Complete Problems for Multi-Pseudodeterministic Computations

Peter Dixon
Department of Computer Science, Iowa State University, USA.
tooplark@iastate.edu

A. Pavan
Department of Computer Science, Iowa State University, USA.
pavan@cs.iastate.edu

N. V. Vinodchandran
Department of Computer Science and Engineering, University of Nebraska, Lincoln USA.
vinod@cse.unl.edu

Abstract

We exhibit several computational problems that are *complete* for multi-pseudodeterministic computations in the following sense: (1) these problems admit 2-pseudodeterministic algorithms (2) if there exists a pseudodeterministic algorithm for any of these problems, then any multi-valued function that admits a k-pseudodeterministic algorithm for a constant k, also admits a pseudodeterministic algorithm. We also show that these computational problems are complete for *Search-BPP*: a pseudodeterministic algorithm for any of these problems implies a pseudodeterministic algorithm for all problems in Search-BPP.

2012 ACM Subject Classification Theory of computation → Probabilistic Computation; Theory of computation → Problems, reductions and completeness

Keywords and phrases Pseudodeterminism, Completeness, Collision Probability, Circuit Acceptance, Entropy Approximation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2021.71

Funding Supported in part by NSF grants 1934884, 1849053, 1849048

Acknowledgements We thank Oded Goldrecih for comments and suggestions on an earlier draft of this paper. We also thank anonymous reviewers for helpful comments.

1 Introduction

Consider the search problem of producing a witness that two multi-variate polynomials f and g over a field are different. A simple probabilistic polynomial-time algorithm for this problem randomly picks an element t from the domain and outputs it if $f(t) \neq g(t)$. Even though this algorithm is simple and efficient and the error probability can be made arbitrarily small, Gat and Goldwasser [3] pointed out a deficiency: two different runs of the algorithm can produce two different witnesses with very high probability. Well-known probabilistic algorithms for several search problems, such as finding a large prime number or computing generators of cyclic groups, also exhibit this deficiency. This raises the question of whether we can design a probabilistic algorithm for search problems that will output the same witness on multiple executions, with high probability. Motivated by the above question, Gat and Goldwasser [3] introduced the notion of *pseudodeterministic algorithms*[1]. Informally, a probabilistic algorithm M is pseudodeterministic if for every x, there exists a

[1] Originally termed Bellagio algorithms

unique 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 an algorithms 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 computation. It is not known whether derandomizing BPP to P implies derandomization of probabilistic search algorithms. However, BPP = P implies that pseudodeterministic search algorithms can be made deterministic [5].

Prior Work

Since its introduction, pseudodeterminism and its generalizations have received considerable attention. In particular, designing pseudodeterministic algorithms for problems that admit polynomial-time randomized algorithms but without known deterministic algorithms continues to be a key line of research with some success. Gat and Goldwasser showed that there exist polynomial-time pseudodeterministic algorithms for algebraic problems such as finding quadratic non-residues and finding certificates that two multivariate polynomials are different [3]. Goldwasser and Grossman exhibited a pseudodeterministic NC algorithm for computing matchings in bipartite graphs [8]. Grossman designed a pseudodeterministic algorithm for computing primitive roots whose runtime matches the best known Las Vegas algorithm [11]. Oliveira and Santhanam [13] showed that there is a sub-exponential time pseudodeterministic algorithm for generating primes that works at infinitely many input lengths.

Subsequent works extended pseudodeterminism to several other scenarios. Works of Goemans, Goldwasser, Grossman, and Holden investigated pseudodeterminism in the context of interactive proofs [9, 4]. Goldwasser, Grossman, Mohanty and Woodruff [10] investigated pseudodeterminism in the data stream model. They showed that certain streaming problems admit faster pseudodeterministic algorithms in comparison to their deterministic counterparts. They also obtain space lower bounds for sketch based pseudodeterministic estimation of ℓ_2 norm. Goldreich, Goldwasser, and Ron [5] investigated pseudodeterminism in the context of sublinear-time algorithms. Dixon, Pavan, and Vinodchandran [2] studied pseudodeterminism in the context of approximation algorithms and showed that making Stockmeyer's [16] well-known approximate counting algorithm pseudodeterministic will yield new circuit lower bounds. Oliveira and Santhanam studied pseudodeterminism in the context of learning algorithms and showed that some randomized learning algorithms can be made pseudodeterministic under certain complexity theoretic assumptions [14]. Since then a few generalizations of pseudodeterminism such as *reproducible algorithms*, *influential bit algorithms* and *multi-pseudodeterministic algorithms* have been introduced [12, 7].

Multi-Pseudodeterminism

Our main focus is on the notion of *multi-pseudodeterminism* recently introduced by Goldreich [7]. Consider the problem of estimating the average value of a function that is defined over a large but finite universe. It is well known that there is an efficient additive error, probabilistic approximation for this problem. So far, we do not know how to make this algorithm pseudodeterministic. Suppose that we relax the restriction of pseudodeterminism - instead of requiring that the algorithm outputs an unique approximation, the algorithm must output *one of two approximate values* with high probability. Then it is very easy to obtain such probabilistic algorithms [7]. Motivated by this, Goldreich introduced the notion of *multi-pseudodeterminism*.

For a positive integer k, a probabilistic algorithm A is k-pseudodeterministic if, for every input x, there exists a set S_x of size at most k such that $A(x)$ belongs to S_x with probability at least $\frac{k+1}{k+2}$. Thus a pseudodeterministic algorithm is 1-pseudodeterministic. Goldreich's work established several key properties of multi-pseudodeterministic algorithms. This work showed that, as with the case of probabilistic and pseudodeterministic algorithms, error reduction is possible for multi-pseudodeterministic algorithms. Goldreich's work also established an equivalence between multi-pseudodeterminism and reproducible algorithms introduced by Grossman and Liu [12] and presented a composition result for multi-pseudodeterministic algorithms.

Our Contributions

Our main focus is on the notion of *multi-pseudodeterminism*. The notion of multi-pseudodeterminism is especially interesting because there are computational problems that admit 2-pseudodeterministic algorithms for which we do not know of any pseudodeterministic algorithms. As mentioned earlier, it can be shown that randomized approximation algorithms can be made 2-pseudodeterministic (see Section 2). Thus it is significant to investigate the possibility of designing pseudodeterministic algorithms for problems that admit k-pseudodeterministic algorithms for small values of k.

Our main contribution is to show the existence of complete problems for multi-pseudodeterministic computations in the following sense: (1) these computational problems admit 2-pseudodeterministic algorithms, and (2) if there exists a pseudodeterministic algorithm for any of these problems, then all multi-valued functions that admit k-pseudodeterministic algorithms for a constant k, also admit pseudodeterministic algorithms.

The computational problems we consider are the following. We note that all of these problems admit 2-pseudodeterministic algorithms.

▶ **Definition 1** (Computational Problems).

- Collision Probability Estimation Problem. *Given a Boolean circuit* $C : \{0,1\}^n \to \{0,1\}$*, give a* (ε, δ)*-additive approximation of the collision probability of* C*.*
- Acceptance Probability Estimation Problem: *Given a Boolean circuit* $C : \{0,1\}^n \to \{0,1\}$*, give a* (ε, δ)*-additive approximation for* $\Pr_{x \in U_n}[C(x) = 1]$*.*
- Entropy Estimation Problem: *Given a Boolean circuit* $C : \{0,1\}^n \to \{0,1\}$*, give an* (ε, δ)*-additive approximation of the entropy of the distribution* $C(U_n)$*.*

We first show that if any of the above problems have a pseudodeterministic algorithm, then all 2-pseudodeterministic algorithms can be made pseudodeterministic and thus any (ε, δ)-approximation algorithm for a function f can be made pseudodeterministic.

Result 1: *If any of the problems from Definition 1 admits a pseudodeterministic algorithm, every function that has a* (ε, δ)*-approximation algorithm has a pseudodeterministic* (ε, δ)*-approximation algorithm.*

Next we extend this result to k-pseudodeterminism.

Result 2: *If any of the problems from Definition 1 admits a pseudodeterministic algorithm, then any multivalued function that admits a k-pseudodeterministic algorithm also admits a pseudodeterministic algorithm.*

Note that the above result holds for any multivalued function computation. *Search problems* are multivalued functions whose outputs have an efficient verification procedure. Much of the work on pseudodeterminism focuses on problem in the class *Search-BPP*: search

problems that have randomized algorithms whose outputs can be verified in BPP [6]. We extend the above result to problems in Search-BPP.

Result 3: *If any of the problems from Definition 1 admits a pseudodeterministic algorithm, then any problem in Search-BPP has a pseudodeterministic algorithm.*

Extending this result on Search-BPP to other classes of multivalued functions is an interesting question. Note that there are natural multivalued functions that admit efficient probabilistic algorithms but are not known to be in Search-BPP (because of the lack of an efficient verification procedure). For example the problem of outputting a Boolean function with high circuit complexity has a simple probabilistic algorithm but not known to be in Search-BPP.

2 Pseudodeterminism in Approximation Algorithms

In this section, we establish that Collision Probability Estimation Problem is complete. We first define the notions of pseudodeterminism and multi-pseudodeterminism for approximation algorithms formally. In general, an approximation algorithm can output different good approximations on different random choices. For an approximation algorithm A to be k-pseudodeterministic, A has to output with high probability at most k good approximations for any input.

▶ **Definition 2** (Multiplicative Approximation)**.** *Let f be a function whose range is the integers. We say that a probabilistic algorithm A is an (ε, δ)-multiplicative approximation algorithm for f if for every x, the random variable $A(x)$ has the following property:* $\Pr\left[\frac{f(x)}{(1+\varepsilon)} \leq A(x) \leq (1+\varepsilon)f(x)\right] \geq (1-\delta)$. *We say that A is an (ε, δ)* k-pseudodeterministic multiplicative approximation algorithm for f *if for every x there exists a set of integers V_x such that $|V_x| \leq k$ and for every $v \in V_x$*

$$\frac{f(x)}{(1+\varepsilon)} \leq v \leq (1+\varepsilon)f(x) \text{ and } \Pr[A(x) \in V_x] \geq 1-\delta.$$

When $k = 1$, we call the algorithm pseudodeterministic*.*

▶ **Definition 3** (Additive Approximation)**.** *Let f be a function whose range is $[0, 1]$. We say that a probabilistic polynomial-time algorithm A is an (ε, δ)-additive approximation algorithm for f if for every x, the random variable $A(x)$ has the following property:* $\Pr\left[(f(x) - \varepsilon \leq A(x) \leq f(x) + \varepsilon\right] \geq (1-\delta)$. *We say that A is an (ε, δ)* k-pseudodeterministic additive approximation algorithm for f *if for every x there exists a set $V_x \subseteq [0, 1]$ such that for all $v \in V_x$*

$$f(x) - \varepsilon \leq v \leq f(x) + \varepsilon \text{ and } \Pr[A(x) \in V_x] \geq 1-\delta.$$

When $k = 1$, we call the algorithm pseudodeterministic*.*

Remark. In general, for (ε, δ) approximation algorithms (additive or multiplicative), error reduction is possible when $\delta < 1/2$ and is bounded away from $1/2$. This is done by repeating the algorithm multiple times and taking the median value. Thus without loss of generality, we may assume that for every (ε, δ) approximation algorithm, $\delta \leq 1/2^n$. Similarly, the error probability of (ε, δ) pseudodeterministic approximation algorithms can be reduced to less than $1/2^n$.

Goldreich [7] observed that every additive error approximation algorithm can be made 2-pseudodeterministic; this extends to multiplicative approximation algorithms as well.

▶ **Proposition 4.** *Every optimization problem that admits an (ε, δ) multiplicative approximation algorithm admits a $(2\varepsilon + \varepsilon^2, \delta)$ 2-pseudodeterministic multiplicative approximation algorithm. Similarly, every optimization problem that admits an (ε, δ) additive approximation algorithm admits a $(2\varepsilon, \delta)$ 2-pseudodeterministic additive approximation algorithm.*

Proof. We give a proof for the multiplicative case. A similar argument holds for additive approximation algorithms. Let f be a function and let A be an (ε, δ)-multiplicative approximation algorithm for f. The proposed 2-pseudodeterministic algorithm B will first run $A(x)$ to get an approximation v. B outputs v_i from a set of points P (defined next) by choosing the smallest $v_i \in P$ that is larger than v. P is defined as the set of points $\{v_i \mid i \in \mathbb{N}\}$ where $v_i = \lceil (1+\varepsilon)^i \rceil$.

Observe that $v_{i+1} = (1+\varepsilon)v_i$ for all i. For any input x, $A(x)$ outputs a value in the range $\left[\frac{f(x)}{(1+\varepsilon)}, (1+\varepsilon)f(x)\right]$ and this range will contain at most 2 values from P. Hence B is 2-pseudodeterministic. Rounding up to the nearest value in P makes the approximation factor at most $(1+\varepsilon)^2 = 1 + 2\varepsilon + \varepsilon^2$.

◀

2.1 Completeness of Collision Probability Estimation

In this section, we prove that Collision Probability Estimation Problem is complete for approximation algorithms in the context of pseudodeterminism. We start with the following observation.

▶ **Proposition 5.** Collision Probability Estimation Problem *admits (ε, δ) 2-pseudodeterministic additive approximation algorithms for every $\varepsilon < 1$ and $\delta < 1$.*

Proof. We can estimate collision probability of C by generating $O(1/\varepsilon^2 \log 1/\delta)$ independent pairs of strings $\langle x_i, y_i \rangle$ and counting the number of times $C(x_i) = C(y_i)$. Simple application of Chernoff bound implies that this is a (ε, δ) additive approximation algorithm. By the previous proposition, we can convert this algorithm into a 2-pseudodeterministic algorithm.

◀

We now prove the main theorem of the section.

▶ **Theorem 6.** *There exists $\varepsilon' > 0$ such that if* Collision Probability Estimation Problem *has an (ε', δ)-pseudodeterministic additive approximation algorithm, then every function f that admits a (ε, δ) multiplicative approximation algorithm (resp. additive), has a $(2\varepsilon + \varepsilon^2, \delta)$-pseudodeterministic multiplicative approximation algorithm (resp. additive).*

Proof. We first provide intuition behind the proof. By Proposition 4, we can assume that f has a (ε', δ) 2-pseudodeterministic algorithm A, where $\varepsilon' = 2\varepsilon + \varepsilon^2$. The idea is to combine two strategies. For an input x, let a and b be the two good outputs of $A(x)$. Consider the case when one of the good outputs, say a, has a noticeably higher probability of occurrence than b. In that case we can run A several times and output the most frequent output. With high probability $A(x)$ will output a. Another case is when both a and b appear with roughly equal probability. In this case, we can run A several times and output the smallest value. Since a and b appear with equal probability, the probability that in several runs of A we will see both a and b is very high and hence, this strategy will output $\min\{a, b\}$. The challenge is to decide which of these cases holds (note that these cases are not disjoint). However, if we have a pseudodeterministic algorithm for Collision Probability Estimation Problem, then we show that we can pseudodeterministically choose a good strategy. Now we provide details.

Let B be a pseudodeterministic algorithm that estimates collision probabilities of Boolean circuits. In particular, assume that B can pseudodeterministically estimate the collision probability of a circuit, $\mathrm{CP}(C)$, with additive error $1/49$ with probability $\geq \frac{11}{12}$. Without loss of generality we assume that the success probability of A is $\geq 48/49$. Assume that A uses m random bits on input x. Consider the following algorithm A'.

Algorithm A': On input x, $A'(x)$ constructs a Boolean circuit $C_x : \{0,1\}^m \to \{0,1\}$ that on input r, simulates $A(x,r)$, where $A'(x)$ runs B on C_x to obtain an estimate of $\mathrm{CP}(C_x)$. If $B(C_x) \geq \frac{51}{98}$, A' runs $A(x)$ *three* times and outputs the most frequent result. Otherwise, if $B(C_x) < \frac{51}{98}$, A' runs $A(x)$ 16 times and outputs the smallest value. **End Of Algorithm.**

Now we will show that A' is a pseudodeterministic algorithm for f. Since A is 2-pseudodeterministic, there exists a set S_x of size at most 2 such that $\Pr[A(x) \in S_x] \geq 48/49$ and every element in S_x is a $(1+\varepsilon')$ multiplicative approximation of $f(x)$. We first consider the case where the size of S_x is 1, say $S_x = \{a\}$. Note that in this case $\mathrm{CP}(C_x)$ is at least $(48/49)^2 > 0.9$. Thus $A'(x)$ runs A three times and outputs the most frequent result, which is a with probability at least $(48/49)^3 \geq 0.9$. Thus $A'(x)$ is pseudodeterministic. Thus, in the rest of the proof, assume that $S_x = \{a,b\}$. Let $p = \Pr[A(x) = a]$ and $q = \Pr[A(x) = b]$. Assume without loss of generality that $p \geq q$. We first establish a relationship among p, q and $\mathrm{CP}(C_x)$.

▷ Claim 7. If $\mathrm{CP}(X_x) > \frac{25}{49}$, then $p > q + 1/7$

Proof. We prove the contrapositive: if $p \leq q+1/7$, then $\mathrm{CP}(C_x) \leq \frac{25}{49}$. Notice that $\mathrm{CP}(C_x)$ is maximized when $p = q + \frac{1}{7}$ and $\delta = 0$, where δ is the error probability of A. Since $p+q+\delta = 1$, it follows that $p = 4/7$ and $q = 3/7$. Thus, $\mathrm{CP}(C_x) \leq (4/7)^2 + (3/7)^2 = 25/49$. ◀

▷ Claim 8. If $\mathrm{CP}(C_x) < \frac{26}{49}$, then $q > \frac{1}{7}$

Proof. We prove the contrapositive: if $q \leq \frac{1}{7}$, then $\Pr[\mathrm{CP}(C_x)] \geq \frac{26}{49}$. Note that $\mathrm{CP}(C_x)$ is minimized when p is as close to q as possible, and all other outputs are different from a and b. Then $q = \frac{1}{7}$ and $p = 1 - \frac{1}{7} - \frac{1}{49} = \frac{41}{49} \geq \frac{5}{7}$. Thus $\mathrm{CP}(C_x) \geq \left(\frac{1}{7}\right)^2 + \left(\frac{5}{7}\right)^2 = \frac{26}{49}$ ◀

▷ Claim 9. If $p > q + 1/7$, then the probability that in 3 independent runs $A(x)$ outputs a at least twice is $\geq 57/100$.

Proof. The worst case is when $p = q+1/7$ and $\delta = 1/49$. In this case, $p = \frac{55}{98}$ and $q = \frac{41}{98}$. The probability that $A(x)$ outputs a at least twice in three runs is $\geq \left(\frac{55}{98}\right)^3 + 3\left(\frac{57}{98}\right)^2 \frac{44}{98} \geq \frac{57}{100}$. ◀

▷ Claim 10. Let E be following event: "Among 16 independent runs of $A(x)$, every run outputs either a or b and at least one run outputs a and at least one run outputs b". If $q \geq \frac{1}{7}$, then $\Pr[E] \geq 3/5$.

Proof. Note that the probability of $\overline{E}$ is at most the sum of the probabilities of i) $A(x) \notin \{a,b\}$, ii) Every run of $A(x)$ outputs only a, and iii) Every run of $A(x)$ outputs only b. This sum is maximized when $q = 1/7$, $\delta = 1/49$, $p = 1 - 1/7 - 1/49$. In this case, $\Pr\left[\overline{E}\right] \leq 1 - (1-\delta)^{16}p^{16} + q^{16} \leq \left(\frac{6}{7}\right)^{16} + \left(\frac{1}{7}\right)^{16} + 1 - \left(\frac{48}{49}\right)^{16} < \frac{2}{5}$. ◀

▷ Claim 11. A' is pseudodeterministic

Let y be the pseudodeterministic output of the probability estimator B on input C_x. If $y > \frac{51}{98}$, then $\mathrm{CP}(C_x) \geq \frac{25}{49}$. By Claim 7, $p > q + 1/7$. By Claim 9, the probability that $A(x)$

outputs a more often than b among three runs is at least $\frac{57}{100}$. Since $y > \frac{51}{98}$, A' will run A 3 times and output the most frequent result; this will output a with probability $\geq \frac{57}{100}$.

On the other hand, if $y \leq \frac{51}{98}$, then $\mathrm{CP}(C_x) \leq \frac{53}{98}$. By Claim 8, $q \geq 1/7$. Since $y \leq \frac{51}{98}$, A' will run A 16 times and output the lexicographically smallest result. By Claim 10, $\Pr[A$ outputs only a and b, and outputs a and b at least once$] \geq \frac{3}{5}$, so $\Pr[A'$ outputs $\min(a, b)] \geq \frac{3}{5}$.

So, given that B outputs y, A' will output one particular value x with probability $\geq \frac{57}{100}$. $\Pr[A'$ outputs $x] \geq \Pr[B$ outputs $y] \Pr[A'$ outputs $x | B$ outputs $y] \geq \frac{11}{12}\frac{57}{100} \geq \frac{52}{100}$. This can be increased to $1 - \frac{1}{n}$ using standard amplification techniques. ◀

3 Pseudodeterminism for Multi-valued Functions

In this section, we generalize the results from the previous section to k-pseudodeterminism. Goldreich [7] defined the notion of k-pseudodeterminism for search problems and this definition can be extended to multivalued functions. A function f is multivalued if $f(x)$ is a subset of the range (possibly empty set). Note that search problems can be cast as multivalued functions: Let R be a binary relation associated with a search problem, and define $f(x)$ as the set of all y such that $\langle x, y \rangle \in R$.

▶ **Definition 12.** *Let f be a multivalued function, i.e, $f(x)$ is a 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 $2/3$. 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 $\frac{k+1}{k+2}$.*

Goldreich [7] showed that if we threshold success probability to at least $\frac{k+1}{k+2}$, the the success probability for k-pseudodeterministic algorithms can be amplified to $1 - 1/2^{p(n)}$ for any polynomial $p(\cdot)$.

We show that if Collision Probability Estimation Problem can be made pseudodeterministic, then any k-pseudodeterministic algorithm for a multi-valued function problem can be made pseudodeterministic for a constant k. We first show how to reduce the size of the output set from k to $k - 1$.

▶ **Theorem 13.** *If* Collision Probability Estimation Problem *has a (ε, δ)-pseudodeterministic additive approximation algorithm with $\varepsilon = 1/100$, then for every multi-valued function f that admits a k-pseudodeterministic algorithm, f has a $(k-1)$-pseudodeterministic algorithm.*

Proof. Let B be a pseudodeterministic algorithm for Collision Probability Estimation Problem. In particular, assume that B, given a circuit C, estimates $\mathrm{CP}(C)$ to within $\frac{1}{100}$ additive error with probability $1 - \delta$, where n is the length of the input to C.

Let A be a k-pseudodeterministic algorithm for a multi-valued function f with error probability $\delta \leq \frac{1}{72k}$. That is, A, on input x, outputs from a set $S_x \subseteq f(x)$ of size $\leq k$, with probability $\geq 1 - \frac{1}{72k}$. We call the elements of S_x *good outputs* of A. Let m be the number of random bits used by A. We will design a $(k-1)$-pseudodeterministic algorithm A' as follows:
Algorithm A': On input x, first construct a circuit C_x that gets as input r and outputs $A(x, r)$. Then compute $B(C_x)$.

If $B(C_x) < \frac{65}{100}$, run A n times on independent random bits. Output the *lexicographically smallest* element that appears at least $\frac{n}{12k}$ times.

If $B(C_x) \geq \frac{65}{100}$, run A n times using independent random bits. Output the most frequent value. **End of Algorithm.**

As in the proof of Theorem 6, it is easy to see that if the size of S_x is 1, $A'(x)$ is pseudodeterministic. Thus in the rest of the proof, we assume that the size of S_x is at least 2. For the proof of correctness, we first establish certain claims relating the collision probability of C_x and the behavior of A.

▷ Claim 14. If $\mathrm{CP}(C_X) \leq \frac{2}{3}$, then there exist two *good* outputs a and b of A so that $\Pr[A(x) = a] \geq \frac{1}{6k}$ and $\Pr[A(x) = b] \geq \frac{1}{6k}$.

Proof. Suppose only one good output a of A has probability $\geq \frac{1}{6k}$. Then that output has probability $\geq \frac{5}{6} - \delta$. This is because if the other $(k-1)$ good outputs have probability $< \frac{1}{6k}$, the total probability of outputs other than a is $< 1/6 + \delta$. Thus $\Pr[A(x) = a] \geq \frac{5}{6} - \delta$.

$$\begin{aligned}\mathrm{CP}(C_x) &\geq \Pr[A \text{ outputs } a \text{ on both runs}] \\ &\geq \left(\frac{5}{6} - \delta\right)^2 \\ &\geq \frac{25}{36} - 2\delta \geq \frac{2}{3}\end{aligned}$$

The last inequality holds since $\delta \leq \frac{1}{72k}$ and $k \geq 2$. ◀

▷ Claim 15. Let a, b be the most and least likely good outputs of A with probabilities p and q respectively. If $\mathrm{CP}(C_x) > \frac{64}{100}$, then $p > q + \frac{1}{8k}$.

Proof. If $p \leq q + \frac{1}{8k}$, then $p \leq \frac{9}{8k} - \frac{\delta}{k} \leq \frac{9}{8k}$. Then $\mathrm{CP}(C_x) \leq k \cdot \frac{81}{64k^2} + \delta^2 = \frac{81}{64k} + \delta^2$. For $k \geq 2$, and $\delta \leq \frac{1}{72k}$, this quantity is $\leq \frac{64}{100}$. ◀

Now we will prove correctness of A'. On input x, let y be the pseudodeterministic output of $B(C_x)$. We will consider two cases: $y \leq \frac{65}{100}$ and $y > \frac{65}{100}$.

Case: $y \leq \frac{65}{100}$

▷ Claim 16. If $y \leq \frac{65}{100}$, then there exists a set $S'(x) \subset S_x$ of size at most $k-1$ such that

$$\Pr[A'(x) \in S'_x] \geq 1 - 2e^{-\frac{n}{72k^2}} - \delta$$

Proof. Note that $B(C_x)$ outputs y with probability at least $1 - \delta$. Thus, if $y \leq \frac{65}{100}$, then with probability $\geq 1 - \delta$, A' will run A n times and output the lexicographically smallest result that appears at least $\frac{n}{12k}$ times. Since $y \leq \frac{65}{100}$, $\mathrm{CP}(C_x) \leq \frac{66}{100} < \frac{2}{3}$. Therefore by Claim 14, there are at least 2 elements a and b from S_x that A outputs with probability $\geq \frac{1}{6k}$. Let b be the lexicographically larger of the two. We set $S'_x = S_x \setminus \{b\}$. Clearly S'_x contains at most $k-1$ elements. Thus the probability that $A'(x)$ outputs an element outside of S'_x is at most the sum of the probabilities of the following events: i) A' outputs b ii) A' outputs an element that is not in S_x, iii) $B(C_x)$ does not output y.

We first bound the probability that A' outputs b. For A' to output b, it must be the case that in the n runs of A the value b is output at least $n/12k$ times and the value a is output at most $n/12k$ times. Thus

$$Pr[A'(x) = b] \leq \Pr[\ A(x) \text{ outputs } a \text{ less than } n/12k \text{ times among } n \text{ runs}]$$

Since the probability that $A(x)$ outputs a is least $n/6k$, the expected number of times $A(x)$ outputs a in n runs is $\geq \frac{n}{6k}$. Thus by Chernoff bound, this is at most $e^{-n/72k^2}$. We now bound the probability of the second event. The probability that $A(x)$ does not belong to S_x

is at most $1/72k$. For $A'(x)$ to output an element c that is not in S_x, it must be the case that c is output $\geq n/12k$ times among n runs of A. Again by Chernoff bound, this is at most $e^{-n/18k^2}$. Finally, the probability that $B(C_x)$ does not output y is at most δ. Thus $A'(x) \in S'_x$ with probability at least $1 - 2e^{-n/72k^2} - \delta$.

◀

Case: $y > \frac{65}{100}$

▷ Claim 17. If $y > \frac{65}{100}$, there exists a sets $S'_x \subseteq S_x$ of size at most $k-1$ such that

$$\Pr[A'(x) \in S'_x] \geq 1 - 2e^{-\frac{n}{8k}} - \delta$$

Proof. If $y > \frac{65}{100}$, then with probability $\geq 1-\delta$, A' will run A n times and output the most frequent result. Also, $\mathrm{CP}(C_x) > \frac{64}{100}$. By Claim 15, $p > q + \frac{1}{8k}$ where p is the probability of the most likely element a from S_x and q is the probability of least likely element b from S_x. We define S'_x as $S_x - \{b\}$. As before, the probability that the output of A' does not belong to S'_x is at most the sum of the probabilities of: i) $A'(x)$ outputs b ii) the output of $A'(x)$ does not belong to S_x iii) $B(C_x)$ does not output y.

We will first analyze the probability of the event that A' outputs b. For this, consider the event $E =$ 'A outputs b more often than a in n trials'. Clearly, the probability that $A'(x)$ outputs b is at most $\Pr[E]$. Define random variables X_i that take value 0 if $A(x)$ outputs a in i^{th} run, 1 if $A(x)$ outputs b in the i^{th} run, and $1/2$ otherwise. Note that $E[X_i] = \frac{1}{2} - \frac{p-q}{2}$, which is at most $\frac{1}{2} - \frac{1}{16k}$. Let $X = \sum_{i=1}^{n} X_i$. Now, $\Pr[E]$ is $\Pr[X > n/2]$.

$$\begin{aligned}
\Pr\left[X > n/2\right] &= \Pr\left[\frac{\sum X_i}{n} > 1/2\right] \\
&\leq \Pr\left[\left|\frac{\sum X_i}{n} - E[X_i]\right| \geq \frac{1}{16k}\right] \\
&\leq e^{-n/256k^2} \text{ by Chernoff bound}
\end{aligned}$$

To bound the probability of the second event, consider the probability that $A(x)$ outputs an element not in S_x more frequently than a in n runs. Since the probability that $A(x)$ outputs an element that is not in S_x is at most $1/72k$, by the same argument the probability of this event is at most $e^{-n/256k^2}$. Finally the probability that $B(C_x)$ does not output y is at most δ. The claim follows. ◀

Combining Claims 16 and 17, we have that A' outputs a value from S'_x with probability at least $1 - 2e^{-n/72k^2} - \delta \geq \frac{k}{k+1}$ for large enough n, since δ can be made exponentially small.

◀

We now state the main result of this section.

▶ **Theorem 18.** *If* Collision Probability Estimation Problem *has a (ε, δ)-pseudodeterministic additive approximation algorithm with $\varepsilon = 1/100$, every multi-valued function f that admits a k-pseudodeterministic algorithm, has a pseudodeterministic algorithm.*

Proof. Let B be the pseudodeterministic algorithm for Collision Probability Estimation Problem, where B runs in time n^c with error probability $\leq \delta$. To convert a $k-$pseudodeterministic algorithm to a pseudodeterministic algorithm, we repeatedly apply Theorem 13. We start with a k-pseudodeterministic algorithm A_k whose runtime is bounded by n^t. On input x, A_{k-1} constructs $C_{A_k,x}$ with size $\leq 4n^{2t}$. A_{k-1} computes $B(C)$, which takes $\leq 4^c n^{2tc}$ time, then runs $A_k(x)$ n times. In total, $A_{k-1}(x)$ takes $\leq n^{t+1} + 4^c n^{2tc} \leq n^{4tc}$.

Applying this conversion $(k-1)$ times, we obtain A_1, a pseudodeterministic algorithm with runtime of $O(n^{t(4c)^k})$. Since k is a constant the runtime is polynomial. Note that in each iteration, the error probability remains the same. Thus A_1 is pseudodeterministic. ◀

3.1 Circuit Probability Acceptance

In this subsection we observe the equivalence of Collision Probability Estimation Problem and Acceptance Probability Estimation Problem in the context of pseudodeterminism.

▶ **Proposition 19.** *There exist $\varepsilon, \varepsilon' > 0$ such that* Collision Probability Estimation Problem *has an (ε, δ)-pseudodeterministic additive approximation algorithm if and only if* Acceptance Probability Estimation Problem *has am (ε', δ)-pseudodeterministic additive approximation algorithm.*

Proof. It is easy to see that Acceptance Probability Estimation Problem admits an (ε, δ) additive approximation algorithm. Thus by Proposition 4, it has a 2-pseudodeterministic (ϵ, δ) approximation algorithm. By Theorem 6, if Collision Probability Estimation Problem admits a pseudodeterministic algorithm, then Acceptance Probability Estimation Problem admits a pseudodeterministic algorithm.

Let B be a pseudodeterministic algorithm for Acceptance Probability Estimation Problem. Consider the following algorithm to estimate the collision probability of a circuit C: If $B(C)$ outputs v, output $v^2 + (1-v)^2$. Let $p = \Pr[C(U_n) = 1]$. If $v \in (p-\varepsilon, p+\varepsilon)$, then the output of B belongs to $(\mathrm{CP}(C) - 8\varepsilon, \mathrm{CP}(C) + 8\varepsilon)$. Clearly B is pseudodeterministic. ◀

The following result is a corollary of the above proposition and Theorem 18.

▶ **Theorem 20.** *There exists $\varepsilon' > 0$ such that if* Acceptance Probability Estimation Problem *admits an (ε', δ) pseudodeterministic additive approximation algorithm, then every function f that admits an (ε, δ) multiplicative approximation algorithm has a $(3\varepsilon, \delta)$-pseudodeterministic multiplicative approximation algorithm.*

3.2 Entropy Estimation

In this subsection we show that Entropy Estimation Problem and Acceptance Probability Estimation Problem are equivalent in the context of pseudodeterminism. We first observe that Entropy Estimation Problem admits an (ε, δ), 2-pseudodeterministic additive approximation algorithm.

▶ **Proposition 21.** *There is an (ε, δ) 2-pseudodeterministic approximation algorithm for the* Entropy Estimation Problem

Proof. Given a circuit C, let $p = \Pr[C(U_n) = 1]$. As in Proposition 5, compute an approximate value q of p and output $H(q)$. It follows that $H(q)$ is an approximation of $H(p)$ due to the known result that entropy can be approximated by the empirical distribution obtained from sampling, for example see [15, 1]. By Proposition 4, we obtain a 2-pseudodeterministic algorithm. ◀

Note that the above proof yields the following.

▶ **Proposition 22.** *There exist ε and ε' such that if* Acceptance Probability Estimation Problem *has an (ε, δ) pseudodeterministic additive error approximation algorithm, then* Entropy Estimation Problem *has (ε', δ) pseudodeterministic, additive error approximation algorithm.*

Next we reduce Acceptance Probability Estimation Problem to Entropy Estimation Problem. Thus a pseudodeterministic algorithm for Entropy Estimation Problem implies a pseudodeterministic algorithm for Acceptance Probability Estimation Problem.

The main technical result that we show is that an approximation of the entropy of $C(U_n)$ can be used to approximate the probability that $C(U_n) = 1$. It is possible that this technical result is known or is a folklore; we could not find a reference. Thus, for completeness a proof is provided in the appendix.

▶ **Theorem 23.** *Suppose that there is a (ε, δ) pseudodeterministic approximation algorithm for* Entropy Estimation Problem *for a sufficiently small ε. Then there is a $(1/100, \delta + e^{-O(n)})$ pseudodeterministic approximation algorithm for* Acceptance Probability Estimation Problem.

Using Proposition 19 and Theorem 23, we obtain that both Acceptance Probability Estimation Problem and Entropy Estimation Problem are complete for k-pseudodeterministic computations.

▶ **Theorem 24.** *There exist $\varepsilon > 0$, such that if either of* Acceptance Probability Estimation Problem *or* Entropy Estimation Problem *admit (ε, δ)-pseudodeterministic, additive approximation algorithm, then every multivalued function that has a k-pseudodeterministic algorithm has a pseudodeterministic algorithm.*

4 Pseudodeterminism for Search Problems

In this section we show that if any of the 3 computational problems we consider has pseudodeterministic approximation schemes then every problem in Search-BPP has pseudodeterministic algorithms. The class Search-BPP was formally introduced by Goldrecich [6]

▶ **Definition 25** (Search BPP [6])**.** *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 $\{y \mid (x,y) \in R\}$. A search problem R is in* search-BPP *(1) if 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 $\geq 2/3$, (2) and 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 $< 1/3$.*

We will first show that if Acceptance Probability Estimation Problem has a pseudodeterministic, additive, approximation scheme, then Search-BPP problems can be made pseudodeterministic. Then we will use Theorem 20 to prove that if Acceptance Probability Estimation Problem has a (ϵ, δ) pseudodeterministic approximation algorithm, then Search-BPP problems admit pseudodeterministic algorithms. We first recall the definition of approximation scheme.

▶ **Definition 26.** *A function $f : \Sigma^* \to \mathbb{Q}$ has an additive, approximation scheme if there is a probabilistic polynomial time algorithm A that gets x, ϵ, and δ as input and*

$$\Pr[f(x) - \epsilon \leq A(x, \epsilon, \delta) \leq f(x) + \epsilon] \geq 1 - \delta$$

▶ **Theorem 27.** *If* Acceptance Probability Estimation Problem *has a pseudodeterministic, additive, approximation algorithm scheme, then every problem in Search-BPP has a pseudodeterministic algorithm.*

Proof. Let R be a problem in Search-BPP and let A and B be probabilistic algorithms that witness R in search-BPP according to the definition.

The idea is to use the method of conditional probabilities to construct a good random choice z_A for A on input x first, and then output $A(x, z_A)$. The search for z_A will be aided by the pseudodeterministic approximation algorithm for Acceptance Probability Estimation Problem.

Consider the following probabilistic algorithm B' that, on input x of length n, first simulates A to get an output y and then runs $B(x, y)$ and accepts if B accepts. Then $\Pr[B'(x) \text{ accepts}] \geq 2/5$. Let $m = p(n)$ be the polynomial bounding the length of the random string of B'. We will view the random string r that B' uses as $r_A r_B$ where r_A is the random string that B' uses to simulate A and r_B is to simulate B. Let A_{ape} be a pseudodeterministic approximation algorithm for Acceptance Probability Estimation Problem. We will use A_{ape} with error $\epsilon \leq \frac{1}{n \cdot p(n)}$ and confidence $1 - \delta \geq 1 - \frac{1}{n \cdot p(n)}$.

For an input x let $C(r_A r_B)$ be the Boolean circuit that simulates B' on x using random string $r = r_A r_B$ and outputs 1 if and only if B' accepts x on r. Thus for any x where $W_x \neq \phi$, $\Pr[C = 1] \geq 2/5$. For a binary string $z \in \{0,1\}^l$, let $C_z : \{0,1\}^{m-l} \to \{0,1\}$ be the circuit obtained by fixing the first l bits of C's input to z. We now describe the algorithm A_R for the search problem that pseudodeterministically outputs a $y \in W_x$.

Algorithm $\mathsf{PseudoA_R}$: On input x, construct the circuit C that gets $r = r_A r_B$ as input and outputs 1 if B' accepts (x, r) on random string $r = r_A r_B$. Initialize $z = \lambda$, the empty string. Iterate from $i = 1$ to $m = p(n)$. At the i^{th} iteration, simulate $A_{\text{ape}}(C_z 0)$ (pseudodeterministically) to approximate $\Pr[C_{z0}(r) = 1]$ up to an additive error ϵ and confidence $(1 - \delta)$ to get a value v. If $v \geq 2/5 - (2i + 1)\epsilon$ then $z \leftarrow z0$ otherwise $z \leftarrow z1$. Continue to the next iteration. After the m^{th} iteration let $z = z_A z_B$ be the binary string of length m constructed. Output $A(x, z_A)$. **End-of-Algorithm**

Correctness: Since error probability of A_{ape} is $\leq \frac{1}{n \cdot m}$, and we are making m calls to A_{ape}, by the union bound, the probability that any one of the calls makes an error is $\leq 1/n$. For the rest of the argument we assume all the calls to A_{ape} pseudodeterministically output an approximation to acceptance probability within an additive error of ϵ.

▷ Claim 28. For every i, for the string z constructed at the end of the i^{th} iteration, $\Pr[C_z = 1] \geq \frac{2}{5} - 2i\epsilon$.

Proof. We prove this by induction on i. For $i = 0$, the hypothesis holds since $\Pr[C = 1] \geq 2/5$. Assume the hypothesis holds for i. Consider the $(i + 1)^{th}$ iteration. Using conditional probabilities, after the i^{th} iteration, $\Pr[C_{z0} = 1] \geq \frac{2}{5} - 2i\epsilon$ or $\Pr[C_{z1} = 1] \geq \frac{2}{5} - 2i\epsilon$. Suppose at the $(i + 1)^{th}$ iteration the value v returned by $A_{\text{ape}}(C_{z0})$ is $\geq 2/5 - (2i + 1)\epsilon$. Then z is updated to $z0$ by the algorithm and from the approximation guarantee of A_{ape} we have that $\Pr[C_{z0} = 1] \geq 2/5 - (2i + 1)\epsilon - \epsilon = 2/5 - 2(i + 1)\epsilon$. On the other hand suppose $v < 2/5 - (2i + 1)\epsilon$. Then z is updated to $z1$. Also $\Pr[C_{z0} = 1] < v + \epsilon = 2/5 - 2i\epsilon$ and hence $\Pr[C_{z1} = 1] \geq 2/5 - 2i\epsilon \geq 2/5 - 2(i + 1)\epsilon$ ◀

Thus for any $1 \leq i \leq m$ $\Pr[C_z = 1] \geq 2/3 - 1/n$ and hence the algorithm outputs a z so that $B'(x, z)$ accepts. Hence the output of the algorithm $A(x, z_A) \in W_x$.

The algorithm $\mathsf{PseudoA}_R$ can be seen as a deterministic algorithm making subroutine calls to the pseudodeterministic algorithm A_{ape}. Hence the overall algorithm is pseudodeterministic. The probability of error is bounded by any one of the calls to A_{ape} making an error which is $\leq 1/n$. ◀

Next we will show that if Acceptance Probability Estimation admits (ε, δ) pseudodeterministic additive approximation, then admits pseudodeterministic additive approximation scheme.

▶ **Proposition 29.** *There exists $\varepsilon > 0$ such that if* Acceptance Probability Estimation *admits an (ϵ, δ) pseudodeterministic additive approximation, then it admits a pseudodeterministic approximation scheme.*

Proof. We first note that Acceptance Probability Estimation Problem admits an additive, approximation scheme. By Theorem 20, there is an $\epsilon' > 0$ such that if Acceptance Probability Estimation Problem has an (ϵ', δ) pseudodeterministic approximation algorithm, then every (ϵ, δ)-additive approximation algorithm for a function f can be made into a $(3\epsilon, \delta)$ pseudodeterministic, additive approximation algorithm. The same proof shows that if f admits an approximation scheme, then it can be made into a pseudodeterministic approximation scheme.

◀

The main result of this section is a corollary of the above proposition and Theorem 27.

▶ **Theorem 30.** *There exists $\varepsilon > 0$ such that if* Acceptance Probability Estimation Problem *has a (ε, δ) pseudodeterministic approximation algorithm algorithm, then every problem in Search-BPP has a pseudodeterministic algorithm.*

References

1. Jayadev Acharya, Sourbh Bhadane, Piotr Indyk, and Ziteng Sun. Estimating entropy of distributions in constant space. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 5163–5174, 2019.
2. 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.
3. E. Gat and S. Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:136, 2011.
4. Michel Goemans, Shafi Goldwasser, and Dhiraj Holden. Doubly-efficient pseudo-deterministic proofs. *arXiv*, 2019.
5. 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.
6. 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.
7. Oded Goldreich. Multi-pseudodeterministic algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:12, 2019.
8. 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.
9. S. Goldwasser, O. Grossman, and D. Holden. Pseudo-deterministic proofs. *CoRR*, abs/1706.04641, 2017.

10 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.

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

12 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.

13 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.

14 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.

15 Liam Paninski. Estimating entropy on m bins given fewer than m samples. *IEEE Trans. Inf. Theory*, 50(9):2200–2203, 2004.

16 L. Stockmeyer. The complexity of approximate counting (preliminary version). In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 118–126, 1983.

A Proof of Theorem 23

We first start with the following lemma. Here H is the binary entropy function $H(p) = -p\log p - (1-p)\log(1-p)$.

▷ Claim 31. Let $0 \le a+b \le 1$. $H(a+b) \ge H(a) + b\log\frac{1-a-b}{a+b}$

Proof.

$$\begin{aligned} H(a+b) &= -(a+b)\log(a+b) - (1-a-b)\log(1-a-b) \\ &= -a\log(a+b) - (1-a)\log(1-a-b) - b\log(a+b) - (-b)\log(1-a-b) \\ &\ge -a\log(a) - (1-a)\log(1-a) + b(\log(1-a-b) - \log(a+b)) \\ &= H(a) + b\log\frac{1-a-b}{a+b} \end{aligned}$$

where the third line follows by Gibbs' inequality. ◀

We now provide a proof of Theorem 23.

Proof. Suppose A is a pseudodeterministic algorithm that, given a Boolean circuit C, outputs (ε,δ) approximation of $H(C(U_n))$. Let $r = \Pr[C(U_n)=1]$. Our goal is to design a pseudodeterministic algorithm to estimate r. Let q be the smaller of $1-r$ and r. We will first design a pseudodeterministic algorithm B that outputs a value p such that p is within $1/100$ of q.

$B(C)$ runs $A(C)$ to obtain y. If $y \ge H(\frac{1}{2} - \frac{1}{100}) + \varepsilon$, then output $\frac{1}{2}$. Otherwise, do a binary search for p in the range $(0, 1/2)$ such that $-p\log p - (1-p)\log(1-p)$ lies within $[y, y+1/2^n)$. We consider two cases.

Case 1: $y \ge H(\frac{1}{2} - \frac{1}{100}) + \varepsilon$. In this case, then clearly $\frac{1}{2} - \frac{1}{100} \le r \le \frac{1}{2} + \frac{1}{100}$. $B(C)$ will output $\frac{1}{2}$, which is within $\frac{1}{100}$ of $r = \Pr[C=1]$, with probability $\ge 1-\delta$.

Case 2: $y < H(\frac{1}{2} - \frac{1}{100}) + \varepsilon$. Since $|H(p) - y| \le 1/2^n$ and $|H(q) - y| \le \varepsilon$, we have that $|H(p) - H(q)| \le \varepsilon + 1/2^n = \varepsilon'$. Now we bound how far p is from q. For this we need the following two technical claims.

▷ Claim 32. Let $a \leq \frac{1}{2}$. If $H(a) \leq H(\frac{1}{2} - \frac{1}{100}) + \varepsilon'$, then $a \leq \frac{1}{2} - \frac{1}{100} + \frac{1}{c}$, for any c that satisfies $\varepsilon' \leq \frac{1}{2c} \log(\frac{102c-200}{98c+200})$.

Proof.

$$\begin{aligned} H(a) &\leq H\left(\frac{1}{2} - \frac{1}{100}\right) + 2\varepsilon \\ &\leq H\left(\frac{1}{2} - \frac{1}{100}\right) + \frac{1}{c}\log\left(\frac{102c-200}{98c+200}\right) \\ &= H\left(\frac{1}{2} - \frac{1}{100}\right) + \frac{1}{c}\log\left(\frac{\frac{1}{2} + \frac{1}{100} - \frac{1}{c}}{\frac{1}{2} - \frac{1}{100} + \frac{1}{c}}\right) \\ &\leq H\left(\frac{1}{2} - \frac{1}{100} + \frac{1}{c}\right) \text{ by Claim 31} \end{aligned}$$

Thus $a \leq \frac{1}{2} - \frac{1}{100} + \frac{1}{c}$. ◀

▷ Claim 33. Let $a = b + \ell$ where $a \leq \frac{1}{2}, a, b, \ell \geq 0$. If $H(a) - H(b) \leq \varepsilon'$ and $a \leq \frac{1}{2} - \frac{1}{100} + \frac{1}{c} \leq \frac{1}{2}$, then $\ell \leq \frac{\varepsilon'}{\log \frac{102c-200}{98c+200}}$

Proof.

$$\begin{aligned} \varepsilon' &\geq H(a) - H(b) \\ &= H(b+\ell) - H(b) \\ &\geq H(b) + \ell \log\left(\frac{1-b-\ell}{b+\ell}\right) - H(b) (By Claim\ 31) \\ &= \ell \log\left(\frac{1-a}{a}\right) \end{aligned}$$

Since $a \leq \frac{1}{2}$, the minimum value for this is when a is as close to $\frac{1}{2}$ as possible.

$$\begin{aligned} &\geq \ell \log \frac{1 - (\frac{1}{2} - \frac{1}{100} + \frac{1}{c})}{(\frac{1}{2} - \frac{1}{100} + \frac{1}{c})} \\ &= \ell \log \frac{102c-200}{98c+200} \\ \Rightarrow \ell &\leq \frac{\varepsilon'}{\log \frac{102c-200}{98c+200}} \end{aligned}$$

◀

▷ Claim 34. If ε' is sufficiently small, then there is a constant c satisfying

$$\frac{\varepsilon'}{\log \frac{102c-200}{98c+200}} \leq \frac{1}{100} \text{ and } \varepsilon' \leq \frac{1}{2c}\log\left(\frac{102c-200}{98c+200}\right)$$

Proof. It can be verified that when $\varepsilon' = 1/42000$ and $c = 1100$, the above inequalities are satisfied. ◀

Now we are ready to prove that $|p - q| \leq \frac{1}{100}$

▷ Claim 35. $|p - q| \leq \frac{1}{100}$

Proof. First, suppose $p > q$. Then $p = q + \ell$. Since, $y < H(\frac{1}{2} - \frac{1}{100}) + \varepsilon$ and $|y - H(p)| \leq \frac{1}{2^n}$, so $H(p) \leq H(\frac{1}{2} - \frac{1}{100}) + \varepsilon + \frac{1}{2^n} = H(\frac{1}{2} - \frac{1}{100}) + \varepsilon'$.By Claim 34, we have that $c = 1/1100$. By claim 32, $p \leq \frac{1}{2} - \frac{1}{100} + \frac{1}{c}$. By claim 33, we obtain that $\ell \leq \frac{\varepsilon'}{\log \frac{102c-200}{98c+200}} \leq 1/100$, thus $p \leq q + 1/100$. A similar argument shows that if $p < q$, then $p \geq q - 1/100$.

◀

We found a value p that is $1/100$-close to q, and the goal is to estimate $r = \Pr[C(U_n) = 1]$, where $q = \min\{r, 1 - r\}$. Thus p is either close to r or to $1 - r$. Now we run $C(U_n)$, n times; if there are more 1s than 0s output $1 - p$; else output p. Using Chernoff bounds, it follows that the output is $1/100$-close to q with probability $\leq 1 - e^{-2n/(110^2)}$.

Finally, recall that we needed $\varepsilon' = \varepsilon + 1/2^n \leq 1/42000$. Thus we can take $\varepsilon \leq 1/43000$ (for large enough n). So, if it's possible to pseudodeterministically estimate $H(\Pr[C = 1])$ within $\frac{1}{43000}$ with probability $1 - \delta$, it's possible to pseudodeterministically estimate $\Pr[C = 1]$ within $\frac{1}{100}$ with probability $1 - \delta - e^{-2n/(110^2)}$.

◀