Electronic Colloquium on Computational Complexity, Report No. 172 (2016)

Preserving Randomness for Adaptive Algorithms

William M. Hoza* Adam R. Klivans†

November 2, 2016

Abstract

We introduce the concept of a *randomness steward*, a tool for saving random bits when executing a randomized estimation algorithm Est on many adaptively chosen inputs. For each execution, the chosen input to Est remains hidden from the steward, but the steward chooses the randomness of Est and, crucially, is allowed to modify the output of Est. The notion of a steward is inspired by *adaptive data analysis*, introduced by Dwork et al. [DFH+15c] Suppose Est outputs values in $\mathbb{R}^d$, has ℓ_∞ error ε, has failure probability δ, uses n coins, and will be executed k times. For any $\gamma > 0$, we construct a computationally efficient steward with ℓ_∞ error $O(\varepsilon d)$, failure probability $k\delta + \gamma$, and randomness complexity $n + O(k \log(d + 1) + (\log k) \log(1/\gamma))$. To achieve these parameters, the steward uses a pseudorandom generator for what we call the *block decision tree* model, combined with a scheme for shifting and rounding the outputs of Est. (The generator is a variant of the [INW94] generator for space-bounded computation.) We also give variant stewards that achieve tradeoffs in parameters.

As applications of our steward, we give time- and randomness-efficient algorithms for estimating the acceptance probabilities of many adaptively chosen Boolean circuits and for simulating any algorithm with an oracle for promise-BPP or APP. We also give a randomness-efficient version of the Goldreich-Levin algorithm; our algorithm finds all Fourier coefficients with absolute value at least θ of a function $F : \{0, 1\}^n \to \{-1, 1\}$ using $O(n \log n) \cdot \mathrm{poly}(1/\theta)$ queries to F and $O(n)$ random bits, improving previous work by Bshouty et al. [BJT99] Finally, we prove a randomness complexity lower bound of $n + \Omega(k) - \log(\delta'/\delta)$ for any steward with failure probability δ', which nearly matches our upper bounds in the case $d \leq O(1)$.

*Department of Computer Science, University of Texas at Austin, whoza@utexas.edu

†Department of Computer Science, University of Texas at Austin, klivans@cs.utexas.edu

ISSN 1433-8092

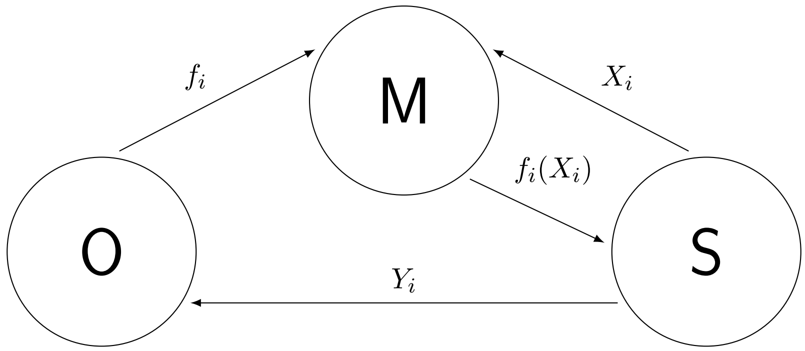

Figure 1: A single round of interaction in the definition of a steward. Here, $f_i(X) \stackrel{\text{def}}{=} \mathsf{Est}(C_i, X)$.

1 Introduction

Let Est be a randomized algorithm that estimates some quantity $\mu(C) \in \mathbb{R}^d$ when given input C. The canonical example is the case when C is a Boolean circuit, $d = 1$, $\mu(C) \stackrel{\text{def}}{=} \Pr_x[C(x) = 1]$, and Est estimates $\mu(C)$ by evaluating C at several randomly chosen points. Suppose that Est uses n random bits, and $\Pr[\|\mathsf{Est}(C) - \mu(C)\|_\infty > \varepsilon] \leq \delta$.

Furthermore, suppose we want to use Est as a *subroutine*, executing it on inputs $C_1, C_2, \ldots, C_k$, where each C_i is chosen adaptively based on the previous outputs of Est. The naïve implementation uses nk random bits and fails with probability at most $k\delta$.

In this work, we show how to generically improve the randomness complexity of any algorithm with this structure, without increasing the number of executions of Est, at the expense of mild increases in the error and failure probability. Our algorithm efficiently finds $Y_1, \ldots, Y_k \in \mathbb{R}^d$ with $\|Y_i - \mu(C_i)\|_\infty \leq O(\varepsilon d)$ for every i, our algorithm has failure probability $k\delta + \gamma$ for any $\gamma > 0$, and our algorithm uses a total of $n + O(k \log(d+1) + (\log k)\log(1/\gamma))$ random bits.

1.1 The randomness steward model

A simple but important observation is that an algorithm that uses Est as a subroutine only sees the *outputs* of Est, not the coin tosses of Est. To make this fact explicit, we introduce the concept of a *randomness steward*. We first describe the model informally.

Imagine that O, the *owner*, plans to execute Est on k adaptively chosen inputs. To improve randomness efficiency, she hires a *steward* S and gives S "stewardship" over her random bits. When O wants to execute Est on an input C_i, she delegates the task to S, who chooses a randomness string $X_i \in \{0,1\}^n$. To prevent O from seeing X_i, O is required to give C_i to a *mediator* M, who evaluates $\mathsf{Est}(C_i, X_i)$ on behalf of S. Based on the output of Est, S chooses a value $Y_i \in \mathbb{R}^d$ to return to O. The steward S judiciously chooses these X_i, Y_i values so as to "spend" as little randomness as possible. The requirement is that with high probability, Y_i is close to $\mu(C_i)$ for every i.

Now, we describe the model rigorously. Say that a function $f : \{0,1\}^n \to \mathbb{R}^d$ is (ε, δ)-*concentrated at* $\mu \in \mathbb{R}^d$ if $\Pr_{X \in \{0,1\}^n}[\|f(X) - \mu\|_\infty > \varepsilon] \leq \delta$. In each round i, the chosen input C_i defines a concentrated function $f_i(X) \stackrel{\text{def}}{=} \mathsf{Est}(C_i, X)$, so it is equivalent to imagine that O picks an arbitrary concentrated function. (See Figure 1.) In the following definition, ε' is the error of the steward, and δ' is its failure probability.

Definition 1. An (ε', δ')-*steward for* k *adaptively chosen* (ε, δ)-*concentrated functions* $f_1, \ldots, f_k : \{0,1\}^n \to \mathbb{R}^d$ is a randomized algorithm S that interacts with an *owner* O through a *mediator* M according to the following protocol.

1. For $i = 1$ to k:
   (a) O chooses $f_i : \{0,1\}^n \to \mathbb{R}^d$ that is (ε, δ)-concentrated at some point $\mu_i \in \mathbb{R}^d$ and gives it to M.
   (b) S chooses $X_i \in \{0,1\}^n$ and gives it to M.
   (c) M gives $f_i(X_i) \in \mathbb{R}^d$ to S.
   (d) S chooses $Y_i \in \mathbb{R}^d$ and gives it to O.

Write $\mathsf{O} \leftrightarrow \mathsf{S}$ ("the interaction of O with S") to denote the above interaction. The requirement on S is that for all O,

$$\Pr[\max_i \|Y_i - \mu_i\|_\infty > \varepsilon' \text{ in } \mathsf{O} \leftrightarrow \mathsf{S}] \leq \delta'.$$

The probability is taken over the internal randomness of S and O.

1.2 Our results

1.2.1 Main result: A steward with good parameters

Our main result is the explicit construction of a steward that simultaneously achieves low error, low failure probability, and low randomness complexity:

Theorem 1. *For any $n, k, d \in \mathbb{N}$ and any $\varepsilon, \delta, \gamma > 0$, there exists an $(O(\varepsilon d), k\delta + \gamma)$-steward for k adaptively chosen (ε, δ)-concentrated functions $f_1, \ldots, f_k : \{0,1\}^n \to \mathbb{R}^d$ with randomness complexity*

$$n + O(k \log(d+1) + (\log k) \log(1/\gamma)).$$

The total running time of the steward is $\mathrm{poly}(n, k, d, \log(1/\varepsilon), \log(1/\gamma))$.

We also give several variant stewards that achieve tradeoffs in parameters. (See Figure 2.)

1.2.2 Application: Acceptance probabilities of Boolean circuits

Our first concrete application of Theorem 1 is a time- and randomness-efficient algorithm for estimating the acceptance probabilities of many adaptively chosen Boolean circuits.

Corollary 1. *There exists a randomized algorithm with the following properties. Initially, the algorithm is given parameters $n, k \in \mathbb{N}$ and $\varepsilon, \delta > 0$. Then, in round i ($1 \leq i \leq k$), the algorithm is given a Boolean circuit C_i on n input bits and outputs a number $Y_i \in [0,1]$. Here, C_i may be chosen adversarially based on $Y_1, \ldots, Y_{i-1}$. With probability $1 - \delta$, every Y_i is $\mu(C_i) \pm \varepsilon$, where $\mu(C_i) \stackrel{\text{def}}{=} \Pr_x[C_i(x) = 1]$. The total running time of the algorithm is*

$$O\left(\frac{\log k + \log(1/\delta)}{\varepsilon^2} \cdot \sum_{i=1}^{k} \mathrm{size}(C_i)\right) + \mathrm{poly}(n, k, 1/\varepsilon, \log(1/\delta)),$$

and the total number of random bits used by the algorithm is $n + O(k + (\log k) \cdot \log(1/\delta))$.

Corollary 1 should be compared to the case when $C_1, \ldots, C_k$ are chosen nonadaptively, for which the randomness complexity can be improved to $n + O(\log k + \log(1/\delta))$ by applying the Goldreich-Wigderson randomness-efficient sampler for Boolean functions [GW97] and reusing randomness. The proof of Corollary 1 works by combining the GW sampler with our steward; the details are in Section 6.1.

ε'	δ'	Randomness complexity	Reference
ε	$k\delta$	nk	Naïve
$O(\varepsilon d)$	$k\delta + \gamma$	$n + O(k \log(d+1) + (\log k)\log(1/\gamma))$	Theorem 1 (main)
$O(\varepsilon)$	$k\delta + \gamma$	$n + O(kd + (\log k)\log(1/\gamma))$	Theorem 4, $d_0 = 1$
$O(\varepsilon d)$	$k\delta + \gamma$	$n + k\log(d+2) + 2\log(1/\gamma) + O(1)$	Theorem 3*
$O(\varepsilon d)$	$2^{O(k\log(d+1))} \cdot \delta$	n	Theorem 5, $d_0 = d$
$O(\varepsilon)$	$2^{O(kd)} \cdot \delta$	n	Theorem 5, $d_0 = 1$
$O(\varepsilon k d/\gamma)$	$k\delta + \gamma$	$n + O(k\log k + k\log d + k\log(1/\gamma))$	Prop. 1 (based on [SZ99])
Any	Any ≤ 0.2	$n + \Omega(k) - \log(\delta'/\delta)$	Theorem 9 (lower bound)

*Computationally inefficient.

Figure 2: Upper and lower bounds for stewards. Recall that ε, δ are the concentration parameters of $f_1, \ldots, f_k$ (i.e. the error and failure probability of the estimation algorithm Est); ε', δ' are the error and failure probability of the steward S; n is the number of input bits to each f_i (i.e. the number of random coins used by Est); k is the number of rounds of adaptivity; d is the dimension of the output of each f_i (i.e. the dimension of the output of Est.) Everywhere it appears, γ denotes an arbitrary positive number.

1.2.3 Application: Simulating an oracle for promise-BPP or APP

Recall that promise-BPP is the class of promise problems that can be decided in probabilistic polynomial time with bounded failure probability. When an algorithm is given oracle access to a promise problem, it is allowed to make queries that violate the promise, and several models have been considered for dealing with such queries. Following Moser [Mos01], we will stipulate that the oracle may respond in any arbitrary way to such queries. (See e.g. [BF99] for two other models.) From these definitions, it is easy to show, for example, that $\mathsf{BPP}^{\mathsf{promise\text{-}BPP}} = \mathsf{BPP}$. Using our steward, we give a time- and randomness-efficient simulation of any algorithm with an oracle for promise-BPP. (As we will discuss in Section 1.4, the corresponding result for BPP-oracle algorithms is trivial.) The algorithm and analysis are almost identical to those used to prove Corollary 1. See Section 6.2 for details.

Recall that APP, introduced by Kabanets et al. [KRC00], is the class of functions $\varphi : \{0,1\}^n \to [0,1]$ that can be approximated to within $\pm\varepsilon$ in probabilistic $\mathrm{poly}(n, 1/\varepsilon)$ time with bounded failure probability. Following Moser [Mos01], we model oracle access to $\varphi \in \mathsf{APP}$ by requiring the oracle algorithm to provide $w \in \{0,1\}^n$ and a unary representation of $1/\varepsilon \in \mathbb{N}$; the oracle is guaranteed to respond with a value that is within $\pm\varepsilon$ of $\varphi(w)$. From these definitions, it is easy to show, for example, that $\mathsf{BPP}^{\mathsf{APP}} = \mathsf{BPP}$. As with promise-BPP, we use our steward to construct a time- and randomness-efficient simulation of any algorithm with an oracle for APP. See Section 6.3 for details.

1.2.4 Application: The Goldreich-Levin algorithm

As a final application, in Section 6.4, we give a randomness-efficient version of the Goldreich-Levin algorithm [GL89] (otherwise known as the Kushilevitz-Mansour algorithm [KM93]) for finding noticeably large Fourier coefficients. Given oracle access to $F : \{0,1\}^n \to \{-1,1\}$, for any $\theta > 0$, we show how to efficiently find a list containing all U with $|\widehat{F}(U)| \geq \theta$. Our algorithm makes $O(n\log(n/\delta)) \cdot \mathrm{poly}(1/\theta)$ queries to F, uses $O(n + (\log n)\log(1/\delta))$ random bits, and has failure probability δ. Notice that the number of random bits *does not depend on* θ. To achieve such a low randomness complexity, we first improve the randomness efficiency of each estimate in the standard Goldreich-Levin algorithm using the GW sampler. Then, we reduce the number of rounds of adaptivity by a factor of $\log(1/\theta)$ by making many estimates within each round. Finally, we apply our steward with $d = \text{poly}(1/\theta)$, unlike our other applications where we choose $d = 1$. (Recall that d is the number of real values estimated in each round.)

1.2.5 Lower bound

We also give a *randomness complexity lower bound* of $n + \Omega(k) - \log(\delta'/\delta)$ for any steward (Section 7). In the case $d \leq O(1)$, this comes close to matching our upper bounds. For example, to achieve $\delta' \leq O(k\delta)$, this lower bound says that $n + \Omega(k)$ random bits are needed; our main steward (Theorem 1) achieves $\varepsilon' \leq O(\varepsilon), \delta' \leq O(k\delta)$ using $n + O(k + (\log k)\log(1/\delta))$ random bits. At the other extreme, if we want a steward that uses only n random bits, this lower bound says that the failure probability will be $\delta' \geq \exp(\Omega(k)) \cdot \delta$; one of our variant stewards (Theorem 5) uses n random bits to achieves $\varepsilon' \leq O(\varepsilon)$ and $\delta' \leq \exp(O(k)) \cdot \delta$.

1.3 Techniques

1.3.1 Block decision trees

A key component in the proof of our main result (Theorem 1) is a pseudorandom generator (PRG) for a new model that we call the *block decision tree* model. Informally, a block decision tree is a decision tree that reads its input from left to right, n bits at a time:

Definition 2. For a finite alphabet Σ, a (k, n, Σ) *block decision tree* is a rooted tree $T = (V, E)$ of height k in which every node v at depth $< k$ has exactly $|\Sigma|$ children (labeled with the symbols in Σ) and has an associated function $v : \{0,1\}^n \to \Sigma$. We identify T with a function $T : (\{0,1\}^n)^{\leq k} \to V$ defined recursively: $T(\text{the empty string}) = \text{the root node}$, and if $T(X_1, \ldots, X_{i-1}) = v$, then $T(X_1, \ldots, X_i)$ is the child of v labeled $v(X_i)$.

The standard nonconstructive argument (Appendix C) shows that there exists a γ-PRG for block decision trees with seed length $n + k\log|\Sigma| + 2\log(1/\gamma) + O(1)$. (See Section 3.1 for the definition of a PRG in this setting.) In Section 3, we explicitly construct a γ-PRG for block decision trees with seed length $n + O(k\log|\Sigma| + (\log k)\log(1/\gamma))$. The generator is constructed by modifying the INW generator for space-bounded computation [INW94].

1.3.2 Shifting and rounding

For a steward S, let $\mathsf{S}(X)$ denote S using randomness X. Our main steward is of the form $\mathsf{S}(X) \stackrel{\text{def}}{=} \mathsf{S}_0(\mathsf{Gen}(X))$. Here, Gen is our PRG for block decision trees, and S_0 is a randomness-inefficient steward. In each round, S_0 queries f_i at a fresh random point $X_i \in \{0,1\}^n$, but S_0 computes the return value Y_i by carefully *shifting and rounding* each coordinate of $f_i(X_i)$. In particular, S_0 finds a single value Δ_i such that after shifting each coordinate of $f_i(X_i)$ according to Δ_i, every coordinate is ε-far from every rounding boundary. Then, S_0 rounds the shifted coordinates to obtain Y_i.

Roughly, the purpose of this shifting and rounding is to reduce the amount of information about X_i that is leaked by Y_i. To make this precise, observe that when any steward and owner interact, it is natural to model the owner's behavior by a decision tree that branches at each node based on the value Y_i provided by the steward. The *branching factor* of this decision tree is a simple measure of the amount of information leaked, and clearly, rounding $f_i(X_i)$ reduces this branching factor. (Blum and Hardt [BH15] used a similar idea in a different setting.)

But more interestingly, we show that the branching factor can be reduced much further by *relaxing* the requirement that the tree perfectly computes $\mathsf{O} \leftrightarrow \mathsf{S}_0$. In particular, for every owner O, we construct a block decision tree T_{O} that merely *certifies correctness* of $\mathsf{O} \leftrightarrow \mathsf{S}_0$. That is, for any $X_1, \ldots, X_k$, if the node $T_{\mathsf{O}}(X_1, \ldots, X_k)$ indicates "success", then the error $\max_i \|Y_i - \mu_i\|_\infty$ in $\mathsf{O} \leftrightarrow \mathsf{S}_0(X_1, \ldots, X_k)$ is small. On the other hand, if $T_{\mathsf{O}}(X_1, \ldots, X_k)$ does not indicate success, then "all bets are off": the error $\max_i \|Y_i - \mu_i\|_\infty$ in $\mathsf{O} \leftrightarrow \mathsf{S}_0(X_1, \ldots, X_k)$ may be small or large.

We show (Lemma 2) that our definition of S_0 ensures the existence of a certification tree T_{O} with a branching factor of only $d+2$ with the additional property that

$$\Pr_{X_1,\ldots,X_k}[T_{\mathsf{O}}(X_1, \ldots, X_k) \text{ indicates success}] \geq 1 - k\delta.$$

Therefore, to save random bits, we don't need to try to fool $\mathsf{O} \leftrightarrow \mathsf{S}_0$. Instead, it suffices for Gen to fool T_{O}. The small branching factor of T_{O} allows Gen to have a correspondingly small seed length.

To construct the tree T_{O}, we think of Δ_i as a *compressed* representation of Y_i. With high probability, given unlimited computation time, O could recover Y_i from Δ_i by computing the *true* vector μ_i, shifting *it* according to Δ_i, and rounding. Each node of the certification tree T_{O}, therefore, just needs to have one child for each possible Δ_i value, along with one $\perp$ child indicating that the compression (and certification) failed.

1.3.3 Lower bound

Our lower bound follows a similar intuition as our upper bounds: we show that in each round, by carefully choosing f_i, the owner can learn $\Omega(1)$ bits of information about the steward's randomness. To conclude that the steward must use $n + \Omega(k) - \log(\delta'/\delta)$ bits of randomness, we show that if the steward has fewer than n bits of randomness remaining from the owner's perspective, then the owner can choose a function that causes the steward's failure probability to be large.

1.4 Why can't we just reuse the random bits?

Notwithstanding our lower bound, the reader might be tempted to think that randomness stewards are trivial: why not just pick $X \in \{0,1\}^n$ uniformly at random *once* and reuse it in every round? For the purpose of discussion, let us generalize, and suppose we are trying to execute an n-coin algorithm A (not necessarily an estimation algorithm) on k inputs $C_1, \ldots, C_k$. If $C_1, \ldots, C_k$ are chosen *non-adaptively* (i.e. all in advance), then we really can use the same X for each execution. By the union bound, the probability that $\mathsf{A}(C_i, X)$ fails for any i is at most $k\delta$.

That argument breaks down in the adaptive case, because C_2 is chosen based on $\mathsf{A}(C_1, X)$, and hence C_2 may be *stochastically dependent* on X, so $\mathsf{A}(C_2, X)$ is not guaranteed to have a low failure probability. Still, the argument can be salvaged in an important special case: Suppose A is a BPP algorithm. Then we can let $\widehat{C}_1, \widehat{C}_2, \ldots, \widehat{C}_k$ be the inputs that would be chosen if A never failed. Then each $\widehat{C}_i$ really is independent of X, so by the union bound, with probability $1 - k\delta$, $\mathsf{A}(\widehat{C}_i, X)$ does not fail for any i. But if $\mathsf{A}(\widehat{C}_i, X)$ does not fail for any i, then by induction, $C_i = \widehat{C}_i$ for every i. So the overall failure probability is once again at most $k\delta$.

The preceding argument applies more generally if A is *pseudodeterministic*, i.e. for each input, there is a unique correct output that A gives with probability $1-\delta$.[1] (Pseudodeterministic algorithms were introduced by Gat and Goldwasser [GG11].) A BPP algorithm is trivially pseudodeterministic.

[1] These two conditions (inputs are chosen nonadaptively, A is pseudodeterministic) are both special cases of the following condition under which the randomness X may be safely reused: for every $1 \leq i \leq k$, C_i is a pseudodeterministic function of $(C_0, C_1, \ldots, C_{i-1})$, where C_0 is a random variable that is independent of X.

Observe, however, that a promise-BPP algorithm is only guaranteed to be pseudodeterministic on inputs that satisfy the promise. This is why the result we mentioned in Section 1.2.3 is in terms of an oracle for promise-BPP. Similarly, estimation algorithms (including APP algorithms) are typically not pseudodeterministic.

In the standard Goldreich-Levin algorithm, randomness is used to estimate $\sum_{U \in \mathcal{U}} \widehat{F}(U)^2$ for certain collections of subsets $\mathcal{U}$. The algorithm's behavior depends on how the estimate compares to $\theta^2/2$. This process is not pseudodeterministic, because if the true value $\sum_{U \in \mathcal{U}} \widehat{F}(U)^2$ is very close to $\theta^2/2$, the estimate falls on each side of $\theta^2/2$ with noticeable probability.

1.5 Related work

1.5.1 Adaptive data analysis

The notion of a randomness steward is inspired by the closely related *adaptive data analysis* problem [DFH$^+$15c, BNS$^+$15, DFH$^+$15a, DFH$^+$15b, HU14, SU14, BH15], introduced by Dwork et al. [DFH$^+$15c]. In the simplest version of this problem, there is an unknown distribution $\mathcal{D}$ over $\{0,1\}^n$ and a *data analyst* who wishes to estimate the mean values (with respect to $\mathcal{D}$) of k adaptively chosen functions $f_1, \ldots, f_k : \{0,1\}^n \to [0,1]$ using as few samples from $\mathcal{D}$ as possible. In this setting, these samples are held by a *mechanism* and not directly accessible by the data analyst. In round i, the data analyst gives f_i to the mechanism, and the mechanism responds with an estimate of $\mathrm{E}_{x \sim \mathcal{D}}[f_i(x)]$. The mechanism constructs the estimate so as to leak as little information as possible about the sample, so that the same sample points can be safely reused for future estimates.

The data analyst and mechanism in the adaptive data analysis setting are analogous to the owner O and steward S in our setting, respectively. In each case, the idea is that the mechanism or steward can intentionally introduce a small amount of error into each estimate to hide information and thereby facilitate future estimates. Note, however, that in the adaptive data analysis problem, there is just one unknown distribution $\mathcal{D}$ and we are concerned with sample complexity, whereas in the randomness stewardship problem, we can think of each concentrated function f_i as defining a new distribution over $\mathbb{R}^d$ and we are concerned with randomness complexity.

1.5.2 The Saks-Zhou algorithm

Another highly relevant construction is the algorithm of Saks and Zhou [SZ99] for simulating randomized logspace algorithms in deterministic space $O(\log^{3/2} n)$. The key component in this algorithm can be reinterpreted as a randomness steward. Using a pseudorandom generator, Saks and Zhou also constructed a randomized algorithm Est that approximates a large power of a given substochastic matrix. (Saks and Zhou used Nisan's generator [Nis92], but any pseudorandom generator for small space can be used – see [Arm98, HU16].) By applying their steward, Saks and Zhou saved random bits when applying Est repeatedly to approximate a much larger power of a given substochastic matrix.

The "Saks-Zhou steward" works by *randomly perturbing and rounding* the output of each f_i, and then reusing the same randomness string X in each round. The perturbation and rounding are somewhat similar to our construction, but note that we shift the outputs of each f_i deterministically, whereas the Saks-Zhou steward uses random perturbations. The rounding parameters are also different. The analysis of the Saks-Zhou steward is similar to the proof that randomness can be safely reused for a pseudodeterministic subroutine; one can show that random perturbation and rounding effectively breaks the dependence between X and Y_i. (See Appendix B for the description and analysis of the Saks-Zhou steward.)

Our steward achieves better parameters than the Saks-Zhou steward (see Figure 2). In particular, to achieve failure probability $k\delta + \gamma$, the error ε' of the Saks-Zhou steward is $O(\varepsilon k d/\gamma)$ – the error grows linearly with k, the number of rounds of adaptivity, as well as with $1/\gamma$ – whereas our steward achieves error $O(\varepsilon d)$. Furthermore, the Saks-Zhou steward uses $n + O(k \log k + k \log d + k \log(1/\gamma))$ random bits, whereas our steward uses only $n + O(k \log(d+1) + (\log k)\log(1/\gamma))$ random bits.

1.5.3 Decision trees and branching programs

In the most common decision tree model, the branching factor $|\Sigma|$ is just 2, and each node reads an arbitrary bit of the input. In the more general *parity decision tree* model, each node computes the parity of some subset of the input bits. Kushilevitz and Mansour showed [KM93] that the Fourier ℓ_1 norm of any Boolean function computed by a parity decision tree is at most 2^k, the number of leaves in the tree. It is well-known (and easy to prove) that this implies that a γ-biased generator is a $(2^k\gamma)$-PRG for parity decision trees. Using e.g. the small-bias generator of Naor and Naor [NN93], this gives an efficient PRG for parity decision trees with asymptotically optimal seed length.

Decision trees in which each node computes a more complicated function have also been studied previously. Bellare [Bel92] introduced the *universal decision tree* model, in which each node computes an arbitrary Boolean function of the input bits. He gave a bound on the ℓ_1 norm of any Boolean function computed by a universal decision tree in terms of the ℓ_1 norms of the functions at each node. Unfortunately, for block decision trees, his bound is so large that it does not immediately imply any nontrivial pseudorandom generators for block decision trees.

A block decision tree can be thought of as a kind of space-bounded computation. Indeed, a block decision tree is a specific kind of *ordered branching program* of width $|\Sigma|^k$ and length k that reads n bits at a time. Hence, we could directly apply a pseudorandom generator for ordered branching programs, such as the INW generator [INW94]. For these parameters, the INW generator has seed length of $n + O(k \log k \log |\Sigma| + \log k \log(1/\gamma))$. This seed length can be slightly improved by instead using Armoni's generator [Arm98] (a generalization of the Nisan-Zuckerman generator [NZ96]), but even that slightly improved seed length is larger than the seed length of the generator we construct.

1.5.4 Finding noticeably large Fourier coefficients

Our randomness-efficient version of the Goldreich-Levin algorithm should be compared to the results of Bshouty et al. [BJT99], who gave several algorithms for finding noticeably large Fourier coefficients, all closely related to one another and based on an algorithm of Levin [Lev93].

- Bshouty et al. gave one algorithm [BJT99, Figure 4] that makes $O(\frac{n}{\theta^2}\log(\frac{n}{\delta\theta}))$ queries and uses $O(n\log(\frac{n}{\theta})\log(\frac{1}{\delta\theta}))$ random bits. Our algorithm has better randomness complexity, but worse query complexity.

- Bshouty et al. gave another algorithm [BJT99, Figure 5] that makes only $O(n/\theta^2)$ queries and uses just $O(\log(n/\theta)\cdot\log(1/\theta))$ random bits, but it merely outputs a list such with probability 1/2, some U in the list satisfies $|\widehat{F}(U)| \geq \theta$, assuming such a U exists.

1.6 Outline of this paper

In Section 2, we describe the shifting and rounding steward S_0 and prove that it admits certification trees with a small branching factor. Then, in Section 3, we construct and analyze our pseudorandom generator for block decision trees. In Section 4, we put these pieces together to prove our main result (Theorem 1). In Section 5, we show how to construct our variant stewards. In Section 6,

1. For $i = 1$ to k:
   (a) O chooses $f_i : \{0,1\}^n \to \mathbb{R}^d$ and gives it to M.
   (b) S_0 picks *fresh randomness* $X_i \in \{0,1\}^n$ and gives it to M.
   (c) M gives $W_i \stackrel{\text{def}}{=} f_i(X_i)$ to S_0.
   (d) S_0 computes Y_i by shifting and rounding W_i according to the algorithm in Section 2.1.
   (e) S_0 gives Y_i to O.

Figure 3: Outline of $\mathsf{O} \leftrightarrow \mathsf{S}_0$.

we explain our applications of our main steward. Finally, in Section 7, we prove our randomness complexity lower bound for stewards.

2 The shifting and rounding steward S_0

As a building block for our main steward constructions, we first construct our randomness-inefficient steward S_0. Recall that any steward makes two choices in each round: the input X_i to f_i and the estimate $Y_i \in \mathbb{R}^d$. The steward S_0 focuses on the second choice: it queries each f_i at a fresh random point $X_i \in \{0,1\}^n$, but it carefully shifts and rounds the output of f_i. (See Figure 3.)

2.1 The shifting and rounding algorithm

We now describe the algorithm by which S_0 computes $Y_i \in \mathbb{R}^d$ from $W_i \stackrel{\text{def}}{=} f_i(X_i)$. Fix $n, k, d \in \mathbb{N}$ and $\varepsilon, \delta > 0$. Let $[d]$ denote the set $\{1, 2, \ldots, d\}$. Partition $\mathbb{R}$ into half-open intervals of length $(d+1) \cdot 2\varepsilon$. For $w \in \mathbb{R}$, let $\mathsf{Round}(w)$ denote the midpoint of the interval containing w. Given $W_i \in \mathbb{R}^d$:

1. Find $\Delta_i \in [d+1]$ such that for every $j \in [d]$, there is a single interval that entirely contains $[W_{ij} + (2\Delta_i - 1)\varepsilon, W_{ij} + (2\Delta_i + 1)\varepsilon]$. (We will show that such a Δ_i exists.)
2. For every $j \in [d]$, set $Y_{ij} = \mathsf{Round}(W_{ij} + 2\Delta\varepsilon)$.

We must show that this algorithm is well-defined:

Lemma 1. *For any $W \in \mathbb{R}^d$, there exists $\Delta \in [d+1]$ such that for every $j \in [d]$, there is a single interval that entirely contains $[W_j + (2\Delta - 1)\varepsilon, W_j + (2\Delta + 1)\varepsilon]$.*

Proof. Consider picking $\Delta \in [d+1]$ uniformly at random. Then for each j, the probability that $[W_j + (2\Delta - 1)\varepsilon, W_j + (2\Delta + 1)\varepsilon]$ intersects two distinct intervals is precisely $1/(d+1)$ by our choice of the length of the intervals. The union bound over d different j values completes the proof. $\square$

2.2 Analysis: Certification trees

As outlined in Section 1.3.2, the key lemma says that for any owner O, there exists a block decision tree T_{O} with a small branching factor that certifies correctness of $\mathsf{O} \leftrightarrow \mathsf{S}_0$:

Lemma 2. *Assume $\delta < 1/2$. Let $\Sigma = [d+1] \cup \{\perp\}$. For any deterministic owner O, there exists a (k, n, Σ) block decision tree T_{O} with the following properties.*

1. *For any internal node v, $\Pr_{X \in \{0,1\}^n}[v(X) = \perp] \leq \delta$.*
2. *Fix $X_1, \dots, X_k \in \{0,1\}^n$, and suppose that the path from the root to $T_{\mathsf{O}}(X_1, \dots, X_k)$ does not include any $\perp$ nodes. Then $\max_i \|Y_i - \mu_i\|_\infty \leq O(\varepsilon d)$ in $\mathsf{O} \leftrightarrow \mathsf{S}_0(X_1, \dots, X_k)$.*

From Lemma 2, it easily follows that if Gen is a γ-PRG for (k, n, Σ) block decision trees, then $\mathsf{S}_0(\mathsf{Gen}(X))$ is an $(O(\varepsilon d), k\delta + \gamma)$-steward: the probability over X that $T_{\mathsf{O}}(\mathsf{Gen}(X))$ passes through any $\perp$ nodes is at most $k\delta + \gamma$. Instantiating Gen with an explicit PRG with a short seed length will complete the proof of our main result. (See Section 4 for details.)

Notice that Lemma 2 does not assert that T_{O} computes the transcript of $\mathsf{O} \leftrightarrow \mathsf{S}_0$. In fact, we will construct T_{O} to compute the transcript of a channel in a *different* protocol involving O and S_0. In this new protocol, each Y_i chosen by S_0 is *compressed and decompressed* before giving it to O, as we suggested in Section 1.3.2. To facilitate the analysis, we introduce *two mediators*: one to compress and another to decompress.

Proof of Lemma 2. T_{O} will be defined in terms of a thought experiment in which S_0 interacts with O through two mediators M_1 and M_2 as described above. The protocol is as follows:

1. For $i = 1$ to k:
   (a) O chooses $f_i : \{0,1\}^n \to \mathbb{R}^d$ that is (ε, δ)-concentrated at some point $\mu_i \in \mathbb{R}^d$ and gives it to M_1, who gives it to M_2.
   (b) M_1 and M_2 both compute the smallest $\widehat{\mu}_i \in \mathbb{R}^d$ (under, say, the lexicographical order) such that f_i is (ε, δ)-concentrated at $\widehat{\mu}_i$. (This exists, because $\{0,1\}^n$ is finite, so the set of points where f_i is concentrated is a closed subset of $\mathbb{R}^d$.)
   (c) S_0 chooses a string $X_i \in \{0,1\}^n$ and gives it to M_2, who gives $f_i(X_i) \in \mathbb{R}^d$ to S_0.
   (d) S_0 chooses $Y_i \in \mathbb{R}^d$. In the ordinary steward protocol, S_0 gives Y_i to O, but in the two-mediator protocol, the message is redirected to M_2.
   (e) Say a value $\Delta \in [d+1]$ is *compatible* with Y_i if $Y_{ij} = \mathsf{Round}(\widehat{\mu}_{ij} + 2\Delta\varepsilon)$ for every $j \in [d]$. M_2 computes

$$\widehat{\Delta}_i = \begin{cases} \text{the smallest } \Delta \in [d+1] \text{ compatible with } Y_i & \text{if any such } \Delta \text{ exists} \\ \perp & \text{otherwise} \end{cases}$$

   and gives it to M_1.
   (f) M_1 computes $\widehat{Y}_i = (\widehat{Y}_{i1}, \dots, \widehat{Y}_{id})$, where for each $j \in [d]$,

$$\widehat{Y}_{ij} = \begin{cases} \mathsf{Round}(\widehat{\mu}_{ij} + 2\widehat{\Delta}_i\varepsilon) & \text{if } \widehat{\Delta}_i \neq \perp \\ 0 & \text{otherwise} \end{cases}$$

   and gives $\widehat{Y}_i$ to O.

(See Figure 4.) Let $\mathsf{O} \xleftrightarrow{2} \mathsf{S}_0$ ("the two-mediator interaction of O with S_0") denote the above interaction between O and S_0.

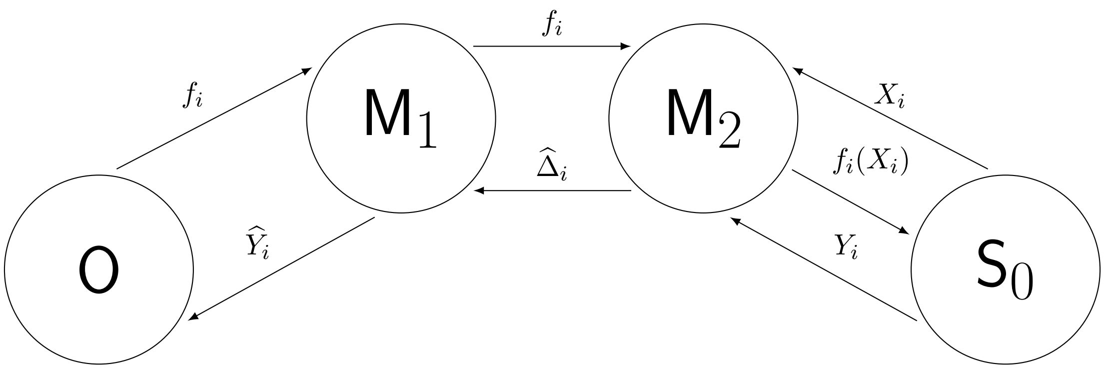

Figure 4: A single round of interaction in the thought experiment in the proof of Lemma 2.

Definition of T_{O} Let V be the vertex set of T_{O}. We first define T_{O} *as a function.* Because $\mathsf{S}_0(X_1, \ldots, X_k)$ looks at X_i only in round i, we can sensibly speak of the first i rounds of $\mathsf{O} \xleftrightarrow{2} \mathsf{S}_0(X_1, \ldots, X_i)$ even for $i < k$. This allows us to define $T_{\mathsf{O}}(X_1, \ldots, X_i)$ to be the node $v \in V$ such that the path from the root to v is described by the values $\widehat{\Delta}_1, \ldots, \widehat{\Delta}_i$ that M_2 sends in $\mathsf{O} \xleftrightarrow{2} \mathsf{S}_0(X_1, \ldots, X_i)$.

Now, we must show that this function T_{O} can be realized as a block decision tree, i.e. that each internal node $v \in V$ can be assigned a transition function $v : \{0,1\}^n \to \Sigma$ that is compatible with the definition of T_{O} as a function. Indeed, observe that $\widehat{\Delta}_1, \ldots, \widehat{\Delta}_{i-1}$ fully determine the state of O after the first $i-1$ rounds of $\mathsf{O} \xleftrightarrow{2} \mathsf{S}_0(X_1, \ldots, X_i)$ and hence determines the function f_i. Furthermore, S_0 is "memoryless", i.e. Y_i is fully determined by f_i and X_i. Because of how M_2 behaves, this shows that $\widehat{\Delta}_i$ is fully determined by $\widehat{\Delta}_1, \ldots, \widehat{\Delta}_{i-1}$ and X_i. So there is a function $\varphi : (\widehat{\Delta}_1, \ldots, \widehat{\Delta}_{i-1}, X_i) \mapsto \widehat{\Delta}_i$, and if the path from the root to v is described by $\widehat{\Delta}_1, \ldots, \widehat{\Delta}_{i-1}$, we can set $v(X_i) \stackrel{\text{def}}{=} \varphi(\widehat{\Delta}_1, \ldots, \widehat{\Delta}_{i-1}, X_i)$.

Analysis of T_{O} By the definition of T_{O} as a function, to prove Condition 1 in the lemma statement, we must show that in each round of $\mathsf{O} \xleftrightarrow{2} \mathsf{S}_0$, $\Pr[\widehat{\Delta}_i = \perp] \le \delta$. Indeed, by concentration, with probability $1 - \delta$, for every j, $|W_{ij} - \widehat{\mu}_{ij}| \le \varepsilon$. In this case, by the construction of S_0, $W_{ij} + 2\Delta_i\varepsilon$ and $\widehat{\mu}_{ij} + 2\Delta_i\varepsilon$ are in the same interval for every $j \in [d]$. Therefore, in this case, there is at least one value Δ compatible with Y_i, namely the value Δ_i used by S_0.

Finally, to prove Condition 2 in the lemma statement, suppose the path from the root node to $T_{\mathsf{O}}(X_1, \ldots, X_k)$ does not include any $\perp$ nodes. Then in $\mathsf{O} \xleftrightarrow{2} \mathsf{S}_0(X_1, \ldots, X_k)$, for every i, $\widehat{\Delta}_i \neq \perp$. This implies that every Y_{ij} is of the form $\mathsf{Round}(\widehat{\mu}_{ij} + 2\widehat{\Delta}_i\varepsilon)$ for some $\widehat{\Delta}_i \in [d+1]$. Therefore, $|Y_{ij} - \widehat{\mu}_{ij}| \le 3(d+1)\varepsilon$, since $2\widehat{\Delta}_i\varepsilon \le 2(d+1)\varepsilon$ and rounding introduces at most $(d+1)\varepsilon$ additional error. Since $\delta < 1/2$, $\|\widehat{\mu}_i - \mu_i\|_\infty \le 2\varepsilon$, so $\|Y_i - \mu_i\|_\infty \le 3(d+1)\varepsilon + 2\varepsilon = (3d+5)\varepsilon$.

Of course, that is still all in $\mathsf{O} \xleftrightarrow{2} \mathsf{S}_0(X_1, \ldots, X_k)$. But the crucial point is, for every i, since $\widehat{\Delta}_i \neq \perp$, we can be sure that $Y_i = \widehat{Y}_i$. Therefore, the values $\mu_1, \ldots, \mu_k, Y_1, \ldots, Y_k$ in $\mathsf{O} \xleftrightarrow{2} \mathsf{S}_0(X_1, \ldots, X_k)$ are *exactly the same* as they are in $\mathsf{O} \leftrightarrow \mathsf{S}_0(X_1, \ldots, X_k)$! Therefore, in $\mathsf{O} \leftrightarrow \mathsf{S}_0(X_1, \ldots, X_k)$, for every i, $\|Y_i - \mu_i\|_\infty \le (3d+5)\varepsilon$. □

Notice that in $\mathsf{O} \xleftrightarrow{2} \mathsf{S}_0(X_1, \ldots, X_k)$, if $\widehat{\Delta}_i = \perp$ for some i, then the interaction might diverge from $\mathsf{O} \leftrightarrow \mathsf{S}_0(X_1, \ldots, X_k)$, in which case $T_{\mathsf{O}}(X_1, \ldots, X_k)$ does not encode the transcript of $\mathsf{O} \leftrightarrow \mathsf{S}_0(X_1, \ldots, X_k)$ in any way.

3 Pseudorandomness for block decision trees

Recall that our goal is to modify the internal parameters of the INW generator, thereby constructing a γ-PRG for (k, n, Σ) block decision trees with seed length $n + O(k \log |\Sigma| + (\log k) \log(1/\gamma)$. The construction and analysis mimic the standard treatment of the INW generator, and the reader who is familiar with the INW generator is encouraged to skip to Section 3.4 to just see the new parameters.

3.1 Formal definitions and theorem statement

Let U_n denote the uniform distribution on $\{0,1\}^n$. For two probability distributions μ, μ' on the same measurable space, write $\mu \sim_\gamma \mu'$ to indicate that μ and μ' have total variation distance at most γ.

Definition 3. We say that $\mathsf{Gen} : \{0,1\}^s \to \{0,1\}^{nk}$ is a *γ-PRG for (k, n, Σ) block decision trees* if for every such tree T, $T(\mathsf{Gen}(U_s)) \sim_\gamma T(U_{nk})$.

Theorem 2. *For every $n, k \in \mathbb{N}$, every finite alphabet Σ, and every $\gamma > 0$, there exists a γ-PRG $\mathsf{Gen} : \{0,1\}^s \to \{0,1\}^{nk}$ for (k, n, Σ) block decision trees with seed length*

$$s \leq n + O(k \log |\Sigma| + (\log k) \log(1/\gamma)).$$

The PRG can be computed in $\mathrm{poly}(n, k, \log |\Sigma|, \log(1/\gamma))$ *time.*

3.2 Concatenating PRGs for block decision trees

Toward proving Theorem 2, for a (k, n, Σ) block decision tree $T = (V, E)$ and a node $v \in V$, let T_v denote the subtree rooted at v, and observe that we can think of T_v as a (k', n, Σ) block decision tree, where $k' = k - \mathrm{depth}(v)$. This simple observation – after a block decision tree has been computing for a while, the remaining computation is just another block decision tree – implies that pseudorandom generators for block decision trees can be *concatenated* with mild error accumulation. This fact and its easy proof are perfectly analogous to the situation with ordered branching programs. We record the details below.

Lemma 3. *Suppose* $\mathsf{Gen}_1 : \{0,1\}^{s_1} \to \{0,1\}^{nk_1}$ *is a γ_1-PRG for (k_1, n, Σ) block decision trees and* $\mathsf{Gen}_2 : \{0,1\}^{s_2} \to \{0,1\}^{nk_2}$ *is a γ_2-PRG for (k_2, n, Σ) block decision trees. Let* $\mathsf{Gen}(x, y) = (\mathsf{Gen}_1(x), \mathsf{Gen}_2(y))$. *Then* Gen *is a $(\gamma_1 + \gamma_2)$-PRG for $(k_1 + k_2, n, \Sigma)$ block decision trees.*

Proof. Fix a $(k_1 + k_2, n, \Sigma)$ block decision tree T. For a node u at depth k_1 and a leaf node v, define

$$\begin{aligned} p(u) &= \Pr[T(U_{nk_1}) = u] & p(v \mid u) &= \Pr[T_u(U_{nk_2}) = v] \\ \widetilde{p}(u) &= \Pr[T(\mathsf{Gen}_1(U_{s_1})) = u] & \widetilde{p}(v \mid u) &= \Pr[T_u(\mathsf{Gen}_2(U_{s_2})) = v]. \end{aligned}$$

To prove correctness of Gen, recall that ℓ_1 distance is twice total variation distance. The ℓ_1 distance between $T(\mathsf{Gen}(U_{s_1+s_2}))$ and $T(U_{n(k_1+k_2)})$ is precisely $\sum_{u,v} |p(u)p(v \mid u) - \widetilde{p}(u)\widetilde{p}(v \mid u)|$. By the triangle inequality, this is bounded by

$$\begin{aligned} &\sum_{u,v} |p(u)p(v \mid u) - p(u)\widetilde{p}(v \mid u)| + \sum_{u,v} |p(u)\widetilde{p}(v \mid u) - \widetilde{p}(u)\widetilde{p}(v \mid u)| \\ = &\sum_{u,v} p(u) \cdot |p(v \mid u) - \widetilde{p}(v \mid u)| + \sum_{u,v} |p(u) - \widetilde{p}(u)| \cdot \widetilde{p}(v \mid u) \\ = &\sum_{u} p(u) \sum_{v} |p(v \mid u) - \widetilde{p}(v \mid u)| + \sum_{u} |p(u) - \widetilde{p}(u)|. \end{aligned}$$

By the correctness of Gen_1 and Gen_2, this is bounded by $(\sum_u p(u) \cdot 2\gamma_2) + 2\gamma_1 = 2(\gamma_1 + \gamma_2)$. $\square$

3.3 Recycling randomness

We find it most enlightening to think of the INW generator in terms of extractors, as suggested by Raz and Reingold [RR99] and in the spirit of the Nisan-Zuckerman generator [NZ96]. The analysis is particularly clean if we work with *average-case extractors*, a concept introduced by Dodis et al. [DORS08].

Definition 4. For discrete random variables X, V, the *average-case conditional min-entropy* of X given V is

$$\widetilde{H}_\infty(X \mid V) = -\log_2 \left(\mathop{\mathrm{E}}_{v \sim V} \left[2^{-H_\infty(X \mid V = v)} \right] \right),$$

where H_∞ is (standard) min-entropy.

Intuitively, $\widetilde{H}_\infty(X \mid V)$ measures the amount of randomness in X from the perspective of someone who knows V. The output of an *average-case extractor* is required to look uniform even from the perspective of someone who knows V, as long as its first input is sampled from a distribution that has high min-entropy conditioned on V:

Definition 5. We say that $\mathsf{Ext} : \{0,1\}^s \times \{0,1\}^d \to \{0,1\}^m$ is an *average-case $(s-t, \beta)$-extractor* if for every X distributed on $\{0,1\}^s$ and every discrete random variable V such that $\widetilde{H}_\infty(X \mid V) \geq s - t$, if we let $Y \sim U_d$ be independent of (X, V) and let $Z \sim U_m$ be independent of V, then $(V, \mathsf{Ext}(X, Y)) \sim_\beta (V, Z)$.

Average-case extractors are the perfect tools for *recycling randomness* in space-bounded computation. We record the details for block decision trees below.

Lemma 4 (Randomness recycling lemma for block decision trees)**.** *Suppose* $\mathsf{Gen} : \{0,1\}^s \to \{0,1\}^{nk}$ *is a γ-PRG for (k, n, Σ) block decision trees and* $\mathsf{Ext} : \{0,1\}^s \times \{0,1\}^d \to \{0,1\}^s$ *is an average-case $(s - k \log|\Sigma|, \beta)$-extractor. Define*

$$\mathsf{Gen}'(x, y) = (\mathsf{Gen}(x), \mathsf{Gen}(\mathsf{Ext}(x, y))).$$

Then Gen' *is a $(2\gamma + \beta)$-PRG for $(2k, n, \Sigma)$ block decision trees.*

Proof. Let T be a $(2k, n, \Sigma)$ block decision tree. Let $X \sim U_s$ and let $V = T(\mathsf{Gen}(X))$. By [DORS08, Lemma 2.2b], the fact that V can be described using $k \log|\Sigma|$ bits implies that $\widetilde{H}_\infty(X \mid V) \geq s - k \log|\Sigma|$. Therefore, by the average-case extractor condition, if we let $Y \sim U_d$ be independent of X and $Z \sim U_d$ be independent of V, then

$$(V, \mathsf{Ext}(X, Y)) \sim_\beta (V, Z).$$

Applying a (deterministic) function can only make the distributions closer. Apply the function $(v, z) \mapsto T_v(\mathsf{Gen}(z))$:

$$T(\mathsf{Gen}'(X, Y)) \sim_\beta T(\mathsf{Gen}(X), \mathsf{Gen}(Z)).$$

By Lemma 3, the right-hand side is (2γ)-close to $T(U_{2nk})$. The triangle inequality completes the proof. □

To actually construct a generator, we will need to instantiate this randomness recycling lemma with an explicit average-case extractor:

Lemma 5. *For every $s, t \in \mathbb{N}$ and every $\beta > 0$, there exists an average-case $(s-t, \beta)$-extractor* $\mathsf{Ext} : \{0,1\}^s \times \{0,1\}^d \to \{0,1\}^s$ *with seed length $d \leq O(t + \log(1/\beta))$ computable in time* $\mathrm{poly}(s, \log(1/\beta))$.

Proof sketch. It is standard (and can be proven using expanders, see e.g. [Vad12]) that there exists an *ordinary* $(s - t - \log(2/\beta), \beta/2)$-extractor $\mathsf{Ext} : \{0,1\}^s \times \{0,1\}^d \to \{0,1\}^s$ with seed length $d \leq O(t + \log(1/\beta))$ computable in time $\mathrm{poly}(s, \log(1/\beta))$. By the same argument as that used to prove [DORS08, Lemma 2.3], Ext is automatically an average-case $(s - t, \beta)$-extractor. □

3.4 The recursive construction

Proof of Theorem 2. Define $\beta = \gamma/2^{\lceil \log k \rceil}$. For $i \geq 0$, define $s_i \in \mathbb{N}$, $d_i \in \mathbb{N}$, $G_i : \{0,1\}^{s_i} \to \{0,1\}^{n \cdot 2^i}$, and $\mathsf{Ext}_i : \{0,1\}^{s_i} \times \{0,1\}^{d_i} \to \{0,1\}^{s_i}$ through mutual recursion as follows. Start with $s_0 = n$ and $G_0(x) = x$. Having already defined s_i and G_i, let Ext_i be the average-case $(s_i - 2^i \log|\Sigma|, \beta)$-extractor of Lemma 5, and let d_i be its seed length. Then let $s_{i+1} = s_i + d_i$, and let

$$G_{i+1}(x, y) = (G_i(x), G_i(\mathsf{Ext}_i(x, y))).$$

We show by induction on i that G_i is a $(\beta \cdot (2^i - 1))$-PRG for $(2^i, n, \Sigma)$ block decision trees. In the base case $i = 0$, this is trivial. For the inductive step, apply Lemma 4, and note that $2\beta(2^i - 1) + \beta = \beta(2^{i+1} - 1)$. This completes the induction. Therefore, we can let $\mathsf{Gen} = G_{\lceil \log k \rceil}$, since $\beta \cdot (2^{\lceil \log k \rceil} - 1) < \gamma$. The seed length $s_{\lceil \log k \rceil}$ of Gen is

$$\begin{aligned} n + \sum_{i=0}^{\lceil \log k \rceil} d_i &\leq n + O\left(\sum_{i=0}^{\lceil \log k \rceil} (2^i \log|\Sigma| + \log k + \log(1/\gamma)) \right) \\ &\leq n + O(k \log|\Sigma| + (\log k) \log(1/\gamma)). \end{aligned}$$

The time needed to compute $\mathsf{Gen}(x)$ is just the time needed for $O(k)$ applications of Ext_i for various $i \leq O(\log k)$, which is $\mathrm{poly}(n, k, \log|\Sigma|, \log(1/\gamma))$. □

4 Proof of main result

Without loss of generality, assume $\delta < 1/2$. (If $\delta \geq 1/2$, then either $k = 1$ or $k\delta \geq 1$; in either case, the result is trivial.) Let S_0 be the steward of Section 2, let Σ be the alphabet of Lemma 2, and let Gen be the γ-PRG for (k, n, Σ) block decision trees of Theorem 2. The steward is $\mathsf{S}(X) \stackrel{\text{def}}{=} \mathsf{S}_0(\mathsf{Gen}(X))$.

Consider any owner O. We may assume without loss of generality that O is deterministic, because a randomized owner is just a distribution over deterministic owners. By Condition 2 of Lemma 2 and the union bound,

$$\Pr[\text{some node in the path from the root to } T_{\mathsf{O}}(U_{nk}) \text{ is labeled } \perp] \leq k\delta.$$

Therefore, when T_{O} reads $\mathsf{Gen}(U_s)$ instead of U_{nk}, the probability is at most $k\delta + \gamma$. By Condition 2 of Lemma 2, this proves the correctness of S. The randomness complexity of S is just the seed length of Gen, which is indeed $n + O(k \log|\Sigma| + (\log k)\log(1/\gamma)) = n + O(k \log(d+1) + (\log k)\log(1/\gamma))$. The total runtime of S is clearly $\mathrm{poly}(n, k, d, \log(1/\varepsilon), \log(1/\gamma))$. □

5 Variant stewards

Theorem 3. *For any $n, k, d \in \mathbb{N}$, for any $\varepsilon, \delta, \gamma > 0$, there exists a (computationally inefficient) $(O(\varepsilon d), k\delta + \gamma)$-steward for k adaptively chosen (ε, δ)-concentrated functions $f_1, \ldots, f_k : \{0,1\}^n \to \mathbb{R}^d$ with randomness complexity*

$$n + k \log(d + 2) + 2 \log(1/\gamma) + O(1).$$

Proof sketch. Mimic the proof of Theorem 1, but use a PRG obtained by the standard nonconstructive argument (Appendix C). $\square$

The shifting and rounding steward S_0 can be generalized to achieve a tradeoff between low error ε' and low branching factor $|\Sigma|$ of the certification tree T_{O}. In particular, for any factorization $d = d_0 d_1$, one can reduce the error from $O(\varepsilon d)$ down to $O(\varepsilon d_0)$ at the cost of increasing the branching factor of T_{O} from $d+2$ up to $(d_0+1)^{d_1}+1$. This is achieved by simply partitioning the d coordinates into d_1 groups of d_0 coordinates and shifting each group individually; the details are in Appendix A. This immediately implies the following generalization of Theorem 1, which achieves a tradeoff between error and randomness complexity:

Theorem 4. *For any $n, k, d, d_0 \in \mathbb{N}$ with $d_0 \leq d$, for any $\varepsilon, \delta, \gamma > 0$, there exists an $(O(\varepsilon d_0), k\delta + \gamma)$-steward for k adaptively chosen (ε, δ)-concentrated functions $f_1, \ldots, f_k : \{0,1\}^n \to \mathbb{R}^d$ with randomness complexity*

$$n + O\left(\frac{kd\log(d_0+1)}{d_0} + (\log k)\log(1/\gamma)\right).$$

The total running time of the steward is $\mathrm{poly}(n, k, d, \log(1/\varepsilon), \log(1/\gamma))$.

Recall from the introduction that if $f_1, \ldots, f_k$ are chosen nonadaptively, then we can reuse randomness and just union bound over the k functions. We now show that we can reuse the randomness in S_0, as long as we union bound over *all the nodes* in the certification tree. (This is similar to the analysis of the Saks-Zhou steward, except that in the Saks-Zhou case, the branching factor of the tree is just 1. It is also similar to the analysis in [BH15].) This gives a steward with very low randomness complexity but large failure probability:

Theorem 5. *For any $n, k, d, d_0 \in \mathbb{N}$ with $d_0 \leq d$, for any $\varepsilon, \delta > 0$, there exists an $(O(\varepsilon d_0), \delta')$-steward for k adaptively chosen (ε, δ)-concentrated functions $f_1, \ldots, f_k : \{0,1\}^n \to \mathbb{R}^d$ with randomness complexity n, where*

$$\delta' \leq \exp\left(O\left(\frac{kd\log(d_0+1)}{d_0}\right)\right) \cdot \delta.$$

The total running time of the steward is $\mathrm{poly}(n, k, d, \log(1/\varepsilon))$.

Proof. Assume without loss of generality that d is a multiple of d_0 and that $\delta < 1/2$. The steward is $\mathsf{S}(X) \stackrel{\text{def}}{=} \mathsf{S}_0(X, X, X, \ldots, X)$, where S_0 is the steward of Section 2 generalized as in Appendix A. To prove correctness, fix any deterministic owner O. Let T_{O} be the block decision tree of Lemma 11. By Condition 1 of Lemma 11, from any internal node, if T_{O} reads X, the probability that it moves to the $\perp$ child is at most δ. Therefore, by the union bound over all nodes, the probability that there is some node from which T_{O} would move to the $\perp$ child upon reading X is at most the value δ' in the lemma statement. By Condition 2 of Lemma 11, if no node in T_{O} takes a $\perp$ transition upon reading X, then $\max_i \|\mu_i - Y_i\|_\infty \leq O(\varepsilon d_0)$ in $\mathsf{O} \leftrightarrow \mathsf{S}(X)$. $\square$

6 Applications

6.1 Acceptance probabilities of Boolean circuits

A (ε, δ)-*sampler* for Boolean functions on n bits is a randomized oracle algorithm Samp such that for any Boolean function $C : \{0,1\}^n \to \{0,1\}$, if we let $\mu(C) \stackrel{\text{def}}{=} 2^{-n}\sum_x C(x)$ be the acceptance probability of C, then $\Pr[|\mathsf{Samp}^C - \mu(C)| > \varepsilon] \leq \delta$. We will use a near-optimal sampler constructed by Goldreich and Wigderson [GW97]:

Lemma 6 ([GW97, Theorem 6.5])**.** *For every $n \in \mathbb{N}$ and every $\varepsilon, \delta > 0$, there is an (ε, δ)-sampler for Boolean functions on n bits that makes $O(\log(1/\delta)/\varepsilon^2)$ queries, uses $n + O(\log(1/\delta))$ random bits, and runs in time* $\mathrm{poly}(n, 1/\varepsilon, \log(1/\delta))$.

Proof of Corollary 1. Let c be the constant under the $O(\cdot)$ of the error ε' in the steward of Theorem 1. When given parameters $n, k, \varepsilon, \delta$, let Samp be the Boolean $(\varepsilon/c, \delta/(2k))$-sampler of Lemma 6, and say it uses m coins. Let S be the (ε, δ)-steward of Theorem 1 for k adaptively chosen $(\varepsilon/c, \delta/(2k))$-concentrated functions $f_1, \ldots, f_k : \{0,1\}^m \to \mathbb{R}$. (So $\gamma = \delta/2$.) When given circuit C_i, define $f_i(X) = \mathsf{Samp}^{C_i}(X)$, i.e. the output Samp^{C_i} with randomness X. Give f_i to S, and output the value Y_i that it returns.

Proof of correctness: The definition of a sampler implies that each f_i is $(\varepsilon/c, \delta/(2k))$-concentrated at $\mu(C_i)$. Furthermore, each f_i is defined purely in terms of C_i, which is chosen based only on $Y_1, \ldots, Y_{i-1}$. Therefore, the steward guarantee implies that with probability $1 - \delta$, every Y_i is within $\pm\varepsilon$ of $\mu(C_i)$.

Randomness complexity analysis: The number of bits m used by the sampler is $n + O(\log(k/\delta))$. Therefore, the number of bits used by the steward is

$$n + O(\log(k/\delta)) + O(k + (\log k)\log(1/\delta)) = n + O(k + (\log k)\log(1/\delta)).$$

Runtime analysis: The runtime of the steward is $\mathrm{poly}(m, k, \log(1/\gamma)) = \mathrm{poly}(n, k, \log(1/\delta))$. The runtime of the sampler is $\mathrm{poly}(n, 1/\varepsilon, \log k, \log(1/\delta))$. The time required to evaluate each query of the sampler in round i is $O(\mathrm{size}(C_i))$ (assuming we work with a suitable computational model and a suitable encoding of Boolean circuits.) The number of queries that the sampler makes in each round is $O(\log(k/\delta)/\varepsilon^2)$. Therefore, the total runtime of this algorithm is

$$O\left(\frac{\log k + \log(1/\delta)}{\varepsilon^2} \cdot \sum_{i=1}^{k} \mathrm{size}(C_i)\right) + \mathrm{poly}(n, k, 1/\varepsilon, \log(1/\delta)). \qquad \square$$

6.2 Simulating a promise-BPP oracle

Theorem 6. *Suppose a search problem Π can be solved by a deterministic* $\mathsf{promise\text{-}BPP}$*-oracle algorithm that runs in time T and makes k queries, and suppose that (regardless of previous oracle responses) each query of this algorithm can be decided by a randomized algorithm that runs in time T', uses n coins, and has failure probability* $1/3$. *Then for any δ, Π can be solved by a randomized (non-oracle) algorithm that runs in time*

$$T + O(T' \cdot k \log(k/\delta)) + \mathrm{poly}(n, k, \log(1/\delta)),$$

has randomness complexity

$$n + O(k + (\log k)\log(1/\delta)),$$

and has failure probability δ.

(Recall that search problems generalize decision problems and function problems. In reality, the theorem generalizes to just about any kind of "problem", but we restrict ourselves to search problems for concreteness.) The theorem can easily be extended to randomized oracle algorithms by considering the problem of executing the randomized oracle algorithm using a given randomness string.

As a reminder, as discussed in Section 1.4, Theorem 6 would be trivial if it involved a BPP oracle instead of a $\mathsf{promise\text{-}BPP}$ oracle. Indeed, in the BPP case, the randomness can be reduced to just $n + O(\log k + \log(1/\delta))$. This is because a BPP algorithm is pseudodeterministic, so the randomness can be safely reused from one query to the next. A $\mathsf{promise\text{-}BPP}$ algorithm is not pseudodeterministic in general – it is only guaranteed to be pseudodeterministic on inputs that satisfy the promise.

Proof sketch of Theorem 6. Let B be the algorithm of Corollary 1 with $\varepsilon = 1/10$ and the desired failure probability δ. When the oracle algorithm makes query i, define $f_i(X)$ to be the value outputted by the $\mathsf{promise\text{-}BPP}$ algorithm on that query string using randomness X. Give B the "circuit" f_i. (The algorithm B treats the circuits as black boxes, so we don't need to bother implementing f_i as a literal Boolean circuit; the important thing is that $f_i(X)$ can be evaluated in time T'.) When B outputs a value Y_i, give the oracle algorithm the response 0 if $Y_i < 1/2$ and 1 if $Y_i \geq 1/2$. □

6.3 Simulating an APP oracle

Theorem 7. *Suppose $\varphi \in \mathsf{APP}$ and a search problem Π can be solved by a deterministic φ-oracle algorithm that runs in time T and makes k queries $(w_1, \varepsilon), \ldots, (w_k, \varepsilon)$ (where w_i depends on previous oracle responses, but ε is the same for every query.) Let c be the constant under the $O(\cdot)$ in the error ε' in Theorem 1. Suppose that (regardless of the oracle responses) $\varphi(w_i)$ can be approximated to within $\pm\varepsilon/c$ by a randomized algorithm that runs in time T', uses n coins, and has failure probability $1/3$. Then for any δ, Π can be solved by a randomized (non-oracle) algorithm that runs in time*

$$T + O(T' \cdot k \log(k/\delta)) + \mathrm{poly}(n, k, \log(1/\delta)),$$

has randomness complexity

$$n + O(k + (\log k)\log(1/\delta)),$$

and has failure probability δ.

The proof of Theorem 7 is similar to the proofs of Corollary 1 and Theorem 6. The difference is that a sampler as defined previously is no longer quite the right tool for deterministic amplification; to amplify an APP algorithm, we are not trying to estimate the *acceptance probability* of a Boolean function, but rather the point where a $[0, 1]$-valued function is *concentrated*. For this, we use an *averaging sampler*.

An *averaging (ε, δ)-sampler* for Boolean functions on n bits is an algorithm $\mathsf{Samp} : \{0,1\}^m \to (\{0,1\}^n)^t$ such that for any Boolean function $C : \{0,1\}^n \to \{0,1\}$, if we let $\mu(C) \stackrel{\text{def}}{=} 2^{-n} \sum_x C(x)$ be the acceptance probability of C, then

$$\Pr_{X \in \{0,1\}^m}\left[\left|\mu(C) - \frac{1}{t}\sum_{i=1}^{t} C(\mathsf{Samp}(X)_i)\right| > \varepsilon\right] \leq \delta.$$

(Note that an averaging sampler induces a sampler of a very specific form: query the oracle at several points and output the empirical mean.) We now show that an averaging sampler can be used to decrease the failure probability of a concentrated function by taking a *median*. This observation (in a different form) is due to Bellare, Goldreich, and Goldwasser [BGG93].

Lemma 7. *Suppose $f : \{0,1\}^n \to \mathbb{R}$ is (ε, δ_0)-concentrated at $\mu \in \mathbb{R}$ and $\mathsf{Samp} : \{0,1\}^m \to (\{0,1\}^n)^t$ is an averaging (ε', δ)-sampler for Boolean functions on n bits, where $\varepsilon' + \delta_0 < 1/2$. Define $g : \{0,1\}^m \to \mathbb{R}$ by*

$$g(x) = \operatorname*{median}_{i \in [t]} f(\mathsf{Samp}(x)_i).$$

Then g is (ε, δ)-concentrated at μ.

Proof. Let $C : \{0,1\}^n \to \{0,1\}$ be the indicator function for $\{x : |f(x) - \mu| \leq \varepsilon\}$. Then by the concentration of f, $2^{-n} \sum_x C(x) \geq 1 - \delta_0$. Therefore, by the averaging sampler condition, with probability $1 - \delta$ over x, $\frac{1}{t} \sum_i C(\mathsf{Samp}(X)_i) \geq 1 - \delta_0 - \varepsilon' > 1/2$. If this is the case, then more than half of the values $f(\mathsf{Samp}(x)_1), \ldots, f(\mathsf{Samp}(x)_t)$ are within $\pm\varepsilon$ of μ, which implies that their median is within $\pm\varepsilon$ of μ. $\square$

The following lemma gives the parameters achieved by the famous "random walk on expanders" averaging sampler; see e.g. [Vad12, Corollary 4.41].

Lemma 8. *For every $n \in \mathbb{N}$ and every $\varepsilon, \delta > 0$, there is an averaging (ε, δ)-sampler for Boolean functions on n bits with $m \leq n + O(\log(1/\delta)/\varepsilon^2)$ and $t \leq O(\log(1/\delta)/\varepsilon^2)$, computable in time* $\mathrm{poly}(n, 1/\varepsilon, \log(1/\delta))$.

Corollary 2 (Deterministic amplification for APP)**.** *Suppose $\varphi \in \mathsf{APP}$ via an algorithm that on input (w, ε) uses n coins and t time steps to compute $\varphi(w) \pm \varepsilon$ with failure probability $1/3$. Then for any δ, is possible to compute $\varphi(w) \pm \varepsilon$ with failure probability δ using $O(t \log(1/\delta)) + \mathrm{poly}(n, \log(1/\delta)$ time steps and $n + O(\log(1/\delta))$ coins.*

Proof. On input (w, ε):

1. Let $\mathsf{Samp} : \{0,1\}^m \to (\{0,1\}^n)^t$ be the averaging $(1/10, \delta)$-sampler for Boolean functions on n bits of Lemma 8.

2. Define $f : \{0,1\}^n \to [0,1]$ by letting $f(X)$ be the output of the 1/3-error-probability algorithm for computing $\varphi(w) \pm \varepsilon$ on randomness X.

3. Pick $X \in \{0,1\}^m$ uniformly at random and return $\mathrm{median}_{i \in [t]} f(\mathsf{Samp}(X)_i)$.

Correctness follows immediately from Lemma 7, since f is $(\varepsilon, 1/3)$-concentrated at $\varphi(w)$. Efficiency follows immediately from Lemma 8. $\square$

Proof of Theorem 7. By Corollary 2, there is an algorithm Φ for computing $\varphi(w_i) \pm \varepsilon/c$ with failure probability $\delta/(2k)$ that runs in time $O(T' \cdot \log(k/\delta)) + \mathrm{poly}(n, \log k, \log(1/\delta))$ and uses $m \leq n + O(\log(k/\delta))$ coins. Let S be the (ε, δ)-steward of Theorem 1 for k adaptively chosen $(\varepsilon/c, \delta/(2k))$-concentrated functions $f_1, \ldots, f_k : \{0,1\}^m \to \mathbb{R}$. (So $\gamma = \delta/2$.) When the oracle algorithm makes query i about string w_i, let $f_i(X) = \Phi(w_i, \varepsilon/c, X)$ and give f_i to S. When S outputs a value Y_i, give it to the oracle algorithm.

Proof of correctness: Each f_i is $(\varepsilon/c, \delta/(2k))$-concentrated at $\varphi(w_i)$. Furthermore, each f_i depends only on the previous oracle responses, i.e. $Y_1, \ldots, Y_{i-1}$. Therefore, the steward guarantee implies that with probability $1 - \delta$, every Y_i is within $\pm\varepsilon$ of $\varphi(w_i)$. If this occurs, then the oracle algorithm is guaranteed to give a correct output.

Randomness complexity analysis: The number of bits used by the steward is

$$m + O(k + (\log k)\log(1/\delta)) = n + O(k + (\log k)\log(1/\delta)).$$

Runtime analysis: The runtime of the steward is $\mathrm{poly}(m, k, \log(1/\gamma)) = \mathrm{poly}(n, k, \log(1/\delta))$. Therefore, the total runtime is bounded by

$$T + k \cdot (O(T' \cdot \log(k/\delta)) + \mathrm{poly}(n, \log k, \log(1/\delta))) + \mathrm{poly}(n, k, \log(1/\delta)),$$

which is bounded by the expression in the theorem statement. $\square$

6.4 The Goldreich-Levin algorithm

Theorem 8 (Randomness-efficient Goldreich-Levin algorithm)**.** *There is a randomized algorithm that, given oracle access to* $F : \{0,1\}^n \to \{-1,1\}$ *and given input parameters* $\delta, \theta > 0$*, outputs a list* L *of subsets of* $[n]$ *such that with probability* $1-\delta$*,*

1. *every* U *satisfying* $|\widehat{F}(U)| \geq \theta$ *is in* L*, and*
2. *every* $U \in L$ *satisfies* $|\widehat{F}(U)| \geq \theta/2$*.*

The number of queries made by the algorithm is

$$O\left(\frac{n}{\theta^{11}\log(1/\theta)}\log\left(\frac{n}{\delta\theta}\right)\right),$$

the number of random bits used by the algorithm is

$$O(n + (\log n)\log(1/\delta)),$$

and the runtime of the algorithm is $\mathrm{poly}(n, 1/\theta, \log(1/\delta))$*.*

For comparison, using standard techniques (the GW sampler, reusing randomness within each round of adaptivity), the Goldreich-Levin algorithm can be implemented in a straightforward way to use $O(\frac{n}{\theta^6}\log(\frac{n}{\delta\theta}))$ queries and $O(n^2 + n\log(\frac{n}{\delta\theta}))$ random bits. So our algorithm significantly improves the randomness complexity at the expense of substantially increasing the exponent of $1/\theta$ in the query complexity.

Toward proving Theorem 8, for a string $x \in \{0,1\}^{\leq n}$, define

$$\mathcal{U}(x) = \{U \subseteq [n] : \forall j \leq |x|, j \in U \iff x_j = 1\}.$$

(That is, we think of $x \in \{0,1\}^\ell$ as specifying $U \cap [\ell]$ in the natural way.) Define $W_x[F] = \sum_{U \in \mathcal{U}(x)} \widehat{F}(U)^2$. One of the key facts used in the standard Goldreich-Levin algorithm is that $W_x[F]$ can be estimated using few queries to F; here, we use the GW sampler to improve the randomness efficiency of that estimation.

Lemma 9. *There is a randomized algorithm that, given oracle access to* F *and inputs* $x \in \{0,1\}^{\leq n}$*,* $\varepsilon, \delta > 0$*, estimates* $W_x[F]$ *to within* $\pm\varepsilon$ *with failure probability* δ*. The number of queries is* $O(\log(1/\delta)/\varepsilon^2)$*, the number of random bits is* $O(n + \log(1/\delta))$*, and the runtime is* $\mathrm{poly}(n, 1/\varepsilon, \log(1/\delta))$*.*

Proof. Let $\ell = |x|$. As shown in the proof of [O'D14, Proposition 3.40],

$$W_x[F] = \mathop{\mathrm{E}}_{\substack{y,y' \in \{0,1\}^\ell \\ z \in \{0,1\}^{n-\ell}}}[F(y,z) \cdot F(y',z) \cdot \chi_x(y) \cdot \chi_x(y')],$$

where $\chi_x(y) \stackrel{\text{def}}{=} \prod_{j : x_j = 1}(-1)^{y_j}$. Let $C : \{0,1\}^{n+\ell} \to \{0,1\}$ be the function

$$C(y,y',z) = \frac{1}{2} + \frac{1}{2} \cdot F(y,z) \cdot F(y',z) \cdot \chi_x(y) \cdot \chi_x(y'),$$

so that $W_x[F] = 2\,\mathrm{E}_{y,y',z}[C(y,y',z)] - 1$. We can estimate the expectation of C to within $\pm\varepsilon/2$ with failure probability δ using the GW sampler of Lemma 6, which implies an estimate of $W_x[F]$ to within $\pm\varepsilon$. The number of queries made by the GW sampler is $O(\log(1/\delta)/\varepsilon^2)$, and each query to C can be evaluated by making 2 queries to F. The randomness complexity of the GW sampler is $n + \ell + O(\log(1/\delta))$, which is $O(n + \log(1/\delta))$. $\square$

The standard Goldreich-Levin algorithm proceeds by finding, for $\ell = 1$ to n, the set of all x with $|x| = \ell$ such that $W_x[F] \gtrsim \theta^2$. In each round, the algorithm estimates $W_x[F]$ for all strings x formed by appending a single bit to a string x' that was previously found to satisfy $W_{x'}[F] \gtrsim \theta^2$. This adaptive structure is exactly suited for saving random bits using a steward. To further drive down the randomness complexity, we reduce the number of rounds of adaptivity by appending $\log(1/\theta)$ bits at a time instead of 1 bit.

Proof of Theorem 8. Algorithm:

1. Let $u = \lfloor \log(1/\theta) \rfloor$, let $k = \lceil n/u \rceil$, and let $d = \lfloor 2^u \cdot 4/\theta^2 \rfloor$.
2. Let S be a $(\theta^2/4, \delta)$-steward for k adaptively chosen $(\varepsilon, \delta/(2n))$-concentrated functions $f_1, \dots, f_k : \{0,1\}^m \to \mathbb{R}^d$, where $\varepsilon \geq \Omega(\theta^2/d)$ and m will become clear later.
3. Set $L_0 := \{\text{empty string}\}$.
4. For $i = 1$ to k:
   (a) If $|L_{i-1}| > d/2^u$, abort and output "fail".
   (b) Observe that every string in L_{i-1} has length $\ell = u(i-1) < n$. Let $x_1, \dots, x_t$ be the set of all strings obtained from strings in L_{i-1} by appending $\min\{u, n-\ell\}$ bits, so $t \leq 2^u |L_{i-1}| \leq d$.
   (c) Define $f_i : \{0,1\}^m \to \mathbb{R}^t$ by letting $f_i(X)_j$ be the estimate of $W_{x_j}[F]$ to within $\pm\varepsilon$ provided by the algorithm of Lemma 9 with failure probability $\delta/(2dn)$ using randomness X. Observe that by the union bound, f_i is $(\varepsilon, \delta/(2n))$-concentrated at $(W_{x_1}[F], \dots, W_{x_t}[F])$.
   (d) By giving f_i to S, obtain estimates $\mu_1, \dots, \mu_t$ for $W_{x_1}[F], \dots, W_{x_t}[F]$.
   (e) Set $L_i := \{x_j : \mu_j \geq \theta^2/2\}$.
5. Output $L \stackrel{\text{def}}{=} \bigcup_{x \in L_k} \mathcal{U}(x)$.

As hopefully became clear, m is the number of random bits used by the algorithm of Lemma 9. With probability $1 - \delta$, all of the responses of S are accurate, i.e. every μ_j value is within $\pm\theta^2/4$ of the corresponding $W_{x_j}[F]$ value. Assume from now on that this has happened.

By the definition of L_i, every x in every L_i satisfies $W_x[F] \geq \theta^2/4$. By Parseval's theorem (see e.g. [O'D14, Section 1.4]), this implies that $|L_i| \leq 4/\theta^2 \leq d/2^u$ for every i. Therefore, the algorithm does not abort. Let ℓ_i be the length of all the strings in L_i, so $\ell_i = ui$ for $i < k$ and $\ell_k = n$. Suppose $\widehat{F}(U)^2 \geq \theta^2$. By induction on i, the unique string $x \in \{0,1\}^{\ell_i}$ with $U \in \mathcal{U}(x)$ is placed in L_i, because the estimate of $W_x[F]$ is at least $3\theta^2/4 > \theta^2/2$. This shows that $U \in L$. Conversely, if U ends up in L, then the estimate of $\widehat{F}(U)^2$ in iteration $i = n$ was at least $\theta^2/2$, so $\widehat{F}(U)^2 \geq \theta^2/4$. This completes the proof of correctness of the algorithm.

Now, observe that the total number of queries to F is at most kd times the $O(\log(nd/\delta)/\varepsilon^2)$ queries that the algorithm of Lemma 9 makes, i.e. the total number of queries to F is

$$O\left(\frac{kd^3 \log(nd/\delta)}{\theta^2}\right) = O\left(\frac{n}{\theta^{11}\log(1/\theta)} \log\left(\frac{n}{\delta\theta}\right)\right).$$

The randomness complexity of the algorithm is just the randomness complexity of S. We will use the steward of Theorem 1 with $\gamma = \delta/2$, so the randomness complexity is $m + O(k\log(d+1) + (\log k)\log(1/\delta))$. Since $m \leq O(n + \log(n/(\delta\theta)))$, the total randomness complexity is

$$O\left(n + \frac{n}{\log(1/\theta)}\log(1/\theta) + (\log n)\log(1/\delta) + \log(1/\theta)\right) = O(n + (\log n)\log(1/\delta) + \log(1/\theta)).$$

To get rid of the $\log(1/\theta)$ term as claimed in the theorem statement, just notice that we can assume without loss of generality that $\theta \geq 2^{-n+1}$, because any nonzero Fourier coefficient of a $\{-1,1\}$-valued function has absolute value at least 2^{-n+1}. The total runtime of the algorithm is clearly $\mathrm{poly}(n, 1/\theta, \log(1/\delta))$. $\square$

7 Randomness complexity lower bound

To understand the following lemma, imagine the perspective of O after $i-1$ rounds of $\mathsf{O} \leftrightarrow \mathsf{S}(Z)$, where Z was chosen uniformly at random from $\{0,1\}^m$. Let R be the set of z such that the hypothesis that S is using randomness string z is compatible with everything that O has seen so far. Then at this point, O's posterior distribution for Z is uniform over R. The following lemma says that with respect to this posterior distribution, O can choose f_i such that either O will learn $\Omega(1)$ bits of information about Z based on Y_i, or else S will have a failure probability of $\Omega(1)$ in round i.

Lemma 10. *Suppose S is an m-coin (ε', δ')-steward for k adaptively chosen (ε, δ)-concentrated functions $f_1, \ldots, f_k : \{0,1\}^n \to \mathbb{R}$ and O is a deterministic owner. Fix $i \in [k]$. For a function $g : \{0,1\}^n \to \mathbb{R}$, let $\mathsf{O}[g]$ be the owner that simulates O for rounds $1, 2, \ldots, i-1$, but chooses g in round i regardless of what O would have chosen. Let $R \subseteq \{0,1\}^m$ be a nonempty set such that the transcript of the first $i-1$ rounds of $\mathsf{O} \leftrightarrow \mathsf{S}(Z)$ is the same for every $Z \in R$. Assume $\delta \geq 2^{-n}$. Then there exists g that is (ε, δ)-concentrated at μ such that either*

1. $\max_{y \in \mathbb{R}} \Pr_{Z \in R}[Y_i = y$ *in* $\mathsf{O}[g] \leftrightarrow \mathsf{S}(Z)] \leq 0.8$, *or*
2. $\Pr_{Z \in R}[|Y_i - \mu_i| > \varepsilon'$ *in* $\mathsf{O}[g] \leftrightarrow \mathsf{S}(Z)] \geq 0.2$.

Proof. For each $j \in \mathbb{Z}$, let $g_j : \{0,1\}^n \to \mathbb{R}$ be constant at εj. If some g_j satisfies Condition 1, we're done. So assume that for each g_j, there is some $y_j \in \mathbb{R}$ such that $\Pr_{Z \in R}[Y_j = y_j$ in $\mathsf{O}[g] \leftrightarrow \mathsf{S}(Z)] > 0.8$. If y_j does not depend on j, then since $0.2 < 0.8$, there is some g_j that satisfies Condition 2, so we are again done. Therefore, assume there is some j such that $y_j \neq y_{j+1}$.

Define $q : R \to \{0,1\}^n$ by letting $q(Z) =$ the value X_i chosen by S in $\mathsf{O} \leftrightarrow S(Z)$. First, assume there is some x^* such that $\Pr_{Z \in R}[q(Z) = x^*] \geq 0.4$. For $s \in \{\pm 1\}$, define $g^s : \{0,1\}^n \to \mathbb{R}$ by

$$g^s(x) = \begin{cases} 0 & \text{if } x = x^* \\ s \cdot 2\varepsilon' & \text{otherwise.} \end{cases}$$

Then $g^s(x)$ is $(0, 2^{-n})$-concentrated at $s \cdot 2\varepsilon'$. Let O' be the randomized owner that tosses a coin to decide whether to simulate $\mathsf{O}[g^{+1}]$ or $\mathsf{O}[g^{-1}]$. Then when $Z \in R$ is chosen uniformly at random, in $\mathsf{O}' \leftrightarrow \mathsf{S}(Z)$, there is a 0.4 chance that $f_i(X_i) = 0$, in which case $\mathsf{S}(Z)$ has only a 50% chance of correctly guessing s. This shows that $\Pr_{Z \in R}[|Y_i - \mu_i| > \varepsilon'$ in $\mathsf{O}' \leftrightarrow \mathsf{S}(Z)] \geq 0.2$, and hence either g^{+1} or g^{-1} satisfies Condition 2, so we are again done. Therefore, assume that for every x^*, $\Pr_{Z \in R}[q(Z) = x^*] < 0.4$.

For $t \in \{j, j+1\}$, let

$$A_t = \{Z \in R : Y_i = y_t \text{ in } \mathsf{O}[g_t] \leftrightarrow \mathsf{S}(Z)\},$$

so that $|A_t| > 0.8|R|$. We define g by the following greedy algorithm. Two players, which we identify with A_j and A_{j+1}, alternate taking turns. When it is A_t's turn, she finds the string $x \in \{0,1\}^n$ such that $g(x)$ is not yet defined that maximizes $q^{-1}(x) \cap A_t$, and defines $g(x) = \varepsilon t$. This continues for 2^n turns until g is defined everywhere.

Clearly, g thus defined is $(\varepsilon, 0)$-concentrated. We will show that g satisfies Condition 1. Proof: Say $z \in \{0,1\}^m$ is *good for* A_t if $z \in A_t$ and $g(q(z)) = \varepsilon t$. In these terms, on A_t's turn, she defines g on one more point in order to maximize the number of z that become good for A_t. Say that $z \in \{0,1\}^m$ is *bad for* A_t if $z \in A_t$ and $g(q(z)) \neq \varepsilon t$. When it is not A_t's turn, some z may become bad for A_t, but the crucial point is that the number of z that become bad for A_t is *at most* the number of z that became *good* for A_t in the *previous* turn (simply because of the greedy choice that A_t made in the previous turn.) This would show that half of A_t is good for A_t, except for one annoyance: the first turn, where some z become bad for the second player with no corresponding previous turn, when z became good. But we already showed that for every x, $|q^{-1}(x)| \leq 0.4|R|$, so the first turn does not matter too much: at the end of the construction, for each t, the number of z that are good for A_t is at least

$$\frac{1}{2}(|A_t| - 0.4|R|) \geq \frac{1}{2}(0.8|R| - 0.4|R|) = 0.2|R|.$$

By construction, if z is good for A_t, then $Y_i = y_t$ in $\mathsf{O}[g] \leftrightarrow \mathsf{S}(z)$. Therefore, for each t, $\Pr_{Z \in R}[Y_i = y_t \text{ in } \mathsf{O}[g] \leftrightarrow \mathsf{S}(Z)] \geq 0.2$, which implies Condition 1 since $y_j \neq y_{j+1}$. □

Having proved Lemma 10, we are ready to prove our randomness complexity lower bound. The idea is that O will spend the first $k-1$ rounds learning as much information as possible about S's randomness string using Lemma 10 (unless she gets lucky and is able to cause S to have an $\Omega(1)$ failure probability in one of these rounds, in which case she will take the opportunity.) Then, in round k, O uses everything she's learned about S's randomness string to choose f_k so as to maximize S's failure probability in that round.

Theorem 9. *Suppose* S *is an* m*-coin* (ε', δ')*-steward for* k *adaptively chosen* (ε, δ)*-concentrated functions* $f_1, \ldots, f_k : \{0,1\}^n \to \mathbb{R}^d$*. Assume* $\delta' < 0.2$ *and* $\delta \geq 2^{-n}$*. Then* $m \geq n + \Omega(k) - \log_2(\delta'/\delta)$*.*

Proof. Without loss of generality, assume $d = 1$. Let O be the following owner:

1. For $i = 1$ to k:
   (a) Let $y_1, y_2, \ldots, y_{i-1}$ be the responses received so far.
   (b) Let $R \subseteq \{0,1\}^m$ be the set of z such that in $\mathsf{O} \leftrightarrow \mathsf{S}(z)$, $Y_j = y_j$ for every $j < i$. (By induction, we have already defined the behavior of O in rounds $1, 2, \ldots, i-1$, so R is well-defined. In other words, R is the set of z that are compatible with what O has seen so far.)
   (c) If $i < k$:
      i. Choose $f_i = g$, where g is the function guaranteed by Lemma 10. (Again, O is already defined and deterministic for rounds $1, 2, \ldots, i-1$, so we can sensibly apply the lemma.)
   (d) Otherwise, if $i = k$:
      i. Pick $S \subseteq R$, $|S| = \min\{\lfloor \delta 2^n \rfloor, |R|\}$ uniformly at random, pick $s \in \{\pm 1\}$ independently and uniformly at random, and choose

$$f_k(x) = \begin{cases} 0 & \text{if } x \in q(S) \\ s \cdot 2\varepsilon' & \text{otherwise.} \end{cases}$$

      (Note that f_k is $(0, \delta)$-concentrated at $s \cdot 2\varepsilon'$, because $|q(S)| \leq |S| \leq \delta 2^n$.)

To analyze O, in $\mathsf{O} \leftrightarrow \mathsf{S}$, say that O *tries to win* in round i if either $i = k$ or else $i < k$ and the function f_i chosen satisfies Condition 2 in Lemma 10. For a string $z \in \{0,1\}^m$, let $w(z) \in [k]$ be the index of the first round in which O tries to win in $\mathsf{O} \leftrightarrow \mathsf{S}(z)$, and let $\tau(z)$ be the transcript of rounds $1, 2, \dots, w(z) - 1$ in $\mathsf{O} \leftrightarrow \mathsf{S}(z)$. Note that since O is deterministic in rounds $1, 2, \dots, k-1$, $w(z)$ and $\tau(z)$ are not random variables. Define an equivalence relation on $\{0,1\}^m$ by saying that $z \sim z'$ if and only if $\tau(z) = \tau(z')$. Say O uses v random bits. We first show that for each equivalence class $\overline{z}$,

$$\Pr_{Z \in \overline{z}, V \in \{0,1\}^v}[\|Y_{w(\overline{z})} - \mu_{w(\overline{z})}\|_\infty > \varepsilon' \text{ in } \mathsf{O}(V) \leftrightarrow \mathsf{S}(Z)] \geq \min\{0.2, \delta \cdot (1/0.8)^{k-1} \cdot 2^{n-m-2}\}. \tag{1}$$

Proof: Observe that in round $w(\overline{z})$, O's set R is precisely $\overline{z}$. If $w(\overline{z}) < k$, then Condition 2 of Lemma 10 immediately implies that the failure probability in Equation 1 is at least 0.2. Suppose instead that $w(\overline{z}) = k$. Then in every previous round, O did not try to win, i.e. O chose a function satisfying Condition 1 of Lemma 10. This implies that in every previous round, O's set R decreased in size by a factor of 0.8. So at the beginning of round k, $|R| \leq 0.8^{k-1} \cdot 2^m$. The probability (over $Z \in \overline{z}$) that S chooses X_k such that $f_k(X_k) = 0$ is

$$\begin{aligned}
\frac{|S|}{|R|} &= \frac{\min\{\lceil \delta 2^n \rceil, |R|\}}{|R|} \\
&\geq \min\left\{1, \delta 2^{n-m-1}(1/0.8)^{k-1}\right\}.
\end{aligned}$$

Conditioned on $f_k(X_k) = 0$, the probability of the event in Equation 1 is at least 0.5, because conditioned on $f_k(X_k) = 0$, s is independent of everything S has seen. Therefore, the probability of the event in Equation 1 is at least $\min\{0.5, \delta 2^{n-m-2}(1/0.8)^{k-1}\}$, completing the proof of Equation 1.

Now, to prove the theorem, observe that

$$\begin{aligned}
\delta' &\geq \Pr_{Z \in \{0,1\}^m, V \in \{0,1\}^v}[\max_i \|\mu_i - Y_i\|_\infty > \varepsilon' \text{ in } \mathsf{O}(V) \leftrightarrow \mathsf{S}(Z)] \\
&\geq \Pr_{Z \in \{0,1\}^m, V \in \{0,1\}^v}[\|\mu_{w(Z)} - Y_{w(Z)}\|_\infty > \varepsilon' \text{ in } \mathsf{O}(V) \leftrightarrow \mathsf{S}(Z)] \\
&= \sum_{\overline{z}} \Pr_{Z \in \{0,1\}^m}[Z \in \overline{z}] \cdot \Pr_{Z' \in \overline{z}, V \in \{0,1\}^v}[\|\mu_{w(\overline{z})} - Y_{w(\overline{z})}\|_\infty > \varepsilon' \text{ in } \mathsf{O}(V) \leftrightarrow \mathsf{S}(Z')] \\
&\geq \sum_{\overline{z}} \Pr_{Z \in \{0,1\}^m}[Z \in \overline{z}] \cdot \min\{0.2, \delta \cdot (1/0.8)^{k-1} 2^{n-m-2}\} \\
&= \min\{0.2, \delta \cdot (1/0.8)^{k-1} 2^{n-m-2}\}.
\end{aligned}$$

We assumed that $\delta' < 0.2$, so we can conclude that $\delta' \geq \delta \cdot (1/0.8)^{k-1} 2^{n-m-2}$. Rearranging proves that

$$\begin{aligned}
m &\geq (n-2) + (k-1)\log_2(1/0.8) - \log_2(\delta'/\delta) \\
&\geq n + \Omega(k) - \log_2(\delta'/\delta),
\end{aligned}$$

completing the proof. □

8 Acknowledgments

We thank David Zuckerman for observations about block decision trees. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1610403.

References

[Arm98] R. Armoni. On the derandomization of space-bounded computations. In *Randomization and Approximation Techniques in Computer Science*, pages 47–59. Springer, 1998.

[Bel92] M. Bellare. A technique for upper bounding the spectral norm with applications to learning. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 62–70. ACM, 1992.

[BF99] H. Buhrman and L. Fortnow. One-sided versus two-sided error in probabilistic computation. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 100–109. Springer, 1999.

[BGG93] M. Bellare, O. Goldreich, and S. Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 3(4):319–354, 1993.

[BH15] A. Blum and M. Hardt. The ladder: A reliable leaderboard for machine learning competitions. *arXiv preprint arXiv:1502.04585*, 2015.

[BJT99] N. H. Bshouty, J. C. Jackson, and C. Tamon. More efficient PAC-learning of DNF with membership queries under the uniform distribution. In *Proceedings of the 12th Annual Conference on Computational Learning Theory*, pages 286–295. ACM, 1999.

[BNS+15] R. Bassily, K. Nissim, A. Smith, T. Steinke, U. Stemmer, and J. Ullman. Algorithmic stability for adaptive data analysis. *arXiv preprint arXiv:1511.02513*, 2015.

[DFH+15a] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems*, pages 2350–2358, 2015.

[DFH+15b] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015.

[DFH+15c] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. L. Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 117–126. ACM, 2015.

[DORS08] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008.

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

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, STOC '89, pages 25–32. ACM, 1989.

[GW97] O. Goldreich and A. Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Structures & Algorithms*, 11(4):315–343, December 1997.

[HU14] M. Hardt and J. Ullman. Preventing false discovery in interactive data analysis is hard. In *Proceedings of the 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, pages 454–463. IEEE, 2014.

[HU16] W. M. Hoza and C. Umans. Targeted pseudorandom generators, simulation advice generators, and derandomizing logspace. *arXiv preprint arXiv:1610.01199*, 2016.

[INW94] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Annual Symposium on Theory of Computing*, STOC '94, pages 356–364. ACM, 1994.

[KM93] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[KRC00] V. Kabanets, C. Rackoff, and S. Cook. Efficiently approximable real-valued functions. In *Electronic Colloquium on Computational Complexity (ECCC) Report TR00*, volume 34, 2000.

[Lev93] L. A. Levin. Randomness and nondeterminism. *Journal of Symbolic Logic*, 58(3):1102–1103, 1993.

[Mos01] P. Moser. Relative to P $\mathsf{promise\text{-}BPP}$ equals APP. In *Electronic Colloquium on Computational Complexity (ECCC) Report TR01*, volume 68, 2001.

[Nis92] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[NN93] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM journal on computing*, 22(4):838–856, 1993.

[NZ96] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[O'D14] R. O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

[RR99] R. Raz and O. Reingold. On recycling the randomness of states in space bounded computation. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, STOC '99, pages 159–168. ACM, 1999.

[SU14] T. Steinke and J. Ullman. Interactive fingerprinting codes and the hardness of preventing false discovery. *arXiv preprint arXiv:1410.1228*, 2014.

[SZ99] M. Saks and S. Zhou. $\mathsf{BP_HSPACE}(S) \subseteq \mathsf{DSPACE}(S^{3/2})$. *Journal of Computer and System Sciences*, 58(2):376–403, 1999.

[Vad12] S. P. Vadhan. *Pseudorandomness*, volume 56. Now, 2012.

A Generalized shifting and rounding algorithm

In this section, we show how to generalize the steward S_0 to achieve a tradeoff between its error and the branching factor of the certification tree T_{O}. Fix any factorization of d into $d = d_0 d_1$. Partition $[d]$ as $[d] = J_1 \cup J_2 \cup \cdots \cup J_{d_1}$, where $|J_t| = d_0$ for each t. Instead of partitioning $\mathbb{R}$ into intervals of length $2(d+1)\varepsilon$, partition $\mathbb{R}$ into intervals of length $2(d_0+1)\varepsilon$. The following algorithm for computing Y_i from W_i generalizes that of Section 2.1:

1. For each $t \in [d_1]$:

   (a) Find $\Delta_{it} \in [d_0 + 1]$ such that for every $j \in J_t$, there is a single interval that entirely contains $[W_{ij} + (2\Delta_{it} - 1)\varepsilon, W_{ij} + (2\Delta_{it} + 1)\varepsilon]$. (Such a Δ_{it} exists by Lemma 1.)

   (b) For every $j \in J_t$, set $Y_{ij} = \mathsf{Round}(W_{ij} + 2\Delta_{it}\varepsilon)$.

The following lemma is the appropriate generalization of Lemma 2:

Lemma 11. *Assume $\delta < 1/2$. Let $\Sigma = [d_0 + 1]^{d_1} \cup \{\perp\}$. For any deterministic owner O, there exists a (k, n, Σ) block decision tree T_{O} with the following properties.*

1. *For any internal node v, $\Pr_{X \in \{0,1\}^n}[v(X) = \perp] \leq \delta$.*

2. *Fix $X_1, \ldots, X_k \in \{0,1\}^n$, and suppose that the path from the root to $T_{\mathsf{O}}(X_1, \ldots, X_k)$ does not include any $\perp$ nodes. Then in $\mathsf{O} \leftrightarrow \mathsf{S}_0(X_1, \ldots, X_k)$, $\max_i \|Y_i - \mu_i\|_\infty \leq O(d_0\varepsilon)$.*

The proof of Lemma 11 is essentially the same as the proof of Lemma 2; we record the details below.

Proof of Lemma 11. Consider a thought experiment in which S_0 interacts with O through two mediators M_1 and M_2 as follows:

1. For $i = 1$ to k:

   (a) O chooses $f_i : \{0,1\}^n \to \mathbb{R}^d$ that is (ε, δ)-concentrated at some point $\mu_i \in \mathbb{R}^d$ and gives it to M_1, who gives it to M_2.

   (b) M_1 and M_2 both compute the lexicographically smallest $\widehat{\mu}_i \in \mathbb{R}^d$ such that f_i is (ε, δ)-concentrated at $\widehat{\mu}_i$.

   (c) S_0 chooses a string $X_i \in \{0,1\}^n$ and gives it to M_2, who gives $f_i(X_i) \in \mathbb{R}^d$ to S_0.

   (d) S_0 chooses $Y_i \in \mathbb{R}^d$ and gives it to M_2.

   (e) Say a vector $(\Delta_1, \ldots, \Delta_{d_1})$ is *compatible* with Y_i if $Y_{ij} = \mathsf{Round}(\widehat{\mu}_{ij} + 2\Delta_t\varepsilon)$ for every $t \in [d_1]$ and every $j \in J_t$. M_2 computes

   $$\widehat{\Delta}_i = \begin{cases} \text{the first } (\Delta_1, \ldots, \Delta_{d_1}) \in [d_0+1]^{d_1} \text{ compatible with } Y_i & \text{if any exist} \\ \perp & \text{otherwise} \end{cases}$$

   and gives it to M_1.

   (f) M_1 computes $\widehat{Y}_i = (\widehat{Y}_{i1}, \ldots, \widehat{Y}_{id})$, where for each $t \in [d_1]$ and each $j \in J_t$,

   $$\widehat{Y}_{ij} = \begin{cases} \mathsf{Round}(\widehat{\mu}_{ij} + 2\widehat{\Delta}_{it}\varepsilon) & \text{if } \widehat{\Delta}_i \neq \perp \\ 0 & \text{otherwise} \end{cases}$$

   and gives $\widehat{Y}_i$ to O.

The definition of T_{O} is exactly the same as in the proof of Lemma 2, except that the notation $\mathsf{O} \xleftrightarrow{2} \mathsf{S}_0$ now refers to the above modified two-mediator thought experiment. To prove Condition 1 in the lemma statement, we must show that in each round of $\mathsf{O} \xleftrightarrow{2} \mathsf{S}_0$, $\Pr[\widehat{\Delta}_i = \perp] \leq \delta$. Indeed, by concentration, with probability $1 - \delta$, for every j, $|W_{ij} - \widehat{\mu}_{ij}| \leq \varepsilon$. In this case, by the construction of S_0, $W_{ij} + 2\Delta_{it}\varepsilon$ and $\widehat{\mu}_{ij} + 2\Delta_{it}\varepsilon$ are in the same interval for every $t \in [d_1]$ and every $j \in J_t$. Therefore, in this case, there is at least one vector $(\Delta_1, \ldots, \Delta_{d_1})$ compatible with Y_i, namely the vector of Δ_{it}

values used by S_0. To prove Condition 2 in the lemma statement, suppose the path from the root node to $T_{\mathsf{O}}(X_1, \ldots, X_k)$ does not include any $\perp$ nodes. Then in $\mathsf{O} \overset{2}{\longleftrightarrow} \mathsf{S}_0(X_1, \ldots, X_k)$, for every i, $\widehat{\Delta}_i \neq \perp$. This implies that every Y_{ij} is of the form $\mathsf{Round}(\widehat{\mu}_{ij} + 2\widehat{\Delta}_{it}\varepsilon)$ for some $\widehat{\Delta}_{it} \in [d_0 + 1]$. Therefore, $|Y_{ij} - \widehat{\mu}_{ij}| \leq 3(d_0 + 1)\varepsilon$, since $2\widehat{\Delta}_{it}\varepsilon \leq 2(d_0 + 1)\varepsilon$ and rounding introduces at most $(d_0 + 1)\varepsilon$ additional error. Since $\delta < 1/2$, $\|\widehat{\mu}_i - \mu_i\|_\infty \leq 2\varepsilon$, so by the triangle inequality, for every i, $\|Y_i - \mu_i\|_\infty \leq 3(d_0 + 1)\varepsilon + 2\varepsilon = (3d_0 + 5)\varepsilon$. Just as in the proof of Lemma 2, the same bound holds in $\mathsf{O} \leftrightarrow \mathsf{S}_0$. □

B The Saks-Zhou steward

In this section, for completeness, we give the description and analysis of the Saks-Zhou steward. This algorithm and analysis are the same in spirit as what appears in [SZ99], but the presentation has been changed to match our framework. None of our results use this steward, but it is interesting to see how the stewards compare.

Proposition 1. *For any $n, k, d \in \mathbb{N}$ and any $\varepsilon, \delta, \gamma > 0$, there exists an $(O(kd\varepsilon/\gamma), k\delta + \gamma)$-steward for k adaptively chosen (ε, δ)-concentrated functions $f_1, \ldots, f_k : \{0,1\}^n \to \mathbb{R}^d$ with randomness complexity*

$$n + O(k \log k + k \log d + k \log(1/\gamma)).$$

The total running time of the steward is $\mathrm{poly}(n, k, d, \log(1/\varepsilon), \log(1/\gamma))$.

Proof. Let u be the smallest power of two such that $u \geq 2kd/\gamma$. (The only reason we choose a power of two is so that we can cleanly draw a uniform random element of $[u]$ using $\log u$ random bits.) Partition $\mathbb{R}$ into half-open intervals of length $\ell = u\varepsilon$. For $w \in \mathbb{R}$, let $\mathsf{Round}(w)$ be the midpoint of the interval containing w. Algorithm S:

1. Pick $X \in \{0,1\}^n$ uniformly at random *once*.
2. For $i = 1$ to k:
   (a) Obtain $W_i = f_i(X) \in \mathbb{R}^d$.
   (b) Pick $\Delta_i \in [u]$ uniformly at random.
   (c) Return $Y_i = (Y_{i1}, \ldots, Y_{id})$, where $Y_{ij} = \mathsf{Round}(W_i + \Delta\varepsilon)$.

Proof of correctness: Fix any deterministic owner O. Consider a thought experiment in which O interacts with a party P who, from the perspective of O, seems to act like the mediator M. (What O does not realize is that there is no steward in this thought experiment.) This party P behaves as follows:

1. For $i = 1$ to k:
   (a) Pick $\Delta_i \in [u]$ uniformly at random.
   (b) Obtain f_i, and find some point $\widehat{\mu}_i \in \mathbb{R}^d$ where f_i is (ε, δ)-concentrated.
   (c) Return $Y_i = (Y_{i1}, \ldots, Y_{id})$, where $Y_{ij} = \mathsf{Round}(\widehat{\mu}_{ij} + \Delta_i\varepsilon)$.

Write $\mathsf{O} \leftrightarrow \mathsf{P}$ to denote the above interaction. For a vector $\vec{\Delta} = (\Delta_1, \ldots, \Delta_k) \in [u]^k$, let $f_1^{[\vec{\Delta}]}, \ldots, f_k^{[\vec{\Delta}]}$ be the functions that O chooses in $\mathsf{O} \leftrightarrow \mathsf{P}(\vec{\Delta})$, and let $\widehat{\mu}_i^{[\vec{\Delta}]}$ be the point at which $f_i^{[\vec{\Delta}]}$ is concentrated that P chooses in $\mathsf{O} \leftrightarrow \mathsf{P}(\vec{\Delta})$. Observe that

$$\Pr_{\substack{\vec{\Delta}\in[u]^k \\ X\in\{0,1\}^n}}[\text{for some } i, \|f_i^{[\vec{\Delta}]}(X) - \widehat{\mu}_i^{[\vec{\Delta}]}\|_\infty > \varepsilon] \le k\delta. \tag{2}$$

(Imagine picking $\vec{\Delta}$ first, and then apply the union bound over the k different values of i.) Next, observe that

$$\Pr_{\substack{\vec{\Delta}\in[u]^k \\ X\in\{0,1\}^n}}[\text{for some } i, j, [\widehat{\mu}_{ij}^{[\vec{\Delta}]}+(\Delta_i-1)\varepsilon, \widehat{\mu}_{ij}^{[\vec{\Delta}]}+(\Delta_i+1)\varepsilon] \text{ is not entirely contained in one interval}] \le \gamma. \tag{3}$$

(Indeed, for each i, j, the probability is just $2/u$, so by the union bound, the probability is at most $2kd/u \le \gamma$.) Now, by the union bound, assume from now on that $\vec{\Delta}, X$ are such that neither the event of Equation 2 nor the event of Equation 3 takes place. Assume without loss of generality that $\delta < 1/2$. We will show that in $\mathsf{O} \leftrightarrow \mathsf{S}(X, \vec{\Delta})$, for every i, $\|Y_i - \mu_i\|_\infty \le 1.5\ell + 3\varepsilon$.

We first show by induction on i that in $\mathsf{O} \leftrightarrow \mathsf{S}(X, \vec{\Delta})$, every f_i is precisely $f_i^{[\vec{\Delta}]}$. In the base case $i = 1$, this is trivial. For the inductive step, since the bad event of Equation 2 did not occur, we know that $f_i(X)$ is $\widehat{\mu}_i^{[\vec{\Delta}]} \pm \varepsilon$. Therefore, since the bad event of Equation 3 did not occur, for every j, $\mathsf{Round}(f_{ij}(X) + \Delta_i\varepsilon) = \mathsf{Round}(\widehat{\mu}_{ij}^{[\vec{\Delta}]} + \Delta_i\varepsilon)$. Therefore, the value Y_i in $\mathsf{O} \leftrightarrow \mathsf{S}(X, \vec{\Delta})$ is the same as the value Y_i in $\mathsf{O} \leftrightarrow \mathsf{P}(\vec{\Delta})$, and hence O chooses the same f_{i+1} in both cases. This completes the induction.

Again using the fact that the bad event of Equation 2 did not occur, this immediately implies that in $\mathsf{O} \leftrightarrow \mathsf{S}(X, \vec{\Delta})$, every $f_i(X)$ is within ℓ_∞ distance ε of a point where f_i is (ε, δ)-concentrated. Since $\delta < 1/2$, this implies that every $f_i(X)$ is within ℓ_∞ distance 3ε of μ_i. Shifting by $\Delta_i\varepsilon$ and rounding introduce at most 1.5ℓ additional error, showing that $\|Y_i - \mu_i\| \le 1.5\ell + 3\varepsilon$ as claimed. To complete the proof of correctness, note that $1.5\ell + 3\varepsilon \le O(kd\varepsilon/\gamma)$.

The randomness complexity of this steward is n bits (for X) plus the randomness needed for $\vec{\Delta}$, for a total randomness complexity of

$$n + k\log u \le n + O(k\log k + k\log d + k\log(1/\gamma)).$$

The steward clearly runs in $\mathrm{poly}(n, k, d, \log(1/\varepsilon), \log(1/\gamma))$ time. □

C Nonconstructive PRG for block decision trees

For completeness, we record the details of the standard nonconstructive argument that there exists a PRG for block decision trees with a small seed length.

Lemma 12. *Suppose $\mathcal{C}$ is a class of Boolean functions $f : \{0,1\}^n \to \{0,1\}$ such that a function in $\mathcal{C}$ can be specified using t bits, i.e. $|\mathcal{C}| \le 2^t$. Then for any γ, there exists a γ-PRG $\mathsf{Gen} : \{0,1\}^s \to \{0,1\}^n$ for $\mathcal{C}$ with seed length*

$$s \le \log t + 2\log(1/\gamma) + O(1).$$

Proof. Consider picking Gen uniformly at random from the set of all functions $\{0,1\}^s \to \{0,1\}^n$. Fix $C \in \mathcal{C}$, and let $\mu(C) = \Pr_x[C(x) = 1]$. Then for each fixed seed $x \in \{0,1\}^s$, the probability (over Gen) that $C(\mathsf{Gen}(x)) = 1$ is precisely $\mu(C)$. Therefore, the expected fraction of x such that $C(\mathsf{Gen}(x)) = 1$ is precisely $\mu(C)$, and by Hoeffding's inequality,

$$\Pr_{\mathsf{Gen}}\left[\left|\frac{\#\{x : C(\mathsf{Gen}(x)) = 1\}}{2^s} - \mu(C)\right| > \gamma\right] \le 2^{-\Omega(\gamma^2 2^s)}.$$

Therefore, by the union bound, the probability that the above bad event holds for *any* C is at most $2^{t-\Omega(\gamma^2 2^s)}$. If we choose s large enough, this probability will be less than 1, showing that there exists a Gen that works for all C. How large do we need to choose s? There is some constant c such that it is sufficient to have $c\gamma^2 2^s > t$. Taking logarithms completes the proof. $\square$

Proposition 2. *For any $k, n \in \mathbb{N}$, any finite alphabet Σ, and any $\gamma > 0$, there exists a γ-PRG $\mathsf{Gen} : \{0,1\}^s \to \{0,1\}^{nk}$ for (k, n, Σ) block decision trees with seed length*

$$s \leq n + k \log |\Sigma| + 2 \log(1/\gamma) + O(1).$$

Proof. Let $\mathcal{C}$ be the class of all Boolean functions $f : \{0,1\}^{nk} \to \{0,1\}$ of the form $f(x) = g(T(x))$, where T is a (k, n, Σ) block decision tree. To specify a function $f \in \mathcal{C}$, we need to specify (1) a bit for each leaf of T and (2) a function $v : \{0,1\}^n \to \Sigma$ for each internal node of T. In total, this number of bits t is

$$\begin{aligned} t &= |\Sigma|^k + 2^n \lceil \log |\Sigma| \rceil \cdot \sum_{i=0}^{k-1} |\Sigma|^i \\ &\leq |\Sigma|^k + 2^{n+1} \log |\Sigma| \cdot \frac{|\Sigma|^k - 1}{|\Sigma| - 1} \\ &\leq |\Sigma|^k + 2^{n+1} |\Sigma|^k \\ &\leq 2^{n+2} |\Sigma|^k. \end{aligned}$$

By Lemma 12, this implies that there is a γ-PRG $\mathsf{Gen} : \{0,1\}^s \to \{0,1\}^{nk}$ for $\mathcal{C}$ with seed length $n + k \log |\Sigma| + 2 \log(1/\gamma) + O(1)$. The "operational" characterization of total variation distance implies that Gen is also a γ-PRG for (k, n, Σ) block decision trees as defined in Section 3. $\square$

ECCC ISSN 1433-8092

http://eccc.hpi-web.de