Electronic Colloquium on Computational Complexity, Report No. 20 (2001)

Lower Bounds for OBDDs and Nisan's pseudorandom generator

N.S. Narayanaswamy[*], C.E. Veni Madhavan[†]

Abstract

We present a new boolean function for which any Ordered Binary Decision Diagram (OBDD) computing it has an exponential number of nodes. This boolean function is obtained from Nisan's pseudorandom generator to derandomize space bounded randomized algorithms. Though the relation between hardness and randomness for computational models is well known, to the best of our knowledge, this is the first study relating the problem of proving lower bounds for OBDDs to the issue of pseudorandom generators for space bounded computation. Using the same technique to prove the OBDD size lower bound, we place a lower bound on the size of families of hash functions used in Nisan's pseudorandom generator. This lower bound rules out one method of obtaining improved derandomizations of space bounded randomized algorithms.

1 Introduction

OBDDs are a restricted type of branching programs which were introduced by Bryant [2] as a data structure allowing efficient operations and compact representation for boolean functions. The interesting lower bound question with respect to OBDDs is regarding the number of nodes in the smallest OBDD that represents a boolean function. Explicit functions that have exponential OBDD complexity are known. For a comprehensive reference on specific functions with exponential OBDD complexity and the current state of research on OBDDs, see the recent book by Wegener [7]. Another important property of OBDDs is that they model the computation of a randomized space bounded algorithm in a non-uniform way. By a space S bounded randomized algorithm, we mean one that uses S bits of workspace, halts in 2^S steps, has an input tape, a random tape and reads every random bit exactly once. See [5] for a survey of the role of randomness to get space efficient algorithms. This connection between OBDDs and randomized space bounded algorithms has been used by Nisan [3] to design a pseudorandom generator to derandomize space bounded randomized algorithms. The design of the pseudorandom generator is based on an algorithm to approximate the square of stochastic matrices (the measure of approximation is presented below) It is known from the seminal work by Nisan and Wigderson [4] that pseudorandom generators for circuits yield hard functions for circuits and vice versa. We explore this connection for OBDDs and obtain a function with exponential OBDD complexity from Nisan's algorithm to approximate the square of a stochastic matrix.

Nisan's pseudorandom geneartor uses a set of functions called Universal Family of Hash functions (defined below). The size of this set of functions determines the space used by Nisan's pseudorandom generator. Consequently, if one could construct a smaller set of functions that can

[*]Institut Für Informatik, Ludwig Maximilians Universität, Oettingenstrasse 67, 80538, Munich, Germany, Email: `swamy@informatik.uni-muenchen.de`. Supported by DFG Grant No. Jo 291/2-1

[†]Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India, Email: `cevm@csa.iisc.ernet.in`.

ISSN 1433-8092

be used by Nisan's pseudorandom generator to yield a derandomization, then it would be possible to derandomize randomized space bounded algorithms with lesser space than currently known by the pseudorandom generator approach. This would also improve the space bounds of algorithms [1, 6] for undirected connectivity, and approximate powering of stochastic matrices. We deal with this question in this paper and prove a lower bound on the size of families of functions that can be used in the algorithm to approximate the square of a stochastic matrix. The proof technique to obtain this lower bound follows closely the method of obtaining a hard function from Nisan's pseudorandom generator.
Our Results: We obtain from Nisan's pseudorandom generator for space bounded computation a new boolean function that has exponential OBDD complexity. This boolean function is very natural as it can be seen as hardness arising from pseudorandomness for OBDDs. The other question that we address in this paper is, *Does there exist a family of functions that are significantly smaller than and of the same quality, as the family used in Nisan's algorithm to approximate the square of a stochastic matrix.* An affirmative result would imply an improved derandomization of space bounded randomized algorithms. We show that the answer is negative, thus ruling out one approach towards getting improved derandomizations of space bounded randomized algorithms.
The paper is organized as follows. Section 2 presents the relevant definitions, notation, and tools for this paper. Nisan's algorithm for finding the approximate square of a stochastic matrix is presented in Section 3. In Section 4 and Section 5 we prove the claimed results. Finally, we summarize the results in Section 6.

2 Preliminaries

Here we present the preliminaries relating to pseudorandom generators, OBDDs and circuits.
Notation related to Matrices: A *stochastic* matrix is a matrix of non-negative entries in which the elements of each row add up to 1. We consider stochastic matrices of dimension 2^S, for some integer $S \geq 0$. We let Q denote a stochastic matrix no matter what its subscript or superscript is. Unless stated otherwise, all the entries of these matrices are of the form $\frac{a}{2^t}, t \in Z_+$. With Q we associate a directed graph G_Q in which each vertex has out-degree 2^t, the edges leaving a vertex are labeled bijectively with $\{0,1\}^t$, and the number of edges from i to j in $G_Q = Q_{ij}2^t$. It is easy to see that such a graph exists. We refer to each such graph G_Q as a graph associated with Q. The *norm* of a matrix Q is defined by $\|Q\| = \max_i(\sum_j |Q_{ij}|)$.
Universal Hashing: Let H_t be a set of functions $\{h : \{0,1\}^t \to \{0,1\}^t\}$. H_t is called a *universal family of hash functions* if for any $x_1 \neq x_2 \in \{0,1\}^t$, and $y_1, y_2 \in \{0,1\}^t$ we have that

$$\Pr_{h \in H_t}(h(x_1) = y_1 \wedge h(x_2) = y_2) = \frac{1}{2^{2t}}$$

An example for a universal family of functions is the set $H_t = \{h_{a,b} : \{0,1\}^t \to \{0,1\}^t | a, b \in \{0,1\}^t, h_{a,b}(x) = ax + b\}$, where the addition and multiplication operations are the ones defined over $GF(2^t)$. Each function in this set has a $2t$ bit description. Conversely, each $2t$ bit string represents a function from H_t. Consequently, for simplicity we will consider $\{0,1\}^{2t}$ as the universal family of hash functions defined on $\{0,1\}^t$.
OBDDs: A *branching program P* for computing a boolean function $f : \{0,1\}^n \to \{0,1\}$ is a layered directed acyclic multi-graph with a distinguished source node s and two distinguished sink nodes. One is an accepting sink which is called a 1-sink and, the other is called a 0-sink. All the edges leaving a sink are self-loops. The out-degree of each node is exactly 2, and the outgoing edges are labeled by $x_i = 0$ and $x_i = 1$ for an input variable x_i associated uniquely with this node. The label

$x_i = \delta$ indicates that only inputs satisfying $x_i = \delta$ may follow this edge in the computation. The branching program P computes a function f in the obvious way: for each $\sigma \in \{0,1\}^n$ we let the output of P to be 1 if and only if there is a directed path starting at the source and leading to the 1-sink such that all labels $x_i = \sigma_i$ along this path are consistent with $\sigma = \sigma_1\sigma_2\ldots\sigma_n$. A path from the source to the 1-sink is called an accepting path. A *read-once* branching program is a branching program in which no variable appears more than once on any computation path. An *OBDD* is a read-once branching program in which, the nodes at the same distance from the source have the same input variable associated with it. For a branching program P, the size of P is defined as the number of internal nodes in P. For an OBDD P, the length of P is defined as the number of layers in the directed acyclic multi-graph that represents P.

OBDDs model randomized space bounded algorithms in a non-uniform way, that is, the computation graph of a randomized algorithm on a fixed input x is an OBDD. In this OBDD the variables in question corresponds to the random bits used by the algorithm while computing on x. By definition, the random bits are read only once and in a fixed order from the tape. The set of accepting paths in this OBDD correspond exactly to the set of strings on the random tape that accept the input.

With an OBDD P on N-nodes, for $1 \le t \le N$, we associate an $N \times N$-matrix which we refer to as the t-step Transition Probability Matrix Q of the OBDD. Q is defined as follows: $Q_{ii} = 1$, for each sink i, $Q_{ij} = \Pr_{x\in\{0,1\}^t}(x_0, \ldots, x_{t-1}$ are the edge labels on a walk from i to $j), 1 \le i \ne j \le N$. For the t-step transition probability matrix Q, of an OBDD P, we fix the directed graph G_Q associated with Q. The graph G_Q is obtained from P as follows: The vertex of G_Q is same as the vertex set of P, and (i,j) is a directed edge in G_Q if and only if there is walk of t edges from i to j in P. The edges in P are labeled by t bit strings. A t bit string is a label on an edge between i and j if and only if it forms a walk from i to j in P.

3 Nisan's pseudorandom generator

Nisan's pseudorandom generator is a function that takes as input $t(2k+1)$ bits and outputs $t2^k$ bits for $t, k \in Z_+$. We denote the pseudorandom generator by $R_k(x, h_1, \ldots, h_k)$ where, x is a t bit seed, and $h_i \in \{0,1\}^{2t}, 1 \le i \le k$ are functions that map t bit strings to t bit strings. It is defined recursively as follows: $R_0(x) = x, R_k(x, h_1, \ldots, h_k) = R_{k-1}(x, h_1 \ldots, h_{k-1}) \circ R_{k-1}(h_k(x), h_1, \ldots, h_{k-1})$. Here $\circ$ denotes the concatenation operation. $R_k(x, h_1, \ldots, h_k)$ can be seen as a vector $(S_1, \ldots, S_{2^k})$ where each entry is from $\{0,1\}^t$. This sequence naturally defines a walk in a directed graph in which each vertex has out-degree 2^t and the edges leaving a vertex are labeled bijectively with $\{0,1\}^t$. This walk visits 2^k vertices, not necessarily distinct, and the $i+1$st vertex in this walk is the end point of that edge labeled S_i which leaves ith vertex of the walk. Therefore, $R_k(x, h_1, \ldots, h_k)$ is defined to take p to q if the above mentioned walk starting at p has q as its final vertex. The main application of the pseudorandom generator is the following theorem proved in [3]:

Theorem 1. *Let Q be a $2^S \times 2^S$ stochastic matrix, and G_Q be a directed graph associated with Q. Let H_t be a universal family of hash functions on $\{0,1\}^t$. For $k \ge 0$ and $\epsilon \ge 0$,*

$$\Pr_{h_1,\ldots,h_k\in H_t}(\|Q_{h_1,\ldots,h_k} - Q^{2^k}\| > (2^k-1)2^{2S}\epsilon) \le \frac{2^{3S}k}{\epsilon^2 2^t}.$$

Where, for $h_1, \ldots, h_k$, $Q_{h_1,\ldots,h_k}(ij) = \Pr_{x\in\{0,1\}^t}(R_k(x, h_1, \ldots, h_k)$ takes i to j in $G_Q)$.

For $\epsilon = \frac{1}{(2^k-1)2^{4S}}, 0 \leq k \leq S, t = 14S$, the above theorem guarantees that

$$\Pr_{h_1,\ldots,h_k}(\|Q_{h_1,\ldots,h_k} - Q^{2^k}\| > \frac{1}{2^{2S}}) \leq \frac{2^{3S}k(2^k-1)^2 2^{8S}}{2^t} \leq \frac{2^{13S}}{2^t} = \frac{1}{2^S}.$$

In particular, for $k = 1$, the theorem gives a randomized algorithm to approximate Q^2 for any stochastic matrix Q. The algorithm is: pick a h uniformly at randomly from H_t and compute Q_h. The theorem guarantees that with probability at least $1 - \frac{1}{2^S}$, Q_h approximates Q^2 to within a distance of $\frac{1}{2^{2S}}$ in the matrix norm we have defined. Since H_t has 2^{28S} elements in it, the theorem says that at most 2^{27S} will yield a matrix whose distance from Q^2 is more than $\frac{1}{2^{2S}}$.

3.1 Application to OBDDs and related questions

Here we present an application of the theorem, which we use to explicitly define a boolean function of exponential OBDD complexity. The application is that, R_k can be used to find accepting paths in OBDDs of size 2^S in which, the probability that a randomly chosen path is an accepting path is more than $\frac{1}{2^{2S}}$. We state this formally in the following theorem:

Theorem 2. *Let P be an OBDD on 2^S nodes and of length L such that, the probability that a randomly chosen path in P is an accepting path is more than $\frac{1}{2^{2S}}$. Let Q be the t-step transition probability matrix of P, and G_Q the graph associated with Q. Let H_t be a universal family of functions on $\{0,1\}^t$. For $t = 14S, \log(\frac{L}{t}) \leq k \leq S$, the set $\{R_k(x,h_1,\ldots,h_k) | x \in \{0,1\}^t, h_i \in H_t, 1 \leq i \leq k\}$ has non-empty intersection with the set of strings accepted by P.*

Proof. Let s be the source node of P, and A the 1-sink. For $k \geq 0$, $Q^{2^k}_{sA}$ is exactly the probability that a randomly chosen string of length $t2^k$ is accepted by P. Also, the strings obtained from the edge labels of s to A walks in G_Q are exactly the set of strings accepted by P. From theorem 1 it follows that for $t = 14S$ and $0 \leq k \leq S$, there is a choice of $h_1, \ldots, h_k$ such that $\|Q_{h_1,\ldots,h_k} - Q^{2^k}\| \leq \frac{1}{2^{2S}}$. This is because the probability that for a randomly chosen $h_1, \ldots, h_k$, $\|Q_{h_1,\ldots,h_k} - Q^{2^k}\| \leq \frac{1}{2^{2S}}$, is at least $1 - \frac{1}{2^S}$. Let $g_1, \ldots, g_k$ be such that $\|Q_{g_1,\ldots,g_k} - Q^{2^k}\| \leq \frac{1}{2^{2S}}$. By the definition of the matrix norm, it follows that for all $1 \leq i,j \leq 2^S$, $|(Q_{g_1,\ldots,g_k})_{ij} - Q^{2^k}_{ij}| \leq \frac{1}{2^{2S}}$. In particular, $|(Q_{g_1,\ldots,g_k})_{sA} - Q^{2^k}_{sA}| \leq \frac{1}{2^{2S}}$. Therefore, $(Q_{g_1,\ldots,g_k})_{sA} \geq Q^{2^k}_{sA} - \frac{1}{2^{2S}}$. We know that $Q^{2^k}_{sA}$ is exactly the probability that a randomly chosen string of length $t2^k$ is accepted by P. From the hypothesis, for $k \geq \log(\frac{L}{t})$, $Q^{2^k}_{sA}$ is more than $\frac{1}{2^{2S}}$. Consequently, $0 < (Q_{g_1,\ldots,g_k})_{sA} = \Pr_{x \in \{0,1\}^t}(R_k(x,g_1,\ldots,g_k)$ takes s to A in $G_Q)$. Since s to A paths in G_Q correspond exactly to the set of strings accepted by P, it follows that there is an $x \in \{0,1\}^t$ such that $R_k(x,g_1,\ldots,g_k)$ is accepted by P. Therefore,the set $\{R_k(x,h_1,\ldots,h_k) | x \in \{0,1\}^t, h_i \in H_t, 1 \leq i \leq k\}$ has non-empty intersection with the set of strings accepted by P. Hence the theorem. □

One application of this theorem is to derandomize space S bounded randomized algorithms which accept languages with one-sided error. As we have said before, these algorithms are modeled non-uniformly by OBDDs of 2^S nodes. To derandomize we are faced with the problem of deciding if an OBDD of size 2^S identically evaluates to 0 or evaluates to 1 on more than half of its input set. This problem is solved by checking if an entry from the set $\{R_S(x,h_1,\ldots,h_S) | x \in \{0,1\}^{14S}, h_i \in H_{14S}, 1 \leq i \leq S\}$ evaluates to 1. The time taken to perform this check is $O(2^{28S^2})$ as there are only 2^{28S^2} elements in this set. The space taken is $28S^2 + 15S$ which is consumed to store $x, h_1, \ldots, h_S$ and one node of the OBDD. This is a significant reduction in time and space used by the naive

approach of performing the check on all possible inputs to the OBDD. Observe that the space and time complexity of the algorithm based on Nisan's pseudorandom generator is dependent on the size of the universal family of hash functions used. Consequently, if one could design a smaller family of functions that guarantees theorem 1 then we will be able to derandomize with smaller space and time complexity. This motivation leads to the following questions that have lead to the results in this paper:
Question 1: For a stochastic matrix Q and an associated directed graph G_Q, define the boolean function f_Q on H as follows: $f_Q(h) = 1$ if $\|Q_h - Q^2\| \leq \frac{1}{2^{2S}}$. Otherwise $f_Q(h) = 0$. The question is: Is there a Q for which the OBDD complexity of f_Q is exponential in the input size? The reason to ask this question is, if the OBDD complexity of f_Q is *small* for each Q, then we can apply Nisan's pseudorandom generator to a small OBDD computing f_Q and construct a smaller family of hash functions that guarantees the consequences of Theorem 1 for $k = 1$.
Question 2: How small can a family of functions get and yet guarantee Theorem 1 for $k = 1$?

4 A Hard Boolean Function for OBDDs

Using the features of the Nisan's pseudorandom generator, we define in this section a boolean function that has exponential OBDD complexity. Let $X = \{R_k(x, h_1, \ldots, h_k) | x \in \{0,1\}^{\frac{14S}{32^2}}$, each $h_i \in \{0,1\}^{\frac{14S}{32^2}}\}$. k is set to be 11 so that the generator outputs $28S$ bits. As we have proved in theorem 2, as R_k is a pseudorandom generator, X will intersect the accepting set of any OBDD of size $2^{\frac{S}{32^2}}$ and of length $28S$ which accepts more than $\frac{1}{2^{2S}}$ fraction of its inputs. Now we define the boolean function f on $\{0,1\}^{28S}$:

$$f(h) = \begin{cases} 1 & \text{if } h \notin X, \\ 0 & \text{otherwise} \end{cases}$$

The number of distinct elements in X is at most the number of possible inputs to R_k. The number of bits in a input to R_k is $(2k+1)\frac{14S}{32^2}$ and, for $k = 11$, it follows that $|X| \leq 2^{\frac{S}{2}}$. Therefore, the probability that for a randomly chosen h, $f(h) = 0$ is $\frac{|X|}{2^{28S}}$ which is at most $\frac{2^{\frac{S}{2}}}{2^{28S}}$. Therefore, f evaluates to 1 on more than $\frac{1}{2^{\frac{2S}{32^2}}}$ of its inputs. In fact, it evaluates to 1 on almost all the inputs. In the following theorem, we claim that every OBDD computing this boolean function has exponential size.

Theorem 3. *Any OBDD computing f has size more than $2^{\frac{S}{32^2}}$.*

Proof. The proof is by contradiction. Let us assume that there is an OBDD P of size $2^{\frac{S}{32^2}}$ that computes f. If this were the case, then X would have a non empty intersection with the set of points that evaluate to 1 under f. This is because the probability that for a randomly chosen h, $f(h) = 1$ is at least $1 - \frac{2^{\frac{S}{2}}}{2^{28S}}$. As P computes f, it follows that P evaluates to 1 on more than $\frac{1}{2^{2S}}$ of its inputs. Therefore, by theorem 2 X should have non-empty intersection with the set of strings accepted by P, that is there is a point in X that evaluates to 1 under f. But we know from our construction that every $h \in X$ evaluates to 0 under f. Hence we have arrived at a contradiction. Therefore, our assumption is wrong: Every OBDD computing f has size more than $2^{\frac{S}{32^2}}$. □

f can be evaluated at an input h by a deterministic algorithm using space S. The algorithm is just to check membership in X, which is done by cycling through all inputs to Nisan's pseudorandom generator and checking for equality with h. Based on this observation we state the following theorem.

Theorem 4. *For every $S \in Z_+$, the boolean function $f : \{0,1\}^{28S} \to \{0,1\}$ defined above can be computed deterministically using space $O(S)$, and every OBDD computing f has size more than $2^{\frac{S}{32^2}}$.*

Proof. The proof follows from Theorem 3, and the properties of f described above. □

In the construction of functions above, we have ensured that the boolean function evaluates to 0 on the elements in the range of the pseudorandom generator. In the next section, we use the same approach to prove a lower bound on the size of a family of functions that can be used in Nisan's pseudorandom generator.

5 A lower bound on the size of the family of functions

Nisan's pseudorandom generator based approach, has given us an $O(S^2)$ space algorithm to approximate the higher powers of a stochastic matrix. The basic idea behind the design of the algorithm is to repeatedly approximate the square of a matrix using a function picked from a universal family of functions. We have seen that the major component of the space usage is in storing S hash functions, each with a 28S bit description. Clearly, if we can construct space efficiently a family of functions that is smaller than the universal family of hash function, and is useful for approximate matrix squaring, then we will be able to run Nisan's algorithm with space lesser than $O(S^2)$. We show that this approach cannot work. The reason is that the size of the family of functions is related to the order of the matrix whose square we want to approximate. In particular, for a family of functions H with N elements, we construct a graph G with $4N-1$ vertices such that, no function in the family can be used to approximate the square of the transition probability matrix Q of G. In other words, we show that $\Pr_{h \in H}(\|Q_h - Q^2\| < \frac{1}{4}) = 0$.

Let H be the given family of functions with N elements such that each function in the family maps t bit strings to t bit strings. We construct a graph L with the following properties:

- **Graph** The graph is a labeled directed multi-graph with $4N-1$ vertices, and each vertex has out-degree 2^t. It is composed of gadgets denoted by $D_h, h \in H$, each of which has three vertices. The gadgets are arranged into a chain according to an arbitrary order of the elements of H, and two consecutive gadgets are connected through an intermediate vertex. Let $h_1, \ldots, h_N$ be an ordering of the functions in H and let $t_1, \ldots, t_{N-1}$ be the intermediate vertices.

- **Gadget** Let $h = h_r, 1 < r < N$, be the rth element in the ordering. D_h is a graph on three vertices. Let us call these vertices v_h^1, v_h^2, v_h^3. v_h^1, and v_h^3 are the *end points* of the gadget and v_h^2 is the *mid point* of the gadget. All edges leaving the midpoint go to one of the end points, and the number of edges are equally distributed among the two end points. Of the 2^t edges leaving the left(resp. right) endpoint, half of them go to the midpoint and remaining half goes to t_{r-1} (resp. t_r). For the case when $r = 1$ (resp. $r = N$), all the edges from the left (resp. right) end point go to the mid point. For $1 \leq i \leq N-1$, half the edges leaving t_i go to the right end point of D_{h_i}, and the rest go to the left end point of $D_{h_{i+1}}$.

- **Edge labels** The edges leaving $t_i, 1 \leq i \leq N-1$ are labeled bijectively with $\{0,1\}^t$. Similarly, the edges leaving the midpoint of every gadget are labeled bijectively with $\{0,1\}^t$. Now we have to define the labels on the edges leaving the end points of a gadget. Let us consider the gadget corresponding to h. $h(x)$ is the label on an edge from the left (resp. right) endpoint to the mid point if and only if x is the label on an edge from the midpoint to the left (resp. right) end point. The elements not in the range of h are assigned bijectively to the remaining edges leaving the left (resp. right) endpoint.

Let Q be the transition probability matrix associated with the graph that we have constructed. For each $h \in H$, $\|Q_h - Q\| \geq \frac{1}{4}$. This follows from the following lemma where, for each $h \in H$ we exhibit a pair of vertices such that $|(Q_h)_{ij} - Q^2_{ij}| = \frac{1}{4}$. For every pair of vertices i, l in the graph, let B_{il} be the set of labels on the edges from i to l in the graph.

Lemma 5. *For every $h \in H$, let i denote the mid point of D_h, l the right end point of D_h, and j the only point outside the gadget that l is adjacent to. Then, $|(Q_h)_{ij} - Q^2_{ij}| = \frac{1}{4}$.*

Proof. Let $h \in H$. We know that $(Q_h)_{ij} = \Pr_{x \in \{0,1\}^t}(R_1(x,h) \text{ takes } i \text{ to } j)$. $R_1(x,h) = x \circ h(x)$ and all walks of length 2 from i to j have l as their intermediate vertex. Therefore, $(Q_h)_{ij} = \mu(x \in B_{il} \wedge h(x) \in B_{lj})$. Similarly, $Q^2_{ij} = Q_{il}Q_{lj} = \mu(B_{il})\mu(B_{lj})$. Furthermore,

$$|\mu(x \in B_{il} \wedge h(x) \in B_{lj}) - \mu(B_{il})\mu(B_{lj})| = \frac{1}{4},$$

This is because, for every ordered triple of vertices (i, l, j) that is a walk of length two in L, $\mu(B_{il})\mu(B_{lj}) = \frac{1}{4}$, and by construction $\mu(x \in B_{il} \wedge h(x) \in B_{lj}) = 0$. Therefore, $|(Q_h)_{ij} - Q^2_{ij}| = \frac{1}{4}$. Hence the lemma. □

From the above construction we then have for a family of size N, a graph of size $4N - 1$, such that no function in the family can be used in Nisan's approximate squaring algorithm to get a matrix which is within an error lesser than $\frac{1}{4}$. In the matrix powering algorithm, we work with stochastic matrices whose dimension is 2^S. Consequently, if H is a family of functions such that, for every stochastic matrix Q of order 2^S, there is a function $h \in H$ such that $\|Q_h - Q^2\| < \frac{1}{4}$ then $2^S < 4|H| - 1$. This gives the lower bound on the size of the family of functions. The lower bound is $|H| > 2^{S-2}$. We summarize the above as the following theorem.

Theorem 6. *If H is a family of functions such that for every stochastic matrix Q of size 2^S there is a $h \in H$ such $\|Q_h - Q^2\| < \frac{1}{4}$, then $|H| > 2^{S-2}$.*

Proof. The discussion preceding the statement proves the theorem. □

6 Conclusion

One contribution of this paper is that we have explored and ruled out one way of getting improvements in the space used by Nisan's algorithm to derandomize space bounded randomized algorithms. In particular, we have answered the second question posed in Section 3 by showing that there do not exist significantly smaller families of hash functions that can be used in place of the one used in Nisan's algorithm. This observation provides useful insights into one of the many stumbling blocks in the way of designing pseudorandom generators to derandomize space S bounded randomized algorithms in $o(S^2)$ space.

Using the same technique to prove a lower bound on the size of hash families, we obtain an explicit boolean function that has exponential OBDD complexity. While functions with exponential lower bounds for OBDDs are known, our result is the first that obtains a hard function from a derandomization for this computation model. An open question that is of interest is whether it is possible to obtain from pseudorandom generators, functions of exponential complexity for generalizations of OBDDs like Randomized OBDDs and Nondeterministic OBDDs. Apart from proving a lower bound for OBDDs, the function we have constructed answers the first question we have asked in Section 3 in the affirmative. One natural interesting question in this line of research is whether a *hard* function for OBDDs can be used to generate pseudorandom bits that cannot be distinguished by space bounded randomized algorithms. Such results are well known in the circuit setting, where hard functions (with varying degrees of hardness) are used to design pseudorandom generators whose output cannot be distinguished from random by polynomial-time randomized algorithms.

Acknowledgements The first author would like to thank Jan Johannsen for his encouragement and criticisms of earlier versions of this report, Vinay and Ramesh for commenting on many ideas and, Avi Wigderson for answering questions regarding the issue of Hardness vs Randomness.

References

[1] Roy Armoni, Amnon Ta-Shma, Avi Wigderson, Shiyu Shou, $SL \subseteq L^{\frac{4}{3}}$, *Proc. 29th ACM Symposium on Theory of Computing, 1997,pp. 230-239.*

[2] R.E. Bryant, On the Complexity of VLSI implementation and Graph Representations of Boolean functions with application to Integer multiplication, *IEEE Trans. Computers, C-40(2):205-213, Feb 1991.*

[3] N. Nisan, Pseudorandom generators for space bounded computation, *Proc. 22nd ACM Symposium on Theory of Computing, 1990,pp. 204-212.*

[4] N. Nisan, A. Wigderson, Hardness vs. Randomness, *Journal of Computer and System Sciences, October 1994, 49(2), pp. 149-167.*

[5] M. Saks, Randomization and Derandomization in Space-bounded Computation, *Computational Complexity, 1996, pp. 128-149.*

[6] M. Saks and S. Zhou, $RSPACE(S) \subseteq DSPACE(S^{\frac{3}{2}})$, *Proc. 36th Symposium on Foundations of Computer Science, 1995,pp. 344-353.*

[7] I. Wegener, Branching Programs and Binary Decision Diagrams. Theory and Applications, *SIAM Monographs on Discrete Mathematics and Applications, 2000*