A Sample of Samplers: A Computational Perspective on Sampling

Oded Goldreich

Abstract. We consider the problem of estimating the average of a huge set of values. That is, given oracle access to an arbitrary function $f : \{0,1\}^n \to [0,1]$, we wish to estimate $2^{-n} \sum_{x \in \{0,1\}^n} f(x)$ upto an additive error of ϵ. We are allowed to employ a randomized algorithm that may err with probability at most δ.
We survey known algorithms for this problem and focus on the ideas underlying their construction. In particular, we present an algorithm that makes $O(\epsilon^{-2} \cdot \log(1/\delta))$ queries and uses $n + O(\log(1/\epsilon)) + O(\log(1/\delta))$ coin tosses, both complexities being very close to the corresponding lower bounds.

Keywords: Sampling, randomness complexity, saving randomness, pairwise independent random variables, Expander graphs, random walks on graphs, information theoretic lower bounds.

An earlier version of this survey appeared as TR97-020 of *ECCC*. The current version includes a quantitative improvement in Theorem 6.1, which is obtained by the subsequent work of [26].

Preface. The idea of writing this survey occurred to me when finding out that a brilliant, young researcher who has worked in very related areas was unaware of the Median-of-Averages Sampler (of [7]). It then occurred to me that many of the results surveyed here have appeared in papers devoted to other subjects (indeed, the Median-of-Averages Sampler is an excellent example), and have thus escaped the attention of a wider community, which might have cared to know about them. I thus decided to write a survey that focuses on these very basics.

1 Introduction

In many settings repeated sampling is used to estimate the average value of a huge set of values. Namely, one has access to a value function ν, which is defined over a huge space (say, $\nu : \{0,1\}^n \to [0,1]$), and wishes to approximate $\bar{\nu} \stackrel{\text{def}}{=} \frac{1}{2^n} \sum_{x \in \{0,1\}^n} \nu(x)$ without having to inspect the value of ν on the entire domain. It is well-known that sampling ν at sufficiently many (random) points yields such an approximation, but we are interested in the complexity of the approximation. Specifically, (1) how many samples are required? (2) how much randomness is required to generate these samples? and (3) is this generation procedure efficient?

We comment that it is essential to have the range of ν be bounded (or else no reasonable approximation may be possible). Our convention of having $[0,1]$ be the range of ν is adopted for simplicity, and the problem for other (predetermined) ranges can be treated analogously.

1.1 Formal Setting

Our notion of approximation depends on two parameters: *accuracy* (denoted ϵ) and *error probability* (denoted δ). We wish to have an algorithm that, with probability at least $1-\delta$, gets within ϵ of the correct value. This leads to the following definition.

Definition 1.1 (sampler): *A* sampler *is a randomized algorithm that on input parameters* n (length)*,* ϵ (accuracy) *and* δ (error)*, and oracle access to* any *function* $\nu : \{0,1\}^n \to [0,1]$*, outputs, with probability at least* $1-\delta$*, a value that is at most* ϵ *away from* $\bar{\nu} \stackrel{\text{def}}{=} \frac{1}{2^n}\sum_{x\in\{0,1\}^n}\nu(x)$*. Namely,*

$$\Pr\left[|\text{sampler}^{\nu}(n,\epsilon,\delta) - \bar{\nu}| > \epsilon\right] \;<\; \delta, \tag{1}$$

where the probability is taken over the internal coin tosses of the sampler.

We are interested in "the complexity of sampling" quantified as a function of the parameters n, ϵ and δ. Specifically, we will consider three complexity measures:

1. Sample Complexity: The number of oracle queries made by the sampler.
2. Randomness Complexity: The number of (unbiased) coin tosses performed by the sampler.
3. Computational Complexity: The running-time of the sampler.
   We say that a sample is efficient if its running-time is polynomial in the total length of its queries (i.e., polynomial in both its sample complexity and in the length parameter, n).

We will focus on efficient samplers. Furthermore, we will focus on efficient samplers that have optimal (upto a constant factor) sample complexity, and will be interested in having the randomness complexity be as low as possible.

1.2 Overview

The straightforward method (or the naive sampler) consists of *uniformly and independently* selecting sufficiently many sample points (queries), and outputting the average value of the function on these points. Using Chernoff Bound one can easily show that $O(\frac{\log(1/\delta)}{\epsilon^2})$ sample points suffice. The naive sampler is optimal (upto a constant factor) in its sample complexity, but is quite wasteful in randomness. In Section 2, we discuss the naive sampler and present lower (and upper) bounds on the sample and randomness complexities of samplers. These bounds will guide our quest for improvements.

Pairwise-independent sampling yields a great saving in the randomness complexity. In Section 3 we present the Pairwise-Independent Sampler, and discuss its advantages and disadvantages. Specifically, for constant $\delta > 0$, the Pairwise-Independent Sampler is optimal upto a constant factor in both its sample and randomness complexities. However, for small δ (i.e., $\delta = o(1)$), its sample complexity is wasteful.

An additional idea is required for going further, and a relevant tool – random walks on expander graphs (see Appendix A) – is also used. In Section 4, we combine the Pairwise-Independent Sampler with the Expander Random Walk Technique to obtain a new sampler. Loosely speaking, the new sampler uses a random walk on an expander to generate a sequence of $\ell \stackrel{\text{def}}{=} O(\log(1/\delta))$ (related) random pads for ℓ invocations of the Pairwise-Independent Sampler. Each of these invocations returns an ϵ-close approximation with probability at least 0.99. The expander walk technique yields that, with probability at least $1-\exp(-\ell) = 1-\delta$, most of these ℓ invocations return an ϵ-close approximation. Thus, the median value is an (ϵ, δ)-approximation to the correct value (i.e., an approximation that, with probability at least $1-\delta$, is within an additive term of ϵ of the correct value). The resulting sampler, called the Median-of-Averages Sampler, has sample complexity $O(\frac{\log(1/\delta)}{\epsilon^2})$ and randomness complexity $2n + O(\log(1/\delta))$.

In Section 5 we present an alternative sampler that improves over the pairwise-independent sampler. Maintaining the sample complexity of the latter (i.e., $O(1/\delta\epsilon^2)$), the new sampler has randomness complexity $n+O(\log(1/\delta\epsilon))$ (rather than $2n$). Combining this new sampler with the Expander Random Walk Technique, we obtain sample complexity $O(\frac{\log(1/\delta)}{\epsilon^2})$ and randomness complexity $n + O(\log(1/\delta)) + O(\log(1/\epsilon))$. Better bounds are obtained for the case of "Boolean samplers" (i.e., algorithms that must only well-approximate Boolean functions). In addition, in Section 5 we present two general techniques for improving existing samplers.

We conclude with some open problems (see Section 6). In particular, we discuss the notion of "oblivious" (or "averaging") samplers, which is closely related to the notion of randomness extractors (see Section 7.2 and more details in [28]).[1] Section 7 sketches the outline of an alternative survey that focuses on the notion of "averaging" samplers and on their relation to general samplers, on the one hand, and to randomness extractors, on the other hand.

The Hitting Problem. In order to distinguish the all-zero function from a function having at least an ϵ fraction of non-zero values, the sampler must query the function at a non-zero value (or "hit" some non-zero value). Thus, any sampler solves the *hitting problem*, as surveyed in Appendix C. That is, given an oracle to a Boolean function having at least an ϵ fraction of 1's, the "hitter" is required to

[1] Indeed, the current text focuses on general samplers, which are not necessarily of the "averaging" type (e.g., the aforementioned Median-of-Averages Sampler). Thus, this survey barely mentions the vast body of work that focuses on randomness extractors, and the interested reader is indeed referred to [28].

find an input that evaluates to 1. As noted above, each sampler can be used for this purpose, but this is an over-kill. Indeed, all results and techniques regarding samplers (presented in the main text of this survey) have simpler analogues for the hitting problem. Thus, Appendix C can be read as a warm-up towards the rest of the survey.

2 The Information Theoretic Perspective

The Naive Sampler, presented below, corresponds to the information theoretical (or statistician) perspective of the problem. We augment it by a lower bound on the *sample complexity* of samplers, which is in the spirit of these areas. We conclude with lower and upper bounds on the *randomness complexity* of samplers. The latter lower bound is also information theoretic in nature, but it refers to a concern that is more common in computer science.

2.1 The Naive Sampler

The straightforward sampling method consists of randomly selecting a small sample set S and outputting $\frac{1}{|S|}\sum_{x\in S}\nu(x)$ as an estimate to $\bar{\nu}$. More accurately, we select m *independently and uniformly distributed* strings in $\{0,1\}^n$, denoted $s_1,...,s_m$, and output $\frac{1}{m}\sum_{i=1}\nu(s_i)$ as our estimate. Setting $m=\frac{\ln(2/\delta)}{2\epsilon^2}$, we refer to this procedure as to the Naive Sampler.

To analyze the performance of the Naive Sampler, we use the Chernoff Bound. Specifically, we define m independent random variables, denoted $\zeta_1,...,\zeta_m$, such that $\zeta_i \stackrel{\text{def}}{=} \nu(s_i)$, where the s_i's are independently and uniformly distributed in $\{0,1\}^n$. By Chernoff Bound:

$$\Pr\left[\left|\bar{\nu}-\frac{1}{m}\sum_{i=1}^{m}\zeta_i\right|>\epsilon\right] \leq 2\exp\left(-2\epsilon^2 m\right) \tag{2}$$

$$< \delta \tag{3}$$

where Eq. (3) is due to $m=\ln(2/\delta)/2\epsilon^2$. Observing that $\frac{1}{m}\sum_{i=1}^{m}\zeta_i$ represents the estimate output by the Naive Sampler, we have established that the Naive Sampler indeed satisfies Definition 1.1 (i.e., is indeed a sampler). We now consider the complexity of the Naive Sampler

- Sample Complexity: $m \stackrel{\text{def}}{=} \frac{\ln(2/\delta)}{2\epsilon^2} = \Theta(\frac{\log(1/\delta)}{\epsilon^2})$.
- Randomness Complexity: $m\cdot n = \Theta(\frac{\log(1/\delta)}{\epsilon^2}\cdot n)$.
- Computational Complexity: indeed efficient.

In light of Theorem 2.1 (below), the sample complexity of the Naive Sampler is optimal upto a constant factor. However, as we will shortly see, it is extremely wasteful in its usage of randomness. In fact, the rest of this survey is devoted to presenting ways for redeeming the latter aspect.

2.2 A Sample Complexity Lower Bound

We first assert that the Naive Sampler is quite good as far as sample complexity is concerned. The following theorem is analogous to many results known in statistics, though we are not aware of a reference prior to [10] where it can be found.

Theorem 2.1 [10]: *Any sampler has sample complexity bounded below by*

$$\min\left\{2^{(n-4)/2}, \frac{\ln(1/O(\delta))}{4\epsilon^2}\right\}$$

provided $\epsilon \leq \frac{1}{8}$ *and* $\delta \leq \frac{1}{6}$.

Note that a (constant factor) gap remains between the lower bound asserted here and the upper bound established by the Naive Sampler. We conjecture that the lower bound can be improved. Motivated by the lower bound, we say that a sampler is sample-optimal if its sample complexity is $O(\frac{\log(1/\delta)}{\epsilon^2})$.

2.3 Randomness Complexity Lower and Upper Bounds

We first assert that the Naive Sampler is quite bad as far as randomness complexity is concerned. First evidence towards our claim is provided by a non-explicit (and so inefficient) sampler:

Theorem 2.2 [10]: *There exists a* (non-efficient) *sampler with sample complexity* $\frac{2\ln(4/\delta)}{\epsilon^2}$ *and randomness complexity* $n + 2\log_2(2/\delta) + \log_2\log_2(1/\epsilon)$.

The proof is by a probabilistic argument that, given the Naive Sampler, asserts the existence of a relatively small set of possible coin tosses under which this sampler behaves almost as under all possible coin tosses (with respect to any possible function ν). Actually, the randomness bound can be improved to $n + \log_2(1/\delta) - \log_2\log_2(1/\delta)$ while using a constant factor larger sample complexity and more sophisticated techniques [30]. More generally:

Theorem 2.3 [30]: *For every function* $s : [0,1]^2 \to \mathbb{R}$ *such that* $s(\epsilon,\delta) \geq \frac{2\log_2(1/\delta)}{\epsilon^2}$, *there exists a* (non-efficient) *sampler with sample complexity* $s(\epsilon,\delta)$ *and randomness complexity*

$$n + \log_2(1/\delta) + 2\log_2(4/\epsilon) - \log_2 s(\epsilon,\delta)$$

This gets us very close to the following lower bound.

Theorem 2.4 [10]: *Let* $s : \mathbb{N} \times [0,1]^2 \to \mathbb{R}$. *Any sampler that has sample complexity at most* $s(n,\epsilon,\delta)$, *has randomness complexity at least*

$$n + \log_2(1/\delta) - \log_2 s(n,\epsilon,\delta) - \log_2(1-2\epsilon)^{-1} - 2,$$

provided $\epsilon, \delta < 0.5$ *and* $s(n,\epsilon,\delta) \leq 2^{n-1}$.

The dependency of the lower bound on the sample complexity should not come as a surprise. After all, there exists a deterministic sampler that queries the function on the entire domain. Furthermore, the upper bound of Theorem 2.3 does express a similar trade-off between randomness complexity and sample complexity. Similarly, one should not be surprised at the effect of $1-2\epsilon$ on the bound: For example, when $\epsilon = 0.5$, a sample may merely output $\widetilde{\nu} = \frac{1}{2}$ as its estimate and always be within ϵ of the average of any function $\nu : \{0,1\}^n \to [0,1]$.

Using Theorem 2.4, we obtain a lower bound on the randomness complexity of any *sample-optimal* sampler:

Corollary 2.5 [10]: *Any sampler that has sample complexity* $O(\frac{\log(1/\delta)}{\epsilon^2})$*, has randomness complexity at least*[2]

$$n + (1 - o(1)) \cdot \log_2(1/\delta) - 2\log_2(1/\epsilon),$$

provided $\epsilon, \delta < 0.4$ *and* $\frac{\log(1/\delta)}{\epsilon^2} = o(2^n)$.

3 The Pairwise-Independent Sampler

To motivate the Pairwise-Independent Sampler, let us confront two well-known central limit theorems: Chernoff Bound, which refers to *totally independent* random variables, and Chebyshev's Inequality, which refers to *pairwise-independent* random variables

Chernoff Bound: Let $\zeta_1, ..., \zeta_m$ be *totally independent* random variables, each ranging in $[0,1]$ and having expected value μ. Then,

$$\Pr\left[\left|\mu - \frac{1}{m}\sum_{i=1}^{m}\zeta_i\right| > \epsilon\right] \;\leq\; 2\exp\left(-2\epsilon^2 m\right)$$

Chebyshev's Inequality: Let $\zeta_1, ..., \zeta_m$ be *pairwise-independent* random variables, each ranging in $[0,1]$ and having expected value μ. Then,

$$\Pr\left[\left|\mu - \frac{1}{m}\sum_{i=1}^{m}\zeta_i\right| > \epsilon\right] \;\leq\; \frac{1}{4\epsilon^2 m}$$

Our conclusion is that these two bounds essentially agree when $m = O(1/\epsilon^2)$. That is, in both cases $\Theta(1/\epsilon^2)$ identical random variables are necessary and sufficient to guarantee a concentration within ϵ with constant probability. Thus, if this is what we want, then there is no point in using the more sophisticated Chernoff Bound, which requires more of the random variables.

In the context of sampling, our conclusion is that for achieving an approximation to within ϵ accuracy with constant error probability, using $O(1/\epsilon^2)$ pairwise-independent random sample points is as good as using $O(1/\epsilon^2)$ totally independent random sample points. Furthermore, in the first case we may be save a lot in terms of randomness.

[2] The $o(1)$ term is actually $\frac{\log_2 O(\log(1/\delta))}{\log_2(1/\delta)}$.

The Pairwise-Independent Sampler [12]: On input parameters n, ϵ and δ, set $m \stackrel{\text{def}}{=} \frac{1}{4\epsilon^2\delta}$ and generate a sequence of m *pairwise-independently and uniformly distributed* strings in $\{0,1\}^n$, denoted $s_1, ..., s_m$. Using the oracle access to ν, output $\frac{1}{m}\sum_{i=1}\nu(s_i)$ as the estimate to $\bar{\nu}$. Using Chebyshev's Inequality, one can easily see that the Pairwise-Independent Sampler indeed satisfies Definition 1.1 (i.e., is indeed a sampler).

There are two differences between the Naive Sampler and the Pairwise-Independent Sampler. Whereas the former uses independently selected sample points, the latter uses a sequence of pairwise independent sample points. As we shall see, this allows the latter sampler to use much less randomness. On the other hand, the Naive Sampler uses $O(\frac{\log(1/\delta)}{\epsilon^2})$ samples (which is optimal upto a constant factor), whereas the Pairwise-Independent Sampler uses $O(\frac{1}{\epsilon^2\delta})$ samples. However, for constant δ, both samplers use essentially the same number of sample points. Thus, for constant δ, the Pairwise-Independent Sampler offers a saving in randomness while being sample-optimal.

Generating a Pairwise-Independent sequence: Whereas generating m totally independent random points in $\{0,1\}^n$ requires $m \cdot n$ unbiased coin flips, one can generate m ($m \leq 2^n$) pairwise-independent random points using only $O(n)$ unbiased coin flips. We present two well-known ways of doing this.

1. Linear functions over finite fields: We associate $\{0,1\}^n$ with the finite field $F \stackrel{\text{def}}{=} \mathrm{GF}(2^n)$. Let $\alpha_1, ..., \alpha_m$ be $m \leq |F|$ distinct elements of F. To generate a (pairwise-independent) sequence of length m, we uniformly and independently select $s, r \in F$, and let the i^{th} element in the sequence be $e_i \stackrel{\text{def}}{=} r + \alpha_i s$ (where the arithmetic is that of F). The analysis of this construction "reduces" the stochastic independence of e_i and e_j to the linear independence of the vectors $(1\ ,\ \alpha_i)$ and $(1\ ,\ \alpha_j)$: For every $i \neq j$ and every $a, b \in F$, we have

$$\begin{aligned}\Pr_{r,s}\left[e_i = a \ \wedge\ e_j = b\right] &= \Pr_{r,s}\left[\begin{pmatrix}1 & \alpha_i\\ 1 & \alpha_j\end{pmatrix}\begin{pmatrix}r\\ s\end{pmatrix} = \begin{pmatrix}a\\ b\end{pmatrix}\right]\\ &= \Pr_{r,s}\left[\begin{pmatrix}r\\ s\end{pmatrix} = \begin{pmatrix}1 & \alpha_i\\ 1 & \alpha_j\end{pmatrix}^{-1}\begin{pmatrix}a\\ b\end{pmatrix}\right]\\ &= \frac{1}{|F|^2}.\end{aligned}$$

   Only $2n$ random coins are required in this construction, but the drawback is that we need a representation of the field F (i.e., an irreducible polynomial of degree n over GF(2)) which may not be easy to find in general.[3] Still, for specific values of n a good representation exists: Specifically, for $n = 2 \cdot 3^\ell$ (with ℓ integer), the polynomial $x^n + x^{n/2} + 1$ is irreducible [17, p. 96], and so we obtain a representation of $\mathrm{GF}(2^n)$ for such n's.

[3] Things are not better if we wish to work with a large field of prime cardinality; since we need to find such a prime.

2. Toeplitz matrices: To avoid problems with non-trivial representation, one may use the following construction. We associate $\{0,1\}^n$ with the n-dimensional vector space over GF(2). Let $v_1, ..., v_m$ be $m \leq 2^n$ distinct vectors in this vector space. A Toeplitz matrix is a matrix with all diagonals being homogeneous; that is, $T = (t_{i,j})$ is a Toeplitz matrix if $t_{i,j} = t_{i+1,j+1}$, for all i, j. Note that a Toeplitz matrix is determined by its first row and first column (i.e., the values of $t_{1,j}$'s and $t_{i,1}$'s). To generate a (pairwise-independent) sequence of length m, we uniformly and independently select an n-by-n Boolean Toeplitz matrix, T, and an n-dimensional Boolean vector u. We let the i^{th} element in the sequence be $e_i \stackrel{\text{def}}{=} Tv_i + u$ (where the arithmetic is that of the vector space). The analysis of this construction is given in Appendix B. Here, we merely note that $3n - 1$ random coins suffice for this construction,

Plugging-in either of these constructions, we obtain the following complexities for the Pairwise-Independent Sampler

- Sample Complexity: $\frac{1}{4\delta\epsilon^2}$.
- Randomness Complexity: $2n$ or $3n-1$, depending on which of the constructions is used.
- Computational Complexity: Indeed efficient.

We note that for constant δ, the sample and randomness complexities match the lower bounds upto a constant factor. However, as δ decreases, the sample complexity of the Pairwise-Independent Sampler increases faster than the corresponding complexity of the Naive Sampler. Redeeming this state of affairs is our next goal.

4 The (Combined) Median-of-Averages Sampler

Our goal here is to decrease the sample complexity of the Pairwise-Independent Sampler while essentially maintaining its randomness complexity. To motivate the new construction we first consider an oversimplified version of it.

Median-of-Averages Sampler (oversimplified): On input parameters n, ϵ and δ, set $m \stackrel{\text{def}}{=} \Theta(\frac{1}{\epsilon^2})$ and $\ell \stackrel{\text{def}}{=} \Theta(\log(1/\delta))$, generate ℓ independent m-element sequences, each being a sequence of m *pairwise-independently and uniformly distributed* strings in $\{0,1\}^n$. Denote the sample points in the i^{th} sequence by $s^i_1, ..., s^i_m$. Using the oracle access to ν, compute $\widetilde{\nu}^i \stackrel{\text{def}}{=} \frac{1}{m}\sum_{j=1}^{m} \nu(s^i_j)$, for $i = 1, ..., \ell$, and output the *median value* among these $\widetilde{\nu}^i$'s. Using Chebyshev's Inequality (as in previous section), for each i, it holds that

$$\Pr[|\widetilde{\nu}^i - \bar{\nu}| > \epsilon] < 0.1$$

and so

$$\Pr\left[|\{i : |\widetilde{\nu}^i - \bar{\nu}| > \epsilon\}| \geq \frac{\ell}{2}\right] < \sum_{j=\ell/2}^{\ell} \binom{\ell}{j} \cdot 0.1^j \cdot 0.9^{\ell-j}$$

$$< 2^{\ell} \cdot 0.1^{\ell/2}$$
$$\leq \delta,$$

where the last inequality is due to the choice of ℓ. Thus, the oversimplified version described above is indeed a sampler and has the following complexities

- Sample Complexity: $\ell \cdot m = O(\frac{\log(1/\delta)}{\epsilon^2})$.
- Randomness Complexity: $\ell \cdot O(n) = O(n \cdot \log(1/\delta))$.
- Computational Complexity: Indeed efficient.

Thus, the sample complexity is optimal (upto a constant factor), but the randomness complexity is higher than what we aim for. To reduce the randomness complexity, we use the same approach as above, but take dependent sequences rather than independent ones. The dependency we use is such that essentially preserves the probabilistic behavior of independent choices. Specifically, we use random walks on expander graphs (cf., Appendix A) to generate a sequence of ℓ "seeds" each of length $O(n)$. Each seed is used to generate a sequence of m pairwise independent elements in $\{0,1\}^n$, as above. Let us generalize this construction as follows.

Theorem 4.1 (general median-composition [7]): *Suppose we are given an efficient sampler of sample complexity $s(n,\epsilon,\delta)$ and randomness complexity $r(n,\epsilon,\delta)$. Then:*

1. *There exists an efficient sampler with sample complexity $O(s(n,\epsilon,0.01) \cdot \log(1/\delta))$ and randomness complexity $r(n,\epsilon,0.01) + O(\log(1/\delta))$.*
2. *For any $c > 4$, there exists an $\alpha > 0$ and an efficient sampler with sample complexity $O(s(n,\epsilon,\alpha) \cdot \log(1/\delta))$ and randomness complexity $r(n,\epsilon,\alpha) + c \cdot \log_2(1/\delta)$.*

Proof: For Item 1, let $r \stackrel{\text{def}}{=} r(n,\epsilon,0.01)$. We use an explicit construction of expander graphs with vertex set $\{0,1\}^r$, degree d and second eigenvalue λ so that $\lambda/d < 0.1$. We consider a random walk of (edge) length $\ell - 1 = O(\log(1/\delta))$ on this expander, and use each of the ℓ vertices along the path as random coins for the given sampler. Thus, we obtain ℓ estimates to $\bar{\nu}$ and output the median value as the estimate of the new sampler. To analyze the performance of the resulting sampler, we let W denote the set of coin tosses (for the basic sampler) that make the basic sampler output an estimate that is ϵ-far from the correct value (i.e., $\bar{\nu}$). Thus, W denotes the set of coin tosses that are bad for the basic sampler, and by the hypothesis $\frac{|W|}{2^r} \leq 0.01$. Using Theorem A.4 (with some W_i's set to W and the others set to $\{0,1\}^r$), we infer that the probability that at least $\ell/2$ vertices of the path reside in W is smaller than

$$\sum_{j=\ell/2}^{\ell} \binom{\ell}{j} \cdot 0.02^{j/2} < 2^{\ell} \cdot 0.02^{\ell/4}$$
$$\leq \delta.$$

Note that we have used $\ell \cdot s(n,\epsilon,0.01)$ samples and $r + (\ell-1)\cdot\log_2 d = r + O(\log(1/\delta))$ coin tosses. Item 1 follows.

Item 2 is proved using the same argument but using Ramanujan Graphs (and slightly more care). Specifically, we use Ramanujan graphs (i.e., expanders with $\lambda \leq 2\sqrt{d-1}$) with vertex set $\{0,1\}^r$, where $r \stackrel{\text{def}}{=} r(n,\epsilon,\alpha)$ and $\alpha = (\frac{\lambda}{d})^2$. Repeating the foregoing argument, with $\ell - 1 = \frac{2\log_2(1/\delta)}{\log_2(\alpha/8)}$, we obtain an efficient sampler that uses $\ell \cdot s(n,\epsilon,\alpha)$ samples and $r + (\ell-1)\cdot\log_2 d = r + (4 + \frac{16}{(\log_2 d)-8})\cdot \log_2(1/\delta)$ coin tosses. Since this can be done with a sufficiently large d, Item 2 follows. ■

Combining the Pairwise-Independent Sampler with Theorem 4.1, we get

Corollary 4.2 (The Median-of-Averages Sampler [7]): *There exists an efficient sampler with*

- Sample Complexity: $O(\frac{\log(1/\delta)}{\epsilon^2})$.
- Randomness Complexity: $O(n + \log(1/\delta))$.

Furthermore, we can obtain randomness complexity $2n + (4 + o(1)) \cdot \log_2(1/\delta))$.

In the next section, we further reduce the randomness complexity of samplers (from $2n + O(\log(1/\delta))$) to $n + O(\log(1/\epsilon) + \log(1/\delta))$, while maintaining the sample complexity (up-to a multiplicative constant).

Generalizing Theorem 4.1. A close look at the proof of Theorem 4.1 reveals the fact that the median value obtained via an expander random walk (on the vertex set $\{0,1\}^r$) is used as a sampler of accuracy 0.49 and error probability δ. This suggests the following generalization of Theorem 4.1: *Suppose we are given two efficient samplers such that the* i^{th} *sampler has sample complexity* $s_i(n,\epsilon,\delta)$ *and randomness complexity* $r_i(n,\epsilon,\delta)$*. Then, for every* $\delta_0 \in (0,0.5)$*, there exists an efficient sampler of sample complexity* $s_2(r,0.5-\delta_0,\delta)\cdot s_1(n,\epsilon,\delta_0)$ *and randomness complexity* $r_2(r,0.5-\delta_0,\delta)$*, where* $r \stackrel{\text{def}}{=} r_1(n,\epsilon,\delta_0)$. Theorem 4.1 is derived as a special case, when using the expander random walk as the second sampler and setting $\delta_0 = 0.01$.

5 The Expander Sampler and Two Generic Transformations

The main result of this section is the following:

Theorem 5.1 [7, 16]: *There exists an efficient sampler that has*

- Sample Complexity: $O(\frac{\log(1/\delta)}{\epsilon^2})$.
- Randomness Complexity: $n + \log_2(1/\epsilon) + O(\log(1/\delta))$.

The theorem is proved by applying Theorem 4.1 to a new efficient sampler that makes $O(\frac{1}{\delta\epsilon^2})$ oracle queries and tosses $n+\log_2(1/\epsilon)$ coins. We start by presenting a sampler for the special case of Boolean functions.

Definition 5.2 (Boolean sampler): *A* Boolean sampler *is a randomized algorithm that on input parameters n, ϵ and δ, and oracle access to* any Boolean *function $\nu:\{0,1\}^n\to\{0,1\}$, outputs, with probability at least $1-\delta$, a value that is at most ϵ away from $\bar{\nu}\stackrel{\text{def}}{=}\frac{1}{2^n}\sum_{x\in\{0,1\}^n}\nu(x)$. Namely,*

$$\Pr[|\text{sampler}^{\nu}(n,\epsilon,\delta)-\bar{\nu}|>\epsilon] \;<\; \delta$$

where the probability is taken over the internal coin tosses of the sampler.

That is, unlike (general) samplers, a *Boolean sampler* is required to work well only when given access to a *Boolean* function. The rest of this section is organized as follows:

- ***In Section 5.1*** we present the Expander Sampler, which is a Boolean sampler of sample complexity $O(1/\delta\epsilon^2)$ and randomness complexity n. This sample complexity is obtained by using Ramanujan Graphs (rather than arbitrary expanders).
- ***In Section 5.2*** we present a (general) transformation of Boolean samplers to general ones.
- ***In Section 5.3*** we revisit the Expander Sampler, while using an arbitrary expander. More importantly, we present another generic composition of samplers, and obtain an alternative construction by using this composition in conjunction with the aforementioned sampler. Unlike the composition method that underlies Theorem 4.1, which reduces the error complexity (in an efficient manner), the current composition reduces the sample complexity.

Theorem 5.1 is proved by combining the ideas of Sections 5.1 and 5.2. An alternative proof of a somewhat weaker result is obtained by combining the ideas of Sections 5.1 and 5.3.

5.1 A Sampler for the Boolean Case

We start by presenting a sampler for the special case of Boolean functions. Our sampling procedure is exactly the one suggested by Karp, Pippinger and Sipser for hitting a witness set [22] (cf., Appendix C), yet the analysis is somewhat more involved. Furthermore, to get an algorithm that samples the universe only on $O(1/\delta\epsilon^2)$ points, it is crucial to use a Ramanujan graph in role of the expander in the Karp-Pippinger-Sipser method.

The sampler. We use an expander of degree $d=4/\delta\epsilon^2$ second eigenvalue bounded by λ and associate the vertex set of the expander with $\{0,1\}^n$. The sampler consists of uniformly selecting a vertex, v, (of the expander) and averaging over the

values assigned (by ν) to all the neighbors of v; that is, *the algorithm outputs the estimate*

$$\widetilde{\nu} \stackrel{\text{def}}{=} \frac{1}{d} \sum_{u \in \Gamma(v)} \nu(u), \tag{4}$$

where $\Gamma(v)$ *denotes the set of neighbors of vertex* v.

This algorithm has

- Sample Complexity: $O(\frac{1}{\delta\epsilon^2})$.
- Randomness Complexity: n.
- Computational Complexity: Indeed efficient; that is, polynomial in n, ϵ^{-1} and δ^{-1}.

Lemma 5.3 [16]: *The foregoing algorithm constitutes an efficient Boolean sampler.*

Proof: We denote by B the set of *bad* choices for the algorithm; namely, the set of vertices that once selected by the algorithm yield a wrong estimate. That is, $v \in B$ if

$$\left| \frac{1}{d} \sum_{u \in \Gamma(v)} \nu(u) - \bar{\nu} \right| > \epsilon. \tag{5}$$

Denote by B' the subset of $v \in B$ for which

$$\frac{1}{d} \sum_{u \in \Gamma(v)} \nu(u) > \bar{\nu} + \epsilon. \tag{6}$$

It follows that each $v \in B'$ has ϵd too many neighbors in the set $A \stackrel{\text{def}}{=} \{u : \nu(u) = 1\}$; namely,

$$|\{u \in \Gamma(v) : u \in A\}| > (\rho(A) + \epsilon) \cdot d, \tag{7}$$

where $\rho(A) \stackrel{\text{def}}{=} \frac{|A|}{N}$ and $N \stackrel{\text{def}}{=} 2^n$. Using the Expander Mixing Lemma (i.e., Lemma A.2), we get that

$$\begin{aligned} \epsilon \cdot \rho(B') &= \left| \frac{|B'| \cdot (\rho(A) + \epsilon)d}{dN} - \rho(B') \cdot \rho(A) \right| \\ &\leq \left| \frac{|(B' \times A) \cap E|}{|E|} - \frac{|A|}{|V|} \cdot \frac{|B'|}{|V|} \right| \\ &\leq \frac{\lambda}{d} \cdot \sqrt{\rho(A) \cdot \rho(B')}. \end{aligned}$$

Thus,

$$\rho(B') \leq \left(\frac{\lambda}{d\epsilon} \right)^2 \cdot \rho(A). \tag{8}$$

Using $\lambda \leq 2\sqrt{d}$ and $d = \frac{4}{\delta\epsilon^2}$, we get $\rho(B') \leq \delta \cdot \rho(A)$. Using a similar argument,[4] we can show that $\rho(B \setminus B') \leq \delta \cdot (1 - \rho(A))$. Thus, $\rho(B) \leq \delta$, and the claim follows. ■

Comment 5.4 [16]: *Observe that if we were to use an arbitrary d-regular graph with second eigenvalue λ, then the foregoing proof would hold provided that*

$$\frac{\lambda}{d} \leq \sqrt{\delta\epsilon^2}. \tag{9}$$

This yields, for any such d-regular graph, an efficient Boolean sampler with sample complexity d *and* randomness complexity n.

5.2 From Boolean Samplers to General Samplers

The following generic transformation was suggested to us by Luca Trevisan.

Theorem 5.5 (Boolean samplers imply general ones): *Suppose we are given an efficient Boolean sampler of sample complexity $s(n, \epsilon, \delta)$ and randomness complexity $r(n, \epsilon, \delta)$. Then, there exists an efficient sampler with sample complexity $s(n + \log_2(1/\epsilon), \epsilon/2, \delta)$ and randomness complexity $r(n + \log_2(1/\epsilon), \epsilon/2, \delta)$.*

Proof: As a mental experiment, given an arbitrary function $\nu : \{0,1\}^n \to [0,1]$, we define a Boolean function $\mu : \{0,1\}^{n+\ell} \to \{0,1\}$, where $\ell \stackrel{\text{def}}{=} \log_2(1/\epsilon)$, as follows: For every x and $i = 1, ..., \epsilon^{-1}$, we set $\mu(x,i) \stackrel{\text{def}}{=} 1$ if and only if $\nu(x) > (i - 0.5) \cdot \epsilon$ (i.e., iff $i < \epsilon^{-1}\nu(x) + 0.5$). Then, for every x, it holds that $|\nu(x) - \epsilon \cdot \sum_{i=1}^{1/\epsilon} \mu(x,i)| \leq \epsilon/2$. Thus, if we were to sample μ and obtain an $\epsilon/2$-approximation of $\bar{\mu}$ then we get an ϵ-approximation of $\bar{\nu}$. Now, although we don't have actual access to μ we can emulate its answers given an oracle to ν.

Given a Boolean sampler, B, we construct a general sampler, A, as follows. On input n, ϵ, δ and access to an arbitrary ν as above, algorithm A sets $n' = n + \ell$, $\epsilon' = \epsilon/2$, and $\delta' = \delta$, and invoke B on input n', ϵ', δ'. When B makes a query $(x,i) \in \{0,1\}^n \times \{0,1\}^\ell$, algorithm A queries for $\nu(x)$ and returns 1 if and only if $\nu(x) > (i - 0.5) \cdot \epsilon$. When B halts with output v, algorithm A does the same. The theorem follows. ■

Combining the sampler of Section 5.1 with Theorem 5.5, we get

Corollary 5.6 (The Expander Sampler, revisited): *There exists an efficient sampler that has*

- Sample Complexity: $O(\frac{1}{\delta\epsilon^2})$.
- Randomness Complexity: $n + \log_2(1/\epsilon)$.

Theorem 5.1 follows by combining Corollary 5.6 with Theorem 4.1.

[4] That is, we consider the set $B'' \stackrel{\text{def}}{=} B \setminus B'$, and observe that every $v \in B''$ has ϵd too many neighbours in $A'' \stackrel{\text{def}}{=} \{0,1\}^n \setminus A$. Hence, we conclude that $\rho(B'') \leq \delta \cdot \rho(A'')$.

5.3 An Alternative Construction

Using an arbitrary expander graph (with $d = \text{poly}(1/\epsilon\delta)$ and $\frac{\lambda}{d} < \sqrt{\delta\epsilon^2}$) and invoking Comment 5.4, we have an efficient Boolean sampler with sample complexity $\text{poly}(1/\epsilon\delta)$ and randomness complexity n. Using Theorem 5.5, we get

Corollary 5.7 (The Expander Sampler, revisited again): *There exists an efficient sampler with* sample complexity $\text{poly}(1/\epsilon\delta)$ *and* randomness complexity $n + \log_2(1/\epsilon)$.

To derive (a weaker form of) Theorem 5.1 via the foregoing sampler, we first need to reduce its sample complexity. This is done via the following general transformation. We say that a sampler is of the averaging type if its output is the average value obtained on its queries, which in turn are determined as a function of its own coin tosses (independently of the answers obtained on previous queries).

Theorem 5.8 (reducing sample complexity (or "sampling the sample")): *Suppose we are given two efficient samplers such that the i^{th} sampler has sample complexity $s_i(n,\epsilon,\delta)$ and randomness complexity $r_i(n,\epsilon,\delta)$. Further suppose that the first sampler is of the averaging type. Then, there exists an efficient sampler of sample complexity $s_2(\log_2 s_1(n,\epsilon/2,\delta/2),\epsilon/2,\delta/2)$ and randomness complexity $r_1(n,\epsilon/2,\delta/2) + r_2(\log_2 s_1(n,\epsilon/2,\delta/2),\epsilon/2,\delta/2)$. Furthermore, if also the second sampler is of the averaging type, then so is the resulting sampler.*

Proof: We compose the two samplers as follows. Setting $m \stackrel{\text{def}}{=} s_1(n,\epsilon/2,\delta/2)$, we invoke the first sampler and determine the m queries it would have asked (given a particular choice of its coins).[5] We then use the second sampler to sample these m queries (invoking it with parameters $\log_2 m, \epsilon/2$ and $\delta/2$). Specifically, we let the second sampler make virtual queries into the domain $[m] \stackrel{\text{def}}{=} \{1,...,m\}$ and answer a query $q \in [m]$ by the value of the function at the i^{th} query specified by the first sampler. That is, given access to a function $\nu : \{0,1\}^n \to [0,1]$, and determining a sequence r of coins for the first sampler, we consider the function $\nu_r : [m] \to [0,1]$ defined by letting $\nu_r(i) = \nu(q_{r,i})$ where $q_{r,i}$ is the i^{th} query made by the first sampler on coins r. We run the second sampler providing it virtual access to the function ν_r in the obvious manner, and output its output. Thus, the complexities are as claimed and the combined sampler errs if either $|\bar{\nu} - \frac{1}{m}\sum_{i=1}^m \nu(q_{r,i})| > \frac{\epsilon}{2}$ or $|\frac{1}{m}\sum_{i=1}^m \nu(q_{r,i}) - \widetilde{\nu}_r| > \epsilon/2$, where $\widetilde{\nu}_r$ is the estimate output by the second sampler when given virtual access to ν_r. Observing that the first event means that the first sampler errs (here we use the hypothesis that this sampler is averaging) and that the second event means that the second sampler errs (here we use $\sum_{i=1}^m \nu(q_{r,i}) = \bar{\nu}_r$), we are done. ■

[5] Here we use the hypothesis that the first sampler is non-adaptive; that is, its queries are determined by its coin tosses (independently of the answers obtained on previous queries).

It is tempting to try to improve the sample complexity of the sampler asserted in Corollary 5.7 by combining it with the Pairwise-Independent Sampler, via Theorem 5.8. The problem is that the former sampler, which we wish to use in the role of the outer sampler, is not of the averaging type. Indeed, the expander sampler (of Comment 5.4) is of the averaging type, but the proof of Theorem 5.5 does not preserve this feature. Instead, as shown in Theorem 5.10 (below), any Boolean sampler of the averaging type is a general sampler of the averaging time, except that its accuracy and error probability may increase by a constant factor. Thus, combining the sampler of Comment 5.4 with the Pairwise-Independent Sampler, via Theorem 5.8, we obtain:

Corollary 5.9 (sampling the Expander Sampler): *There exists an efficient sampler that has*

- Sample Complexity: $O(\frac{1}{\delta\epsilon^2})$.
- Randomness Complexity: $n + O(\log(1/\epsilon)) + O(\log(1/\delta))$.

Indeed, the sampler asserted in Corollary 5.9 operates by selecting a random vertex in an expander and taking a pairwise-independent sample of its neighbor set. A weaker form of Theorem 5.1 (i.e., with an $O(\log(1/\epsilon)$ term rather than with a $\log_2(1/\epsilon)$ term) follows by combining Corollary 5.9 with Theorem 4.1.

It is left to establish the aforementioned claim by which any Boolean sampler of the averaging type is a general sampler (of the averaging time), except that its accuracy and error probability may increase by a constant factor. (A similar statement was proved in [30].)

Theorem 5.10 (Boolean vs general samplers of the averaging type): *Every Boolean sampler of the averaging type, having sample complexity $s(n,\epsilon,\delta)$ and randomness complexity $r(n,\epsilon,\delta)$, is a general sampler* (of the averaging type) *with sample complexity $s(n,\epsilon/4,\delta/3)$ and randomness complexity $r(n,\epsilon/4,\delta/3)$.*

Proof: For any function $\nu : \{0,1\}^n \to [0,1]$, we consider a random function $\rho:\{0,1\}^n\to\{0,1\}$ such that, for every x, we set $\rho(x) = 1$ with probability $\nu(x)$, independently of the setting of all other arguments. Clearly, with probability $1 - \exp(-2\epsilon^2 2^n) > 1 - \delta$, it holds that $|\bar{\nu} - \bar{\rho}| < \epsilon$. Furthermore, fixing any possible outcome of the sampler's coins, with probability at least $1-\exp(-8\epsilon^2 s)$ over the choice of ρ, the average of the ρ-values queried by the sampler is 2ϵ-close to the average of the ν-values, where s denotes the number of queries. Since (by Theorem 2.1) $s > \epsilon^{-2}\log(1/\delta)/8$, with probability at least $1-\delta$ over the choice of ρ, the average that the Boolean sampler outputs when given access to ν is 2ϵ-close to the average it would have output on a random ρ, which in turn (with probability at least $1-\delta$ over the sampler's coins) is ϵ-close to $\bar{\rho}$. Thus, with probability at least $1-3\delta$ (over the sampler's coins), the Boolean sampler outputs a value that is 4ϵ-close to $\bar{\nu}$, ■

6 Conclusions and Open Problems

The main results surveyed in the text are summarized in Figure 1. The first row tabulates $\Omega(\epsilon^{-2}\log(1/\delta))$ as a lower bound on sample complexity and the subsequent three rows refer to sample-optimal samplers (i.e., samplers of sample complexity $O(\epsilon^{-2}\log(1/\delta))$). The last row refers to a sampler (cf., Thm. 6.1 below) that has randomness complexity closer to the lower bound. However, this sampler is not sample-optimal.

	sample complexity	randomness complexity	pointer
lower bound	$\Omega(\frac{\log(1/\delta)}{\epsilon^2})$		Thm. 2.1
lower bound	for $O(\frac{\log(1/\delta)}{\epsilon^2})$	$n + (1-o(1)) \cdot \log_2(1/\delta) - 2\log_2(1/\epsilon)$	Cor. 2.5
upper bound	$O(\frac{\log(1/\delta)}{\epsilon^2})$	$n + \log_2(1/\delta)$	Thm. 2.3
algorithm	$O(\frac{\log(1/\delta)}{\epsilon^2})$	$n + O(\log(1/\delta)) + \log_2(1/\epsilon)$	Thm. 5.1
algorithm	$\mathrm{poly}(\epsilon^{-1}, \log(1/\delta))$	$n + (1+\alpha) \cdot \log_2(1/\delta)$, $\forall \alpha > 0$	Thm. 6.1

Fig. 1. Summary of main results.

The randomness complexity of sample-optimal samplers. A closer look at the randomness complexity of sample-optimal samplers is provided in Figure 2. The first two rows tabulate lower and upper bounds, which are $2\log_2(1/\epsilon) + O(1)$ apart. Our conjecture is that the lower bound can be improved to match the upper bound.[6] The efficient samplers use somewhat more than $n + 4 \cdot \log_2(1/\delta)$ coins, where one factor of 2 is due to the use of expanders and the other to the "median-of-averages paradigm". As long as we stick to using expanders in the Median-of-Averages Sampler, there is no hope to reduce the first factor, which is due to the relation between the expander degree and its second eigenvalue. In fact, achieving a factor of 4 rather than a bigger factor is due to the use of Ramanujan Graphs (which have the best possible such relation).

Boolean samplers vs general ones. Another fact presented in Figure 2 is that we can currently do better if we are guaranteed that the oracle function is Boolean (rather than mapping to the interval $[0,1]$). We stress that the lower bound holds also with respect to samplers that need only to work for Boolean functions.

Adaptive vs non-adaptive. All known samplers are non-adaptive; that it, they determine the sample points (queries) solely as a function of their coin tosses. In contrast, *adaptive* samplers may determine the next query depending on the

[6] Partial support for this conjecture was offered to us recently by Ronen Shaltiel (priv. comm., 2010). He observed that one $\log_2(1/\epsilon)$ term can be shaved off the lower bound in the special case of averaging samplers (see below), by using the connection to randomness extractors and a lower bound on entropy loss due to [25].

lower bound (even for Boolean)	$n + \log_2(1/\delta) - 2\log_2(1/\epsilon) - \log_2\log_2(1/\delta) - O(1)$
upper bound	$n + \log_2(1/\delta) - \log_2\log_2(1/\delta)$
efficient samplers	$n + (4+\alpha)\log_2(1/\delta) + \log_2(1/\epsilon)$, for any $\alpha > 0$
efficient Boolean samplers	$n + (4+\alpha)\log_2(1/\delta)$, for any $\alpha > 0$

Fig. 2. The randomness complexity of samplers that make $\Theta(\frac{\log(1/\delta)}{\epsilon^2})$ queries.

value of the function on previous queries. Intuitively, adaptivity should not help the sampler. Indeed, all lower bounds refer also to adaptive samplers, whereas all upper bound only utilizes non-adaptive samplers. This indicates that the difference between adaptive samplers and non-adaptive ones can not be significant. In a preliminary version of this survey we advocated providing a direct and more tight proof of the foregoing intuition. When referring to the sample complexity, such a simple proof was provided in [6, Lem. 9]: It amounts to observing that adapting queries made to a random isomorphic copy of a function f are equivalent to uniformly and independently distributed queries made to f. Thus, adaptivity offers no advantage in this setting.

Averaging (or oblivious) *samplers.* A special type of non-adaptive samplers are ones that output the average value of the function over their sample points. Such samplers were first defined in [9], where they were called "oblivious", but we prefer the term *averaging*. (Recall that we have already defined and used such samplers in Section 5.3.) We mention that averaging samplers have some applications not offered by arbitrary non-adaptive samplers (cf., [9] and [29]). More importantly, averaging samplers are very appealing, since averaging over a sample seem *the natural thing to do*. Furthermore, as pointed out in [30], averaging samplers are closely related to randomness extractors (see Section 7 and more details in [28]). Note that the Naive Sampler, the Pairwise-Independent Sampler, and the Expander Sampler are all averaging samplers, although they differ in the way they generate their sample. However, the Median-of-Averages Sampler, as its name indicates, is not an averaging sampler. Thus, obtaining an averaging sampler of relatively low sample and randomness complexities requires an alternative approach. The best results are obtained via the connection to randomness extractors, and are summarized below.

Theorem 6.1 (efficient averaging samplers [26, Cor. 7.3]):[7] *For every constant $\alpha > 0$, there exists an efficient* averaging *sampler with*

- Sample Complexity: $\mathrm{poly}(\epsilon^{-1}, \log(1/\delta))$.
- Randomness Complexity: $n + (1+\alpha)\cdot\log_2(1/\delta)$.

We stress that this sampler is not sample-optimal (i.e., the polynomial in ϵ^{-1} is not quadratic). It would be interesting to obtain an efficient sample-optimal *averaging* sampler of low randomness complexity, say, one that uses $O(n+\log(1/\delta))$

[7] The result builds on [30], and uses [18, Thm. 1.5] in order to remove a mild restriction on the value of ϵ.

coins. We mention that non-explicit sample-optimal averaging samplers of low randomness complexity do exist; specifically, Theorems 2.2 and 2.3 holds with averaging-samplers (see [10, 30], resp.).

7 Postscript: A Different Perspective

As stated in the introduction, the intention of the current survey was to provide a wide audience of theoretical computer scientists with a basic tutorial regarding samplers. The focus of this tutorial was on the complexity of sampling, and our aim was to *simultaneously* minimize three complexity measures: (1) the sample complexity, (2) the randomness complexity, and (3) the computational complexity. We actually focused on the minimization of the first two, while requiring that a minimal level of computational efficiency is maintained (i.e., that the sampler works in time that is polynomial in the total length of the queries made).

From our perspective, averaging samplers are of no special interest, except maybe for their natural appeal. An alternative perspective, strongly advocated by Ronen Shaltiel and Amnon Ta-Shma, may put averaging samplers and their relation to general samplers at the main focus. This is likely to yield a very interesting survey, which we outline in the rest of this section, but it is not the one we set out to write...

7.1 Average Samplers versus General Samplers

The alternative survey will focus on the question of whether non-averaging samplers can outperform averaging samplers. As noted by Amnon and Ronen, a good starting point for such a survey is the observation that the median of averages operation can be used for improving the performance of samplers, but it yields non-averaging samplers. Specifically, the median of averages operation can be combined with simple averaging samplers (e.g., the pairwise independent ones) to yield very strong and simple non-averaging samplers. Another interesting observation is that the currently known lower bound on the randomness complexity of sample-optimal averaging samplers is higher than the currently know bound for general samplers (see Footnote 6). Finally, when viewing the minimization of sample complexity as the primary goal and the minimization of the randomness complexity as the secondary goal, the median of averages operation enables constructing *efficient* samplers that are by far better (and also much simpler) than the currently known *efficient* averaging samplers.

Another interesting parameter is the Boolean versus general distinction, which was discussed in prior sections. Recall that in the case of averaging samplers, the two notions are almost identical (see Theorem 5.10), whereas for general sampler we currently lose a $\log_2(1/\epsilon)$ term in the randomness complexity (see Theorem 5.5). Focusing on sample-optimal samplers, we summarize the currently known results in Figure 3, where the three first rows ignore the question

of efficiency (and the last row of Figure 3 is justified by combining Theorems 6.1 and 5.8).[8]

lower bound (even for Boolean)	$n + \log_2(1/\delta) - 2\log_2(1/\epsilon) - \ell - O(1)$
lower bound for averaging samplers	$n + \log_2(1/\delta) - \log_2(1/\epsilon) - \ell - O(1)$
upper bound (by averaging samplers)	$n + \log_2(1/\delta) - \ell$
efficient samplers	$n + (4+\alpha) \cdot \log_2(1/\delta) + \log_2(1/\epsilon)$, $\forall \alpha > 0$
efficient averaging samplers	$n + (1+\alpha) \cdot \log_2(1/\delta) + \widetilde{O}(s)$, $\forall \alpha > 0$

Fig. 3. The randomness complexity of samplers that make $s \stackrel{\text{def}}{=} \Theta(\frac{\log(1/\delta)}{\epsilon^2})$ queries, where ℓ denotes $\log_2 \log_2(1/\delta)$.

7.2 Average Samplers versus Randomness Extractors

We start by recalling the basic definition of randomness extractors, while (slightly) changing some common conventions to better fit our discussion.[9] Loosely speaking, a randomness extractor is a function $\text{Ext} : \{0,1\}^r \times [s] \to \{0,1\}^n$ that uses an $(\log_2 s)$-bit long random seed in order to transform an r-bit long (outcome of a) weak source of randomness into an n-bit long string that is almost uniformly distributed in $\{0,1\}^n$. Specifically, we consider arbitrary weak sources that are restricted (only) in the sense that, for a parameter k, no string appears as the source outcome with probability that exceeds 2^{-k}. Such sources are called (r,k)-sources (and k is called the min-entropy). A special type of (r,k)-sources are (r,k)-flat sources, which are sources in which each string appears with probability that equals either 2^{-k} or 0. We say that two distributions are ϵ-close if the statistical difference (a.k.a variation distance) between them is at most ϵ. Now, Ext is called a (k,ϵ)-extractor if for any (r,k)-source X it holds that $\text{Ext}(X, U_s)$ is ϵ-close to the uniform distribution over n-bit strings, where U_s denotes the uniform distribution over $[s]$.

There is a close relationship between extractors and averaging samplers. In order to discuss this relationship, it will be more convenient to state the performance guarantees of the sampler (i.e., ϵ and δ) in terms of its complexities (i.e., s and r), rather than the other way around (as done in the rest of this survey). Thus, we may say that a certain oracle machine (which has certain sample and randomness complexities) is an (ϵ,δ)-sampler if it satisfies Eq. (1) for these particular values of ϵ and δ.

We shall first show that any averaging sampler gives rise to an extractor. Let $G : \{0,1\}^r \to (\{0,1\}^n)^s$ be the sample generating algorithm of an averaging

[8] Specifically, we invoke Theorem 5.8 when using the sampler of Theorem 6.1 as the first (i.e., "outer") sampler, and the Naive Sampler as the second (i.e., "inner") sampler.

[9] Typically, extractors are defined as mapping $\{0,1\}^n \times \{0,1\}^s$ to $\{0,1\}^m$.

(ϵ, δ)-sampler. That is, G uses r bits of randomness and generates s sample points in $\{0,1\}^n$ such that, for every $f : \{0,1\}^n \to [0,1]$ with probability at least $1-\delta$, the average of the f-values of these s pseudorandom points resides in the interval $[\overline{f} \pm \epsilon]$, where $\overline{f} \stackrel{\text{def}}{=} \sum_{x \in \{0,1\}^n} f(x)/2^n$. Define $\text{Ext} : \{0,1\}^r \times [s] \to \{0,1\}^n$ such that $\text{Ext}(\omega, i)$ is the i^{th} sample generated by $G(\omega)$. We shall prove that Ext is a $(k, 2\epsilon)$-extractor, for $k = r - \log_2(\epsilon/\delta)$.

Suppose towards the contradiction that there exists a (r,k)-source X such that for some $S \subset \{0,1\}^n$ it is the case that $\Pr[\text{Ext}(X, U_s) \in S] > 2^{-n} \cdot |S| + 2\epsilon$. Then, without loss of generality, X is (r,k)-flat, and we consider the set

$$B = \{x \in \{0,1\}^r : \Pr[\text{Ext}(x, U_s) \in S] > 2^{-n} \cdot |S| + \epsilon\}.$$

Then, $|B| > \epsilon \cdot 2^k = \delta \cdot 2^r$, where the inequality holds since $\Pr[\text{Ext}(X, U_s) \in S] \leq \Pr[X \in B] + 2^{-n} \cdot |S| + \epsilon$. Defining $f(z) = 1$ if $z \in S$ and $f(z) = 0$ otherwise, it holds that $\overline{f} = |S|/2^m$. But, for every $\omega \in B$, the f-average of the sample $G(\omega)$ is greater than $\overline{f} + \epsilon$, in contradiction to the hypothesis that the sampler has error probability δ (with respect to accuracy ϵ).

We now turn to show that extractors give rise to averaging samplers. Let $\text{Ext} : \{0,1\}^r \times [s] \to \{0,1\}^n$ be a (k,ϵ)-extractor. Consider the sample generation algorithm $G : \{0,1\}^r \to (\{0,1\}^n)^s$ defined by $G(\omega) = (\text{Ext}(\omega, i))_{i \in [s]}$. We prove that G corresponds to an averaging (ϵ, δ)-sampler, for $\delta = 2^{-(r-k-1)}$.

Suppose towards the contradiction that there exists a function $f : \{0,1\}^n \to [0,1]$ such that for $\delta 2^r = 2^{k+1}$ strings $\omega \in \{0,1\}^r$ the average f-value of the sample $G(\omega)$ deviates from $\overline{f}$ by more than ϵ. Suppose, without loss of generality, that for at least half of these ω's the average is greater than $\overline{f} + \epsilon$, and let B denote the set of these ω's. Then, for X that is uniformly distributed on B (and is thus a (r,k)-source), we have

$$\text{Exp}[f(\text{Ext}(X, U_s))] > \text{Exp}[f(U'_n)] + \epsilon,$$

where U'_n denotes the uniform distribution on n-bit long strings. But, since $|f(z)| \leq 1$ for every z, this contradicts the hypothesis that $\text{Ext}(X, U_s)$ is ϵ-close to U'_n, because $|\text{Exp}[f(Y)] - \text{Exp}[f(Z)]|$ is upper bounded by the statistical difference between Y and Z (times $\max_z\{|f(z)|\}$). Summarizing the foregoing discussion, we obtain:

Theorem 7.1 (averaging samplers vs randomness extractors): *Let $r, s, k \in \mathbb{N}$ and $\epsilon, \delta \in [0,1]$. Then:*

1. *If $\text{Ext} : \{0,1\}^r \times [s] \to \{0,1\}^n$ is a (k,ϵ)-extractor, then the sample generating algorithm $G : \{0,1\}^r \to (\{0,1\}^n)^s$ defined by $G(\omega) = (\text{Ext}(\omega, i))_{i \in [s]}$ yields an averaging (ϵ, δ)-sampler for $\delta = 2^{-(r-k-1)}$* (i.e., $r - k = \log_2(1/\delta) + 1$).
2. *If $G : \{0,1\}^r \to (\{0,1\}^n)^s$ is the sample generating algorithm of an averaging (ϵ, δ)-sampler, then the algorithm $\text{Ext} : \{0,1\}^r \times [s] \to \{0,1\}^n$ defined by $\text{Ext}(\omega, i) = G(\omega)_i$ is a $(k, 2\epsilon)$-extractor, for $k = r - \log_2(\epsilon/\delta)$* (i.e., $r - k = \log_2(1/\delta) - \log_2(1/\epsilon)$).

Note that starting with a $(k, 2\epsilon)$-extractor and applying both parts of Theorem 7.1, we obtain a $(k', 2\epsilon)$-extractor for $k' = k + 1 + \log_2(1/\epsilon)$. Thus, the translation offered by Theorem 7.1 is not optimal, yet the bounds provided in both directions are (in general) tight.[10]

The connection to averaging samplers and the desire to have averaging samplers of optimal sample and randomness complexities calls attention to a research direction regarding extractors that did not receive much attention. We refer to the construction of extractors with strongly optimal seed length and almost optimal extraction rate. That is, the seed length, which is $\log_2 s$ in terms of this section, should be optimal up to a *constant additive term*, whereas the extraction rate (i.e., n/k) (or rather the inverse loss rate (i.e., $(r-k)/(n-k)$)) should be close to 1.

Acknowledgments

I would like to thank Noga Alon, Nabil Kahale, Ronen Shaltiel, Amnon Ta-Shma, Luca Trevisan, and Salil Vadhan for useful discussions.

References

1. M. Ajtai, J. Komlos, E. Szemerédi, "Deterministic Simulation in LogSpace", *Proc. 19th STOC*, 1987, pages 132–140.
2. N. Alon, "Eigenvalues, Geometric Expanders, Sorting in Rounds and Ramsey Theory", *Combinatorica*, 6 (1986), pages 231–243.
3. N. Alon, J. Bruck, J. Naor, M. Naor and R. Roth, "Construction of Asymptotically Good, Low-Rate Error-Correcting Codes through Pseudo-Random Graphs", *IEEE Transactions on Information Theory* **38** (1992), pages 509–516.
4. N. Alon and V.D. Milman, λ_1, Isoperimetric Inequalities for Graphs and Superconcentrators, *J. Combinatorial Theory, Ser. B* 38 (1985), pages 73–88.

[10] To see the tightness of Part 1, consider an arbitrary (k, ϵ)-extractor, $\mathrm{Ext} : \{0,1\}^r \times [s] \to \{0,1\}^n$, and modify it such that, for every $x' \in \{0,1\}^k$ and $i \in [3\epsilon \cdot s]$, it holds that $\mathrm{Ext}(0^{r-k}x', i) = 0^n$. Then, the modified extractor is a $(k+2, 2\epsilon)$-extractor, but the resulting averaging sampler has error probability *at least* 2^{-r+k} with respect to deviation 2ϵ. (Recall that Part 1 asserts that the resulting averaging sampler has error probability at most $2^{-(r-k-3)}$ with respect to deviation 2ϵ.) To see the tightness of Part 2, consider an arbitrary avearging (ϵ, δ)-sampler with a sample generating algorithm $G : \{0,1\}^r \to (\{0,1\}^n)^s$, and modify the latter to be identically zero on $\delta 2^r$ seeds; that is, for an arbitrary $B \subset \{0,1\}^r$ of size $\delta 2^r$, redefine G such that for every $x \in B$ it holds that $G(x) = (0^n)^s$. Then, the modified averaging sampler is an $(\epsilon, 2\delta)$-sampler, but (as shown next) the resulting extractor can be a $(k', c\epsilon)$-extractor only if $k' > k + \log_2(1/\epsilon) - c'$, where $k \stackrel{\mathrm{def}}{=} r - \log_2(1/\delta)$ and $c' = \log_2(c+1)$. The lower bound on k' holds because a (k', r)-source may assign B probability $2^{k-k'}$, whereas 0^n should be assigned probability at most $c\epsilon + 2^{-n}$. Thus, $2^{k-k'} \leq c\epsilon + 2^{-n}$, which implies $k' - k > \log_2(1/\epsilon) - c'$. (Recall that Part 2 asserets that the resulting construct is a $(k', 2\epsilon)$-extractor for $k' = k + \log_2(1/\epsilon) + 1$.)

5. N. Alon and J.H. Spencer, *The Probabilistic Method*, John Wiley & Sons, Inc., 1992.
6. Z. Bar-Yossef, R. Kumar, and D. Sivakumar, "Sampling Algorithms: Lower Bounds and Applications", *33rd STOC*, pages 266–275, 2001.
7. M. Bellare, O. Goldreich, and S. Goldwasser, "Randomness in Interactive Proofs", *Computational Complexity*, Vol. 4, No. 4 (1993), pages 319–354. Extended abstract in *31st FOCS*, 1990, pages 318–326.
8. M. Bellare, O. Goldreich, and S. Goldwasser. Addendum to [7], available from `http://theory.lcs.mit.edu/~oded/papers.html`, May 1997.
9. M. Bellare, and J. Rompel, "Randomness-efficient oblivious sampling", *35th FOCS*, 1994.
10. R. Canetti, G. Even and O. Goldreich, "Lower Bounds for Sampling Algorithms for Estimating the Average", *IPL*, Vol. 53, pages 17–25, 1995.
11. L. Carter and M. Wegman, "Universal Classes of Hash Functions", *J. Computer and System Sciences*, Vol. 18, pages 143–154 (1979).
12. B. Chor and O. Goldreich, "On the Power of Two–Point Based Sampling," *Jour. of Complexity*, Vol 5, 1989, pages 96–106.
13. A. Cohen and A. Wigderson, "Dispensers, Deterministic Amplification, and Weak Random Sources", *30th FOCS*, 1989, pages 14–19.
14. O. Gaber and Z. Galil, "Explicit Constructions of Linear Size Superconcentrators", *JCSS*, **22** (1981), pages 407–420.
15. O. Goldreich, R. Impagliazzo, L.A. Levin, R. Venkatesan, and D. Zuckerman, "Security Preserving Amplification of Hardness", *31st FOCS*, pages 318–326, 1990.
16. O. Goldreich and A. Wigderson. Tiny Families of Functions with Random Properties: A Quality–Size Trade–off for Hashing. *Journal of Random structures and Algorithms*, Vol. 11, Nr. 4, December 1997, pages 315–343.
17. S. W. Golomb, *Shift Register Sequences*, Holden-Day, 1967. (Aegean Park Press, Revised edition, 1982.)
18. V. Guruswami, C. Umans, and S. Vadhan. Unbalanced Expanders and Randomness Extractors from Parvaresh-Vardy Codes. *JACM*, Vol. 56 (4), Article No. 20, 2009. Preliminary version in *22nd CCC*, 2007.
19. S. Hoory, N. Linial, and A. Wigderson. *Expander Graphs and their Applications. Bull. AMS*, Vol. 43 (4), pages 439–561, 2006.
20. R. Impagliazzo and D. Zuckerman, "How to Recycle Random Bits", *30th FOCS*, 1989, pages 248–253.
21. N. Kahale, "Eigenvalues and Expansion of Regular Graphs", *Journal of the ACM*, 42(5):1091–1106, September 1995.
22. R.M. Karp, N. Pippinger and M. Sipser, "A Time-Randomness Tradeoff", *AMS Conference on Probabilistic Computational Complexity*, Durham, New Hampshire (1985).
23. A. Lubotzky, R. Phillips, P. Sarnak, "Explicit Expanders and the Ramanujan Conjectures", *Proc. 18th STOC*, 1986, pages 240–246.
24. G.A. Margulis, "Explicit Construction of Concentrators", *Prob. Per. Infor.* 9 (4) (1973), 71–80. (In Russian, English translation in *Problems of Infor. Trans.* (1975), 325–332.)
25. J. Radhakrishnan and A. Ta-Shma: Bounds for Dispersers, Extractors, and Depth-Two Superconcentrators. *SIAM J. Discrete Math.*, Vol. 13 (1), pages 2–24, 2000.
26. O. Reingold, S. Vadhan, and A. Wigderson. Entropy Waves, the Zig-Zag Graph Product, and New Constant-Degree Expanders and Extractors. *ECCC*, TR01-018, 2001. Preliminary version in *41st FOCS*, pages 3–13, 2000.

27. M. Sipser, "Expanders, Randomness or Time vs Space", *Structure in Complexity Theory* (proceedings), 1986.
28. R. Shaltiel. Recent Developments in Explicit Constructions of Extractors. In *Current Trends in Theoretical Computer Science: The Challenge of the New Century, Vol 1: Algorithms and Complexity*, World scietific, 2004. (Editors: G. Paun, G. Rozenberg and A. Salomaa.) Preliminary version in *Bulletin of the EATCS 77*, pages 67–95, 2002.
29. L. Trevisan, "When Hamming meets Euclid: The Approximability of Geometric TSP and MST", *29th STOC*, pages 21–29, 1997.
30. D. Zuckerman. Randomness-Optimal Oblivious Sampling. *Journal of Random Structures and Algorithms*, Vol. 11, Nr. 4, December 1997, pages 345–367. Preliminary version in *28th STOC*, pages 286–295, 1996.

Appendix A: Expanders and Random Walks

This appendix provides more background on expanders than the very minimum that is needed for the main text. On the other hand, there is much more to be learned about this subject (see, e.g., [19]).

A.1 Expanders

An (N, d, λ)-expander is a d-regular graph with N vertices such that the absolute value of all eigenvalues (except the biggest one) of its adjacency matrix is bounded by λ. A (d, λ)-family is an infinite sequence of graphs so that the n^{th} graph is a $(2^n, d, \lambda)$-expander. We say that such a family is *efficiently constructible* if there exists a polynomial-time algorithm that given a vertex, v, in the expander and an index $i \in [d] \stackrel{\text{def}}{=} \{1, ..., d\}$, returns the i^{th} neighbor of v. We first recall that for $d = 16$ and some $\lambda < 16$, efficiently constructible $(16, \lambda)$-families do exist (cf., [14]).[11]

In our applications we use (parameterized) expanders satisfying $\frac{\lambda}{d} < \alpha$ and $d = \text{poly}(1/\alpha)$, where α is an application-specific parameter. Such (parameterized) expanders are also efficiently constructible. For example, we may obtain them by taking paths of length $O(\log(1/\alpha))$ on an expander as above. Specifically, given a parameter $\alpha > 0$, we obtain an efficiently constructible (D, Λ)-family satisfying $\frac{\Lambda}{D} < \alpha$ and $D = \text{poly}(1/\alpha)$ as follows. We start with a constructible $(16, \lambda)$-family, set $k \stackrel{\text{def}}{=} \log_{16/\lambda}(1/\alpha) = O(\log 1/\alpha)$ and consider the paths of length k in each graph. This yields a constructible $(16^k, \lambda^k)$-family, and indeed both $\frac{\lambda^k}{16^k} < \alpha$ and $16^k = \text{poly}(1/\alpha)$ hold.

[11] These are minor technicalities, which can be easily fixed. Firstly, the Gaber–Galil expanders are defined (only) for graph sizes that are perfect squares [14]. This suffices for even n's. For odd n's, we may use a trivial modification such as taking two copies of the graph of size 2^{n-1} and connecting each pair of corresponding vertices. Finally, we add multiple edges so that the degree becomes 16, rather than being 14 for even n's and 15 for odd n's.

Comment: To obtain the best constants in Sections 4 and 5, one may use efficiently constructible Ramanujan Graphs [23]. Furthermore, using Ramanujan Graphs is essential for our proof of the second item of Theorem 4.1 as well as of Lemma 5.3. Ramanujan Graphs satisfy $\lambda \leq 2\sqrt{d-1}$ and so, setting $d = 4/\alpha$, we obtain $\frac{\lambda}{d} < \alpha$, where α is an application-specific parameter. Here some minor technicalities arise since these graphs are given only for certain degrees and certain sizes. Specifically, they can be efficiently constructed for $\frac{1}{2} \cdot q^k \cdot (q^{2k}-1)$ vertices, where q is a prime such that $q \equiv d-1 \equiv 1 \bmod 4$ and $d-1$ is a prime that is a quadratic residue modulo q (cf., [3, Sec. II]). This technical difficulty may be resolved in two ways:

1. Fixing d and ϵ, N, we may find q and k satisfying the foregoing conditions with $\frac{1}{2} \cdot q^k \cdot (q^{2k}-1) \in [(1-\epsilon) \cdot N, N]$, in time polynomial in $1/\epsilon$ (and in $\log N$). This defines a Ramanujan Graph that is adequate for all our applications (since it biases the desired sample in $[N]$ only by ϵ).
2. Fixing d and ϵ, N, we may find q and k satisfying the foregoing conditions with $\frac{1}{2} \cdot q^k \cdot (q^{2k}-1) \in [N, 2N]$, in time polynomial in $\log N$. We may easily modify our applications so that whenever we obtain a vertex not in $[N]$ we just ignore it. One can easily verify that the analysis of the application remains valid.

A.2 The Expander Mixing Lemma

The following lemma is folklore and has appeared in many papers. Loosely speaking, the lemma asserts that expander graphs (for which $d \gg \lambda$) have the property that the fraction of edges between two large sets of vertices approximately equals the product of the densities of these sets. This property is called *mixing*.

Lemma A.2 (Expander Mixing Lemma): *Let $G = (V, E)$ be an expander graph of degree d and λ be an upper bound on the absolute value of all eigenvalues, except the biggest one, of the adjacency matrix of the graph. Then, for every two subsets, $A, B \subseteq V$, it holds*

$$\left| \frac{|(A \times B) \cap E|}{|E|} - \frac{|A|}{|V|} \cdot \frac{|B|}{|V|} \right| \leq \frac{\lambda \sqrt{|A| \cdot |B|}}{d \cdot |V|} < \frac{\lambda}{d}.$$

The lemma (and a proof) appears as Corollary 2.5 in [5, Chap. 9].

A.3 Random walks on Expanders

A fundamental discovery of Ajtai, Komlos, and Szemerédi [1] is that random walks on expander graphs provide a good approximation to repeated independent attempts to hit any arbitrary fixed subset of sufficient density (within the vertex set). The importance of this discovery stems from the fact that a random walk on an expander can be generated using much fewer random coins than required for generating independent samples in the vertex set. Precise formulations of the foregoing discovery were given in [1, 13, 15] culminating in Kahale's optimal analysis [21, Sec. 6].

Theorem A.3 (Expander Random Walk Theorem [21, Cor. 6.1]): *Let $G = (V, E)$ be an expander graph of degree d and λ be an upper bound on the absolute value of all eigenvalues, except the biggest one, of the adjacency matrix of the graph. Let W be a subset of V and $\rho \stackrel{\text{def}}{=} |W|/|V|$. Then, the fraction of random walks* (in G) *of* (edge) *length ℓ that stay within W is at most*

$$\rho \cdot \left(\rho + (1-\rho)\cdot\frac{\lambda}{d}\right)^{\ell} \tag{10}$$

A more general bound (which is weaker for the above special case) was pointed out to us by Nabil Kahale (personal communication, April 1997):

Theorem A.4 (Expander Random Walk Theorem – general case): *Let $G = (V, E)$, d and λ be as in Theorem A.3. Let $W_0, W_1, ..., W_\ell$ be subsets of V with densities $\rho_0, ..., \rho_\ell$, respectively. Then the fraction of random walks* (in G) *of* (edge) *length ℓ that intersect $W_0 \times W_1 \times \cdots \times W_\ell$ is at most*

$$\sqrt{\rho_0\rho_\ell}\cdot\prod_{i=1}^{\ell}\sqrt{\rho_i + (1-\rho_i)\cdot\left(\frac{\lambda}{d}\right)^2} \tag{11}$$

Theorem A.4 improves over a previous bound of [7] (see [8]). Comments regarding the proofs of both theorems follow.

On the proofs of Theorems A.3 and A.4. The basic idea is viewing events occuring during the random walk as an evolution of a corresponding probability vector under suitable transformations. The transformations correspond to taking a random step in G and to passing through a "sieve" that keeps only the entries that correspond to the current set W. The key observation is that the first transformation shrinks the component that is orthogonal to the uniform distribution, whereas the second transformation shrinks the component that is in the direction of the uniform distribution. Details follow.

Let A be a matrix representing the random walk on G (i.e., A is the adjacency matrix of G divided by the degree, d). Let $\bar{\lambda}$ denote the absolute value of the second largest eigenvalue of A (i.e., $\bar{\lambda} \stackrel{\text{def}}{=} \lambda/d$). Let P (resp., P_i) be a 0-1 matrix that has 1-entries only on its diagonal such that entry (j, j) is set to 1 if and only if $j \in W$ (resp., $j \in W_i$). Then, we are interested in the vector obtained when applying $(PA)^\ell$ (resp., $P_\ell A \cdots P_1 A$) to the vector representing the uniform distribution; that is, the probability that we are interested in is the sum of the component of the resulting vector.

The best bounds are obtained by applying the following technical lemma, which refer to the effect of a single PA application. For any n-by-n stochastic matrix M, we let $\|M\|$ denote the norm of M defined as the maximum of $\|Mx\|$ taken over all normal vectors x (i.e., $x \in \mathbb{R}^n$ with $\|x\| = 1$), where $\|x\|$ denote the Euclidean norm of $x \in \mathbb{R}^n$.

Lemma A.5 ([21, Lem. 3.2] restated): *Let M be a symmetric stochastic matrix and let δ denote the absolute value of the second largest eigenvalue of M. Let P be a 0-1 matrix that has 1's only on the diagonal and let ρ be the fraction of 1's on the diagonal. Then, $\|PMP\| \leq \rho + (1-\rho)\cdot\delta$.*

A proof of a weaker bound is presented below.

Proof of Theorem A.3: Let $u \in \mathbb{R}^n$ be the vector representing the uniform distribution over $V \equiv \{1,...,n\}$ (i.e., $u = (n^{-1},...,n^{-1})$). Let P be a 0-1 matrix such that the only 1-entries are in entries (i,i) with $i \in W$. Thus, the probability that a random walk of length ℓ stays within W is the sum of the entries of the vector

$$x \stackrel{\text{def}}{=} (PA)^{\ell} Pu. \tag{12}$$

In other words, denoting by $\|x\|_1$ the L_1 norm of x, we are interested in an upper bound on $\|x\|_1$. Since x has at most ρn non-zero entries (i.e., $x = Px'$ for some x'), we have $\|x\|_1 \leq \sqrt{\rho n}\cdot\|x\|$. Invoking Lemma A.5 we get

$$\begin{aligned}\|x\|_1 &\leq \sqrt{\rho n}\cdot\|(PA)^{\ell}Pu\| \\ &\leq \sqrt{\rho n}\cdot\|PAP\|^{\ell}\cdot\|Pu\| \\ &\leq \sqrt{\rho n}\cdot\left(\rho+(1-\rho)\cdot\bar{\lambda}\right)^{\ell}\cdot\sqrt{\rho/n}\end{aligned}$$

and the theorem follows. ∎

Proof of Theorem A.4: Using the same argument, we need to upper bound the L_1 norm of x given by

$$x \stackrel{\text{def}}{=} P_{\ell}A\cdots P_1AP_0u. \tag{13}$$

We observe that $\|P_jA\| = \sqrt{\|P_jA^2P_j\|}$ and use Lemma A.5 to obtain $\|P_jA^2P_j\| \leq \rho_j + (1-\rho_j)\cdot\bar{\lambda}^2$. Thus, we have

$$\begin{aligned}\|x\|_1 &\leq \sqrt{\rho_{\ell}n}\cdot\|P_{\ell}A\cdots P_1AP_0u\| \\ &\leq \sqrt{\rho_{\ell}n}\cdot\prod_{j=1}^{\ell}\|P_jA\|\cdot\|P_0u\| \\ &\leq \sqrt{\rho_{\ell}n}\cdot\prod_{j=1}^{\ell}\sqrt{\rho_j+(1-\rho_j)\cdot\bar{\lambda}^2}\cdot\sqrt{\rho_0/n}\end{aligned}$$

and the theorem follows. ∎

Proof of a weak version of Lemma A.5. Rather than proving that $\|PMP\| \leq \rho + (1-\rho)\cdot\delta$, we shall only prove that $\|PMP\| \leq \|PM\| \leq \sqrt{\rho+\delta^2}$. That is, we shall prove that, for every z, it holds that $\|PMz\| \leq (\rho+\delta^2)^{1/2}\cdot\|z\|$. Intuitively, M shrinks the component of z that is orthogonal to the uniform vector u,

whereas P shrinks the component of z that is in the direction of u. Specifically, we decompose $z = z_1 + z_2$ such that z_1 is the projection of z on u and z_2 is the component orthogonal to u. Then, using the triangle inequality and other obvious facts (which imply $\|PMz_1\| = \|Pz_1\|$ and $\|PMz_2\| \leq \|Mz_2\|$), we have

$$\begin{aligned}\|PMz_1 + PMz_2\| &\leq \|PMz_1\| + \|PMz_2\| \\ &\leq \|Pz_1\| + \|Mz_2\| \\ &\leq \sqrt{\rho_i} \cdot \|z_1\| + \delta \cdot \|z_2\|\end{aligned}$$

where the last inequality uses the fact that P shrinks any uniform vector by eliminating $1-\rho_i$ of its elements, whereas M shrinks the length of any eigenvector except u by a factor of at least δ. Using the Cauchy-Schwartz inequality[12], we get

$$\begin{aligned}\|PMz\| &\leq \sqrt{\rho_i + \delta^2} \cdot \sqrt{\|z_1\|^2 + \|z_2\|^2} \\ &= \sqrt{\rho_i + \delta^2} \cdot \|z\|,\end{aligned}$$

where the equality is due to the fact that z_1 is orthogonal to z_2. ■

Appendix B: Analyzing the Toeplitz Matrix Construction

For every $i \neq j$ and $a, b \in \mathrm{GF}(2)^n$, we have

$$\begin{aligned}\Pr_{T,u}\begin{bmatrix}e_i = a \\ e_j = b\end{bmatrix} &= \Pr_{T,u}\left[e_i = a | e_i \oplus e_j = a \oplus b\right] \cdot \Pr_{T,u}\left[e_i \oplus e_j = a \oplus b\right] \\ &= \Pr_{T,u}\left[Tv_i + u = a | Tw = c\right] \cdot \Pr_T\left[Tw = c\right],\end{aligned}$$

where $w = v_i \oplus v_j \neq 0^n$ and $c = a \oplus b$. Clearly, for any $c \in \mathrm{GF}(2)^n$ and any T':

$$\begin{aligned}\Pr_{T,u}[Tv_i + u = a | Tw = c] &= \Pr_u[T'v_i + u = a] \\ &= 2^{-n}\end{aligned}$$

It is thus left to show that, for any $w \neq 0^n$, when T is a uniformly chosen Toeplitz matrix, the vector Tw is uniformly distributed over $\mathrm{GF}(2)^n$. It may help to consider first the distribution of Mw, where M is a uniformly distributed n-by-n matrix. In this case Mw is merely the sum of several (not zero) uniformly and independently chosen column vectors, and so is uniformly distributed over $\mathrm{GF}(2)^n$. The argument regarding a uniformly chosen Toeplitz matrix is slightly more involved.

Let f be the first non-zero entry of $w = (w_1, ..., w_n) \neq 0^n$ (i.e., $w_1 = \cdots = w_{f-1} = 0$ and $w_f = 1$). We make the mental experiment of selecting $T = (t_{i,j})$, by uniformly selecting elements determining T as follows. First we uniformly

[12] That is, we get $\sqrt{\rho_i}\|z_1\| + \delta\|z_2\| \leq \sqrt{\rho_i + \delta^2} \cdot \sqrt{\|z_1\|^2 + \|z_2\|^2}$, by using $\sum_{i=1}^n a_i \cdot b_i \leq \left(\sum_{i=1}^n {a_i}^2\right)^{1/2} \cdot \left(\sum_{i=1}^n {b_i}^2\right)^{1/2}$, with $n = 2$, $a_1 = \sqrt{\rho_i}$, $b_1 = \|z_1\|$, etc.

and independently select $t_{1,n}, ..., t_{1,f}$. Next, we select $t_{2,f}, ..., t_{n,f}$ (here it is important to select $t_{j,f}$ before $t_{j+1,f}$). Finally, we select $t_{n,f-1}, ..., t_{n,1}$. Clearly, this determines a uniformly chosen Toeplitz matrix, denoted T. We conclude by showing that each of the bits of Tw is uniformly distributed given the previous bits. To prove the claim for the j^{th} bit of Tw, consider the time by which $t_{1,n}, ..., t_{1,f}, ..., t_{j-1,f}$ were determined. Note that these determine the first $j-1$ bits of Tw. The key observation is that the value of the j^{th} bit of Tw is a linear combination of the above determined values XORed with the still undetermined $t_{j,f}$. (Here we use the hypothesis that $w_1 = \cdots = w_{f-1} = 0$ and $w_f = 1$.) Thus, uniformly selecting $t_{j,f}$ makes the j^{th} bit of Tw be uniformly distributed given the past. ■

Appendix C: The Hitting problem

The hitting problem is a one-sided version of the Boolean sampling problem. Given parameters n (length), ϵ (density) and δ (error), and oracle access to any function $\sigma : \{0,1\}^n \to \{0,1\}$ such that $|\{x : f(x)=1\}| \geq \epsilon 2^n$, the task to find a string that is mapped to 1. That is:

Definition C.1 (hitter): *A* hitter *is a randomized algorithm that on input parameters n, ϵ and δ, and oracle access to* any *function $\sigma:\{0,1\}^n \to \{0,1\}$, such that $|\{x : f(x)=1\}| \geq \epsilon 2^n$, satisfies*

$$\Pr[\sigma(\text{hitter}^\sigma(n,\epsilon,\delta)) = 1] \;>\; 1-\delta.$$

Observe that, on input parameters n, ϵ and δ, any sampler must be able to distinguish the all-zero function from any function $\sigma:\{0,1\}^n \to \{0,1\}$ such that $|\{x : f(x)=1\}| \geq 2\epsilon 2^n$. Thus, in the latter case, the sampler must obtain (with probability at least $1-\delta$) the value 1 for at least one of its queries, and outputting such a query satisfies the requirement for a hitter (w.r.t parameters n, 2ϵ and δ).

We note that all results and techniques regarding sampling (presented in the main text), have simpler analogous with respect to the hitting problem. In fact, this appendix may be read as a warm-up towards the main text.

C.1 The Information Theoretic Perspective

Analogously to the Naive Sampler, we have the Naive Hitter that *independently* selects $m \stackrel{\text{def}}{=} \frac{\ln(1/\delta)}{\epsilon}$ uniformly distributed sample points and queries the oracle on each. Clearly, the probability that the hitter fails to sample a point of value 1 is at most $(1-\epsilon)^m = \delta$. The complexities of this hitter are as follows

- Sample Complexity: $m \stackrel{\text{def}}{=} \frac{\ln(1/\delta)}{\epsilon} = \Theta(\frac{log(1/\delta)}{\epsilon})$.
- Randomness Complexity: $m \cdot n = \Theta(\frac{log(1/\delta)}{\epsilon} \cdot n)$.
- Computational Complexity: Indeed efficient.

It is easy to prove that the Naive Hitter is sample-optimal. That is:

Theorem C.2 (sample complexity lower bound): *Any hitter has sample complexity bounded below by*

$$\min\left\{2^{n-O(1)}, \frac{\ln(1/2\delta)}{2\epsilon}\right\}$$

provided $\epsilon \leq \frac{1}{8}$.

Proof Sketch: Let A be a hitter with sample complexity $m = m(n, \epsilon, \delta)$ and let σ be a function selected at random by setting its value independently on each argument such that $\Pr(\sigma(x)=1) = 1.5\epsilon$. Then,

$$\Pr_\sigma[\sigma(A^\sigma(n, \epsilon, \delta)) \neq 1] \;=\; (1 - 1.5\epsilon)^m,$$

where the probability is taken over the choice of σ and the internal coin tosses of A. On the other hand, using a Multiplicative Chernoff Bound:

$$\Pr_\sigma[|\{x : \sigma(x)=1\}| < \epsilon 2^n] = 2\exp(-\Omega(\epsilon 2^n)).$$

We may assume that $\Omega(\epsilon 2^n) > \log_2(1/\delta)$ and so the probability that σ has at least ϵ fraction of 1's and yet algorithm A fails is at least $(1 - 1.5\epsilon)^m - \delta > \delta$, unless $m > \frac{\ln(1/2\delta)}{\ln(1-1.5\epsilon)} > \frac{\ln(1/2\delta)}{2\epsilon}$. ■

Theorem C.3 (randomness complexity lower bound): *Let* $s : \mathbb{N} \times [0,1]^2 \to \mathbb{R}$. *Any sampler that has sample complexity at most* $s(n, \epsilon, \delta)$, *has randomness complexity at least*

$$r > n - \log_2 s(n, \epsilon, \delta) + \log_2((1 - \epsilon)/\delta).$$

Proof Sketch: Let A be a hitter with sample complexity $s = s(n, \epsilon, \delta)$, and randomness complexity $r = r(n, \epsilon, \delta)$. Consider any subset of $\delta 2^r$ possible sequence of coin tosses for A and all $\delta 2^r \cdot s$ points that are queried at any of these coin-sequences. We argue that $\delta 2^r \cdot s > (1 - \epsilon)2^n$ must hold, or else there exists a function σ that evaluates to 0 on each of these points and to 1 otherwise (contradicting the requirement that this function be "hit" with probability at least $1 - \delta$). Thus, $r > n + \log_2(1 - \epsilon) - \log_2 s + \log_2(1/\delta)$. ■

C.2 The Pairwise-Independent Hitter

Using a pairwise-independent sequence of uniformly distributed sample points rather than a totally independent one, we obtain the pairwise-independent hitter. Here we set $m \stackrel{\text{def}}{=} \frac{1-\epsilon}{\delta\epsilon}$. Letting ζ_i represent the σ-value of the i^{th} sample point,

considering only σ's with an ϵ-fraction of 1-values,[13] and using Chebyshev's Inequality we have

$$\begin{aligned}\Pr\left[\sum_{i=1}^{m}\zeta_i = 0\right] &\leq \Pr\left[\left|m\epsilon - \sum_{i=1}^{m}\zeta_i\right| \geq \epsilon m\right] \\ &\leq \frac{m\cdot(1-\epsilon)\epsilon}{(\epsilon m)^2} \\ &= \delta.\end{aligned}$$

Recalling that we can generate $2^n - 1$ pairwise-independent samples using $2n$ coins, the pairwise-independent hitter achieves

- Sample Complexity: $\frac{1}{\delta\epsilon}$ (reasonable for constant δ).
- Randomness Complexity: $2n$
- Computational Complexity: Indeed efficient.

C.3 The combined Hitter

Our goal here is to decrease the sample complexity of the Pairwise-Independent Hitter while essentially maintaining its randomness complexity. To motivate the new construction we first consider an oversimplified version of it.

Combined Hitter (oversimplified): On input parameters n, ϵ and δ, set $m \stackrel{\text{def}}{=} \frac{2}{\epsilon}$ and $\ell \stackrel{\text{def}}{=} \log_2(1/\delta)$, generate ℓ independent m-element sequences, each being a sequence of m *pairwise-independently and uniformly distributed* strings in $\{0,1\}^n$. Denote the sample points in the i^{th} sequence by $s^i_1, ..., s^i_m$. We merely try all these $\ell \cdot m$ samples as hitting points. Clearly, for each $i = 1, ..., \ell$,

$$\Pr[(\forall j \in \{1, .., m\})\ \sigma(s^i_j)=0] < \frac{1}{2}$$

and so the probability that none of these s^i_j "hits σ" is at most $0.5^\ell = \delta$. Thus, the oversimplified version described above is indeed a hitter and has the following complexities:

- Sample Complexity: $\ell \cdot m = O(\frac{\log(1/\delta)}{\epsilon})$.
- Randomness Complexity: $\ell \cdot O(n) = O(n \cdot \log(1/\delta))$.
- Computational Complexity: Indeed efficient.

Thus, the sample complexity is optimal (upto a constant factor), but the randomness complexity is higher than what we aim for. To reduce the randomness complexity, we use the same approach as above, but take dependent sequences

[13] Considering only σ's with *exactly* an ϵ-fraction of 1-values implies that $\text{Var}[\zeta_i] = (1-\epsilon)\epsilon$. Needless to say, if the hitter works well for all these functions, then it works well for all functions having *at least* an ϵ-fraction of 1-values.

rather than independent ones. The dependency we use is such that essentially preserves the probabilistic behavior of independent choices. Specifically, we use random walks on expander graphs (cf., Appendix A) to generate a sequence of ℓ "seeds" each of length $O(n)$. Each seed is used to generate a sequence of m pairwise independent elements in $\{0,1\}^n$, as above. Thus, we obtain:

Corollary C.4 (The Combined Hitter): *There exists an efficient hitter with*

- Sample Complexity: $O(\frac{\log(1/\delta)}{\epsilon})$.
- Randomness Complexity: $2n + O(\log(1/\delta))$.

Furthermore, we can obtain randomness complexity $2n + (2 + o(1)) \cdot \log_2(1/\delta))$.

Proof Sketch: We use an explicit construction of expander graphs with vertex set $\{0,1\}^{2n}$, degree d and second eigenvalue λ so that $\lambda/d < 0.1$. We consider a random walk of (edge) length $\ell - 1 = \log_2(1/\delta)$ on this expander, and use each of the ℓ vertices along the path as random coins for the Pairwise-Independent Hitter, which in turn makes $m \stackrel{\text{def}}{=} \epsilon/3$ trials. To analyze the performance of the resulting algorithm, we let W denote the set of coin tosses (for the basic hitter) on which the basic hitter fails to output a point that evaluates to 1. By the hypothesis, $\frac{|W|}{2^{2n}} \leq 1/3$, and using Theorem A.3, the probability that all vertices of a random path reside in W is bounded above by $(0.34 + 0.1)^\ell < \delta$. The furthermore clause follows by using a Ramanujan Graph and an argument as in the proof of Item 2 of Theorem 4.1. ■

C.4 The Expander Hitter

Our goal here is to decrease the randomness complexity of hitters from $2n + O(\log(1/\delta))$ to $n + O(\log(1/\delta))$, while preserving the sample complexity of $O(\epsilon^{-1}\log(1/\delta))$. The first step is to get an analogous improvement with respect to the Pairwise-Independent Hitter (which has sample complexity $O(1/\delta\epsilon)$).

We use a Ramanujan Graph of degree $d = O(1/\epsilon\delta)$ and vertex-set $\{0,1\}^n$. The hitter uniformly selects a vertex in the graph and use its neighbors as a sample. Suppose we try to hit a 1-value of a function σ and let $S \stackrel{\text{def}}{=} \{u : \sigma(u) = 1\}$. Let $B \stackrel{\text{def}}{=} \{v : \mathrm{N}(v) \cap S = \emptyset\}$ be the set of bad vertices (i.e., choosing any of these results in not finding a preimage of 1). Using the Expander Mixing Lemma we have

$$\begin{aligned}\rho(B)\rho(S) &= \left|\frac{|(B \times S) \cap E|}{|E|} - \rho(B)\rho(S)\right| \\ &\leq \frac{\lambda}{d} \cdot \sqrt{\rho(B)\rho(S)}\end{aligned}$$

Hence, $\rho(B)\rho(S) \leq (\lambda/d)^2 = \epsilon\delta$ and using $\rho(S) \geq \epsilon$ we get $\rho(B) \leq \delta$. The complexities of this hitter are as follows:

- Sample Complexity: $O(\frac{1}{\delta\epsilon})$

- Randomness Complexity: n
- Computational Complexity: Indeed efficient.

Adapting the argument in the proof of Corollary C.4, we obtain

Corollary C.5 (The Combined Hitter, revisited): *There exists an efficient hitter with*

- Sample Complexity: $O(\frac{\log(1/\delta)}{\epsilon})$.
- Randomness Complexity: $n + (2 + o(1)) \cdot \log_2(1/\delta))$.