Recycling Random Bits in Parallel

Katalin Friedl*

Shi-Chun Tsai†

Abstract

We show that r pseudo-random bits can be obtained by concatenating t blocks of r/t pseudo-random bits, where the blocks are generated in parallel. This can be considered as a parallel version of [8] – recycling random bits by doing a random walk on an expander. The proof is based on the fact that t simultaneous independent random walks on an expander graph is equivalent to a random walk on a much larger expander. In [8] generating the random walk requires arithmetic operation with long integers. In the parallel version the integers involved are much shorter, and different segments of the pseudo-random bits are generated independently in parallel. Optimal speedup is also achieved.

1 Introduction

It is well known that randomness plays an important role in computations. Randomized algorithms are often simpler and faster than their deterministic versions. The best known example is the primality test [14]. Random bits are an expensive resource. Lately much work has been done on "derandomization" of algorithms. The tricks to save random bits are not only useful in designing efficient algorithms but in other areas as well; for example they also play an important role in the theory of transparent proofs [5, 4].

In some simulations computers can consume thousands of random bits per second. Since the parallel machines are getting more pervasive, the topic of generating pseudo-random numbers in parallel has drawn more and more attention. Because of the increasing demand of randomness in computation, we need to use the random bits not only more economically but also supply the random bits more efficiently.

A randomized algorithm which uses r random bits and has error probability less than $1/2$ can be repeated k times using independent random bits to reduce the error to less than $1/2^k$. This uses rk random bits. Impagliazzo and Zuckerman [8] and independently Cohen and Wigderson [6] showed that the error can be reduced by using fewer random bits, namely $O(r+k)$ random bits are sufficient to achieve error less then $1/2^k$. From a few truly random bits (seed) a much longer sequence of pseudo-random bits is generated, and these pseudo-random bits are used in the computation where random bits are needed. The method to obtain the pseudo-random bits is based on random walks on explicitly given expanders. The idea of using expanders to reduce the error was proposed in [15, 1]. There are different kinds of explicit expander constructions [12, 13, 7, 10], the first one given by Margulis [12]. Because of its simplicity we shall use the expander from [7]. A short description of this graph can be found at the end of Section 2.1.

To generate the pseudo-random sequence the algorithm of [8] performs a random walk on an expander with constant degree of size 2^r. In the construction, computing the next transition in the random walk involves arithmetic operations on integers of length r (r can be a polynomial of the input size n). The parallel version we propose has the advantage that the length of the numbers involved is reduced. The idea is straightforward: to obtain r pseudo-random bits we generate t segments of r/t pseudo-random bits simultaneously (wlog we can assume that r/t is an integer). Then store the t segments in a shared memory, which can be accessed later by other applications. Each generated segment can be stored in a pre-assigned block

*Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest, Lágymányosi u. 11. H-1111, Hungary. Email: kati@ilab.sztaki.hu. Most of the work was done while at the Department of Computer Science, University of Chicago. Supported in part by OTKA Grant No. 2581.

†Department of Computer Science, University of Chicago, 1100 E. 58th St., Chicago, IL 60637, USA, Email: tsai@cs.uchicago.edu.

of memory. To do so, we make t independent random walks on an expander with $2^{r/t}$ vertices. Thus the r-bit arithmetics in [8] are reduced to t independent r/t-bit arithmetics. Our method uses $O(r+tk)$ random bits to reduce the error probability to 2^{-k}. The correctness of the idea is based on the fact that t simultaneously independent random walks on a smaller expander is equivalent to a random walk on a much larger expander. Formally, we prove that t simultaneously independent random walks on an s-regular expander of $2^{r/t}$ vertices can be simulated by a random walk on an $((s+1)^t-1)$-regular expander with 2^r vertices. The relation is bridged by the fact that the eigenvalues of the tensor product of two matrices can be obtained from the product of the eigenvalues of the two matrices. Therefore the quality of the pseudo-random bits generated by our approach is guaranteed by the results in [8, 6]. Meanwhile we get an optimal (t times) speedup.

The rest of the paper is organized as follows. In section 2 we present some definitions and facts. In section 3 we prove some technical results about graph products. In section 4 we show how to recycle random bits in parallel.

2 Preliminaries and notations

Let $G(V,E)$ be a graph with n vertices. The boundary of a set $W \subseteq V$ is denoted by $N(W)$, it contains those vertices of $V(G)\backslash W$ that are adjacent to some vertex in W. The adjacency matrix A of the graph G is defined as an $n \times n$ matrix with entries $a_{ij}=1$, if there is an edge between the i-th and the j-th vertices; 0 otherwise ($1 \le i,j \le n$). By the eigenvalue of a graph we mean the eigenvalue of its adjacency matrix.

2.1 Expanders

Definition. [3] A graph G is called an (n,d,c)-*expander* if it has n vertices, the maximum degree of a vertex is d, and for every set of vertices $W \subset V(G)$ of cardinality $|W| \le \frac{n}{2}$, the inequality $|N(W)| \ge c|W|$ holds.

The constant c is called the *expansion coefficient* or *expansion rate* of the graph.

If G is d-regular then its largest eigenvalue is d. The expansion coefficient is connected to the second largest eigenvalue, namely

Theorem 1 *[2] If λ is the second largest eigenvalue of the d-regular graph G with n vertices, then G is an (n,d,c)-expander for $c=\frac{2(d-\lambda)}{3d-2\lambda}$.*

Note that if we consider the matrix $\frac{1}{d}A$ then its largest eigenvalue is 1 and if λ_2 denotes the second largest eigenvalue then $c = 2(1-\lambda_2)/(3-2\lambda_2)$.

Sometimes a slightly different definition of expander is used. For example in [7], an (n,d,c)-expander is defined as a bipartite graph with n inputs, n outputs and at most dn edges, such that for every subset X of inputs, $|N(X)| \ge (1+c(1-|X|/n))|X|$. These graphs are called strong (n,d,c)-expanders in [2]. There is a relation between expanders and strong expanders: if $G=(I,O;E)$ is a strong (n,d,c)-expander, then G is a $(2n,d,c/16)$-expander [2]. The expansion rate can be improved from $c/16$ to $c/6$ by a more careful estimate in the proof.

Gabber and Galil [7] explicitly constructed strong $(n,7,(2-\sqrt{3})/2)$-expanders, where $n=m^2$ and m is any positive integer, which with our notation are $(2n,7,(2-\sqrt{3})/12)$-expanders. For completeness we repeat here their construction. Let V_n be $\{0,1,\cdots,m-1\} \times \{0,1,\cdots,m-1\}$. The 7-regular expander G_n is obtained from 7 permutations on V_n. For each vertex $(x,y) \in V_n$ there are 7 neighbors, determined by 7 permutations. The 7 permutations are $\rho_0(x,y)=(x,y), \rho_1(x,y)=(x,2x+y), \rho_2(x,y)=(x,2x+y+1), \rho_3(x,y)=(x,2x+y+2), \rho_4(x,y)=(x+2y,y), \rho_5(x,y)=(x+2y+1,y)$, and $\rho_6(x,y)=(x+2y+2,y)$, where $+$ is mod m. Note that this expander can be constructed efficiently by an algorithm.

2.2 Graph product

To show that t independent random walks on a smaller expander can be simulated with a random walk on a larger expander we need the following definitions [11, 9].

Definition. The *tensor (or Kronecker) product* of two matrices A and B in coordinate form is defined as

$$A_{n\times p} \otimes B_{m\times q} = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1p}B \\ a_{21}B & a_{22}B & \dots & a_{2p}B \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ a_{n1}B & a_{n2}B & \dots & a_{np}B \end{pmatrix}.$$

Definition. Let G_1 and G_2 be two graphs. G is the *strong direct product* of G_1 and G_2, denoted as $G = G_1 \times G_2$, if

$$V(G) \quad = \quad V(G_1) \times V(G_2)$$

$$
\begin{aligned}
E(G) \quad = \quad & \{((u_1,u_2),(v_1,v_2))| \\
& ((u_1,v_1)\in E(G_1)\text{and } (u_2,v_2)\in E(G_2)) \\
& \text{or } (u_1=v_1 \text{ and } (u_2,v_2)\in E(G_2)) \\
& \text{or } ((u_1,v_1)\in E(G_1) \text{ and } u_2=v_2)\}.
\end{aligned}
$$

Let A_i, for $i=1$ and 2, be the adjacency matrix of G_i and G'_i be the graph G_i with a self-loop on each vertex. A quick observation by the definition of expander is that G is an expander iff G' is an expander. Let $A'_i = A_i + I$, then it is clear that A'_i is the adjacency matrix of G'_i, $A'_1 \otimes A'_2$ is the adjacency matrix of $G'_1 \times G'_2$, and $A'_1 \otimes A'_2 - I$ is the adjacency matrix of $G_1 \times G_2$.

For convenience we use $G(t)$ to denote the graph $\overbrace{G\times G\times\cdots\times G}^{t}$ and $A(t)$ for its adjacency matrix $[\overbrace{A'\otimes A'\otimes\cdots\otimes A'}^{t} - I_{n^t\times n^t}]$, where t is any positive integer.

Since we are interested in expanders and the expansion rate is connected to the second largest eigenvalue the following well known fact will be useful.

Lemma 2 *[11] Given two matrices A and B, the eigenvalues of $A\otimes B$ are given by all possible products $\lambda\mu$, where λ and μ are eigenvalues of A and B, respectively.*

2.3 Random walks

In a random walk on the graph G a particle starting from an initial vertex u, and in each step it either stays or moves to one of its neighbors with some given probabilities. A random walk is described by the transition matrix and the initial position of the particle. We usually define the random walk by its transition matrix. We shall also talk about t *independent random walks* on the same graph when starting from t arbitrary vertices each of the walks has the same transition matrix and they move independently of each other.

Now consider the s-regular bipartite graph G. Its adjacency matrix A has s as the largest eigenvalue. Let $P = \frac{1}{2}(I + \frac{A}{s})$ and consider the random walk with transition matrix P. In this random walk, the particle stays at the same vertex with probability $\frac{1}{2}$ and moves to an arbitrary neighbor with probability $\frac{1}{2s}$. This so called lazy random walk is used in [8].

Instead of P we use another random walk on the s-regular graph G given by the following transition matrix $P' = \frac{1}{s+1}(I+A) = \frac{1}{s+1}A'$. It is not hard to see that $P'\otimes P'$ is also a transition matrix, and it defines a random walk on the graph $G\times G$. It is worth noting that the second largest eigenvalue of $P'\otimes P'$ by Lemma 2 is the same as $P's$. It is clear that $\frac{1}{(s+1)^t}(I+A(t))$ is a transition matrix, too. We can use it to define a random walk on the $((s+1)^t-1)$-regular graph $G(t)$.

3 Technical results

In this section we prove that t independent random walks on a smaller expander can be simulated by a random walk on a larger expander by looking at the second largest eigenvalue of the associated transition matrix.

Theorem 3 *Given an s-regular bipartite graph G and its adjacency matrix A, let λ_i be the i-th eigenvalue of $\frac{1}{s}A$ and ϵ be a positive constant. If $\lambda_2 < 1-\epsilon$ then all the eigenvalues of $\frac{1}{(s+1)^t}(I+A(t))$, except the largest one (i.e. 1), have absolute value $\le 1-\gamma$, where $\gamma = Min\ \{\frac{s\epsilon}{s+1}, \frac{2}{s+1}\}$.*

Proof. With the assumption on G, we have $1 = \lambda_1 > \lambda_2 \ge \cdots \ge \lambda_n = -1$, since for a bipartite graph the eigenvalues appear in pairs $\pm\lambda$. The eigenvalues of $\frac{1}{s+1}(I+A)$ are $\frac{1+s\lambda_i}{s+1}$, for $1\le i\le n$. So for $i\ge 2$, we have $|\frac{1+s\lambda_i}{s+1}| \le 1-\gamma$. While the largest eigenvalue is still 1. Now consider the case $t=2$. By Lemma 2 we know 1 is the largest eigenvalue of the transition matrix $\frac{1}{(s+1)^2}(I+A)\otimes(I+A)$ and the rest of the eigenvalues have absolute values $\le 1-\gamma$. Then by induction the theorem follows. $\Box$

Corollary 4 *With the same assumption as above, G is an expander with $c = 2\epsilon/(1+2\epsilon)$ and $G(t)$ is an expander with $c = 2\gamma/(1+2\gamma)$.*

Proof. By Theorem 1, G is an (n,s,c)-expander with $c = 2\epsilon/(1+2\epsilon)$. The absolute value of each eigenvalues of $I + A(t)$, except the largest one, is less than $(s+1)^t(1-\gamma)$. For the matrix $\frac{A(t)}{(s+1)^t-1}$, it is less than $\frac{(s+1)^t(1-\gamma)-1}{(s+1)^t-1} = 1 - \frac{(s+1)^t\gamma}{(s+1)^t-1}$. Then again by Theorem 1, $G(t)$ is an $(n^t,(s+1)^t-1,c)$-expander with $c = 2\gamma/(1+2\gamma)$. $\Box$

Recall that we define t independent random walks on a graph as: starting from t arbitrary vertices, for each random walk it proceeds independently at next step to an adjacent vertex chosen uniformly at random or stays. We show in the following that t independent random walks on G can be simulated by a random walk on $G(t)$. This is of interest especially when G is an expander.

Theorem 5 *Let A be the adjacency matrix of a graph G. Then t independent random walks on G with $\frac{1}{s+1}(I+A)$ as the transition matrix can be simulated by a random walk on $G(t)$ with $\frac{1}{(s+1)^t}(I+A(t))$ as the transition matrix.*

Proof. Let the t random walks start at $u_{01}, u_{02}, \cdots, u_{0t}$, which means the i-th random walk starts with vertex u_{0i}, for $1 \leq i \leq t$. The initial vertices are chosen uniformly at random. We associate a particle with each random walk to indicate the current position of the walk. The random variables X_i indicate the vertex on which the i-th particle stays. Then Prob$[X_i = v_i, 1 \leq i \leq t$, at the $(k+1)$-th step $\mid X_i = u_i, 1 \leq i \leq t$ at the k-th step$]= \frac{1}{(s+1)^t}$, if each v_i is u_i or u_i's neighbor; 0 otherwise. From G define a graph H, where $V(H) = \{< u_1, u_2, \cdots, u_t > | u_i \in V(G)\}$, and $E(H) = \{(< u_1, u_2, \cdots, u_t >, < v_1, v_2, \cdots, v_t >) | u_i = v_i \text{ or } (u_i, v_i) \in E(G)\}$. By removing the self-loops in H we get the graph $G(t)$. The corresponding transition matrix is nothing but $\frac{1}{(s+1)^t}(I+A(t))$. Thus the theorem follows. □

Now it is clear that t independent random walks with proper transition matrices on an s-regular expander is nothing but a random walk with a random walk on a bigger expander. This allows us to parallelize the computation proposed in [8]. The details are in the next section.

4 How to recycle random bits in parallel

The work in [8] is to reduce the error probability with fewer random bits for the membership problems in BBP. Recall that BPP is the set of languages $L \subseteq \{0,1\}^*$ such that there exists a deterministic polynomial time Turing machine $M_L(a,x)$ for which

$a \in L \Rightarrow Pr[M_L(a,x) = 1] \geq \frac{2}{3}$

$a \notin L \Rightarrow Pr[M_L(a,x) = 0] \geq \frac{2}{3}$

where the probabilities are for an x picked uniformly in $\{0,1\}^{p(|a|)}$ for some polynomial p. $M_L(a,x) = 1$ indicates M_L accepts a by using random bits x; $M_L(a,x) = 0$ otherwise. Let $L \in BPP$, $a \in \{0,1\}^n$, and let $r = p(n)$ be the number of random bits used. To amplify the success probability from $\frac{2}{3}$ to $1-2^{-k}$, as usual we can test it $O(k)$ times and take a majority vote, but this procedure needs $O(rk)$ random bits. The success probability can be amplified with $O(r+k)$ random bits by using an expander on 2^r vertices to generate a pseudo-random sequence [8]. The idea is to perform a random walk on the expander and from time to time take the label of the vertex being visited as next segment of the pseudo-random string, meanwhile make enough moves between generating two segments to ensure that the resulting string is pseudo-random. With our approach, we make t independent random walks on an expander with $2^{\frac{r}{t}}$ vertices. We use the same 7-regular expander G but now only on $2^{\frac{r}{t}}$ vertices. Let A be the adjacency matrix of G, $t \geq 1$ be an integer and λ_i be the i-th largest eigenvalue of $\frac{1}{8}(I+A)$. There is a positive constant ϵ such that $\lambda_2 < 1-\epsilon$. Perform t independent random walks on G with transition matrix $\frac{1}{8}(I+A)$. By Theorem 5, we know it corresponds to a random walk on an (8^t-1)-regular expander with $\frac{1}{8^t}(I+A(t))$ as the transition matrix. By Theorem 3, we know that the second largest eigenvalue of $\frac{1}{8^t}(I+A(t))$ is at most $1-\gamma$, where $\gamma = \text{Min}\{\frac{7\epsilon}{8}, \frac{2}{8}\}$.

Algorithm: *Let ℓ be the smallest integer such that $(1-\gamma)^\ell < 1/10$. Use r bits to find t starting vertices uniformly. Perform t independent random walks on G with the transition matrix $\frac{1}{8}(I+A)$ for $7\ell k$ steps. At every ℓ steps, output the concatenation (from walk 1 to walk t) of the labels of the t vertices on which the t particles are sitting.*

The above algorithm generates $7k$ pseudo-random strings of length r each, which can be used to reduce the error probability to 2^{-k} for the membership problems in BPP. Each r-bit pseudo-random string is generated in parallel. It is not hard to see that we use $O(r+tk)$ random bits in the above algorithm. The correctness of the algorithm can be shown by following the proof in [8] with the (8^t-1)-regular expander and the transition matrix $\frac{1}{8^t}(I+A(t))$. Note that the expander has 2^r vertices and we will use the label of each vertex as pseudo-random bits. For completeness, we include the proof in this paper, i.e. we will show that by using the pseudo-random bits generated by our algorithm the error probability is at most 2^{-k}. Before applying our algorithm, as in [8] we boost the success probability from 2/3 to .99 by using the usual technique (i.e. repeat the algorithm a few times and take the majority answer as the output). This step will use at most $O(r)$ random bits. From now on we are dealing with a new probabilistic algorithm with error probability .01 for the same membership problem. In other words, for any input, there are at most $2^r/100$ vertices of the expander that will not witness the correct answer and the probabilistic algorithm for the membership problem will give an incorrect answer by using them as random bits. Let U denote the set of these vertices and W the set of good vertices which make the algorithm give a correct answer. Let p_i be the probability for a particle sitting at vertex i during the random walk. Then the 2^r dimensional vector $p = (p_1, \cdots, p_{2^r})$ represents the probabilities of being at the different vertices, where $\sum p_i = 1$. For any vector v, define $\| v \| = \sqrt{\sum_i v_i^2}$, and $|v| = \sum_i |v_i|$. Let N be a $2^r \times 2^r$ diagonal matrix, where $N_{i,i} = 1$ if $i \in U$; 0 otherwise. $M = I - N$, where I is the identity matrix. It is clear that the vector Np keeps only the entries in p corresponding to the vertices in U and has the rest of the entries as zeroes. Similarly, Mp keeps the entries corresponding to the vertices in W. Let $B = (\frac{1}{8^t}(I + A(t)))^\ell$, which is a transition matrix whose (i, j) entry indicates the probability of starting from vertex i and stopping at vertex j in ℓ steps. Note that the second largest eigenvalue of B is at most $(1-\gamma)^\ell$ and its largest eigenvalue is still 1. Also B is a symmetric matrix. Let $p_0 = (1/2^r, \cdots, 1/2^r)$ be a 2^r-dimensional vector, which defines the initial probability of the random walk. NBp_0 represents the probabilities of being at the different vertices in U and $|NBp_0|$ represents the probability of reaching the vertices in U at the ℓ-th step. More generally, for example, $|(NB)(NB)(MB)p_0|$ indicates the probability of generating good pseudo-random bits, bad pseudo-random bits and again bad pseudo-random bits in order. In the sequel, we follow the proof technique used in [1, 8].

Lemma 6 *[8] Let $p = (p_1, \cdots, p_{2^r})$, where $p_i \geq 0$ and $\sum_{i=1}^{2^r} p_i = 1$. Then*

1. $\| MBp \| \leq \| p \|$.
2. $\| NBp \| \leq \| p \| /5$.

Proof. B has real eigenvalues $1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_{2^r}$, since it is a symmetric real matrix. Let u_i be the corresponding eigenvector of λ_i, where $\{u_i\}$ forms an orthonormal basis and u_1 has the same value for each entry. It is clear that p can be written as $\sum_i a_i u_i$, where a_i's are real numbers. Then $Bp = \sum_i a_i(Bu_i) = \sum_i a_i \lambda_i u_i$. $\| Bp \| = \sqrt{\sum_i a_i^2 \lambda_i^2} \leq \sqrt{\sum_i a_i^2} = \| p \|$, since $|\lambda_i| \leq 1$. By the definition of M, we know $\| MBp \| \leq \| Bp \|$. Thus the first part is proved.

By the triangle inequality,

$$\| NBp \| \quad \leq \quad \| NB(a_1 u_1) \| + \| NB(\sum_{i \geq 2} a_i u_i) \| .$$

Since $Bu_1 = u_1$ and there are at most $2^r/100$ non-zero entries on the diagonal of N, we have $\| NB(a_1 u_1) \| = \| N(a_1 u_1) \| \leq \frac{\| a_1 u_1 \|}{10} \leq \frac{\| p \|}{10}$. By the selection of ℓ in the above algorithm, we know that $\lambda_i < 1/10$, for $i \geq 2$. Thus

$$\begin{aligned} \| NB(\sum_{i \geq 2} a_i u_i) \| &\leq \| B(\sum_{i \geq 2} a_i u_i) \| \\ &= \| \sum_{i \geq 2} a_i \lambda_i u_i \| \\ &\leq \| \sum_{i \geq 2} a_i u_i \| /10 \\ &\leq \| p \| /10. \end{aligned}$$

Therefore $\| NBp \| \leq \| p \| /5$. □

Let S be a sequence of length $7k$ consisting of two elements (NB) and (MB), i.e. we treat (NB) and (MB) as single letters. Actually, S is the product of a sequence of matrices. Consider the sequence S with at least $7k/2$ (NB)'s. Then $|Sp_0|$ represents the probability that the majority of the pseudo-random bits give wrong answers, i.e. visits the vertices in U more than $7k/2$ times in the walk. We can prove that this probability is small.

Theorem 7 *[8] In a random walk given by the transition matrix B, the probability that it visits the vertices in U more than $7k/2$ times is less than 2^{-k}.*

Proof. $|Sp_0| \leq 2^{\frac{r}{2}} \| Sp_0 \| \leq 2^{\frac{r}{2}} \| (NB)^{\frac{7k}{2}} p_0 \| \leq 2^{\frac{r}{2}} (\frac{1}{5})^{\frac{7k}{2}} \| p_0 \| = 5^{-7k/2}$. In total, there are at most 2^{7k} such sequences S. Thus the probability is bounded by $2^{7k} 5^{-7k/2} < 2^{-k}$. □

We now show how to deal with the overhead of long integer arithmetics involved in [8]. The smaller expander makes it easier and more feasible to calculate the neighbors in the random walk, since each vertex has a shorter label. More precisely, the computation of $\rho(x, y)$, for $x, y \in \{0, 1\}^{\frac{r}{2}}$, can be decomposed by computing $\rho_{i1}(x_1, y_1) \circ \rho_{i2}(x_2, y_2) \circ \cdots \circ \rho_{it}(x_t, y_t)$, where each ρ_{ij} and ρ can be any one of the 7 permutations described in Section 2.1, and x_i's, y_i's are in $\{0, 1\}^{\frac{r}{2t}}$. Note that the $\rho_{ij}(x_i, y_i)$'s are computed in parallel. Thus in a multi-processor system or a PRAM machine, processor j computes $\rho_{ij}(x_j, y_j)$, for $j = 1..t$, and the r pseudo-random bits are simply the concatenation of their outputs. It is clear that we obtain an optimal speed-up by t times. In the sequential version $O(r+k)$ random bits are used to reduce the error probability down to 2^{-k}. For the parallel version, $O(r+tk)$

random bits are used and this makes sense as long as $O(r+k) \leq O(r+tk) \leq O(rk)$. Thus, to achieve the optimal speedup and save the random bits simultaneously, we need $t \leq O(r)$. From the practical point of view, t is going to be constant in real applications.

Note that the ℓ in the algorithm can be very small, if we can find an expander which has a very small second largest eigenvalue and can be efficiently generated. Also note that the intermediate ℓ steps between two outputs can be done in one step by simply using the distance-ℓ graph of the original expander. In this case we will use more space to store the even bigger set of permutations, which can be obtained by composing the earlier mentioned 7 permutations. The transition matrix is simply B.

The naive way to parallelize [8] would be doing the arithmetic in parallel. Our method is faster than that: assume that r/t-bit operations take one unit time – then we do one step in unit time; the naive parallel version with t processors will take more time, because of the carries between the blocks (see NC addition). Another advantage is that in our method the processors are independent, thus there is no need to communicate among processors. What we pay for this is in the need for more random bits. Our method can be further decomposed by recursion as long as we can keep the use of random bits under control. It is an interesting question whether this proposed technique could be useful in existing highly parallel computers.

Acknowledgements

We would like to thank Janos Simon for helpful discussion and the anonymous referees for helpful comments.

References

[1] M. Ajtai, J. Komlós, and E. Szemerédi. Deterministic simulation of logspace. In *19th STOC*, pages 132–140, 1987.

[2] N. Alon. Eigenvalues and expanders. *Combinatorica*, 6:83–96, 1986.

[3] N. Alon, J. Spencer, and P. Erdős. *The Probabilistic Method.* John Wiley and Sons Inc., 1991.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *33rd FOCS*, pages 14–23, 1992.

[5] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[6] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *30th FOCS*, pages 14–19, 1989.

[7] O. Gabber and Z. Galil. Explicit construction of linear sized superconcentrators. *J. Comp. and Sys. Sci.*, 22:407–420, 1981.

[8] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *30th FOCS*, pages 248–253, 1989.

[9] L. Lovász. *Combinatorial Problems and Exercises.* North-Holland, Amsterdam, second edition, 1993.

[10] A. Lubotzky, R. Phillips, and N. Sarnak. Explicit expanders and the ramanujan conjecture. In *18th STOC*, pages 261–277, 1986.

[11] M. Marcus and H. Minc. *A Survey of Matrix Theory and Matrix Inequalities.* Allyn and Bacon Inc., 1964.

[12] G. A. Margulis. Explicit construction of expanders. *Probl. Peredachi Informatsii*, 9/4:71–80(in Russian), 1973.

[13] G. A. Margulis. Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Probl. Peredachi Informatsii(Problems of Information Transmission)*, 24/1:51–60(in Russian), 39–46(English translation), 1988.

[14] M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12:128–138, 1980.

[15] M. Sipser. Expanders, randomness or time vs space. In *Structure in Complexity Theory*, pages 325–329, 1986.