with probability at least 2/3. Indeed, this algorithm works in the *uniform query model*. We consider the k-dimensional version of this problem, which we call the k-Dimensional Hamming Weight problem. Let x be a string of length n where n is divisible by k. View x as $x_1x_2\cdots x_k$ where each x_i is an n/k-bit substring of x. Consider the multivalued function f_ϵ defined as follows: $\langle a_1, a_2, \cdots, a_k\rangle \in f_\epsilon(x)$ if for every $1 \le i \le k$, $a_i \in [h(x_i) - \epsilon, h(x_i) + \epsilon]$. We will establish that f_ϵ has $\Omega(n)$ k-pseudodeterministic query complexity and has constant $(k+1)$-pseudodeterministic query complexity (in the uniform query model).

Consider the k-dimensional cube with side length n/k, partitioned into $(n/k)^k$ unit cubes. A vertex v of the partition is a lattice point $\langle a_1, \cdots, a_k\rangle$ where $0 \le a_i \le n/k$. The cubical Sperner's lemma states that for every *proper coloring* (see Definition 7.3 for definition of proper coloring) of the vertices of the partition with $k+1$ colors, there is a unit cube whose vertices have all $k+1$ colors.

With each lattice point $\vec{v} = \langle a_1, \cdots, a_k\rangle$ we associate a string $\vec{v}_x = y_1y_2\cdots y_k$, where each y_i is an n/k bit string. For each y_i, the first a_i bits are all ones and the rest of the bits are all zeros. Let A be a k-pseudodeterministic algorithm for the k-Dimensional Hamming Weight problem in the uniform query model. We first prove a *distance lemma* that states that if $\vec{u}$ and $\vec{v}$ are two adjacent vertices of the partition, then the output distribution of A with oracle access to $\vec{u}$ must be very close to the output distribution of A with oracle access to $\vec{v}$. Next, we color the lattice points of the partition using $(k+1)$ colors based on the most likely output of A. To define the coloring, we first design a suitable partition of the continuous unit cube $[0,1]^k$ into $(k+1)$ regions. Now, for the vertex $\vec{v} = \langle a_1, \cdots, a_k\rangle$, consider the most likely output of A with oracle access to the input string $\vec{v}_x$. If this output falls in the region i of the unit cube, then the vertex $\vec{v}$ of the partition gets color i.

We establish that the coloring is *proper* so that the conditions of Sperner's lemma holds. By the cubical Sperner's lemma, there is a unit cube of the partition whose vertices have all $k+1$ colors. Let $\vec{v_1}\cdots\vec{v_{k+1}}$ be such vertices of such a unit cube (i.e. each $\vec{v_i}$ has a distinct color). Since every $\vec{v_i}$ has a distinct color, it must be the case that the most likely outputs of $A^{\vec{v_i}x}$, $1 \le i \le k+1$ must be all distinct. Let us denote these most likely outputs with $o_1, o_2, \cdots, o_k$. Since A is a k-pseudodeterministic algorithm, $A^{\vec{v_i}x}$ must output o_i with probability at least $\frac{k+1}{k(k+2)}$. However, since the vertices $\vec{v_1}, \cdots, \vec{v_k}$ belong to the same k-dimensional unit hyper cube, the L_1 distance between any two vertices is bounded by k. Thus, by the distance lemma, the output distribution of $A^{\vec{v_1}x}$ must be close to the output distributions of $A^{\vec{v_2}x}, \cdots A^{\vec{v_k}x}$. This implies that $A^{\vec{v_1}x}$ must output o_1 with probability at least $\frac{k+1}{k(k+2)}$ and each of $o_2, \cdots o_k$ with probability close to $\frac{k+1}{k(k+2)}$. Thus the probability that $A^{\vec{v_1}x}$ outputs a member of $\{o_1, \cdots o_{k+1}\}$ is close to $\frac{(k+1)^2}{k(k+2)}$. Since $\frac{(k+1)^2}{k(k+2)} > 1$, we obtain a contradiction. The upper bound on the query complexity of $(k+1)$-pseudodeterministic algorithm follows due to a result from Goldreich [18].

□

We can generalize the lower bound result to a *nonadaptive query* model where the underlying algorithm is allowed to make nonadaptive queries to the input string x.

An earlier version of this paper with a subset of results (and a subset of authors) appeared as a technical report [11].

3 PRELIMINARIES

We assume standard notation and definitions from complexity theory [4]. 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 $1 - \delta$.

3.1 Pseudodeterminism

Definition 3.1. Acceptance Probability Estimation Problem: $\text{APEP}_{(\varepsilon,\delta)}$: 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 3.2 ([14, 18]). 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 $v_x \in f(x)$ such that $A(x) = v_x$ with probability at least 2/3. The function 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 [10, 18] for a proof).

Proposition 3.3. *For every $0 < \varepsilon, \delta < 1$, there is a 2-pseudodeterministic algorithm for* $\text{APEP}_{(\varepsilon,\delta)}$.

Goldreich, Goldwasser and Ron [19] studied the notion of *safe pseudodeterministic algorithms* that can be adapted to multi-valued functions and thus approximation algorithms.

Definition 3.4. A multi-valued function f has *safe pseudodeterministic algorithm*, if there is a probabilistic polynomial-time algorithm A such that for every x, there exist $v_x \in f(x)$ such that

- $\Pr[A(x) \in \{v_x, \perp\}] = 1$, and
- $\Pr[A(x) = v_x] \ge 2/3$.

We will need the following characterization of pseudodeterminism proved by Gat and Goldwasser [14].

Theorem 3.5. *A function admits a pseudodeterministic algorithm if and only if it is computable in* PF^{BPP}.

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

Proposition 3.6. *If* $\text{APEP}_{(1/100,1/8)}$ *has a pseudodeterministic algorithm, then for every* $0 < \varepsilon, \delta < 1$, $\text{APEP}_{(\varepsilon,\delta)}$ *has a pseudodeterministic algorithm.*

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

3.2 Promise Problems

*Definition 3.7 (*PromiseBPP*).* 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 3.8. Let C be a complexity class. We say that a promise (Π_y, Π_n) has a solution in C if there exists a language L in C such that $\Pi_y \subseteq L$ and $L \cap \Pi_n = \emptyset$.

Definition 3.9. 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 $\forall x$

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

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

The notion of $\exists \cdot L$ can be defined similarly.

Definition 3.10. 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 $\Pi = \exists \cdot \Pi'$. We say that a language $L \in \exists \cdot$ BPP, if there is a language $L' \in$ BPP such that $L = \exists \cdot L$.

Definition 3.11. 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 $< 1/3$.

3.3 Search Problems

In this work, all the relations considered are total, although the results established will hold for relations that are not necessarily total. We start with the most standard definition of a probabilistic search problem.

*Definition 3.12 (*SearchBPP *[19]).* A search problem R is in SearchBPP if there is a pair of probabilistic polynomial time algorithms A and B so that,

- For every x, $A(x) \in R(x)$ with probability $\geq 2/3$.
- B witnesses $R \in$ BPP; that is, if $(x, y) \in R$, then $B(x, y)$ accepts with probability $> 2/3$, and if $(x, y) \notin R$ then $B(x, y)$ rejects with probability $> 2/3$.

Definition 3.13. For a total multi-valued function f, we say that f is in SearchBPP if there is a relation R in SearchBPP so that $\forall x$, the witness set $R(x) \subseteq f(x)$.

We use the following result from [10].

Theorem 3.14. *If* APEP *admits a pseudodeterministic algorithm, then every problem in* SearchBPP *has a pseudodeterministic algorithm.*

In Section 6, we study certain generalized definitions of SearchBPP. We will give the needed definitions in that section.

3.4 Query Complexity

In this section, we define pseudodeterministic query complexity models. In these models, the algorithm cannot access input directly, but only through queries. Such models are used in the realm of sub-linear time computations.

Query Complexity Models. Let f be a multi-valued function. In the query complexity model, the algorithm A gets n as input and has oracle access to $x \in \Sigma^n$. The algorithm at any stage during the computation can make a *query* $i \in \{1, 2, \cdots, n\}$ to the oracle $x \in \Sigma^n$ and receives x_i as the answer. We denote the computation of A on $x \in \Sigma^n$ as $A^x(n)$. We say that a probabilistic algorithm A is a k-pseudodeterministic algorithm (k-PD in short) for f, if for every $x \in \Sigma^n$, there exists a set $S_x \subseteq f(x)$ of size $\leq k$, such that $A^x(n)$ outputs an element of S_x with probability at least $\frac{k+1}{k+2}$. For a $Q : \mathbb{N} \to \mathbb{N}$, we say that the query complexity of A is $Q(n)$, if $A^x(n)$ makes at most $Q(n)$ queries for every $x \in \Sigma^n$, for all random choices of A.

We consider the following three models for accessing the oracle x: the *uniform model*, the *non-adaptive model* and the *adaptive model*. In the uniform model, the queries made by the probabilistic algorithm A are generated uniformly at random from $\{1, \ldots, n\}$. In the non-adaptive model, at the beginning of the computation, the algorithm $A^x(n)$, picks a random string r and generates queries $i_1, i_2, \ldots, i_\ell$ and obtain answers $x_{i_1}, \ldots x_{i_\ell}$. After that, the algorithm does not make any more queries. Note that the queries generated can depend on the randomness of A. In the adaptive model, the algorithm picks a random string r, and makes adaptive queries to x (that is, the i^{th} query can depend on the the answer to the $(i-1)^{th}$ query). We use $Q^{unif}(A)$ to denote the query complexity of A in the uniform model, $Q^{||}(A)$ to denote the query complexity of A in the non-adaptive model, and $Q(A)$ to denote the query complexity in the adaptive model.

4 PSEUDODETERMINISM AND PROMISE PROBLEMS

4.1 Main Equivalence

Theorem 4.1. APEP *has a pseudodeterministic algorithm if and only if* PromiseBPP *has a solution in* BPP.

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:

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

Note that given x, we can construct C_x in time 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 4.2. *B has* BPP*-type behavior.*

Proof of Claim. 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 4.3. *L is a solution to the promise problem* Π.

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

Suppose $x \in \Pi_n$. Then $\Pr[C_x(r) = 1] < 1/3$. Thus $A_{\text{ape}}(C_x)$ outputs a canonical value $v \leq 1/3 + 1/100 < 1/2$ with probability at least $7/8$, so 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 3.3, 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 [17].

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

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

We make the following two critical observations.

Observation 4.4. *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 4.5. *For every circuit C, there exists v such that $\langle C, v\rangle \in \Pi_y$ and $v = k\epsilon$ for some $k > 0$.*

Proof of Observation. 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 4.6. $\Pi \in$ PromiseBPP.

Proof of Claim. 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 of the theorem 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 4.5, 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 4.4, we have that $\ell\varepsilon \in (p_c + 2\varepsilon, p_c - 2\varepsilon)$. Thus APEP has a $(2\varepsilon, \delta)$, $\mathsf{PF}^{\mathsf{BPP}}$ approximation algorithm. This implies that APEP has a $(2\varepsilon, \delta)$ pseudodeterministic algorithm by Theorem 3.5. □

4.2 Safe Pseudodeterministic Algorithms

Definition 4.7 ([13, 19]). A multi-valued function f has a *safe pseudodeterministic algorithm*, if there is a probabilistic polynomial-time algorithm A such that for every x, there exist $v_x \in f(x)$ such that

- $\Pr[A(x) \in \{v_x, \perp\}] = 1$, and
- $\Pr[A(x) = v_x] \geq 2/3$.

Theorem 4.8 ([19]). *A multi-valued function f has a safe pseudodeterministic algorithm if and only if f is in* $\mathsf{PF}^{\mathsf{ZPP}}$.

We can extend Theorem 4.1 to safe pseudodeterministic algorithms (we omit the proof).

Theorem 4.9. APEP *admits safe-pseudodeterministic algorithms if and only if* PromiseBPP *has a solution in* ZPP.

4.3 Equivalence in Space Bounded Computations

We observe that the equivalence between promise problems and pseudodeterminism also holds in probabilistic space bounded computations. We assume standard definitions of log space classes including BPL where the associated probabilistic machines halt on all random choices. We also use standard notions of space bounded transducers when considering function computation using space bounded Turing machines.

*Definition 4.10 (*PromiseBPL*).* A promise problem $\Pi = (\Pi_y, \Pi_n) \in$ PromiseBPL if there exists a probabilistic logspace machine M such that $\forall x$

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

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

*Definition 4.11 (Automata Acceptance Probability Estimation Problem (*AAPEP*)).* Given a finite automata M over binary alphabet, and an integer n, compute an (ε, δ)-additive estimation of the probability of acceptance of M: $\Pr_{r\in\{0,1\}^n}[M(r) \text{ accepts}]$.

Theorem 4.12. *There is an randomized logspace algorithm that given an automata M, ε, δ, and n in unary, outputs an (ε, δ)-additive approximation of* $\Pr_{r\in\{0,1\}^n}[M(r)$ accepts$]$.

Techniques used to prove Theorem 4.1 can be adapted to prove the following equivalence.

Theorem 4.13. AAPEP *admits a pseudodeterministic approximation algorithm if and only if* PromiseBPL = BPL.

5 CONSEQUENCES OF THE EQUIVALENCES

5.1 Circuit Lower Bounds and Derandomization

Theorem 5.1. *If* APEP *admits pseudodeterministic approximation algorithms, then*

(1) *Every promise problem* $\Pi = (\Pi_Y, \Pi_N)$ *in* PromiseMA *has a solution in* MA.
(2) MA = $\exists \cdot$ BPP.
(3) *For any* k, MA $\not\subseteq$ SIZE(n^k)

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 4.1, 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 $\exists$BPP. Thus it follows that MA = $\exists \cdot$ BPP.
(3) Santhanam [40] 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.

Theorem 5.2. *If* APEP *admits safe-pseudodeterministic algorithms, then* MA = NP *and* NP *has problems that cannot be solved using $O(n^k)$-size Boolean circuits.*

Proof. Since MA = $\exists \cdot$ PromiseBPP, if APEP admits safe pseudodeterministic algorithms, then by Theorem 4.9, PromiseBPP has a solution in ZPP. Thus MA = $\exists \cdot$ ZPP which equals NP. The second part of the theorem follows by combining this with Theorem 5.1. □

We observe that there is a relativized world in which the hypothesis APEP has a safe-pseudodeterministic algorithm holds, but the hypothesis E does not have $2^{\epsilon n}$-size circuits does not hold.

Observation 5.3. *There is an oracle A relative to which* APEP *has pseudodeterministic algorithms, but* EXP *has polynomial-size circuits.*

Proof. Consider an oracle relative to which EXP equals ZPP []. Since APEP can be solved by $\mathsf{PF}^{\mathsf{EXP}}$, it follows that APEP is in $\mathsf{PF}^{\mathsf{ZPP}}$ and this has a safe-pseudodeterministic algorithm. Since ZPP $\subseteq$ P/poly, we have that the EXP has polynomial-size circuits. □

5.2 Complete Problems and Hierarchy Theorems

In this section, we show that hierarchy theorems for BPTIME and ZPTIME follow as consequences of Theorems 4.1 and 4.9. For BPTIME, this gives an alternate proof of the result in Lu, Olivera and Santhanam [35]. We need the following result due to Barak [5].

Theorem 5.4. *If* PromiseBPP *has a solution in* BPP*, then there exists a constant c such that for every time constructible function $t(n)$,* BPTIME$(t(n))$ *is a proper subset of* BPTIME$(t(n)^c)$.

Theorem 5.5. *If* APEP *has pseudodeterministic algorithms, then the hierarchy theorems for* BPTIME *hold. In particular,* BPTIME$(n^\alpha) \subsetneq$ BPTIME(n^β) *for constant $1 \le \alpha < \beta$.*

Proof. By Theorem 4.1, the hypothesis implies that PromiseBPP has a solution in BPP and by Theorem 5.4, there exists a $c > 0$ such that for every a, BPTIME(n^a) is a proper subset of BPTIME(n^{ac}). The theorem follows from applying standard padding arguments. □

We next observe that if APEP admits pseudodeterministic algorithms, then MA has complete problems.

Theorem 5.6. *If* APEP *admits pseudodeterministic algorithms, then* MA *has a complete language and thus there is a hierarchy theorem for* MA*, i.e.,* MATIME(n^α) *is a proper subset of* MATIME(n^β) *for $1 \le \alpha < \beta$.*

Proof. By Theorem 5.1, the hypothesis implies that MA = $\exists \cdot$ BPP, thus it suffices to exhibit a language that is complete for the class $\exists \cdot$ BPP. We consider circuits with two types of inputs nondeterministic inputs and regular inputs. We use C_{n+m} to denote circuits with $n+m$ inputs and for such curcuits we use C_y to denote the circuits obtained by fixing the first n inputs to y. Consider the following language

$$L = \{C_{n+m} \mid \exists y \in \Sigma^n \, A_{\text{ape}}(C_y) \ge 1/2\}$$

Let L' be a language in $\exists$BPP. There exists a language $L'' \in$ BPP and a polynomial p such that $x \in L$ if and only if there exists $y \in \Sigma^{p(|x|)}$ such that $\langle x, y\rangle \in L''$. Let M be a probabilistic machine that witness that L'' is in BPP. Let C_x be the circuit obtained by converting this machine into a circuit and hard wiring x. Note that C_x has two types of inputs y and the random strings of M.

Now if $x \in L$, there exists y such that M accepts $\langle x, y\rangle$ with probability at least $2/3$. Thus the acceptance probability of C_{xy} is at least $2/3$ and the canonical output of $A_{\text{ape}}(C_{xy})$ is at least $1/2$. Thus $C_x \in L$. If $x \notin L$, for every y, M accepts $\langle x, y\rangle$ with probability at

most 1/3 and the canonical output of $A_{\text{ape}}(C_{xy})$ for every y is less than 1/2. Thus L' reduces to L. It is easy to see that L is in $\exists \cdot \text{BPP}$, as A_{ape} outputs a canonical correct answer with probability at least 2/3. Using the ideas from [5], it follows that if MA has complete problems, then hierarchy results hold. □

A similar proof establishes the following.

Theorem 5.7. *If* APEP *has safe pseudodeterministic algorithms, then* ZPP *has a complete problem and hierarchy theorems for* ZPTIME *hold. In particular,* $\text{ZPTIME}(n^{\alpha}) \subsetneq \text{ZPTIME}(n^{\beta})$ *for constant* $1 \leq \alpha < \beta$.

5.3 Completeness for Poly-Pseudodeterminism

Theorem 5.8. *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 5.9. *$\exists v^* \in S_x$ such that $\Pr\left[M_f(x) = v^*\right] \geq \frac{1+1/k(n)}{k(n)+2}$. 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 3.6 and standard error reduction techniques.

Claim 5.10. *$\forall x, W_x \subseteq f(x)$ 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 $W_x \neq \phi$ □

Claim 5.11. *$R \in$ 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. □

Claim 5.12. *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 $v \notin W_x$

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

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

By Claim 5.9, 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. □

This completes the proof that f is in SearchBPP. □

Using the above result, we obtain the following corollary, which improves a result from [10].

Theorem 5.13. *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 3.14, if APEP admits pseudodeterministic algorithms, every problem in SearchBPP has a pseudodeterministic algorithm. □

6 EQUIVALENCE OF PROBABILISTIC SEARCH PROBLEMS

Several variants of probabilistic search problems have been considered in the literature. We define them here. We will show that making these variants pseudodeterministic is equivalent to making APEP pseudodeterministic.

Definition 6.1. A *search problem* is a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$. For every x, the witness set of x with respect to R is $R(x) = \{y \mid (x, y) \in R\}$. R is *total* if for every x, there exists a y such that

$\langle x, y\rangle \in R$. A *promise search problem* is a disjoint pair of relations (R_Y, R_N) where $R_Y, R_N \subseteq \{0,1\}^* \times \{0,1\}^*$. A promise search problem (R_Y, R_N) is *total* if R_Y is total.

Goldreich studied probabilistic, promise search problems [17].

*Definition 6.2 (*PrSearchBPP *[17]).* A promise search problem (R_Y, R_N) is in PrSearchBPP if there is a pair of probabilistic polynomial time algorithms A and B so that,

- For every x, $A(x) \in R_Y(x)$ with probability $\geq 2/3$.
- B witnesses $(R_Y, R_N) \in$ PromiseBPP, i.e. if $(x, y) \in R_X$, then $B(x, y)$ accepts with probability $> 2/3$, and if $(x, y) \in R_N$ then $B(x, y)$ rejects with probability $> 2/3$.

The following variation of SearchBPP is considered in [35] and is obtained by relaxing the requirement on the decision algorithm B. We denote this as WeakSearchBPP.

*Definition 6.3 (*WeakSearchBPP *[35]).* A search problem R is in WeakSearchBPP, if there exist a pair of probabilistic algorithms A and B such that

- For every x, $A(x) \in R_x$ with probability $\geq 2/3$
- For every $(x, y) \notin R$, $B(x, y)$ rejects with probability $\geq 2/3$. For every x, with probability more than $1/2$ over random choices of A, $B(x, A(x))$ accepts with probability $\geq 2/3$.

In all the above class of search problems, we say that problem is in the class witnessed by the pair (A, B) of probabilistic algorithms.

Here we show that APEP is "complete for" both PrSearchBPP and WeakSearchBPP with respect to pseudodeterminism. Earlier works have shown that a APEP can be cast as PrSearchBPP [17] and WeakSearchBPP [35].

Theorem 6.4. *The following statements are equivalent.*

(1) APEP *admits a pseudodeterministic approximation algorithm*
(2) PrSearchBPP *admits pseudodeterministic algorithms.*
(3) WeakSearchBPP *admits pseudodeterministic algorithms.*

7 QUERY COMPLEXITY LOWERBOUNDS

In this section, we show a fine separation result for multi-pseudodeterministic computations in the query complexity model. We start with the definition of the candidate problems that we use to exhibit the separation. For a string y, let Hamming(y) denote the fraction of 1's in y.

Definition 7.1 (k-Dimensional Hamming Weight*).* Let $x = y_1 y_2 \cdots y_k$ be an n bit string where each y_i is of length n/k. Given $\epsilon < 1/2$, output $\langle a_1, a_2, \ldots, a_k\rangle$ such that for every $1 \leq i \leq k$, $|a_i - \text{Hamming}(y_i)| \leq \epsilon$.

For the rest of the section we work with $\epsilon = 1/4$.

7.1 Lower Bound in the Uniform Query Model

Theorem 7.2. *For the* k-Dimensional Hamming Weight *problem*

- *There exists a* $(k+1)$*-pseudodeterministic algorithm with query complexity* $O(k^5/\epsilon^2)$ *in the uniform query model.*
- *For every* k*-pseudodeterministic algorithm* A *that solves the* k-Dimensional Hamming Weight *problem,* $Q^{unif}(A) \in \Omega(n/k^4)$.

Clearly, for $k = 1$, there is 2-pseudodeterministic algorithm B such that for every y, the algorithm B^y makes $O(1/\epsilon^2)$ uniformly chosen queries and outputs a value h with $|h - \text{Hamming}(y)| \leq \epsilon$. Using this, it is easy to design a 2^k-pseudodeterminsitic algorithm for the k-Dimensional Hamming Weight problem that makes $\tilde{O}(k/\epsilon^2)$ queries. Goldreich [18], using a randomized rounding technique showed that there is a $(k+1)$-pseudodeterministic algorithm makes $O(k^5/\epsilon^2)$ queries in the uniform query model.

To prove the lower bound, we focus our attention on an explicit family of strings which we define now. For $0 \leq i \leq n$, let y_i be the string whose first i bits are 1, and the rest of the bits are 0. We will show that every k-pseudodeterministic algorithm for the k-Dimensional Hamming Weight problem has high query complexity even when restricted to the above family of input strings.

Consider the k-dimensional hypercube with side length n/k. Consider the standard partition of this hypercube into $(n/k)^k$-many unit hypercubes. Note that the set of vertices of these hypercubes is $\{0, 1, 2, \ldots, \frac{n}{k}\}^k$ (a subset of the integer lattice). For each such vertex/lattice point $\vec{v} = \langle a_1, \ldots, a_k\rangle$, we associate the string $\vec{v_x} = y_{a_1} y_{a_2} \cdots y_{a_k}$.

Sperner's Lemma. An essential ingredient of the proof of the theorem is the cubical Sperner's lemma [34, 42] which we state next.

Definition 7.3. Let $\mathcal{G}_k$ be the standard partition of the k-dimensional hypercube with integer side length p into unit cubes. Let Δ_p be the set of vertices/lattice points of the partition. For any vertex $\vec{v} \in \Delta_p$, let $\vec{v}[i]$ denote the i^{th} coordinate of $\vec{v}$. Let $C : \Delta_p \to \{1, \ldots, k+1\}$ be a function. We say that C is *proper coloring* function if it satisfies both of the following:

(1) if $\vec{v}[i] = 0$, then $C(\vec{v}) \leq i$
(2) if $\vec{v}[i] = p$, then $C(\vec{v}) \neq i$.

Lemma 7.4 (Cubical Sperner's Lemma [34, 42]). *For every proper coloring of* $\mathcal{G}_k$*, there is a unit cube of the partition whose vertices contain all the* $k+1$ *distinct colors.*

Now we proceed with the proof of Theorem 7.2.

Proof of Theorem 7.2. Let A be a k-pseudodeterministic algorithm for the k-Dimensional Hamming Weight in the uniform query model that makes $s(n)$ queries. Let $\vec{u}$ and $\vec{v}$ be two adjacent lattice points (i.e. they differ by 1 at exactly one coordinate, and at all other coordinates they have the same value). We claim the following:

Claim 7.5. $d_{TV}(A^{v_x}, A^{u_x}) \leq \frac{s(n)}{n}$.

Proof of Claim. Assume that $\vec{u} = \langle a_1, a_2, \ldots, a_{j-1}, a, a_{j+1}, \ldots, a_k\rangle$ and $\vec{v} = \langle a_1, a_2 \ldots, a_{j-1}, a+1, \ldots, a_k\rangle$ (so $\vec{u}$ and $\vec{v}$ differ on the jth coordinate by 1).

Recall that $u_x = y_{a_1} \cdots y_{a_{j-1}} y_a \cdots y_k$ and $v_x = y_{a_1} \cdots y_{a_{j-1}} y_{a+1} \cdots y_k$. The $j$the substring u_x is y_a and the jth substring of v_x is y_{a+1}. All other substrings of u_x and v_x are the same. By definition, $y_a = 1^a 0^{n/k-a}$ and $y_{a+1} = 1^{a+1} 0^{n/k-(a+1)}$. Thus the strings u_x and v_x differ at exactly one index (index $\frac{n}{k}(j-1) + (a+1)$). Now, observe that the behaviour of A^{v_x} and A^{u_x} differ only when A generates this index as a query. Since A is generating its queries uniformly at random, the probability that A queries this index is at most $s(n)/n$. Thus, the claim follows. □

From the above claim, we get the following corollary using the triangle inequality.

Corollary 7.6. *Let $\vec{u}$ and $\vec{v}$ be two lattice points. If* $\Pr[A^{u_x} = \vec{z}] = \rho$, *then*

$$\Pr[A^{v_x} = \vec{z}] \geq \rho - \left(||\vec{u} - \vec{v}||_1 \times \frac{s(n)}{n}\right)$$

.

We now give an outline of how Sperner's lemma is used. Consider the k-dimensional cube with the side length $p = n/k$ and let $\mathcal{G}_k$ be the partition of it into unit cubes and Δ_p be the set vertices of the partition. We will give a proper coloring of $\mathcal{G}_k$ based on the output of A. More precisely, for any lattice point $\vec{v}$, let $\vec{z}$ be the most likely output of A^{v_x} (if there are multiple, use any one of them). We will first give a coloring C_{aux} on $[0,1]^k$ which we call a *auxiliary coloring*. Define $C(\vec{v}) = C_{aux}(\vec{z})$. We will first argue that C is a proper coloring. Hence by Sperner's lemma, there is a unit cube containing all $k+1$ distinct colors. Hence for these $k+1$ vertices, A has to output at least $k+1$ distinct values each with probability $> \frac{(k+1)}{k(k+2)}$. However, since all the points on vertices of the unit cube are close to each other, we can arrive at a contradiction using Corollary 7.6. Now we give details.

Define $C_{aux} : [0,1]^n \to \{1, 2, \ldots, k+1\}$ as follows: for a $\vec{z} = \langle z_1, \ldots, z_k \rangle$

$$C_{aux}(z_1, \ldots, z_k) = \begin{cases} i : & z_1, \ldots, z_{i-1} \geq \frac{1}{2}, z_i < \frac{1}{2} \\ k+1 : & z_1, \ldots, z_k \geq \frac{1}{2} \end{cases}$$

Now we define the coloring of Δ_p. Consider a lattice point $\vec{v}$ of the partition, and and let $\vec{z}$ be the most probable out of A^{v_x} (break the ties arbitrarily if there are multiple most probable outputs).

$$C(\vec{v}) = C_{aux}(\vec{z})$$

See Figure 2 for an illustration of C_{aux} and C in the two dimensional case.

Lemma 7.7. *C is a proper coloring of Δ_p.*

Proof of Lemma. Consider a lattice point $\vec{v} = \langle a_1, \ldots, a_k \rangle$, and let $\vec{z} = \langle z_1, z_2, \ldots, z_k \rangle$ be the most likely output of A^{v_x}. We need to show (1) $a_i = 0 \Rightarrow C(\vec{v}) \leq i$ and (2) $a_i = n/k \Rightarrow C(\vec{v}) \neq i$.

Assume $a_i = 0$. Note that the ith substring of v_x is the all-zeros string y_0 whose hamming weight is 0. Note that since $|z_i - \text{Hamming}(y_0)| \leq \epsilon$, we have $z_i \leq \epsilon < \frac{1}{2}$. Suppose that $C_{aux}(\langle z_1, \cdots z_k \rangle) = j > i$, then it must be the case that $z_1, \ldots, z_{j-1} \geq 1/2$, thus in particular $z_i \geq 1/2$, and this is a contradiction. Thus $C(\vec{v}) = C_{aux}(\vec{z}) \leq i$.

Suppose that $a_i = p = n/k$. Now, the ith substring of v_x is the all ones string $y_{n/k}$ whose hamming weight is 1. Thus it must be the case that the ith component, z_i, of the output of A^{v_x} is $\geq 1 - \epsilon > 1/2$. Suppose that $C(\vec{v}) = i$, this means that $C_{aux}(\langle z_1, \ldots, z_k \rangle) = i$. However $C_{aux}(\langle z_1, \cdots, z_k \rangle) = i$ implies that $z_i < 1/2$ and this is a contradiction. Thus $C(\vec{v}) \neq i$. □

By cubical Sperner's Lemma, there must be a unit cube in the partition with vertices where all $k+1$ colors are present. Let $\vec{v}_1, \ldots, \vec{v}_{k+1}$ be vertices of such a unit cube with colors $1, \ldots, k+1$ respectively. Let $\vec{z}_1, \ldots \vec{z}_{k+1}$ be the most likely outputs of $A^{v_{1x}}, A^{v_{2x}} \cdots A^{v_{kx}}$ respectively. Since the colors of $\vec{v}_1, \cdots \vec{v}_k$ are all distinct from each other, it must be the case that $\vec{z}_1, \ldots, \vec{z}_{k+1}$ are all distinct. We have the following:

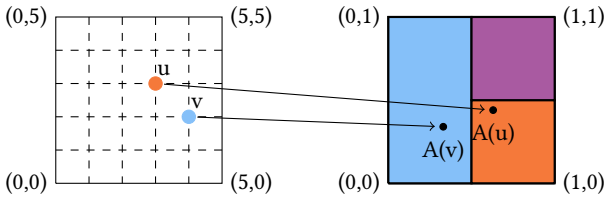

Figure 2: $\vec{u}$ is colored orange because $A(\vec{u})$ is in the orange area. Likewise, $\vec{v}$ is colored blue because $A(\vec{u})$ is in the blue area.

$$\Pr[A^{v_{1x}} = \vec{z}_1] \geq \frac{k+1}{k(k+2)}$$

$$\Pr[A^{v_{1x}} = \vec{z}_j] \geq \frac{k+1}{k(k+2)} - \frac{k \cdot s(n)}{n} \text{ for all } 2 \leq j \leq k+1$$

The first inequality is due to the fact that A is k-pseudodeterministic and hence the most likely output appears with probability at least $\frac{(k+1)}{k(k+2)}$. The second set of inequalities are a consequence of Corollary 7.6 and the fact that L_1 distance between any two points on the k-dimensional unit cube is at most k. Thus,

$$\Pr[A^{v_x} = \vec{z} : \vec{z} \in \{\vec{z}_1, \ldots, \vec{z}_{k+1}\}] \geq \frac{(k+1)^2}{k(k+2)} - \frac{k^2 s(n)}{n}$$

When $s(n) < \frac{n}{k^2(k^2+2k)}$, the above probability is bigger than 1, which is a contradiction. Thus the theorem follows. □

7.2 Lower Bound in the Non-Adaptive Query Model

In this section, we extend the above result to the more general non-adaptive query model. For this we combine the ideas from the earlier section with the work of Goldreich, Goldwasser, and Ron [19].

Theorem 7.8. *In the non-adaptive query model, any k-pseudodeterministic algorithm A that solves the k-Dimensional Hamming Weight problem requires $Q^{||}(A) \in \Omega(n/k^4)$.*

Proof. Let A be a k-pseudodeterministic algorithm for the k-Dimensional Hamming Weight problem that makes $s(n)$ non-adaptive queries. Given $0 \leq \delta \leq 1$, an element $i \in \{1, \ldots, n\}$ is *δ-low* if A queries i with probability at most δ. Note that in the non-adaptive query model the distribution of queries is independent of the input/oracle string x.

Observation 7.9. *The number of queries that are δ-low is $\geq 2n\epsilon$ for $s(n) \leq (1 - 2\epsilon)n\delta$.*

Proof of Observation. Since the algorithm makes $s(n)$ queries, the number of queries that are not δ-low is at most $s(n)/\delta$. If $s(n) \leq (1 - 2\epsilon)n\delta$, the number of δ-low queries is at least $2n\epsilon$. □

We now prove that if two strings differ on a low indices, then the algorithm will give similar approximations.

Lemma 7.10. *Let s be a δ-low query. Let p and q be two n/k-bit strings such that p and q differ only at the sth index. Let $p_1, p_2, \ldots, p_{i-1}, p_{i+1}, \ldots, p_k$ be n/k-bit strings.*

$$d_{TV}(A^{p_1 \cdots p_{i-1} p p_{i+1} \cdots p_k}, A^{p_1 \cdots p_{i-1} q p_{i+1} \cdots p_k}) \leq \delta$$

Proof of Lemma. Since s is queried with probability at most δ, and for the rest of the queries the oracle answers are the same, the claim follows. □

First we define a class of functions which is used to prove the lower bound. Let $s_1, s_2, \ldots, s_L$ be a set of δ-low queries (where $L = 2n$) such that $s_1 < s_2 < \cdots < s_L$. For every $1 \leq i \leq L$, we define a string p_i as follows: for $1 \leq j \leq i$, the s_jth bit of p_i is 1 and all remaining bits are 0. Consider the k-dimensional cube with side length L, partitioned into L^k-many unit cubes. Let Δ_L be the set of all vertices/lattice points of the partition. Consider a vertex $\vec{v} = \langle a_1, \ldots, a_k \rangle$ of the partition. We associate a string $\vec{v_x} = p_{a_1} \cdots p_{a_k}$ to $\vec{v}$ as before.

As before, we define an auxiliary coloring $C_{aux} : [0,1]^n \to \{1, 2, \ldots, k+1\}$ as follows. For $\vec{z} = \langle z_1, \ldots, z_k \rangle$,

$$C_{aux}(\langle z_1, \ldots, z_k \rangle) = \begin{cases} i : & z_1, \ldots, z_{i-1} \geq \epsilon, z_i < \epsilon \\ k+1 : & z_1, \ldots, z_k \geq \epsilon \end{cases}$$

We define a coloring of the partition of the hypercube based on C_{aux}. Let $\vec{v}$ be a vertex of the partition, and let $\vec{z}$ be the most probable output of $A(v_x)$.

$$C(\vec{v}) = C_{aux}(\vec{z})$$

The proof of the following lemma is similar to the proof of Lemma 7.7.

Lemma 7.11. *C is a proper coloring of the partition of the hypercube.*

By the Cubic Sperner Lemma, there exists a unit cube where all $k+1$ colors appear on the vertices. Consider such a unit cube and let $\vec{v}_1, \ldots, \vec{v}_{k+1}$ be vertices of the unit cube such that $C(\vec{v}_i) = i$. Let $\vec{z}_1, \ldots, \vec{z}_{k+1}$ be the most likely outputs of $A^{v_{1x}}, \ldots, A^{v_{k+1x}}$. Since the colors of $\vec{v}_1, \ldots, \vec{v}_{k+1}$ differ from each other, it must be the case that all $\vec{z}_i$s $1 \leq i \leq k+1$ are distinct.

Lemma 7.12. *For every i, j: $d_{TV}(A^{v_{ix}}, A^{v_{jx}}) \leq k\delta$.*

Proof of Lemma. Let $\vec{v}_i = \langle a_1, \ldots, a_k \rangle$ and $\vec{v}_j = \langle b_1, \ldots, b_k \rangle$. Note that a_i and b_i differ by at most 1 as $\vec{v}_i$ and $\vec{v}_j$ are the vertices of the same unit hypercube. Note that $v_{ix} = p_{a_1} \cdots p_{a_k}$ and $v_{jx} = \langle p_{b_1} \cdots p_{b_k} \rangle$. Since a_i and b_i differ by at most 1, the strings p_{a_i} and p_{b_i} differ on at most one index s which is a δ-low query. Hence by triangle inequality and Lemma 7.10, we obtain that

$$d_{TV}(A^{v_{ix}}, A^{v_{jx}}) \leq k\delta$$

□

Since A is k-pseudodeterministic and $\vec{z}_i$ is the most probable output of $A^{v_{ix}}$ for all $1 \leq i \leq k+1$, we have

$$\Pr[A^{v_{ix}} = \vec{z}_i] \geq \frac{k+1}{k(k+2)}$$

By Lemma 7.12, for all $2 \leq j \leq k+1$,

$$\Pr[A^{v_{1x}} = \vec{z}_j] \geq \frac{k+1}{k(k+2)} - k\delta$$

Thus

$$\Pr[A^{v_{1x}} \in \{\vec{z}_1, \cdots, \vec{z}_{k+1}\}] \geq \frac{(k+1)^2}{k(k+2)} - k^2\delta$$

We chose $\delta < \frac{1}{k^2(k^2+2k)}$, and the above probability is > 1 leading to a contradiction. Thus A cannot be k-pseudodeterministic if it is making $s(n) \leq (1-2\epsilon)n\delta$ non-adaptive queries. For large enough n, we have $s(n) \leq n/k^4$. □

8 FUTURE DIRECTIONS

It would be interesting to further explore the connections between pseudodeterminism and probabilistic complexity classes. Does a pseudodeterministic algorithm for APEP imply that MA can be derandomized to NP? More generally, does a pseudodeterministic algorithm for APEP imply a safe pseudodeterministic algorithm for APEP? The most important open question is to design a pseudodeterministic algorithm for APEP. For query complexity lower bounds, it is easy to see that the nonadaptive lower bound implies a $\Omega(\log n)$ lower bound in the adaptive query model. Improving this lower bound is an interesting open problem. We conjecture that the correct lower bound in the adaptive query model is indeed $\Omega(n)$.

9 DEDICATION TO ALAN SELMAN (1941–2021)

We dedicate this work to Alan Selman who made seminal contributions to both notions studied in this work—promise problems and pseudodeterminism—in the context of nondeterministic computations. As mentioned in the introduction, the notion of promise problems continues to play a critical role in several areas. Here we briefly discuss Alan's contributions to pseudodeterminism. The notion of pseudodeterminism introduced by Gat and Goldwasser [14] asks whether a probabilistic computation can be made to output a canonical value. Alan and his co-authors considered an analogous question in the context of nondeterministic computations. Let f be a partial multi-valued function. We say that f is in NPSV if there is a nondeterministic polynomial-time machine M such that for every x, if $f(x) \neq \emptyset$, then there is a canonical value $v_x \in f(x)$ such that every path of M either outputs v_x or $\perp$, and at least one path outputs v_x. If $f(x) = \emptyset$, then every path of M outputs $\perp$. Thus, NPSV captures pseudodeterminism in the context of nondeterministic computations. A natural question that arises is whether f_{SAT} is in NPSV? Here $f_{\mathsf{SAT}}(\phi)$ is the set of satisfying assignment of ϕ if $\phi \in$ SAT and is undefined if $\phi \notin$ SAT. A fundamental result due to Alan and his coauthors is that if f_{SAT} is in NPSV, then the polynomial-time hierarchy collapses [28].

In latter works, Alan and his co-authors generalized NPSV to capture multi-pseudodeterminism in the context of nondeterministic computations. A multivalued function f is in NPkV, if there is a nondeterministic polynomial-time machine M such that for every x, if $f(x) \neq \emptyset$, then there is set S_x of size at most k such that every path of M outputs a value in S_x or $\perp$, and at least one path outputs a value from S_x. If $f(x) = \emptyset$, then every path of the M outputs $\perp$. Alan and his co-authors generalized the above result to show that if

every function in NP(k + 1)V is in NPkV, then the polynomial-time hierarchy collapses [36].

On a personal note, Pavan would like to express his gratitude to Alan for introducing him to the beautiful world of complexity theory and for stressing the significance of coming up with "simpler proofs".

ACKNOWLEDGMENTS

We thank the reviewers for helpful comments and suggestions. We thank Jamie Radcliffe for discussions related to Sperner's lemma and related topics. This work is supported in part by NSF grants 1934884, 2130608, and 2130536.

REFERENCES

[1] M. Agrawal, N. Kayal, and N. Saxena. 2004. PRIMES in P. *Ann. of Math. (2)* 160, 2 (2004), 781–793.
[2] Nima Anari and Vijay V. Vazirani. 2020. Matching Is as Easy as the Decision Problem, in the NC Model. In *11th Innovations in Theoretical Computer Science Conference, ITCS (LIPIcs, Vol. 151)*. 54:1–54:25.
[3] Andrea Arcuri and Lionel C. Briand. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *33rd Int. Conf. on Soft. Engg. (ICSE)*. 1–10.
[4] S. Arora and B. Barak. 2009. *Computational Complexity - A Modern Approach*. Cambridge University Press.
[5] Boaz Barak. 2002. A Probabilistic-Time Hierarchy Theorem for "Slightly Non-uniform" Algorithms. In *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002 (LNCS, Vol. 2483)*. Springer, 194–208.
[6] N. H. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. 1996. Oracles and Queries That Are Sufficient for Exact Learning. *J. Comput. Syst. Sci.* 52, 3 (1996), 421–433.
[7] J-Y. Cai. 2001. $S_2^P \subseteq ZPP^{NP}$. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*. 620–629.
[8] J. Y. Cai, R. Lipton, L. Longpré, M. Ogihara, K. Regan, and D. Sivakumar. 1995. Communication Complexity of Key Agreement on Small Ranges. In *STACS*. 38–49.
[9] Peter Dixon, A. Pavan, and N. V. Vinodchandran. 2018. On Pseudodeterministic Approximation Algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018 (LIPIcs, Vol. 117)*. 61:1–61:11.
[10] Peter Dixon, A. Pavan, and N. V. Vinodchandran. 2021. Complete Problems for Multi-Pseudodeterministic Computations. In *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021 (LIPIcs, Vol. 185)*. 66:1–66:16.
[11] Peter Dixon, Aduri Pavan, and N. V. Vinodchandran. 2021. Promise Problems Meet Pseudodeterminism. *Electron. Colloquium Comput. Complex.* 28 (2021), 43. https://eccc.weizmann.ac.il/report/2021/043
[12] Shimon Even, Alan L. Selman, and Yacov Yacobi. 1984. The Complexity of Promise Problems with Applications to Public-Key Cryptography. *Inf. Control.* 61, 2 (1984), 159–173.
[13] Eran Gat. 2009. On the canonization of probabilistic algorithms, MS Thesis, Weizmann Institute of Science.
[14] E. Gat and S. Goldwasser. 2011. Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications. *Electronic Colloquium on Computational Complexity (ECCC)* 18 (2011), 136.
[15] Michel Goemans, Shafi Goldwasser, and Dhiraj Holden. 2019. Doubly-Efficient Pseudo-Deterministic Proofs. *arXiv* (2019).
[16] Oded Goldreich. 2006. On Promise Problems: A Survey. In *Theoretical Computer Science, Essays in Memory of Shimon Even (Lecture Notes in Computer Science, Vol. 3895)*, Oded Goldreich, Arnold L. Rosenberg, and Alan L. Selman (Eds.). Springer, 254–290.
[17] Oded Goldreich. 2011. In a World of P=BPP. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, Oded Goldreich (Ed.). Lecture Notes in Computer Science, Vol. 6650. Springer, 191–232.
[18] Oded Goldreich. 2019. Multi-pseudodeterministic algorithms. *Electronic Colloquium on Computational Complexity (ECCC)* 26 (2019), 12.
[19] O. Goldreich, S. Goldwasser, and D. Ron. 2013. On the possibilities and limitations of pseudodeterministic algorithms. In *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*. 127–138.
[20] O. Goldreich and D. Zuckerman. 1997. Another proof that BPP subseteq PH (and more). *Electronic Colloquium on Computational Complexity (ECCC)* 4, 45 (1997).
[21] Oded Goldreich and David Zuckerman. 2011. Another Proof That BPP ⊆ PH (and More). In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, Oded Goldreich (Ed.). Lecture Notes in Computer Science, Vol. 6650. Springer, 40–53.
[22] S. Goldwasser and O. Grossman. 2017. Bipartite Perfect Matching in Pseudo-Deterministic NC. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*. 87:1–87:13.
[23] S. Goldwasser, O. Grossman, and D. Holden. 2017. Pseudo-deterministic Proofs. *CoRR* abs/1706.04641 (2017).
[24] Shafi Goldwasser, Ofer Grossman, Sidhanth Mohanty, and David P. Woodruff. 2020. Pseudo-Deterministic Streaming. In *11th Innovations in Theoretical Computer Science Conference, ITCS (LIPIcs, Vol. 151)*, Thomas Vidick (Ed.). 79:1–79:25.
[25] Shafi Goldwasser, Russell Impagliazzo, Toniann Pitassi, and Rahul Santhanam. 2021. On the Pseudo-Deterministic Query Complexity of NP Search Problems. In *36th Computational Complexity Conference, CCC 2021 (LIPIcs, Vol. 200)*, Valentine Kabanets (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 36:1–36:22.
[26] O. Grossman. 2015. Finding Primitive Roots Pseudo-Deterministically. *Electronic Colloquium on Computational Complexity (ECCC)* 22 (2015), 207.
[27] Ofer Grossman and Yang P. Liu. 2019. 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*. SIAM, 606–620.
[28] Lane A. Hemaspaandra, Ashish V. Naik, Mitsunori Ogihara, and Alan L. Selman. 1996. Computing Solutions Uniquely Collapses the Polynomial Hierarchy. *SIAM J. Comput.* 25, 4 (1996), 697–708. https://doi.org/10.1137/S0097539794268315
[29] Dhiraj Holden. 2017. A Note on Unconditional Subexponential-time Pseudo-deterministic Algorithms for BPP Search Problems. *CoRR* abs/1707.05808 (2017).
[30] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. 2002. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.* 65, 4 (2002), 672–694.
[31] Russell Impagliazzo and Avi Wigderson. 1997. *P = BPP* if *E* Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*. ACM, 220–229.
[32] R. Kannan. 1982. Circuit-size Lower bounds and Non-reducibility to Sparse sets. *Information and Control* 55 (1982), 40–56.
[33] J. Köbler and O. Watanabe. 1998. New Collapse Consequences of NP Having Small Circuits. *SIAM J. Comput.* 28, 1 (1998), 311–324.
[34] H. W. Kuhn. 1960. Some Combinatorial Lemmas in Topology. *IBM Journal of Research and Development* 4, 5 (1960), 518–524. https://doi.org/10.1147/rd.45.0518
[35] Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. 2021. Pseudodeterministic Algorithms and the Structure of Probabilistic Time. In *STOC*. To Appear. ECCC Tech Report 21-039.
[36] Ashish V. Naik, John D. Rogers, James S. Royer, and Alan L. Selman. 1998. A Hierarchy Based on Output Multiplicity. *Theor. Comput. Sci.* 207, 1 (1998), 131–157.
[37] I. Oliveira and R. Santhanam. 2017. 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*. 665–677.
[38] Igor Carboni Oliveira and Rahul Santhanam. 2018. Pseudo-Derandomizing Learning and Approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018 (LIPIcs, Vol. 116)*. 55:1–55:19.
[39] Amit Sahai and Salil P. Vadhan. 2003. A complete problem for statistical zero knowledge. *J. ACM* 50, 2 (2003), 196–249.
[40] R. Santhanam. 2009. Circuit Lower Bounds for Merlin–Arthur Classes. *SIAM J. Comput.* 39, 3 (2009), 1038–1061.
[41] N. V. Vinodchandran. 2005. A note on the circuit complexity of PP. *Theor. Comput. Sci.* 347, 1-2 (2005), 415–418.
[42] Laurence A Wolsey. 1977. Cubical sperner lemmas as applications of generalized complementary pivoting. *Journal of Combinatorial Theory, Series A* 23, 1 (1977), 78–87. https://doi.org/10.1016/0097-3165(77)90081-4