Electronic Colloquium on Computational Complexity, Report No. 78 (2019)

Pseudo-Mixing Time of Random Walks

Itai Benjamini* Oded Goldreich†

June 1, 2019

Abstract

We introduce the notion of pseudo-mixing time of a graph defined as the number of steps in a random walk that suffices for generating a vertex that looks random to any polynomial-time observer, where, in addition to the tested vertex, the observer is also provided with oracle access to the incidence function of the graph.

Assuming the existence of one-way functions, we show that the pseudo-mixing time of a graph can be much smaller than its mixing time. Specifically, we present bounded-degree N-vertex Cayley graphs that have pseudo-mixing time t for any $t(N) = \omega(\log \log N)$. Furthermore, the vertices of these graphs can be represented by string of length $2 \log_2 N$, and the incidence function of these graphs can be computed by Boolean circuits of size $\text{poly}(\log N)$.

Keywords: Random walks, Mixing time, Cayley Graphs, Pseudorandomness, One-way functions, pseudorandom permutations.

Contents

1 Introduction **1**

2 The main result and its proof **3**

3 Additional comments **6**

*Faculty of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel. Email: itai.benjamini@gmail.com.

†Faculty of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel. Email: oded.goldreich@weizmann.ac.il. Partially supported by the Israel Science Foundation (grant No. 1146/18).

ISSN 1433-8092

1 Introduction

A popular way to sample a huge "structured" set that is endowed with a group structure is to start at a fixed element of the set, which is typically easy to find, and take a random walk on the corresponding Cayley graph. If the length of the random walk exceeds the graph's mixing time, then the end-point of the walk is almost uniformly distributed in the corresponding set.

A couple of comments are in place. First, the aforementioned sampling procedure is beneficial only when the original set S is structured in the sense that its elements can be represented by bit-strings of a certain length, denoted ℓ, but not all ℓ-bit strings are elements of S. Furthermore, S may occupy only a negligible fraction of $\{0,1\}^\ell$ (i.e., $|S| < \mu(\ell) \cdot 2^\ell$, where $\mu : \mathbb{N} \to [0,1]$ is a negligible function (e.g., tends to zero faster than the reciprocal of any polynomial)). In such cases, it is infeasible to sample S (almost) uniformly by selecting uniformly few ℓ-bit long strings and taking the first string that falls in S, and so one needs an alternative.

Second, the set S should be endowed with a feasible group operation, and the group should be generated be a small number of generators that are easy to find (along with their inverses). If this is that case, then we can sample S almost uniformly by taking a sufficiently long random walk on the corresponding Cayley graph, starting at the vertex that corresponds to the identity element (which we assume, without loss of generalization, to be represented by 0^ℓ).[1] Of course, the length of the random walk should slightly exceed the graph's mixing time (i.e., the minimum t such that the total variation distance between the end-vertex of a t-step random walk and a uniformly distributed vertex is negligible).

The requirement that the random walk be longer than the mixing time of the graph is aimed at obtaining a distribution that is *statistically close to the uniform distribution over the set of vertices*. However, for actual applications, which may be modeled as efficient procedures that run in time that is polynomial in the length of the description of a vertex in the graph, it suffices to require that *it is infeasible to distinguish the distribution of the end-point of the walk from a uniformly distributed vertex*. Indeed, here we adopt the notion of computational indistinguishability, which underlies much of modern cryptography and the computational notion of pseudorandomness (cf. [3, Apdx. C] and [3, Chap. 8], resp.), and consider the following notion of "pseudo-mixing time" (which is formulated, as usual in the theory of computation, using asymptotic terms).

Definition 1 (pseudo-mixing time, a naive definition): *For a constant d, let $\{G_\ell = (V_\ell, E_\ell)\}_{\ell \in \mathbb{N}}$ be a sequence of d-regular graphs such that $0^\ell \in V_\ell \subseteq \{0,1\}^\ell$. For a function $t : \mathbb{N} \to \mathbb{N}$, the graphs $\{G_\ell\}_{\ell \in \mathbb{N}}$ have* pseudo-mixing time *at most t if for every probabilistic polynomial-time algorithm A it holds that*

$$|\Pr[A(W_t(G_\ell))=1] - \Pr[A(U(G_\ell))=1]| = \mathtt{negl}(\ell),$$

where $W_t(G_\ell)$ denotes the distribution of the end-point of a $t(|V_\ell|)$-long random walk on G_ℓ starting at 0^ℓ, and $U(G_\ell)$ denotes the uniform distribution over V_ℓ. Indeed, `negl` *denotes a negligible function* (i.e., one that vanishes faster than $1/p$, for any positive polynomial p).

Clearly, if the total variation distance between $W_t(G_ell)$ and $U(G_\ell)$ is negligible (in terms of ℓ), then G_ℓ has pseudo-mixing time at most t. The point is that this sufficient condition is not necessary; as we shall see, G_ℓ may have pseudo-mixing time at most t also in case that the total variation distance between $W_t(G_ell)$ and $U(G_\ell)$ is large (say, larger than 0.99).

Definition 1 is titled "naive" because algorithm A, which represents an observer that examines (or uses) the sampled vertex, does not get (or use) any auxiliary information about the graph G_ℓ. In contrast, the motivating discussion referred to the case that one can take a random walk on the graph.

[1] Given an arbitrary group $(S, *)$, where $S \subseteq \{0,1\}^\ell$ and i representing the identity element, consider the group $(S \oplus i, \diamond)$ such that $x \diamond y = ((x \oplus i) * (y \oplus i)) \oplus i$, where $\oplus$ denotes the bit-by-bit exclusive-or of bit strings.

Hence, it is natural to augment the (efficient) observer by providing it with a device that lists the neighbors of any vertex of its choice. In other words, we provide the observer with oracle access to the incidence function of the graph; this oracle answers the query (v, i) with the i^{th} neighbor of vertex v in the graph. Indeed, such a device constitutes a representation of the graph, and we are most interested in the case that this representation is succinct (i.e., the incidence function can be computed by an efficient algorithm (given some short auxiliary infomation) or by a small Boolean circuit). This leads to the following definition.

Definition 2 (succinct representation of graphs and pseudo-mixing time): *For a constant d, let $\{G_\ell = (V_\ell, E_\ell)\}_{\ell\in\mathbb{N}}$ be a sequence of d-regular graphs such that $0^\ell \in V_\ell \subseteq \{0,1\}^\ell$. The graph G_ℓ is represented by its incidence function $g_\ell : V_\ell \times [d] \to V_\ell$, where $g_\ell(v, i)$ is the i^{th} neighbor of v in G_ℓ.*

Succinct representation: *The graphs $\{G_\ell\}_{\ell\in\mathbb{N}}$ have a* succinct representation *if there exists a family of $\text{poly}(\ell)$-size Boolean circuits that compute their incidence functions.*

Pseudo-mixing time: *For a function $t : \mathbb{N} \to \mathbb{N}$, the graphs $\{G_\ell\}_{\ell\in\mathbb{N}}$ have* pseudo-mixing time *at most t if for every probabilistic polynomial-time oracle machine M it holds that*

$$|\Pr[M^{g_\ell}(W_t(G_\ell))=1] - \Pr[M^{g_\ell}(U(G_\ell))=1]| = \mathtt{negl}(\ell),$$

where $M^g(x)$ denotes the output of M on input x when making queries to the oracle g, and the other notations (e.g., $W_t(G_\ell)$ and $U(G_\ell)$) *are as in Definition 1.*

We stress that pseudo-mixing refers to observers that are efficient in terms of the length of the description of vertices in the graph (i.e., they run for $\text{poly}(\ell)$-time). Recall that our focus is on the case that the size of V_ℓ is exponential in ℓ. Hence, a polynomial-time machine cannot possibly explore the entire graph G_ℓ. On the other hand, if G_ℓ is rapidly mixing (i.e., its mixing time is $O(\log |V_\ell|)$), then a polynomial-time machine may obtain many samples of $W_t(G_\ell)$ and $U(G_\ell)$ (or rather many samples that are distributed almost as $U(G_\ell)$).[2] At this point, we can state our main result.

Theorem 3 (main result): *Assuming the existence of one-way functions, there exists a sequence of bounded-degree Cayley graphs $\{G_\ell = (V_\ell, E_\ell)\}_{\ell\in\mathbb{N}}$ such that the following properties hold:*

1. *The graphs $\{G_\ell\}_{\ell\in\mathbb{N}}$ have a* succinct representation.
2. *For every $t : \mathbb{N} \to \mathbb{N}$ such that $t(N) = \omega(\log\log N)$, the graphs $\{G_\ell\}_{\ell\in\mathbb{N}}$ have* pseudo-mixing time at most t.
3. The vertex set is hard to sample*: Any probabilistic polynomial-time algorithm that is given input 1^ℓ, fails to output a vertex of the graph other than 0^ℓ, except with negligible probability.*

Furthermore, the graph G_ℓ has $2^{(0.5+o(1))\cdot\ell}$ vertices, and its mixing time is $\Theta(\ell) = \Theta(\log |V_\ell|)$.

We stress that these bounded-degree graphs have size $\exp(\Theta(\ell))$, and so their pseudo-mixing time is only slightly larger than logarithmic in their (standard) mixing time. Indeed, their pseudo-mixing time is only slightly larger than double-logarithmic in their size (i.e., $|V_\ell|$). In contrast, *the pseudo-mixing time of bounded-degree graphs cannot be* (strictly) *double-logarithmic* in $|V_\ell|$, since an observer can explore the $O(\log\log |V_\ell|)$-neighborhood of 0^ℓ in $\text{poly}(\ell)$-time (and so distinguish vertices in this $\text{poly}(\ell)$-sized set from the set of all other $\exp(\Theta(\ell))$ vertices of the graph).

We comment that, as shown in Theorem 7, the existence of one-way functions is essential for the conclusion of Theorem 3. Essentially, the existence of one-way function, which implies the existence of

[2] Here we assume that t does not exceed the mixing time; otherwise, the entire discussion is moot.

pseudorandom generators, allows to construct Cayley graphs G_ℓ with succinct representation in which all vertices (except for 0^ℓ) look random. Of course, once we reach a vertex in G_ℓ by following a walk from the vertex 0^ℓ, we learn the label of this vertex, but the labels of vertices that we did not visit in such walks look random to us. Hence, for $t(N) = \omega(\log\log|V_\ell|)$, both the random variable $W_t(G_\ell)$ and $U(G_\ell)$ hit vertices that were visited by such $\mathrm{poly}(\ell)$-many explorations with negligible probability, and otherwise they look random.

A stronger notion of the pseudo-mixing time. Given that our focus is on Cayley graphs, one may wonder whether our upper bound on the pseudo-mixing time holds also when the observer is given access to the group operation (as well as to the corresponding inverse operation).[3] A necessary condition for such a stronger notion of pseudo-mixing time is that, for every ($\mathrm{poly}(\ell)$-time computable)[4] word $w = w[X]$ over the set of generators and a variable X, if w equals the identity when X is in the support of $W_t(G_\ell)$, then it is trivial (i.e., it equals the identity over the entire group). For starters, we suggest the following problem (where $r(\ell)$ plays the role of $t(\exp(\ell))$ of above).

Open Problem 4 (vanishing over a large ball implies vanishing in the group): *For some $r : \mathbb{N} \to \mathbb{N}$ such that $r(\ell) \in [\omega(\log\ell), o(\ell)]$, is it possible to construct an $\exp(\ell)$-sized group having succinct representation that satisfies the following condition: For every word over the group's generators and a variable X that vanishes over a ball of radious $r(\ell)$ centered at the group's indentity, it holds that the word vanishes over the entire group.*

As hinted above (see also Footnote 4), we are actually interested in words that are computable by arithmetic circuits of size $\mathrm{poly}(\ell)$.

2 The main result and its proof

Theorem 3 is proved by combining the following facts:

1. The elements of any finite group can be relabeled arbitrarily yielding a group that is isomorphic to the original (Proposition 5).
2. Assuming the existence of one-way functions, the foregoing relabeling can be pseudorandom and succinct in the sense of Property 1 of Theorem 3.
3. There are explicit Cayley graphs with $2^{(1+o(1))\cdot\ell/2}$ vertices such that the t-neighborhood of each vertex is a tree.[5]

Indeed, the graphs asserted in Theorem 3 are obtained by starting with a Cayley graph as in Fact 3, and relabeling its vertices as suggested by Steps 1–2. Hence, the pivot of our proof is performing such relabeling.

Proposition 5 (relabeling a group): *Let $(S, *)$ be a group such that $S \subset \{0,1\}^\ell$ and 0^ℓ is the identity element, and let π be a permutation over $\{0,1\}^\ell$ such that $\pi(0^\pi) = 0^\pi$. Then, the set $S_\pi = \{\pi(e) : e \in S\}$ combined with the operation $\diamond_\pi : S_\pi \times S_\pi \to S_\pi$ such that $\diamond_\pi(\alpha,\beta) = \pi(\pi^{-1}(\alpha) * \pi^{-1}(\beta))$ forms a group that is isomorphic to $(S, *)$.*

[3] Note that giving oracle access to the incidence function of the graph is equivalent to giving access to an oracle that performs the group operation only when the second operant is in a fixed set of generators (of the group).

[4] By $T(\ell)$-time computable word, we mean a word that is computable by a (uniform) arithmetic circuit of size $T(\ell)$, where the leaves in this circuit are fed by the fixed constants and variables and the internal gates perform the group operations (of the Cayley graph G_ℓ). Note that such a circuit can compute a word of length exponential in $T(\ell)$, but only a negligible fraction of words of exponential length are computable by such (polynomial-size) circuits.

[5] Actually, a much weaker condition suffices.

Proof: One can readily verify that the group axioms hold.

- For every $\alpha, \beta, \gamma \in S_\pi$ it holds that

$$\begin{aligned}
\diamond_\pi(\diamond_\pi(\alpha,\beta),\gamma) &= \pi(\pi^{-1}(\diamond_\pi(\alpha,\beta)) * \pi^{-1}(\gamma)) \\
&= \pi(\pi^{-1}(\pi(\pi^{-1}(\alpha) * \pi^{-1}(\beta))) * \pi^{-1}(\gamma)) \\
&= \pi((\pi^{-1}(\alpha) * \pi^{-1}(\beta)) * \pi^{-1}(\gamma)) \\
&= \pi(\pi^{-1}(\alpha) * (\pi^{-1}(\beta) * \pi^{-1}(\gamma))) \\
&= \diamond_\pi(\alpha, \diamond_\pi(\beta,\gamma))
\end{aligned}$$

  where the last equality is established analogously to the first three equalities.

- For every $\alpha \in S_\pi$ it holds that $\diamond_\pi(\alpha, 0^\ell) = \pi(\pi^{-1}(\alpha) * 0^\ell) = \alpha$ and $\diamond_\pi(0^\ell, \alpha) = \pi(0^\ell * \pi^{-1}(\alpha)) = \alpha$. Indeed, 0^n is the identity element of the new group.

- Denoting by $\mathtt{inv}(a)$ the inverse of $a \in S$ in the original group, we can verify that $\pi(\mathtt{inv}(\pi^{-1}(\alpha)))$ is the inverse of $\alpha \in S_\pi$ in the new group. Specifically,

$$\begin{aligned}
\diamond_\pi(\alpha, \pi(\mathtt{inv}(\pi^{-1}(\alpha)))) &= \pi(\pi^{-1}(\alpha) * \pi^{-1}(\pi(\mathtt{inv}(\pi^{-1}(\alpha))))) \\
&= \pi(\pi^{-1}(\alpha) * \mathtt{inv}(\pi^{-1}(\alpha))) \\
&= 0^\ell
\end{aligned}$$

  and similarly $\diamond_\pi(\pi(\mathtt{inv}(\pi^{-1}(\alpha))), \alpha) = 0^\ell$.

Lastly, note that π is an isomoprphism of $(S, *)$ to $(S_\pi, \diamond_\pi)$, since $\pi(a * b) = \diamond_\pi(\pi(a), \pi(b))$. ■

We now restated Theorem 3, while recalling that a function is negligible if it tends to zero faster than the reciprocal than any polnomial. We use $\mathtt{negl}$ to denote a generic negligible function.

Theorem 6 (main result, restated): *Assuming the existence of one-way functions, there exists a distribution ensemble of bounded-degree Caley Graphs $\{\mathcal{D}_\ell\}_{\ell\in\mathbb{N}}$ such that the following properties hold:*

1. Succinct representation: *For every $\ell \in \mathbb{N}$, each graph in the support of $\mathcal{D}_\ell$ is a $O(1)$-regular graph with $2^{(0.5+o(1))\cdot\ell}$ vertices. Furthermore, the vertex set is a subset of $\{0,1\}^\ell$ and contains 0^ℓ. These graphs are strongly explicit in the sense that each of these d-regular graphs is represented by an ℓ-bit long string, called its* $\mathtt{seed}$*, and there exists a polynomial-time algorithm that on input a seed s, a vertex v in the graph represented by s, and an index $i \in [d]$, returns the i^{th} neighbor of v in the graph.*

2. Pseudo-mixing time: *For every $t : \mathbb{N} \to \mathbb{N}$ such that $t(N) = \omega(\log\log N)$, any probabilistic polynomial-time oracle machine M that is given oracle access to the incidence function of a graph $G = (V, E)$ selected according to $\mathcal{D}_\ell$ cannot distinguish the uniform distribution over G's vertices from the distribution of the end-vertex in a $t(|V|)$-step random walk on G that starts in 0^ℓ. Furthermore, with probability $1 - \mathtt{negl}(\ell)$ over the choice of G in $\mathcal{D}_\ell$, it holds that*

$$|\Pr[M^G(W_t(G))=1] - \Pr[M^G(U(G))=1]| = \mathtt{negl}(\ell),$$

   where $U(G)$ denotes the uniform distribution on G's vertices, and $W_t(G)$ denotes the distribution of the end-vertex of a $t(|V|)$-step random walk on G that starts at 0^ℓ. Moreover, this holds even if the machine is given $\mathrm{poly}(\ell)$ many samples from $U(G)$.

3. Difficulty of sampling: *Any probabilistic polynomial-time oracle machine M that is given oracle access to the incidence function of a graph G selected according to $\mathcal{D}_\ell$ produces, with probability $1-\mathtt{negl}(\ell)$, an output that is either not a vertex or is a vertex obtained as answers to one of its queries or 0^ℓ. Furhermore, with probability $1-\mathtt{negl}(\ell)$ over the choice of $G=(V,E)$ in $\mathcal{D}_n$, it holds that*

$$\Pr[M^G(1^\ell) \in V \setminus (0^\ell \cup A_M^G)] = \mathtt{negl}(\ell),$$

where A_M^G is the set of answers provided to M's queries during the execution of $M^G(1^\ell)$.

Furthermore, these graphs have mixing time $\Theta(\ell)$.

Theorem 3 follows by fixing an arbitrary typical graph in each distribution $\mathcal{D}_\ell$. The furthermore clause of Property 2 refers to a setting in which the observer may obtain samples of $U(G)$ via a different process (which may be more expensive so that we may prefer obtaining samples of $W_t(G)$ instead, since t is much smaller than G's mixing time). Property 3 asserts that we are in a case of interest in the sense that taking walks on G is the only feasible way to obtain vertices of G (other than 0^ℓ).

Proof: Our starting point is a family of explicit d-regular Cayley Graphs $\{G_\ell=(V_\ell,E_\ell)\}_{\ell\in\mathbb{N}}$ such that $0^\ell \in V_\ell \subset \{0,1\}^\ell$ and $|V_\ell| = 2^{(0.5+o(1))\cdot\ell}$. Furthermore, we assume that $W_t(G_\ell)$ has min-entropy $\omega(\log \ell)$, where the min-entropy of a random variable X equals $\min_x\{\log_2(1/\Pr[X=x])\}$. This certainly holds if G_ℓ is an expander graph.

Next, we consider the distribution $\mathcal{D}_\ell$ obtained by selecting a permuation π over $\{0,1\}^\ell$ that is pseudorandom subject to $\pi(0^\ell)=0^\ell$. It is instructive to view π as selected as follows: First, we select a pseudorandom permuation ϕ of $\{0,1\}^\ell$, and then we let $\pi(x)=\phi(x)$ if $x \in \{0,1\}^\ell \setminus \{\phi^{-1}(0^\ell), 0^\ell\}$, and $\pi(\phi^{-1}(0^\ell)) = \phi(0^\ell)$ and $\pi(0^\ell)=0^\ell$ otherwise. Recall that pseudorandom permutations can be constructed based on any one-way function (cf. [5, 4, 6]). These permutations of $\{0,1\}^\ell$ are represented by ℓ-bit long strings, and are coupled with efficient algorithms for evaluating the permutation and its inverse.

Applying Proposition 5 to the group $(V_\ell,*)$ that underlies the Cayley graph G_ℓ, we obtain a group $(S_\pi,\diamond_\pi)$ that is isomorphic to $(V_\ell,*)$, and it follows that $\pi(G_\ell)=(S_\pi,E_\pi)$, where $E_\pi=\{\{\pi(u),\pi(v)\} : \{u,v\}\in E_\ell\}$, is a Cayley Graph of the group $(S_\pi,\diamond_\pi)$. Specifically, letting $g: V_\ell \times [d] \to V_\ell$ denote the incidence function of G_ℓ, the incidence function of $\pi(G_\ell)$, denoted g_π, satisfies $g_\pi(\alpha,i)=\diamond_\pi(\alpha,\pi(g_i))$ (which equals $\pi(\pi^{-1}(\alpha)*g_i)$), where g_i is the i^{th} generator of the set underlying the definition of G_ℓ. (Here and below, we include both the generator and its inverse in the set of generators, so to avoid inverse notations.) This establishes Property 1.

To prove Properties 2 and 3, we analyze an ideal construction in which $\pi: \{0,1\}^\ell \to \{0,1\}^\ell$ is a uniformly distributed permutation that satiosfies $\pi(0^\ell)=0^\ell$; equivalently, ϕ is a totally random permutation. By the pseudoramndomness of the permutations used in the actual construction, if Property 2 (resp., Property 3) holds for the ideal construction, then it holds also for the actual construction, since otherwise we obtain a poly(ℓ)-time oracle machine that distinguishes the truely random permutations from the pseudorandom ones. Hence, we focus on the analysis of the ideal construction.

Starting with Proposition 3, observe that, by making oracle calls to $G'=\pi(G_\ell)$, a machine that runs in poly(ℓ)-time may encounter poly(ℓ) many vertices of G' by taking walks (of various lengths) from the vertex 0^ℓ. As far as such a machine is concerned, the name of each other vertex is uniformly distributed among the remaining $2^\ell - \text{poly}(\ell)$ strings of length ℓ. Hence, if such a machine outputs a string that is neither 0^ℓ nor any of the vertices encounted by it, then this string is a vertex of G' with probability at most $\frac{|V_\ell|}{2^\ell} = \exp(-\Omega(\ell)) = \mathtt{negl}(\ell)$.

Turning to Property 2, note that the relevant machine may encounter vertices of $G'=\pi(G_\ell)$ by taking walks (of various lengths) either from 0^ℓ or from the sample of $U(G')$ or from the test vertex (which is distributed either according to $W_t(G')$ or according to $U(G')$). Each newly encountered vertex looks as being uniformly distributed among the unencountered vertices, and so the difference between

the two cases (regarding the tested vertex) amount to the difference between pattern of collisions that the machine sees, where all walks are oblivious of the names of the encountered vertices (since these are random). Collisions between walks that start at the same vertex do not matter, since these are determined by the corresponding sequence of steps obliviously of the start vertex.[6] Hence, the only collisions that may contribute to distinguishing the two cases are collisions between a walk from the tested vertex and a walk from some other vertex (equiv., a collision between the tested vertex and a walk from some other vertex).[7] The key observation is that both $W_t(G')$ and $U(G')$ have min-entropy $\omega(\log \ell)$; specifically, $\max_v\{\Pr[W_t(G')=v]\} = \exp(-\omega(\log \ell))$ and $\max_v\{\Pr[U(G')=v]\} = \exp(-\Omega(\ell))$. Hence, such a collision occurs with probability at most $\exp(-\omega(\log \ell)) = \mathtt{negl}(\ell)$. It follows that a $\mathrm{poly}(\ell)$-query machine cannot distinguish between $W_t(G')$ or $U(G')$; that is, its distinguishing gap is at most $\mathrm{poly}(\ell) \cdot \mathtt{negl}(\ell) = \mathtt{negl}(\ell)$.

The foregoing analysis refers to what happens in expectation on $\mathcal{D}_\ell$, whereas the two furthermore claims refer to what happens on typical graphs drawn from $\mathcal{D}_\ell$. In the case of Property 3 applying Markov inequality will do, since we actually upper-bounded (by $\frac{|V_\ell|}{2^\ell} + \mathtt{negl}(\ell)$) the probability that the machine outputs a vertex in $\pi(V_\ell) \setminus \{0^\ell\}$ that was not encoutered in its queries. The case of Property 2 requires a slighly more refined argument (since the gap $\Pr[M^G(W_t(G))=1] - \Pr[M^G(U(G))=1]$, for each G, may be either positive or negative).[8] Using the fact that we can (use the sample of $U(G)$ to) approximate both $\Pr[M^G(W_t(G))=1]$ and $\Pr[M^G(U(G))=1]$, we can translate a gap on a non-negligible measure of $\mathcal{D}_\ell$ to a gap in the expectation, which means that the main claim of Property 3 implies its furthermore claim.[9] ■

Digest: Why is Theorem 6 true? Essentially, *a priori* (and with the exception of 0^ℓ), the vertices of the distribution of graphs that we construct look like random ℓ-bit strings. By taking walks from the vertex 0^ℓ (or from the tested or sampled vertices), the observer may discover $\mathrm{poly}(\ell)$ vertices of the graph, but (in both cases) these vertices look random and are unlikely to include the tested vertex. The latter assertion relies on the fact that $\max_v\{\Pr[W_t(G)=v]\}$ is negligible, which requires $t(|V_\ell|) = \omega(\log \ell)$.

3 Additional comments

We first note that our choice to use graphs with $2^{(0.5+o(1))\cdot\ell}$ vertices is quite immaterial. It is merely a natural intermediate point that balanced between having too many vertices and too little vertices.

The pseudorandomness of multiple walks. Using the fact that Property 2 of Theorem 6 refers to a model in which the observer (equiv., potential distinguisher) obtains samples of the uniform distribution over the vertices of the graph, it follows that such observers cannot distinguish multiple samples of the t-step random walk from multiples samples of the graph's vertices. This can be proved

[6] Specifically, suppose that two walks that start at vertex v collide, and that these walks correspond to the sequences of steps $i_1, ..., i_m$ and $j_1, ..., j_n$. Hence, $v \diamond g_{i_1} \diamond \cdots \diamond g_{i_m} = v \diamond g_{j_1} \diamond \cdots \diamond g_{j_n}$, which implies $g_{i_1} \diamond \cdots \diamond g_{i_m} = g_{j_1} \diamond \cdots \diamond g_{j_n}$.

[7] Specifically, suppose that a walk from the test vertex v collides with a walk from vertex w, and that these walks correspond to the sequences of steps $i_1, ..., i_m$ and $j_1, ..., j_n$. Then, $v \diamond g_{i_1} \diamond \cdots \diamond g_{i_m} = w \diamond g_{j_1} \diamond \cdots \diamond g_{j_n}$, which implies $v = w \diamond g_{j_1} \diamond \cdots \diamond g_{j_n} \diamond g_{i_m}^{-1} \diamond \cdots \diamond g_{i_1}^{-1}$.

[8] Hence, even if the absolute value of the gap is large on each specific G, the average gap may be negligible due to cancellations.

[9] An alternative procedure that uses a single sample of $U(G)$ is presented in [1, Prop. 1.1]. It yields a procedure P, which takes an independent sample of $U(G)$, denoted Z, and satisfies

$$\Pr[P^G(W_t(G), Z)=1] - \Pr[P^G(U(G), Z)=1] = (\Pr[M^G(W_t(G))=1] - \Pr[M^G(U(G))=1])^2.$$

In the original presentation the procedure also takes an independent sample of the other distribution (i.e., $W_t(G)$), but in the current setting such a sample can be generated using oracle access to G.

using a hybrid argument (see, e.g., [3, Sec. 8.2.3.3]), and is a special case of a general result that asserts that indistinguishability of one sample (drawn from one of two distributions) implies indistinguishability of multiple samples (which are all drawn from one of two distributions). This generic result presupposes that the (single sample) distinguisher can obtain additional samples of the tester distribution. Clearly, the distinguisher can generate t-step random walks, provided that $t(N) = \mathrm{poly}(\log N)$, and uniformly sampled vertices are available to it by the definition of the model.[10]

On the necessity of one-way functions. A natural question is whether using one-way function in necessary towards establishing Theorem 6. We show that the answer is affirmative, essentially because the existence of efficiently sampleable distribution ensembles that are far apart but are computationally indistinguishable implies the existence of one-way functions [2]. Next, we present a more direct argument (tailored to the current application).

Theorem 7 (one-way functions are necessary for Theorem 6): *The existence of a distribution ensemble of graphs that satisfies Conditions 1 and 2 of Theorem 6 implies the existence of one-way functions*

Proof: Using the hypothesis, we present a "false entropy generator" (as defined in [5]), and use the fact that such a generator implies a pseudorandom generator (see [5]), which in turn implies a one-way function. Details follow.

Fixing a function $t : \mathbb{N} \to \mathbb{N}$ such $t(N) = o(\log N)$, let $t'(\ell) = t(2^{(0.5+o(1))\cdot\ell}) = o(\ell)$. For every ℓ, we consider a function $F : \{0,1\}^\ell \times ([d]^{t'(\ell)})^m \to \{0,1\}^{m\cdot n}$ that takes as input a seed s for a graph in the support of $\mathcal{D}_\ell$ along with the descriptions of $m = O(1)$ random ($t'(\ell)$-step) walks, where each description is an $t'(\ell)$-long sequence over $[d]$, and outputs a sequence of m vertices, $(v_1, v_2, ..., v_m)$, such that v_i is the end-vertex of the i^{th} random walk (on the graph represented by s).

By Property 1 of Theorem 6, the function F is computable in polynomial-time. On the other hand, by Property 2 (and the foregoing comment about the pseudorandomness of multiple walks), it follows that the output of the function is indistinguishable (in polynomial-time) from m uniformly and independently distributed vertices of the graph. The point is that the length of the input to F is $\ell' \stackrel{\text{def}}{=} \ell + m \cdot O(t'(\ell)) = \ell + O(1) \cdot o(\ell) < 2\ell$, whereas the output is computationally indistinguishable from a m-long sequence over $\{0,1\}^\ell$ that has min-entropy at least $k \stackrel{\text{def}}{=} m \cdot \ell/2 > 2\ell'$ (i.e., each possible outcome occurs with probability at most 2^{-k}). Hence, $F : \{0,1\}^{\ell'} \to \{0,1\}m \cdot \ell$ is a "false entropy generator".

To obtain a pseudorandom generator, we may either use a randomness extractor of seed length $o(\ell)$ or a strong randomness extractor of seed length $\mathrm{poly}(\ell)$ (see, e.g., [3, Def. D.8]). Either way, we extract $0.75 \cdot k$ bits (from an input of min-entropy k), while incurring an error of $\exp(-\Omega(k)) = \exp(-\Omega(\ell))$. Applying the extractor on the output of F, we obtain a pseudorandom generator. For example, consider $G(s', s'') = (s'', h_{s''}(F(s'))$, where $h_{s''} : \{0,1\}^{O(|s''|)} \to \{0,1\}^{1.5\cdot|s''|}$ is a pairwise-independent hash function (see, e.g., [3, Apdx. D.2]). ■

Acknowledgments

We are grateful to Tsachik Gelander, Nir Avni, and Chen Meiri for sharing with us their knowledge and conjectures regarding Problem 4.

[10] Alternatively, if the graphs are expanders, then we can generate almost uniformly samples of the graph's vertices by taking an $O(\ell)$-step random walk (using queries to the incidence function of the graph).

References

[1] Z. Brakerski and O. Goldreich. From Absolute Distinguishability to Positive Distinguishability. In *Studies in Complexity and Cryptography*, pages 141–155, Lecture Notes in Computer Science, Vol 6650, Springer, 2011.

[2] O. Goldreich. A Note on Computational Indistinguishability. *Information Processing Letters*, Vol. 34 (6), pages 277–281, 1990.

[3] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[4] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *Journal of the ACM*, Vol. 33, No. 4, pages 792–807, 1986.

[5] J. Hastad, R. Impagliazzo, L.A. Levin and M. Luby. Construction of Pseudorandom Generator from any One-Way Function. A Pseudorandom Generator from any One-way Function. *SIAM Journal on Computing*, Volume 28, Number 4, pages 1364–1396, 1999.

[6] M. Luby and C. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal on Computing*, Vol. 17, 1988, pages 373–386.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il