CS369E: Expanders	April 25, 2005

Lecture 4: Derandomization

Lecturer: Cynthia Dwork *Scribe: Adam Barth & Prahladh Harsha*

In today's (and the next) lecture(s), we will discuss applications of expanders in the context of derandomization. The three applications we will consider are the following:

- Use of random walks on expanders as an error reduction technique for randomized algorithms.
- a pseudo-random generator to fool space bounded machines.
- Derandomized linearity testing

We will discuss the first two applications in this lecture and postpone the linearity testing to the next lecture.

4.1 RP error reduction

Consider an RP algorithm with constant error probability that uses r random bits. We will improve the error probability to 2^{-k} with $r + O(k)$ random bits. Compare this with (1) the brute force k-independent trials, which would require $O(kr)$ random bits to achieve the same error probability and (2) the technique due to Karp, Pippenger and Sipser [KPS] (discussed in Lecture 1) which uses r random bits (i.e., no extra random bits) and achieves an error probability of $1/\text{poly}(r)$. We will use random walks on expanders to reduce the error of RP algorithms. Ajtai, Komlos and Szemeredi first used random walks on expanders in the context of small-space derandomization [AKS]. The proof we present in lecture is due to Impagliazzo and Zuckerman [IZ].

The KPS technique, though great in terms of the number of extra random bits being used is limited by the fact that the running time of the improved algorithm is at least $\text{poly}(1/\delta)$ where δ is the (new reduced) error of the algorithm. Hence, we can reduce the error to at most 1/poly. The technique, discussed today, will further reduce the error to 2^{-k} at the cost of only $O(k)$ extra random bits as opposed $O(rk)$ random bits in the k independent trails, while the algorithm still runs in (randomized) polynomial time.

As in KPS, we will use a d-regular expander with $V = \{0,1\}^r$, thus $|V| = 2^r$ and d is a constant. As in KPS, we will assume that there exists an implicit construction of such expanders in the following sense: given any vertex v and any index i in the range $1 \dots d$ (where d is the degree of the expander), we can in time polynomial in $|v|$ and $|i|$, compute the i^{th}-neighbor of v. The expanders constructions we will discuss later in the course will satisfy such strong properties.

Recall that to find witnesses, KPS began at a random vertex and completely explored all vertices within a ball of radius $O(k)$. Here, we also start at a random vertex but instead of exploring all vertices in a ball, we will walk randomly for k steps and run the original RP algorithm along all vertices along this random walk. Thus the total randomness uses is at most $r + k \log d$ since $\log d$ bits are required to choose a random neighbor.

Clearly, if the input is a NO instance, then this new algorithm will also reject. Our concern is that there might exist YES instances, for which the random walk fails to arrive at even one membership witness. The following theorem shows that this is highly unlikely.

Theorem 4.1 (Hitting Property of Expander Random Walks)**.** *Given a graph $G = (V, E)$ with spectral expansion λ and $B \subset V$, the probability a random walk of length k, starting from a random vertex $r_0 \in_R V$, starts and remains in B is $\leq \left(\mu^2 + \lambda^2\right)^{k/2}$, where $\mu = |B|/|V|$ is the density of B.*

For every RP language L and every $x \in L$, $|W_x|/2^r \geq 3/4$. For our application to derandomization, we set $B = V \setminus W_x$ and obtain the required error-reduction for RP.

Proof. We wish to bound $\Pr_{r_0,\ldots,r_k}\left[r_0, \ldots, r_k \in B\right]$, where r_i is the ith vertex encountered in the random walk on G.

Let A be the normalized adjacency matrix for G, where the vertices of G are ordered such that the first $|B|$ vertices are the elements of B. Fix P to be the projection matrix onto B, that is

$$P = \left(\begin{array}{c|c} I_{|B|\times|B|} & 0_{|B|\times|V\setminus B|} \\ \hline 0_{|V\setminus B|\times|B|} & 0_{|V\setminus B|\times|V\setminus B|} \end{array}\right).$$

In other words, $P_{i,j} = 1$ if $i = j$ and $i, j \in 1, \ldots, |B|$ and is 0 otherwise. For any distribution π on the set of vertices V, note that $\|P\pi\|_1$ is the probability that a vertex chosen according to π is in the set B.

Fix u to be the uniform distribution on V. As mentioned above, $\|Pu\|_1$ is the probability a uniformly randomly selected vertex lies in B, *i.e.* $\|Pu\|_1 = \mu$. Similarly, $\|P(AP)u\|_1$ is the probability r_1 is also in B, *i.e.* the probability both $r_0, r_1 \in B$. By an inductive argument, we seek to bound $\|P(AP)^k u\|_1$. Observe $\|(PAP)^k u\|_1 = \|P(AP)^k u\|_1$, as P is idempotent. It will be more convenient to work with $\|(PAP)^k u\|_1$ than $\|P(AP)^k u\|_1$. We now switch to the L_2 norm and will later return to the L_1 norm.

We first show that a single application of PAP to any vector x reduces its L_2-norm by a factor of $\sqrt{\mu^2 + \lambda^2}$.

Claim 4.2. *$\forall x \in \mathbb{R}^n$, $\|(PAP)x\|_2 = \sqrt{\mu^2 + \lambda^2} \cdot \|x\|_2$.*

Assuming this claim, we complete the proof of the theorem. Applying the claim k times, we obtain

$$\|(PAP)^k x\|_2 \leq \left(\mu^2 + \lambda^2\right)^{k/2} \cdot \|x\|_2.$$

We now return to the L_1 norm.

$$\begin{aligned} \|(PAP)^k u\|_1 &\leq \sqrt{N}\|(PAP)^k u\|_2 && \text{By Cauchy-Schwarz Inequality} \\ &\leq \sqrt{N}\left(\mu^2 + \lambda^2\right)^{k/2}\|u\|_2 \\ &= \left(\mu^2 + \lambda^2\right)^{k/2}. \end{aligned}$$

□

We now prove Claim 4.2.

Proof of Claim 4.2: The main intuition behind the proof is that A reduces the length of component of the vector x that is orthogonal to u while P reduces the length of the component of x along u. Together, they reduce the length of x.

Fix $y = Px$ and split $y = y^{\|} + y^{\perp}$, where $y^{\|}$ is parallel to u and $y^{\perp}$ is perpendicular to u. By the triangle inequality and $Ay^{\|} = y^{\|}$, $\|PAy\|_2 \leq \|Py^{\|}\|_2 + \|PAy^{\perp}\|_2$. Because of the second eigenvalue, $\|Ay^{\perp}\|_2 \leq \lambda\|y^{\perp}\|_2 \leq \lambda\|y\|_2 \leq \lambda\|x\|_2$ (since the length of y is at most that of x, recall that y is the projection of x). Therefore,

$$\|PAy^{\perp}\|_2 \leq \|Ay^{\perp}\|_2 \leq \lambda\|x\|_2 \tag{1}$$

As for the $y^{\|}$ term,

$$y^{\|} = \left(\frac{y \cdot u}{\|u\|_2^2}\right) u, \text{ which implies } y^{\|} = \left(\sum_i y_i\right) u.$$

By observing $y = Px$ and so y has support $|B| = \mu N$.

$$\begin{aligned}
\|y^{\|}\|_2^2 &= \sum_{i=1}^{N} \frac{(\sum_{i=1}^{\mu N} y_i)^2}{N^2} \\
&= \frac{(\sum_{i=1}^{\mu N} y_i)^2}{N} \\
&\leq \frac{\mu N \left(\sum_i y_i^2\right)}{N} \qquad \text{(By Cauchy-Schwarz inequality)} \\
&= \mu\|y\|_2^2
\end{aligned}$$

On the other hand, since $y^{\|} = (\sum_i y_i)\, u$, we have that $(Py^{\|})_j = (\sum_i y_i)/N$ for $j = 1, \ldots, \mu N$ and 0 otherwise. Hence,

$$\begin{aligned}
\|Py^{\|}\|_2^2 &= \sum_{j=1}^{\mu N} \frac{\left(\sum_i y_i\right)^2}{N^2} \\
&= \mu N \frac{\left(\sum_i y_i\right)^2}{N^2} \\
&= \mu\|y^{\|}\|_2^2
\end{aligned}$$

Combining the two we have, $\|Py^{\|}\|_2^2 \leq \mu\|y^{\|}\|_2^2 \leq \mu^2\|y\|_2^2 \leq \mu^2\|x\|_2^2$. Combining equation (1) we have $\|(PAP)x\|_2^2 \leq (\mu^2 + \lambda^2)\|x\|_2^2$. Hence, $\|(PAP)x\|_2 \leq \sqrt{\mu^2 + \lambda^2}\|x\|_2$ □

In this result about RP, we worry about not hitting a witness. For a similar result about BPP, however, we need to show we encounter approximately the correct fraction of appropriate witnesses. For random walks on the complete graph (*i.e.* independent trials), Chernoff bounds tell us the witness fraction will be close to the appropriate ratio with high probability. It is possible to obtain a similar Chernoff bound for random walks on expanders, but we omit the details.

4.2 PRGs for Space Bounded Computation

The general idea of pseudo-random generators is to output a long string from a short, truly random seed such that some restricted class of adversaries (typically time-bounded adversaries) cannot distinguish (with greater than some probability) the long string from a long string of truly random bits. We think of a generator as taking a seed of length $s(n)$ and producing a string of length $f(n)$.

Definition 4.3. *Let M be a randomized Turing machine using, on input w, $f(|w|)$ random bits. The family $\{g_n\}_{n=1}^{\infty}$ of functions $g_n : \{0,1\}^{s(n)} \to \{0,1\}^{f(n)}$ is an ϵ-generator for M if, for all w,*

$$\left| \Pr_{r\in\{0,1\}^{f(|w|)}} \Big[M(w,r) \text{ accepts} \Big] - \Pr_{z\in\{0,1\}^{s(|w|)}} \Big[M(w, g_{|w|}(z)) \text{ accepts} \Big] \right| \le \epsilon$$

Typically, pseudo-random generators are constructed to fool time-bounded adversaries. Here, we consider constructing a generator to fool (randomized) space-bounded adversaries (specifically logspace machines).

4.2.1 Randomized Logspace TM

Before proceeding any further, we have to clarify a point regarding how the random bits are accessed in a randomized space bounded computation (logspace in our case). There are 2 varying definitions of randomized space bounded TMs based on the manner the random bits are accessed

- The TM obtains the random bits as and when required by it.
- The string of random bits is fed as an auxiliary off-line input (on a separate tape) in addition to the regular input to the TM. In this case, the head accessing the random bits on this tape can move back and forth on the tape.

It is to be noted that in the case of randomized time bounded computation it is immaterial which convention we observe. For randomized space bounded computation, we shall consider only TMs of the first kind or equivalently consider TMs of the second kind in which the head on the random tape is restricted in the sense that it can only move right along the tape (i.e., the random tape is one-way read-only tape) Recall that a space S-bounded machine is also effectively time-bounded, for some large time bound 2^S.

4.2.2 Nisan's Generator

In this lecture, we will construct a pseudo-random generator for space bounded machines using expanders. The first such PRG construction was given by Nisan [Nis]. Nisan's construction used hash functions instead of expanders. In this lecture, we give the construction due to Impagliazzo, Nisan and Wigderson [INW], that uses expanders.

Typically, we would like the generator to also run within the space-bound S. If this were the case, we would be able to completely derandomize the randomized space S-bounded machine. Unfortunately, we won't be able to achieve something as strong as that. Instead, we let the generator use more space (specifically S^2 space) than the space S-bounded TM it fools.

We will prove the following theorem in today's lecture.

Theorem 4.4. *There exists a n-space-bounded Turing machine $G : \{0,1\}^{O(\log^2 m)} \to \{0,1\}^m$ such that for all randomized S-space-bounded Turing machines M, G is a $(1/2^S)$-generator for M, where $m = 2^S$ and $n = O(\log^2 m)$.*

4.2.3 Proof of Theorem 4.4

Before going into proving the existence of a PRG as mentioned in Theorem 4.4, we shall first study the structure of the computation tableau of a randomized space S TM and find how this structure can be exploited to reduce the randomness. The computation tableau for a randomized space S TM is as shown in Figure 1, i.e., it is a very thin (width at most S), but possibly very long (length can be as long as 2^S) tableau.

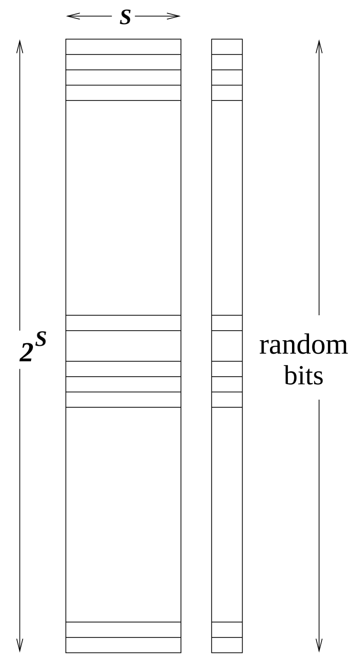

Figure 1: The Computation Tableau of a space S TM

In the original definition of the randomized TM, the computation requires at most 2^S random bits. We will break the tableau into several components (see Figure 2), each of which require exactly R random bits. Thus, there are at most $2^S/R$ such components. For simplicity, we will assume that there are actually 2^S components. R will be typically $\Theta(S)$ for our purposes, but our proof will work even for larger R.

Consider the first two components A_1 and A_2 both of which require random strings r_1 and r_2 respectively each of length R. If r_1 and r_2 are chosen independently, then by definition the 2 components work to give the right results. We would like to choose r_1 and r_2 in such a manner that their behavior is not significantly different from the case when

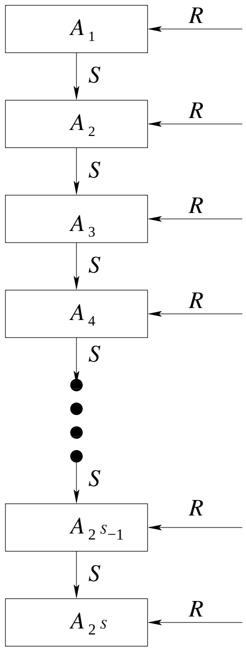

Figure 2: Breaking the tableau into components each requiring R random bits

r_1 and r_2 are chosen independently. We would now use the fact that the computation tableau is very thin (more specifically, at most S bits are communicated between the two components A_1 and A_2) to let us choose r_1 and r_2 in a manner better than independently.

To put things more formally, we have 2 algorithms A_1 and A_2 such that

$$\begin{tikzcd}
A_1 \arrow[d, "b_1"] & x_1 \arrow[l] \\
A_2 \arrow[d] & x_2 \arrow[l] \\
b_2 &
\end{tikzcd}$$

Figure 3: A_1, A_2 with random inputs

- A_1 takes an input x_1 of length r and outputs a string b_1 of length c.
- A_2 takes as input the output b_1 of A_1 and another string x_2 of length r and outputs a string b_2 of length c (see Figure 3).

What we are in search of is a generator that supplies strings x_1 and x_2 in a fashion better than choosing them independently. For notational brevity, given a function g, define functions g^l and g^r such that $g^l(z)$ and $g^r(z)$ denote the left half and the right half of the string $g(z)$ (i.e., $g(z) = g^l(z) \circ g^r(z)$ and $|g^l(z)| = |g^r(z)|$)[1]. See Figure 4.

Definition 4.5. *A function g ($g : \{0,1\}^t \to \{0,1\}^r \times \{0,1\}^r$) is defined to be a $\epsilon-$ generator for communication c if for all functions A_1 and A_2 such that $A_1 : \{0,1\}^r \to \{0,1\}^c$ and $A_2 : \{0,1\}^c \times \{0,1\}^r \to \{0,1\}^c$, we have that*

$$\forall b \qquad \left| \operatorname*{Prob}_{x_1,x_2 \in \{0,1\}^r} [A_2(A_1(x_1), x_2) = b] - \operatorname*{Prob}_{z \in \{0,1\}^t} \left[A_2(A_1(g^l(z)), g^r(z)) = b\right] \right| < \epsilon$$

$$\begin{tikzcd}
A_1 \arrow[d, "b_1"] & g^l(z) \arrow[l] \\
A_2 \arrow[d] & g^r(z) \arrow[l] \\
b_2 &
\end{tikzcd}$$

Figure 4: A_1, A_2 with inputs from generator

For notational convenience, we shall call a $2^{-2c}-$generator for communication c a $c-$generator.

For the present we shall assume the following lemma and present its proof later (in Section 4.3).

Lemma 4.6. *There exists a constant $k > 0$ such that for all r, c, there exists a polynomial time and linear space computable $c-$generator g where g is such that $g : \{0,1\}^{r+kc} \to \{0,1\}^r \times \{0,1\}^r$.*

Given such a c-generator, we can generate pseudo-random strings for every pair of successive components (A_{2i-1} and A_{2i}), such that their behavior is almost similar to the case when pure random strings are fed to all the components. More formally, we let $g_1 : \{0,1\}^{R+kS} \to \{0,1\}^R \times \{0,1\}^R$ be the S-generator guaranteed by lemma 4.6 (i.e., by setting $r = R$ and $c = S$ in the lemma). For every pair of components (A_{2i-1} and A_{2i}), instead of feeding them each with pure random strings of length R, we now take one random string of length $R + kS$, run the generator g_1 on this string and feed the output of the generator to the two components A_{2i-1} and A_{2i} (See Figure 5). By doing so, we require only $2^{S-1} \cdot (R + KS)$ random bits as opposed to $2^S \cdot R$ random bits. Furthermore, each application of the generator g_1 causes an error of at most $1/2^{2S}$ (since g_1 is a S-generator). Hence, the total error incurred is at most $2^{S-1}/2^{2S}$ since we run the generator g_1 at most 2^{S-1} times.

We now have 2^{S-1} components each of which require $R+kS$ random bits (see Figure 5). Furthermore, as before each pair of successive components communicate at most S bits of information. Hence, we can once again apply the generator to reduce the number of random bits required by every pair of components from $R + kS$ each to $R + 2kS$ total. Moreover, we can perform this operation repeatedly till we finally have just one component left. More formally we do the following (also see Figure 6).

[1] $\circ$ denotes the concatenation operator

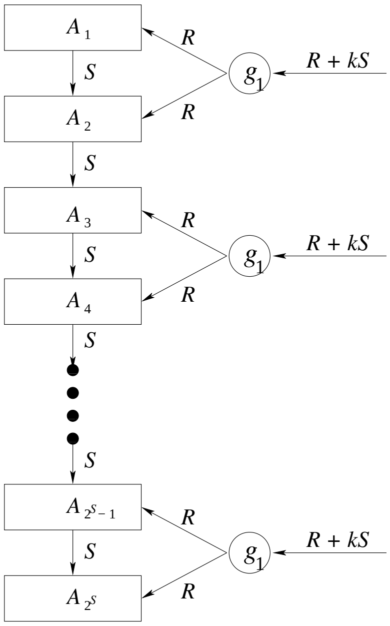

Figure 5: Using S-generator to save randomness

Let $g_i : \{0,1\}^{R+ikS} \to \{0,1\}^{R+(i-1)kS} \times \{0,1\}^{R+(i-1)kS}$ be a $S-$generator for $i = 1, 2, \ldots, S$ as guaranteed by lemma 4.6 (i.e., by setting $r = R + (i-1)kS$ and $c = S$ in the lemma). Define functions $G_i : \{0,1\}^{R+ikS} \to \{0,1\}^{2^i \cdot R}$ for $i = 0, 1, \ldots S$ inductively as follows

$$\begin{aligned} G_0(z) &= z \\ G_i(z) &= G_{i-1}(g_i^l(z)) \circ G_{i-1}(g_i^r(z)) \end{aligned}$$

We shall show that the existence of G_S implies Theorem 4.4. Clearly, by definition of G_S, G_S is a Space(n) TM (i.e., it runs in space $O(R + kS^2)$). We only have to show that G_S fools all randomized space S TMs, which is implied by the following lemma.

Lemma 4.7. *For all* TM*s A that run in space S and in time 2^S, G_S is an $2^{-S}-$generator for A*

Proof. Each application of the generator g_i, for any i, incurs an error of at most $1/2^{2S}$ (as guaranteed by Lemma 4.6). There are at most $2^{S-1} + 2^{S-2} + \cdots + 2 + 1 = 2^S - 1$ applications of the generator g_i (over all i) (see Figure 6). Hence, the maximum error incurred is at most $(2^S - 1)/2^{2S} < 1/2^S$. Thus, proved.

□

4.3 Proof of Lemma 4.6 using Expanders

We prove Lemma 4.6 using the expander mixing lemma

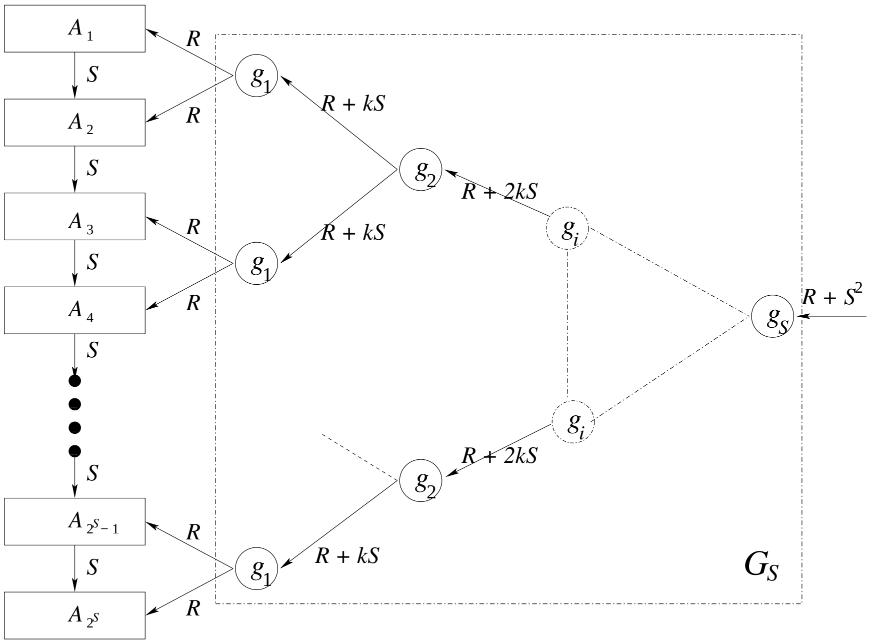

Figure 6: Pseudo-random generator G_S for Space S machines

Lemma 4.8 (Expander Mixing Lemma)**.** *If $G = (V, E)$ is a D-regular graph with spectral expansion λ, then for all sets S and $T \subseteq V$, we have*

$$\left|\frac{e(S,T)}{|E|} - \frac{|S|}{|V|} \cdot \frac{|T|}{|V|}\right| \leq \lambda\sqrt{\frac{|S|}{|V|} \cdot \frac{|T|}{|V|}} \leq \lambda,$$

where $e(S,T)$ denotes the number of edges between the sets S and T.

Lemma 4.6 basically tells us that Figure 3 can be replaced by Figure 4. In other words, the $\epsilon-$generator for communication $r + kc$, g, should construct strings $g^l(z)$ and $g^r(z)$ such that functions A_1 and A_2 are fooled into believing that these strings were random ones. The main idea is to view the set of strings $\{0,1\}^r$ as the vertices of an expander $G = (V, E)$ and choose the strings $g^l(z)$ and $g^r(z)$ to be the endpoints of a random edge of the expander.

The actual construction of the c-generator is as follows: Let $G = (V, E)$ be a $D = 2^{6c}$-regular Ramanujan expander graph on $|V| = 2^r$ vertices. (Recall that a D-regular Ramanujan graph is a D-regular graph with the best possible spectral expansion, namely $\lambda \approx 1/\sqrt{D}$. Such graphs are constructed by Lubotsky, Philips and Sarnark [LPS]). Note that for super-constant c, the above expander has super-constant degree. We can either construct such a Ramanujan expander explicitly or start with a constant degree Ramanujan expander and then take a suitable power of it to increase the degree). The generator $g : \{0,1\}^{r+6c} \to \{0,1\}^r \times \{0,1\}^r$ works as follows: On input $z = (x, i) \in \{0,1\}^r \times \{0,1\}^{d(=6c)}$, output $(g^l(z), g^r(z)) = (x, y)$ where y is the vertex reached by taking the i^{th} edge out of x.
Proof of Lemma 4.6: Let b be any string that is a possible output of the pair of

algorithms (A_1, A_2). For every $b' \in \{0,1\}^c$, define the following:

$$\begin{aligned} S_{b'} &= \left\{x \in \{0,1\}^r | A_1(x) = b'\right\} \\ T_{b'} &= \left\{x \in \{0,1\}^r | A_2(b', x) = b\right\} \end{aligned}$$

i.e., if $x_1 \in S_{b'}$ and $x_2 \in T_{b'}$, $A_1(x_1) = b'$ and $A_2(b', x_2) = b$. Hence,

$$\begin{aligned} \operatorname*{Prob}_{x_1,x_2}[A_2(A_1(x_1), x_2) = b] &= \sum_{b' \in \{0,1\}^c} \operatorname*{Prob}_{x_1,x_2}[x_1 \in S_{b'} \wedge x_2 \in T_{b'}] \\ &= \sum_{b' \in \{0,1\}^c} \frac{|S_{b'}|}{|V|} \cdot \frac{|T_{b'}|}{|V|} \end{aligned}$$

Similarly,

$$\begin{aligned} \operatorname*{Prob}_{z \in \{0,1\}^{r+d}}\left[A_2(A_1(g^l(z)), g^r(z)) = b\right] &= \sum_{b' \in \{0,1\}^c} \operatorname*{Prob}_{(x,i)}[x \in S_{b'} \wedge (\ i\text{th edge out of } x \text{ leads to } T_{b'})] \\ &= \sum_{b' \in \{0,1\}^c} \frac{e(S_{b'}, T_{b'})}{|E|} \end{aligned}$$

Hence,

$$\begin{aligned} &\left|\operatorname*{Prob}_{x_1,x_2}[A_2(A_1(x_1), x_2) = b] - \operatorname*{Prob}_{z \in \{0,1\}^{r+d}}\left[A_2(A_1(g^l(z)), g^r(z)) = b\right]\right| \\ = &\left|\sum_{b' \in 0,1^c} \frac{|S_{b'}|}{|V|} \cdot \frac{|T_{b'}|}{|V|} - \frac{e(S_{b'}, T_{b'})}{|E|}\right| \\ \leq &\sum_{b' \in 0,1^c} \left|\frac{|S_{b'}|}{|V|} \cdot \frac{|T_{b'}|}{|V|} - \frac{e(S_{b'}, T_{b'})}{|E|}\right| \\ \leq &\sum_{b' \in 0,1^c} \lambda \qquad \text{(By Expander Mixing Lemma)} \\ \leq &\ 2^c \lambda \\ \leq &\ 2^c \cdot \frac{1}{\sqrt{D}} \\ = &\ 2^c \cdot \frac{1}{2^{3c}} \\ = &\ \frac{1}{2^{2c}} \end{aligned}$$

Thus, proved. □

References

[AKS] Miklos Ajtai, Janos Komlos, Endre Szemeredi: "Deterministic Simulation in LOGSPACE", STOC 1987: 132-140.

[INW] Russell Impagliazzo, Noam Nisan, Avi Wigderson: “Pseudorandomness for network algorithms”. STOC 1994: 356-364

[IZ] Russell Impagliazzo, David Zuckerman: “How to Recycle Random Bits”, FOCS 1989: 248-253.

[KPS] Richard Karp, Nicholas Pippenger and Michael Sipser: ”A time-randomness tradeoff”, AMS Conference on Probabilistic Computational Complexity, 1985.

[LPS] Alexander Lubotzky, R. Phillips, P. Sarnak: “Ramanujan graphs”. Combinatorica 8(3): 261-277 (1988)

[Nis] Noam Nisan: “Pseudorandom generators for space-bounded computation”. Combinatorica 12(4): 449-461 (1992)