Average Case Lower Bounds for Monotone Switching Networks

Yuval Filmus, Toniann Pitassi, Robert Robere and Stephen A. Cook
Department of Computer Science
University of Toronto
Toronto, Canada
{*yuvalf, toni, robere, sacook*}*@cs.toronto.edu*

***Abstract*—An approximate computation of a Boolean function by a circuit or switching network is a computation in which the function is computed correctly on the majority of the inputs (rather than on all inputs). Besides being interesting in their own right, lower bounds for approximate computation have proved useful in many subareas of complexity theory, such as cryptography and derandomization. Lower bounds for approximate computation are also known as correlation bounds or average case hardness. In this paper, we obtain the first average case monotone depth lower bounds for a function in monotone P. We tolerate errors that are asymptotically the best possible for monotone circuits. Specifically, we prove average case exponential lower bounds on the size of monotone switching networks for the GEN function. As a corollary, we separate the monotone NC hierarchy in the case of errors — a result which was previously only known for exact computations. Our proof extends and simplifies the Fourier analytic technique due to Potechin [21], and further developed by Chan and Potechin [8]. As a corollary of our main lower bound, we prove that the communication complexity approach for monotone depth lower bounds does not naturally generalize to the average case setting.**

***Keywords*-switching-networks**

I. Introduction

In this paper, we study the average case hardness of monotone circuits and monotone switching networks. The first superpolynomial lower bounds on monotone circuit size is the celebrated result due to Razborov [24], who showed that the clique function requires exponential-size monotone circuits, and thus also requires large monotone depth. His result was improved and generalized by many authors to obtain other exponential lower bounds for monotone circuits (for example [1], [2], [4], [14], [15]). All of these bounds are average case lower bounds for functions that lie outside of monotone P. The best known average case lower bound for an explicit function is for Andreev's polynomial problem, which is $(1/2-1/n^{1/6})$-hard for subexponential-size circuits (follows from [4]); that means that there is a subexponential function $f(n)$ such that every circuit of size at most $f(n)$ differs from Andreev's polynomial problem on at least a $(1/2 - 1/n^{1/6})$-fraction of the inputs. For an excellent survey on the applications of lower bounds on approximate computations, see [5].

Beginning in 1990, lower bound research was aimed at proving monotone size/depth tradeoffs for *efficient* functions that lie inside monotone P. The first such result, due to Karchmer and Wigderson [18], established a beautiful equivalence between (monotone) circuit depth and the (monotone) communication complexity of a related communication game. They used this framework to prove that the NL-complete directed connectivity problem requires $\Omega(\log^2 n)$ monotone circuit depth, thus proving that monotone NL (and thus also monotone NC^2) is not contained in monotone NC^1. Subsequently Grigni and Sipser [13] used the communication complexity framework to separate monotone logarithmic depth from monotone logarithmic space. Raz and McKenzie [22] generalized and improved these lower bounds by defining an important problem called the GEN problem, and proved tight lower bounds on the monotone circuit depth of this problem. As corollaries, they separated monotone NC^i from monotone NC^{i+1} for all i, and also proved that monotone NC is a strict subset of monotone P. Unlike the earlier results (for functions lying outside of P), the communication-complexity-based method developed in these papers seems to work only for exact computations.

Departing from the communication game methodology from the 1990's, Potechin [21] recently introduced a new Fourier-analytic framework for proving lower bounds for monotone switching networks. Potechin was able to prove using his framework a $n^{\Omega(\log n)}$ size lower bound for monotone switching networks for the directed connectivity problem. (A lower bound of $2^{\Omega(t)}$ on the size of monotone switching networks implies a lower bound of $\Omega(t)$ on the depth of monotone circuits, and thus the result on monotone switching networks is stronger.) Recently, Chan and Potechin [8] improved on [21] by establishing a $n^{\Omega(h)}$ size lower bound for monotone switching networks for the GEN function, and also for the clique function. Thus, they generalized most of the lower bounds due to Raz and McKenzie to monotone switching networks. However, again their lower bounds apply only for monotone switching networks that compute the function correctly on every input.

In this paper we obtain the first *average case* lower bounds on the size of monotone switching networks (and thus also the first such lower bounds on the depth of monotone circuits) for functions *inside* monotone P. We prove our lower bounds by generalizing the Fourier-analytic technique due to Chan and Potechin. In the process we first give a new

presentation of the original method, which is simplified and more intuitive. Then we show how to generalize the method in the presence of errors, which involves handling several nontrivial obstacles.

We show that GEN is $(1/2 - 1/n^{1/3-\epsilon})$-hard for subexponential-size circuits (under a specific distribution), and that directed connectivity is $(1/2 - 1/n^{1/2-\epsilon})$-hard for $n^{O(\log n)}$-size circuits (also under a specific distribution). In comparison, it is known that *under the uniform distribution*, no monotone function is $(1/2 - \log n/\sqrt{n})$-hard even for $O(n \log n)$-size circuits [20]. A related result shows that for every ϵ there is a (non-explicit) function that is $(1/2-1/n^{1/2-\epsilon})$-hard for subexponential-size circuits under the uniform distribution [17].

As a corollary to the above theorem, we separate the levels of the NC hierarchy, as well as monotone NC from monotone P, in the average case setting. That is, we prove that for all i, there are monotone functions that can be computed exactly in monotone NC^{i+1} but such that any NC^i circuit computing the same function must have large error. And similarly, there are functions in monotone P but such that any monotone NC circuit must have large error.

This leaves open the question of whether or not the original communication-complexity-based approach due to Karchmer and Wigderson can also be generalized to handle errors. This is a very interesting question, since if this is the case, then average case monotone depth lower bounds would translate to probabilistic communication complexity lower bounds for *search problems*. Developing communication complexity lower bound techniques for search problems is an important open problem because such lower bounds have applications in proof complexity, and imply integrality gaps for matrix-cut algorithms (See [3], [16].) We show as a corollary of our main lower bound that the communication complexity approach does not generalize to the case of circuits that make mistakes.

The outline for the rest of the paper is as follows. In Section II we give background information on switching networks, the GEN function, and Fourier analysis. Section III is a summary of our main lower bound, and corollaries, including a strong separation of the monotone NC hierarchy and of monotone NC from monotone P. Section IV gives an overview of the proof of our lower bound. Technical results are left to Sections V and VI: the lower bound for exact computations is presented in Section V, and Section VI extends it to average case lower bounds. This paper is an extended abstract of [11].

II. Preliminaries

In this section we give definitions that will be useful throughout the entire course of the paper. If n is a positive integer then we define the set $[n] = \{1, 2, \ldots, n\}$. If N and m are positive integers, we denote by $\binom{[N]}{m}$ the collection of all subsets of $[N]$ of size m. The notation $\mathbf{1}$ denotes the vector all of whose entries are 1 (the length of the vector will always be clear from the context). For an input $x \in \{0,1\}^n$, we denote the ith index of x by x_i. For a pair of inputs $x, y \in \{0,1\}^n$, we write $x \leq y$ if $x_i \leq y_i$ for all i. A boolean function f is *monotone* if $f(x) \leq f(y)$ whenever $x \leq y$.

Assume f is a monotone boolean function. An input x is called a *maxterm* of f if $f(x) = 0$ and $f(x') = 1$ for the input x' obtained by flipping any 0 in x to a 1. Similarly, x is called a *minterm* if $f(x) = 1$ and $f(x') = 0$ for the input x' obtained by flipping any 1 in x to a 0. If f is a boolean function and $f(x) = 1$ we will call x an *accepting* instance of f, otherwise it is a *rejecting* instance.

If $P(x)$ is a boolean condition depending on an input x, then we write $[P(x)]$ to denote the boolean function associated with that condition. For example, if V is a fixed set, we denote by $[U \subseteq V]$ the function $P(U)$ which is 1 if $U \subseteq V$ and 0 otherwise.

Monotone circuits are circuits in which only AND gates and OR gates are allowed (unrestricted circuits can also use NOT gates). Such circuits always compute monotone boolean functions. Monotone P is the class of languages computed by uniform polynomial size monotone circuits. Monotone NC is the class of languages computed by uniform polynomial size monotone circuits of polylogarithmic depth. Monotone NC^i is the class of languages computed by uniform polynomial size monotone circuits of depth $O(\log^i n)$.

A. The GEN problem

We will prove lower bounds for the GEN function, originally defined by Raz and McKenzie [22].

Definition II.1. Let $N \in \mathbb{N}$, and let $L \subseteq [N]^3$ be a collection of triples on $[N]$ called *variables*. For a subset $S \subseteq [N]$, the set of points *generated* from S by L is defined recursively as follows: every point in S is generated from S, and if i, j are generated from S and $(i, j, k) \in L$, then k is also generated from S. (If L were a collection of pairs instead of a collection of triples, then we could interpret L as a directed graph, and then the set of points generated from S is simply the set of points reachable from S.)

The GEN problem is as follows: given a collection of variables L and two distinguished points $s, t \in [N]$, is t generated from $\{s\}$?

Formally, an instance of GEN is given by a number N, two numbers $s, t \in [N]$, and N^3 boolean values coding the set $L \subseteq [N]^3$. For definiteness, in the remainder of the paper we fix (arbitrarily) $s = 1$ and $t = N$.

We assume that every instance of GEN throughout the rest of the paper is defined on the set $[N]$, and we use s and t to denote the start and target points of the instance. Sometimes we want to distinguish a particular variable in instances of GEN, so, if ℓ is a variable appearing in an instance I then we call (I, ℓ) a *pointed instance*.

Every DAG with in-degree at most 2 naturally defines a minterm of GEN which we call a *graph instance*.

Definition II.2. Let G be a DAG with in-degree at most 2 and a single sink t, and suppose the vertex set of G is a subset of $[N]$ not containing s. The GEN instance corresponding to G contains the following triples: for each source x, the triple (s, s, x); for each vertex z with inbound neighborhood $\{x\}$, the triple (x, x, z); for each vertex z with inbound neighborhood $\{x, y\}$, the triple (x, y, z). Such an instance is called a *graph instance* isomorphic to G. The *underlying vertex set* consists of the vertex set of G with the vertex t removed.

For a graph G, the function G-GEN is the monotone function whose minterms are all graph instances isomorphic to G.

If G does not have a unique source or a unique sink then we can simply add one and connect it to all of the sources or sinks, which is illustrated in Figure 1.

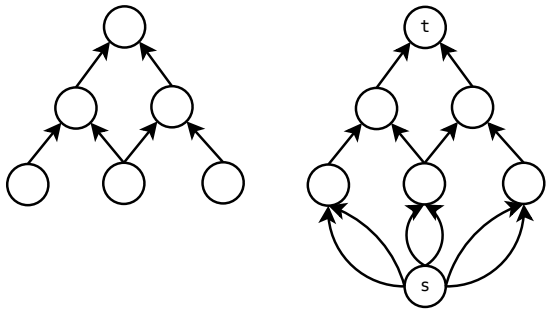

Figure 1. A pyramid graph and the corresponding GEN instance

The problem GEN is monotone: if we have an instance of GEN given by a set of variables L, and L is an accepting input for GEN, then adding any variable $l \notin L$ to L will not make L a rejecting input. Moreover, it can be computed in monotone P (we leave the proof as an easy exercise).

Theorem II.3. GEN *is in monotone* P.

Let $(C, \overline{C})$ be a partition of $[N] \setminus \{s, t\}$ into two sets. We call such a set C a *cut* in the point set $[N]$. We think of the cut C as always containing s and never containing t, and so we define $C_s = C \cup \{s\}$. Then we can define an instance $I(C)$ of GEN, called a *cut instance*, as

$$I(C) = [N]^3 \setminus \{(x, y, z) \in N^3 \mid x, y \in C_s, z \in \overline{C}_s\}.$$

The set of points generated in $I(C)$ from $\{s\}$ is precisely C_s. If $I(C)$ is a cut instance and $\ell = (x, y, z)$ is a variable with $x, y \in C_s$ and $z \in \overline{C}_s$ then we say that ℓ *crosses* C.

It might seem more natural to define C as a subset of $[N]$ containing s and not containing t. However, from the point of view of Fourier analysis, it is more convenient to remove both "constant" vertices s, t from the equations.

The set of cut instances is exactly the set of maxterms of GEN.

Proposition II.4. *Let L be an instance of* GEN. *Then L is rejecting if and only if there exists a cut $C \subseteq [N]$ such that $L \subseteq I(C)$.*

Let $\mathcal{C}$ be the collection of subsets of $[N] \setminus \{s, t\}$, and note that every set $C \in \mathcal{C}$ can be identified with a cut (and therefore a cut instance). We also note that $|\mathcal{C}| = 2^{N-2}$.

B. Vectors and Vector Spaces

In this section we recall some definitions from linear algebra. Consider the set of all cuts $\mathcal{C}$ on the point set $[N]$ of GEN. A *cut vector* is a function $f : \mathcal{C} \to \mathbb{R}$, which we think of as a real-valued vector indexed by cuts $C \in \mathcal{C}$. We define an inner product on the space of cut vectors by

$$\langle f, g \rangle = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} f(C) g(C)$$

for any two cut vectors f, g. Two cut vectors f, g are *orthogonal* if $\langle f, g \rangle = 0$, and a set of vectors $\mathcal{V}$ is orthogonal if every pair of vectors in $\mathcal{V}$ is orthogonal. Using this inner product, we define the *magnitude* of a cut vector f to be $||f|| = \sqrt{\langle f, f \rangle}$.

We will also need some tools from Fourier analysis. Given a cut $U \in \mathcal{C}$, define the cut vector $\chi_U : \mathcal{C} \to \mathbb{R}$ by

$$\chi_U(C) = (-1)^{|U \cap C|}.$$

These vectors form an orthonormal basis for the space of cut vectors called the *Fourier basis*. It follows that we can write any cut vector $f : \mathcal{C} \to \mathbb{R}$ as $f = \sum_{C \in \mathcal{C}} \langle f, \chi_C \rangle \chi_C$, where $\langle f, \chi_C \rangle$ is called the *Fourier coefficient* at C. Following convention, we will denote $\langle f, \chi_C \rangle$ by $\hat{f}(C)$.

We need some useful properties of the Fourier transform. First recall *Parseval's Theorem*: let f be any cut vector, then

$$\langle f, f \rangle = \sum_{C \in \mathcal{C}} \hat{f}(C)^2, \tag{1}$$

Parseval's theorem also holds in a more general setting: If B is an orthonormal set of cut vectors then

$$\langle f, f \rangle \geq \sum_{\phi \in B} |\langle f, \phi \rangle|^2. \tag{2}$$

C. Switching Networks

In this section we introduce switching networks, which are a computation model used to capture space-bounded computation.

Definition II.5. Let $X = \{x_1, \ldots, x_n\}$ be a set of input variables. A *monotone switching network* $\mathbf{M}$ on the variables X is specified as follows. There is an underlying graph undirected $G = (V, E)$ whose nodes are called *states* and whose edges are called *wires*, with a distinguished *start state* $\mathbf{s}$ and a distinguished *target state* $\mathbf{t}$. The wires of $\mathbf{M}$ are labelled with variables from X.

Given an input $x\colon X \to \{0,1\}$ (or an assignment of the variables), the switching network responds as follows. Let $\mathbf{e}$ be a wire in the switching network, and let x_i be the variable labelling $\mathbf{e}$. The edge $\mathbf{e}$ is *alive* if $x_i = 1$, and it is *dead* if $x_i = 0$.

We say that $\mathbf{M}$ *accepts* an input x if there exists a path from $\mathbf{s}$ to $\mathbf{t}$ using only wires which are alive under x. If no such path exists, then $\mathbf{M}$ *rejects* x. The boolean function computed by $\mathbf{M}$ is $f(x) = [\mathbf{M} \text{ accepts } x]$.

Throughout the paper we follow the convention used in the previous definition and use **bold** face to denote objects in switching networks.

Monotone switching networks and monotone circuits are connected by the following result essentially proved by Borodin [6]: a monotone circuit of depth d can be simulated by a monotone switching network of size 2^d, and a monotone switching network of size s can be simulated by a monotone circuit of depth $O(\log^2 s)$. (The result holds also for non-monotone switching networks and non-monotone circuits.) See Appendix A for a proof sketch.

A *switching network for* GEN is a switching network whose input is an instance of GEN. Such a switching network is *complete* if it accepts all yes instances of GEN. It is *sound* if it rejects all no instances of GEN. A switching network which is both complete and sound computes the GEN function.

Let $\mathbf{M}$ be a switching network for GEN. We can naturally identify each state $\mathbf{u}$ in the switching network $\mathbf{M}$ with a *reachability vector* $R_{\mathbf{u}}\colon \mathcal{C} \to \{0,1\}$ for cut instances $I(C)$ defined by

$$R_{\mathbf{u}}(C) := \begin{cases} 1 & \text{if } \mathbf{u} \text{ is reachable on input } I(C), \\ 0 & \text{otherwise.} \end{cases}$$

Here are some basic properties of the reachability vectors.

Theorem II.6. *Let $\mathbf{M}$ be a switching network with start state $\mathbf{s}$ and target state $\mathbf{t}$.*

1) $R_{\mathbf{s}} \equiv 1$.
2) *If $\mathbf{M}$ is sound then $R_{\mathbf{t}} \equiv 0$.*
3) *If $\mathbf{u}$ and $\mathbf{v}$ are two states connected by a wire labelled ℓ and C is a cut with $\ell \in I(C)$ then $R_{\mathbf{u}}(C) = R_{\mathbf{v}}(C)$.*

D. Reversible Pebbling for GEN

Next we discuss the reversible pebbling game on graphs, which is a space-efficient way to perform reachability tests on graphs. The particular form of this test gives an algorithm for GEN by applying it to the underlying graph of a GEN instance.

Definition II.7. Let $G = (V, E)$ be a directed acyclic graph (DAG) with a unique source s and a unique sink t. For a node $v \in V$, let $P(v) = \{u \in V : (u, v) \in E\}$ be the set of all incoming neighbors of v. We define the *reversible pebbling game* as follows. A *pebble configuration* is a subset $S \subseteq V$ of "pebbled" vertices. For every $x \in V$ such that $P(x) \subseteq S$, a *legal pebbling move* consists of either pebbling or unpebbling x, see Figure 2. Since s is a source, $P(s) = \emptyset$, and so we can always pebble or unpebble it.

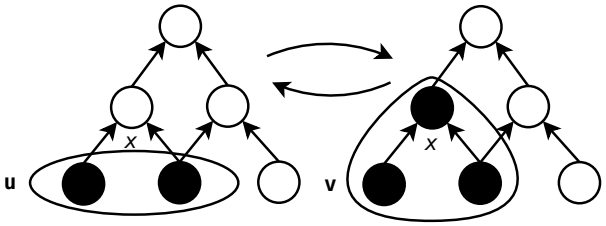

Figure 2. Legal pebbling moves involving x; the corresponding pebbling configurations are u and v

The goal of the reversible pebbling game is to place a pebble on t, using only legal pebbling moves, starting with the empty configuration, while minimizing the total number of pebbles used simultaneously. Formally, we want to find a sequence of pebbling configurations $S_0 = \emptyset, S_1, \ldots, S_n$ such that $t \in S_n$, and for each $i \in \{0, \ldots, n-1\}$, the configuration S_{i+1} is reachable from configuration S_i by a legal pebbling move. We call such a sequence a *valid pebbling sequence* for G. The *pebbling cost* of the sequence is $\max(|S_0|, \ldots, |S_n|)$. The *reversible pebbling number* of a DAG G is the minimal pebbling cost of a valid reversible pebbling sequence for G.

The more common *black pebbling game* has the same rules for pebbling a node $x \in V$, but always allows unpebbling a node. The *black pebbling number* of a dag G is the minimal pebbling cost of a valid black pebbling sequence for G. Every valid pebbling sequence is also a valid black pebbling sequence, and so the black pebbling number is upper-bounded by the reversible pebbling number.

We will need the following classes of directed graphs. A *pyramid graph with h levels* has a vertex set V which can be partitioned into h subsets, $V_1, V_2, \ldots, V_h$ (called levels), where V_i has i vertices. Let $V_i = \{v_{i1}, v_{i2}, \ldots, v_{ii}\}$. For each $i \in [h-1]$, if v_{ij} and $v_{i,j+1}$ are a pair of adjacent vertices in layer V_i, then there are edges $(v_{ij}, v_{i+1,j-i-1})$ and $(v_{i,j+1}, v_{i+1,j-i-1})$. (For example, the graph in Figure 2 is a pyramid with 3 levels.) A *directed path of length n* has vertices $V = \{v_1, \ldots, v_n\}$ and edges $E = \{(v_1, v_2), \ldots, (v_{n-1}, v_n)\}$.

Dymond and Tompa [10] showed that the reversible pebbling number of *any* graph with n vertices and in-degree 2 is $O(n/\log n)$. We have the following theorem regarding pebbling numbers of different classes of graphs.

Theorem II.8.

1) *If P is pyramid graph with h levels then the reversible*

pebbling number of P is $\Theta(h)$ [9]

2) *If P is a directed path of length n then the reversible pebbling number of P is $\Theta(\log n)$ [21]*
3) *There is an explicit family of DAGs G_n with in-degree 2 such that G_n has n vertices and reversible pebbling number $\Omega(n/\log n)$ [12]*

For more information on (non-reversible) pebbling games see the excellent survey of Nordström [19]. For more on reversible pebbling and its applications, see [7].

Raz and McKenzie [22] proved the following result.

Theorem II.9. *For each DAG G there is a polynomial size monotone circuit of depth $\Theta(h \log N)$ for G-GEN, where h is the reversible pebbling number of G.*

III. Statement of Results

In this paper we prove lower bounds for monotone switching networks computing GEN. Our first contribution is a simplified proof of the following theorem [8] which gives an exponential lower bound for monotone switching networks.

Theorem III.1. *Let N, m, h be positive integers satisfying $m \geq h \geq 3$, and let G be a DAG on m vertices with in-degree at most 2 and reversible pebbling number at least h. Any sound monotone switching network for GEN which accepts all graph instances isomorphic to G must have at least $\Omega(hN/m^3)^{(h-2)/3}/O(m)$ states.*

Corollary III.2. *For any $\epsilon > 0$, any monotone switching network which computes GEN must have at least $2^{\Omega(\epsilon N^{1/2-\epsilon})}$ states.*

We also consider monotone switching networks which are allowed to make errors. Let $\mathcal{D}$ be any distribution on instances of GEN. We say that a monotone switching network $\mathbf{M}$ computes GEN with error ϵ if the function computed by $\mathbf{M}$ differs from GEN on an ϵ-fraction of inputs (with respect to $\mathcal{D}$).

The distributions $\mathcal{D}$ we use are parametrized by DAGs. For any DAG G with in-degree at most 2 we define $\mathcal{D}_G$ to be the distribution on instances of GEN which with probability $1/2$ chooses $I(C)$ for a uniformly random cut $C \in \mathcal{C}$, and with probability $1/2$ chooses a uniformly random graph instance isomorphic to G.

Our major result in this paper is a strong extension of Theorem III.1 for switching networks computing GEN with error close to $1/2$.

Theorem III.3. *Let α be a real number in the range $0 < \alpha < 1$. Let m, h, N be positive integers satisfying $324m^2 \leq N^\alpha$ and $3 \leq h \leq m$. Let G be a DAG with m vertices, in-degree 2 and reversible pebbling number at least h.*

Any monotone switching network which computes GEN on $[N+2]$ with error $\epsilon \leq 1/2 - 1/N^{1-\alpha}$ must have at least $\Omega(hN/m^3)^{(h-2)/3}/O(mN)$ states.

Corollary III.4. *For any α in the range $0 < \alpha < 1$, any monotone switching network computing GEN with error at most $1/2 - 1/N^{1-\alpha}$ must have at least $2^{\Omega((1-\alpha)N^{\alpha/2})}$ states.*

Using this theorem, we get the following corollary separating NC^i and NC^{i+1} in the presence of errors. Recall from Theorem II.8 that the pyramid graph with height h has reversible pebbling number $\Theta(h)$ and $m = h(h+1)/2$ nodes.

Theorem III.5. *Let $0 < \delta < 1/3$ be any real constant. For each positive integer i there exists a language L which is computable in monotone NC^{i+1}, but there is no sequence of circuits in monotone NC^i which computes L on inputs of length k with error $\epsilon \leq 1/2 - 1/k^{1/3-\delta}$.*

Similarly, we can separate monotone NC from monotone P.

Theorem III.6. *Let $0 < \delta < 1/3$ be any real constant. There exists a language L which is computable in monotone P, but there is no sequence of circuits in monotone NC which computes L on inputs of length k with error $\epsilon \leq 1/2 - 1/k^{1/3-\delta}$.*

We also get the following result for directed connectivity which approaches the optimal result mentioned in the introduction (albeit with a different input distribution).

Theorem III.7. *Let $0 < \delta < 1/2$ be any real constant. For $k = N^2$, let f be the function whose input is a directed graph on N vertices, and $f(G) = 1$ if in the graph G, the vertex N is reachable from the vertex 1. There exists a distribution $\mathcal{D}$ on directed graphs such that any monotone switching network computing f with error $\epsilon < 1/2 - 1/k^{1/2-\delta}$ (with respect to $\mathcal{D}$) contains at least $N^{\Omega(\log N)}$ states.*

Another corollary of our result concerns randomized Karchmer-Wigderson games. For a boolean function $f\colon \{0,1\}^n \to \{0,1\}$ let $R_f \subseteq \{0,1\}^{2n} \times [n]$ be the following relation associated with f. If $\alpha \in f^{-1}(1)$ and $\beta \in f^{-1}(0)$, then $R_f(\alpha, \beta, i)$ holds if and only if $\alpha(x_i) \neq \beta(x_i)$. Intuitively, if α is a 1-input of the function and β is a 0-input of the function, then i is an index where α and β differ. Similarly, if f is a monotone boolean function, then for $\alpha \in f^{-1}(1)$ and $\beta \in f^{-1}(0)$ we define $R_f^m(\alpha, \beta, i)$ if and only if $\alpha(x_i) = 1$ and $\beta(x_i) = 0$.

Karchmer and Wigderson [18] proved that for any boolean function, the minimum depth of any circuit computing f is equivalent to the communication complexity of R_f, and similarly for any monotone boolean function, the minimum monotone circuit depth for f is equivalent to the communication complexity of R_f^m. We are interested in whether there is an analog of the Karchmer-Wigderson result in the case of circuits and communication complexity protocols that make errors, as an analog of Karchmer-Wigderson in

the average case setting, together with our new lower bounds for monotone circuits, would imply a new technique for obtaining average case communication complexity lower bounds for search problems.

It is not hard to see that the Karchmer-Wigderson protocol still works correctly for circuits with errors. For non-monotone functions, the reverse direction does not hold as R_f has an efficient randomized protocol for all functions f [23], contradicting the fact that almost all boolean functions require exponential-size circuits even to approximate.

Using Theorem III.3 we can show the following (see [11, Section 7] for a proof).

Theorem III.8. *The monotone Karchmer-Wigderson reduction does not hold in the average-case setting. That is, there is a monotone function $f\colon \{0,1\}^n \to \{0,1\}$, a distribution $\mathcal{D}$ and an ϵ satisfying $0 < \epsilon < 1/2$, such that there is an efficient protocol for R_f^m with error at most ϵ with respect to $\mathcal{D}$ but such that any subexponential-size monotone circuit for f of depth n^ϵ has error greater than ϵ with respect to $\mathcal{D}$.*

IV. Overview of Proof

In this section we give an intuitive overview of our proof. We focus on a set of minterms and maxterm over a ground set of size N. The minterms are height h, size m *pyramids*[1], where $m = O(N^{1/3})$, and the maxterms are *cuts*, given by a subset C of the vertices containing s and not containing t. The instance $I(C)$ corresponding to C contains all triples except for (i,j,k) where i and j are in C and k is not in C. Our minterms will consist of a special exponential-sized family of pyramid instances, $\mathcal{P}$, with the property that their pairwise intersection is at most h, and our maxterms, $\mathcal{C}$, will consist of all cuts. Given a monotone switching network $(\mathbf{M}, \mathbf{s}, \mathbf{t})$ solving GEN over $[N]$, for each state $\mathbf{v}$ we let $R_\mathbf{v}$ be it's reachability function.

At the highest level, the proof is a bottleneck counting argument. For each pyramid $P \in \mathcal{P}$, we will construct a function g_P from the set $\mathcal{C}$ of all cuts to the reals. This function will satisfy three properties.

(1) For every $P \in \mathcal{P}$ there is a "complex" state v_P in the switching network such that $\langle g_P, R_{\mathbf{v}_P}\rangle = \Omega(1/|\mathbf{M}|)$.
(2) g_P only depends on coordinates from P, and g_P has zero correlation with any function which depends on at most h coordinates.
(3) Finally, $\|g_P\|$ is upper-bounded by $m^{O(h)}$.

The first property tells us that for every pyramid $P \in \mathcal{P}$, there is a complex state $\mathbf{v}_P$ in the network that is specialized for P. The second property, together with the fact that the P's in $\mathcal{P}$ are pairwise disjoint, will imply that the functions $\{g_P : P \in \mathcal{P}\}$ are orthogonal, and thus we have

$$1 = \langle R_\mathbf{v}, R_\mathbf{v}\rangle \geq \sum_{P\in\mathcal{P}} |\langle R_\mathbf{v}, \frac{g_P}{\|g_P\|}\rangle|^2 = \frac{1}{m^{O(h)}} \sum_{P\in\mathcal{P}} \langle g_P, R_\mathbf{v}\rangle^2.$$

By the third property, $\|g_P\|$ is small, and thus it follows that no state $\mathbf{v}$ cannot be complex for more than $N^2 m^{O(h)}$ different P's (since otherwise, the quantity on the right side of the above equation would be greater than 1.) This together with the fact that $|\mathcal{P}|$ is very large, so that $|\mathcal{P}|/(N^2 m^{O(h)})$ is still exponentially large, imply our lower bound.

It remains to come up with the magical functions g_P. For the rest of this overview, fix a particular pyramid P. How can we show that some state in the switching network is highly specific to P? We will use the original Karchmer-Wigderson intuition which tells us that in order for a monotone circuit to compute GEN correctly (specifically, to output "1" on P and "0" on all cuts), it must for every cut C produce a witness variable $l \in P$ that is not in $I(C)$. And (intuitively) because different variables must be used as witnesses for different cuts, this should imply that a non-trivial amount of information must be remembered in order to output a good witness. This intuition also occurs in many information complexity lower bounds.

The above discussion motivates studying progress (with respect to our fixed pyramid P, and all cuts), by studying progress of the associated search problem. To this end, for each pyramid $P \in \mathcal{P}$ and variable $\ell \in P$, we will consider ℓ-nice functions $g_{P,l}$, where ℓ-nice means that $g_{P,\ell}(C) = 0$ whenever $I(C)(\ell) = 1$. We will think of an ℓ-nice function $g_{P,\ell}$ as a "pseudo" probability distribution over no instances (cuts) that puts zero mass on all cuts C such that $I(C)(\ell) = 1$. ($g_{P,\ell}$ is not an actual distribution since it attains both positive and negative values.) For a state $\mathbf{u}$ in the switching network, the inner product $\langle R_\mathbf{u}, g_{P,\ell}\rangle$ will be our measure of progress of the GEN function with respect to the pyramid P, on no instances where the witness *cannot* be ℓ. In order for $g_{P,\ell}$ to behave like a distribution, we will require that $\langle g_{P,\ell}, 1\rangle = 1$

Because $R_\mathbf{s}$ accepts all cuts, it follows that $\langle R_\mathbf{s}, g_{P,\ell}\rangle = 1$, which we interpret as saying that at the start state, we have made no progress on rejecting the pseudo-distribution defined by $g_{P,\ell}$. Similarly, because $R_\mathbf{t}$ rejects all cuts, it follows that $\langle R_\mathbf{t}, g_{P,\ell}\rangle = 0$, which we interpret as saying that at the final state, we have made full progress since we have successfully rejected the entire pseudo-distribution defined by $g_{P,\ell}$.

For our yes instance P and some variable $\ell \in P$, let $\mathbf{p} = \mathbf{s}, \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_q, \mathbf{t}$ be an accepting computation path on P. If we trace the corresponding inner products along the path, $\langle R_\mathbf{s}, g_{P,\ell}\rangle$, $\langle R_{\mathbf{u}_1}, g_{P,\ell}\rangle$, $\ldots$, $\langle R_\mathbf{t}, g_{P,\ell}\rangle$, they will start with value 1 and go down to 0. Now if $\mathbf{u}$ and $\mathbf{v}$ are adjacent states in the switching network connected by the variable ℓ, then progress at $\mathbf{u}$ with respect to $g_{P,\ell}$ is the same as progress at $\mathbf{v}$ with respect to $g_{P,\ell}$. This is because

[1]Our theorem is presented more generally for any fixed graph of size roughly N^ϵ but for simplicity of the proof overview, we will restrict attention to pyramid yes instances.

the pseudo-distribution defined by $g_{P,\ell}$ ignores inputs where ℓ crosses the cut (they have zero "probability"), and all other cuts reach $\mathbf{u}$ iff they reach $\mathbf{v}$.

This allows us to invoke a crucial lemma that we call the gap lemma, which states the following. Fix an accepting path $\mathbf{p}$ for the yes instance P. Then since for every $\ell \in P$, $\langle g_{P,\ell}, R_{\mathbf{s}} \rangle = 1$ and $\langle g_{P,\ell}, R_{\mathbf{t}} \rangle = 0$, and for every pair of adjacent states $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ along the path, one of these inner products doesn't change, then there must exist some node $\mathbf{v}$ on the path and two variables $\ell_1, \ell_2 \in P$ such that $|\langle g_{P,\ell_1}, R_{\mathbf{v}} \rangle - \langle g_{P,\ell_2}, R_{\mathbf{v}} \rangle| \geq 1/|M|$. Thus the gap lemma for P implies that this state $\mathbf{v}$ behaves significantly differently on the two pseudo-distributions g_{P,ℓ_1} and g_{P,ℓ_2} of cuts, and therefore this node can distinguish between these two pseudo-distributions. We will let $g_P = g_{P,\ell_1} - g_{P,\ell_2}$ be the pseudo-distribution associated with P. In summary, the gap lemma implies that for every yes instance P, we have a pseudo-distribution g_P and a "complex state" in the switching network which is highly specific to g_P. Thus we have shown property (1) above.

In order to boost the "complex state" argument and get an exponential size lower bound, as explained earlier, we still need to establish properties (2) and (3). The construction proceeds in two steps: first we construct functions $g_{P,\ell}$ satisfying properties (2) but whose norm is too large, and then we fix the norm. Our construction is the same as in earlier papers, and this is the essential place in the proof where the pebbling number of the graph comes into play. (For pyramid graphs, the pebbling number is $\Theta(h)$, where h is the height of the pyramid.) The construction, while natural, is technical and so we defer it to the extended version of this paper [11].

Now we want to generalize the above argument to switching networks that are allowed to make errors. Several important things go wrong in the above argument. First, the set $\mathcal{P}$ of pyramids that we start with above may not be accepted by the network, and in fact it might even be that none of them are accepted by the network. Secondly, it is no longer true that $\langle g_{P,\ell}, R_{\mathbf{t}} \rangle = 0$ as required in order to apply the gap lemma, because now there may be many cuts which are incorrectly accepted by the switching network.

The first problem can be easily fixed since if a random pyramid is accepted by the network, then we can still find a large design consisting of good pyramids, by taking a random permutation of a fixed design. Solving the second problem is more difficult. In the worst case, it may be that $\langle g_{P,\ell}, R_{\mathbf{t}} \rangle \neq 0$ for all $g_{P,\ell}$, and so the gap lemma cannot be applied at all. To address this issue, we will say that a pyramid is *good* for a network if it is both accepted by the network, and if $\langle g_{P,\ell}, R_{\mathbf{t}} \rangle$ is small (say less than $1/2$) for *some* $\ell \in P$. We are able to prove (by estimating the second moment) a good upper bound on the probability that $\langle g_{P,\ell}, R_{\mathbf{t}} \rangle$ is large, and thus we show that with constant probability, a random pyramid is good. Then we generalize our gap lemma to obtain an "approximate" gap lemma which essentially states that as long as the inner products with $R_{\mathbf{t}}$ are not too close to 1, then we can still find a complex state for P in the network; in fact, it is enough that *one* inner product is not too close to 1. Using these two new ingredients, we obtain average case exponential lower bounds for monotone switching networks for GEN.

V. Lower Bounds for Exact Computation

In this section we give a proof of Theorem III.1, which is the main result[2] of [8]. As discussed in the overview, the proof makes use of ℓ-nice vectors, which we proceed to define.

Definition V.1. Let $g \colon \mathcal{C} \to \mathbb{R}$ be a cut vector and $\ell \in [N]^3$. Recall that for each cut C in GEN we associate a cut instance $I(C)$. We say that g is *ℓ-nice* if $\langle g, \mathbf{1} \rangle = 1$ and $g(C) = 0$ whenever $\ell \notin I(C)$.

In a sense, vectors which are ℓ-nice are "ignorant" of cuts which are crossed by the variable ℓ. This has the following implication.

Lemma V.2. *Let $\ell \in [N]^3$ and let $\mathbf{M}$ be a monotone switching network for* GEN. *If a cut vector g is ℓ-nice then for any pair of states $\mathbf{u}, \mathbf{v} \in \mathbf{M}$ connected by a wire labelled ℓ we have $\langle R_{\mathbf{u}}, g \rangle = \langle R_{\mathbf{v}}, g \rangle$.*

Proof: Suppose $\mathbf{u}, \mathbf{v} \in \mathbf{M}$ are connected by a wire labelled ℓ. Theorem II.6 shows that $R_{\mathbf{u}}(C) = R_{\mathbf{v}}(C)$ whenever $\ell \in I(C)$. Since $g(C) = 0$ whenever $\ell \notin I(C)$ it follows that $R_u(C)g(C) = R_v(C)g(C)$ for all $C \in \mathcal{C}$ and so it must be that $\langle R_{\mathbf{u}}, g \rangle = \langle R_{\mathbf{v}}, g \rangle$. ∎

For each graph instance P of GEN we will come up with ℓ-nice functions $g_{P,\ell}$ for each $\ell \in P$. By tracing out the inner products of $g_{P,\ell}$ with reachability vectors along an accepting path for P, we will be able to come up with a complex state specific to P. To find the complex state, we make use of the following arithmetic lemma.

Lemma V.3. *Let ℓ, m be integers, and let $x_{t,i}$ be real numbers, where $0 \leq t \leq \ell$ and $1 \leq i \leq m$. Suppose that for all $t < \ell$ there exists i such that $x_{t,i} = x_{t+1,i}$. Then*

$$\max_{t,i,j} |x_{t,i} - x_{t,j}| \geq \frac{1}{2\ell} \max_i |x_{\ell,i} - x_{0,i}|.$$

The next lemma is a generalization of the gap lemma from [8]; it produces states in any switching network computing GEN which are highly specific to the vectors $g_{P,\ell}$.

Lemma V.4 (Generalized Gap Lemma)**.** *Let P be an accepting instance of* GEN*, and let $\{g_\ell\}_{\ell \in P}$ be a collection of vectors indexed by variables in P such that for each $\ell \in P$*

[2]The result proved in [8] is stated only for pyramids, but as noted in [7], the proof works for any DAG with in-degree at most 2 once we replace h with the reversible pebbling number of the graph.

the corresponding vector g_ℓ is ℓ-nice. Let $\mathbf{M}$ be a monotone switching network for GEN *with n states. Let $\{R_{\mathbf{u}}\}_{\mathbf{u}\in\mathbf{M}}$ be the set of reachability vectors for $\mathbf{M}$, and let W be an $\mathbf{s}$ to $\mathbf{t}$ path in $\mathbf{M}$ which accepts P. Suppose that for some $\ell \in P$ we have $\langle R_{\mathbf{t}}, g_\ell\rangle \leq \beta$, where $\mathbf{t}$ is the target state of $\mathbf{M}$. Then there is a node $\mathbf{u}$ on W and two variables $\ell_1, \ell_2 \in P$ for which*

$$|\langle R_{\mathbf{u}}, g_{\ell_1} - g_{\ell_2}\rangle| \geq \frac{|1-\beta|}{2n}.$$

Proof: Denote the nodes on W by $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$, where $\mathbf{u}_1 = \mathbf{s}$ and $\mathbf{u}_m = \mathbf{t}$. Apply Lemma V.3 with $x_{t,\ell} = \langle R_{\mathbf{u}_t}, g_\ell\rangle$ to obtain a node $\mathbf{u}$ and two variables ℓ_1, ℓ_2 such that

$$\begin{aligned}|\langle R_{\mathbf{u}}, g_{\ell_1} - g_{\ell_2}\rangle| &= |\langle R_{\mathbf{u}}, g_{\ell_1}\rangle - \langle R_{\mathbf{u}}, g_{\ell_2}\rangle| \\ &\geq \frac{1}{2m}\max_\ell |\langle R_{\mathbf{s}}, g_\ell\rangle - \langle R_{\mathbf{t}}, g_\ell\rangle| \geq \frac{|1-\beta|}{2n}.\end{aligned}$$

■

The following theorem from [8] gives the desired collection of nice vectors satisfying the required constraints in order to carry out the argument that we outlined earlier. We leave a simplified proof of this theorem to the extended version of this paper [11].

Theorem V.5. *Let m and h be positive integers. Let P be a graph instance of* GEN *with vertex set V_P isomorphic to a graph G with m vertices and reversible pebbling number at least h. There exist cut vectors $g_{P,\ell}$ for each $\ell \in P$ with the following properties:*

1) *For any $\ell \in P$, $\langle g_{P,\ell}, \mathbf{1}\rangle = 1$.*
2) *For any $\ell \in P$, $g_{P,\ell}$ is ℓ-nice.*
3) *For any $\ell \in P$, $g_{P,\ell}$ depends only on vertices in V_P.*
4) *For any $\ell \in P$, $\|g_{P,\ell}\|^2 \leq (9m)^{h+1}$.*
5) *For any $\ell_1, \ell_2 \in P$ and $S \in \mathcal{C}$ of size $|S| \leq h-2$, $\hat{g}_{P,\ell_1}(S) = \hat{g}_{P,\ell_2}(S)$.*

The third property implies that the function constructed by the gap lemma is specific to P. The final property shows that all the small Fourier coefficients of $g_{P,\ell_1} - g_{P,\ell_2}$ vanish. To take advantage of this property, we employ a combinatorial design in which any two sets intersect in fewer than h points.

Lemma V.6 (Trevisan [25])**.** *For any positive integers q, m, h with $h \leq m$, there exist q sets $Q_1, Q_2, \ldots, Q_q \subseteq [N]$, where $N = m^2 e^{1+\ln(q)/h}/h$, such that $|Q_i| = m$ for each i and $|Q_i \cap Q_j| \leq h$ for each $i \neq j$.*

We are now ready to put the pieces together to prove an exponential lower bound on the size of monotone switching networks for GEN that are correct on all inputs.

Proof of Theorem III.1: Lemma V.6 gives a design $\{Q_1, \ldots, Q_q\}$ of size $q = ((h-2)N/e(m-1)^2)^{h-2}$ in which $|Q_i| = m-1$ and $|Q_i \cap Q_j| \leq h-2$ for all $i \neq j$. For each Q_i, choose some graph instance P_i isomorphic to G whose underlying vertex set is Q_i. Apply Theorem V.5 to each instance P_i to get a collection $\{g_{P_i,\ell}\}_{\ell\in P_i}$ of cut vectors. Note that $m \geq h \geq 3$ implies that $q \geq (hN/4em^2)^{h-2}$.

Let $\mathbf{M}$ be a sound monotone switching network for GEN of size n which accepts all graph instances isomorphic to G. We apply the gap lemma (Lemma V.4) to each collection of vectors, which gives a set of vectors $\{g_{P_i}\}_{i=1}^q$ and a collection of states $\{\mathbf{u}_i\}_{i=1}^q$ such that $\langle g_{P_i}, \mathbf{u}_i\rangle \geq 1/2n$ and $g_{P_i} = g_{P_i,\ell_1} - g_{P_i,\ell_2}$ for some pair of variables $\ell_1, \ell_2 \in P_i$.

The third property in Theorem V.5 shows that g_{P_i} depends only on vertices in Q_i, and the final property guarantees that $\hat{g}_{P,i}(C) = 0$ for all $|C| \leq h-2$. It follows that for $i \neq j$, the functions g_{P_i}, g_{P_j} are orthogonal: if $\hat{g}_{P_i}(C), \hat{g}_{P_j}(C) \neq 0$ then $|C| \geq h-1$ and $C \subseteq Q_i, Q_j$, contradicting the fact that $|Q_i \cap Q_j| \leq h-2$. Finally, since $\|g_{P_i,\ell}\| \leq \sqrt{(9(m-1))^{h+1}}$ we get $\|g_{P_i}\| \leq 2\sqrt{(9m)^{h+1}}$.

Since the set $\{g_{P_i}/\|g_{P_i}\| : 1 \leq i \leq q\}$ is orthonormal, Parseval's theorem implies that

$$\begin{aligned}n &= \sum_{\mathbf{u}\in\mathbf{M}} 1 \geq \sum_{\mathbf{u}\in\mathbf{M}} \|R_{\mathbf{u}}\|^2 \geq \sum_{\mathbf{u}\in\mathbf{M}}\sum_{i=1}^{q}\left(\frac{\langle R_{\mathbf{u}}, g_{P_i}\rangle}{\|g_{P_i}\|}\right)^2 \\ &\geq q \cdot \frac{1}{4n^2}\frac{1}{4(9m)^{h+1}}.\end{aligned}$$

We deduce that

$$n^3 \geq \frac{q}{16\cdot(9m)^{h+1}} \geq \left(\frac{hN}{36em^3}\right)^{h-2}(27m)^{-3}.$$

■

VI. Average Case Lower Bounds

In this section we extend the above exponential lower bound to deterministic monotone switching networks which are allowed to make errors on some distribution of inputs[3]. We begin by defining how we measure the error rates of switching networks.

If x is an instance of GEN, then we will write $\mathsf{GEN}(x) = 1$ to denote that x is an accepting instance, and $\mathsf{GEN}(x) = 0$ to denote that x is a rejecting instance.

Definition VI.1. Let $\mathcal{D}$ be a distribution on GEN inputs, and let $0 \leq \epsilon < 1/2$ be a real number. We say that a monotone switching network $\mathbf{M}$ *computes* GEN *with ϵ error on $\mathcal{D}$* if $\Pr_{x\sim\mathcal{D}}[\mathbf{M}(x) \neq \mathsf{GEN}(x)] = \epsilon$.

The error ϵ results from a combination of two errors: the error ϵ_1 on yes instances, and the error ϵ_2 on no instances. According to the definition of $\mathcal{D}$, we have $\epsilon = (\epsilon_1 + \epsilon_2)/2$.

For the rest of the section fix a DAG G on m vertices with a unique sink t, in-degree at most 2, and reversible pebbling number at least h. We will use the following distribution $\mathcal{D}$ over instances of GEN: with probability $1/2$, choose a random graph instance isomorphic to G, and with probability $1/2$ choose $I(C)$ for a random cut C.

[3]By Yao's Minimax Theorem [26] this is equivalent to a lower bound for randomized monotone switching networks.

If $\mathbf{M}$ is a monotone switching network computing GEN with errors on some distribution, say that a instance P is *good* for $\mathbf{M}$ if P is accepted by the network and $\langle R_{\mathbf{t}}, g_{P,\ell}\rangle \leq 1 - 1/N$, where $R_{\mathbf{t}}$ is reachability vector of the target state $\mathbf{t}$ in $\mathbf{M}$, $\ell \in P$ is any variable, and $g_{P,\ell}$ is the ℓ-nice vector given by Theorem V.5.

The probabilistic method can be used to prove the next lemma, which shows that if a uniformly random instance is good with high probability, then we can find a block design with "many" good instances.

Lemma VI.2. *Let N, m, h, q be positive integers. Suppose that there are q sets $S_1, S_2, \ldots S_q \subseteq [N]$ such that the following holds:*

1) *$|S_i| = m$ for all $i \in [q]$, and*
2) *$|S_i \cap S_j| \leq h$ for all $i \neq j$.*

Additionally, suppose that at most a fraction ϵ of the sets $\binom{[N]}{m}$ have some property P. Then there exists a collection of $q' = (1-\epsilon)q$ sets $S'_1, S'_2, \ldots, S'_{q'} \subseteq [N]$ for which both properties stated above hold, and additionally none of the sets S'_i has property P.

For the rest of this section, let N be an integer and let $\mathbf{M}$ be a monotone switching network of size n computing GEN on $[N+2]$ with error ϵ. For convenience we assume that $s = N+1$ and $t = N+2$. Recall that $R_{\mathbf{t}}$ is the reachability vector for the target state of $\mathbf{M}$. For a pointed instance (P, ℓ), $g_{P,\ell}$ is the vector constructed using Theorem V.5.

We prove an upper bound on $\Pr_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell}\rangle \geq \delta]$ by estimating the second moment $\mathrm{E}_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell}\rangle^2]$ and applying Markov's inequality. However, for technical reasons, instead of estimating the second moment directly we will rather estimate a particular sum of tensor squares. The sum we are going to consider includes a pointed instance (P, ℓ) for each subset Q of $[N]$ of size m (recall $s, t \notin [N]$ in this section).

Definition VI.3. A *pointed instance function* $\wp$ associates with each set $Q \in \binom{[N]}{m}$ a graph instance P isomorphic to G and a variable $\ell \in P$.

The *tensor product* of two cut vectors u and v is the vector $u \otimes v \colon \mathcal{C}^2 \to \mathbb{R}$ defined by $(u \otimes v)(C, D) = u(C)v(D)$. It is well known that tensor products satisfy $\langle u \otimes u, v \otimes v\rangle = \langle u, v\rangle^2$ and $\widehat{f \otimes g}(C, D) = \hat{f}(C)\hat{g}(D)$. This makes tensor products a useful tool to estimate the second moment while keeping the assumption of linearity, as the next lemma shows.

Lemma VI.4. *Let $\wp$ be a pointed instance function chosen uniformly at random, and let (P, ℓ) be a random pointed instance. We have*

$$\mathop{\mathrm{E}}_{\wp}[\langle R_{\mathbf{t}} \otimes R_{\mathbf{t}}, \sum_{D \in \binom{[N]}{m}} g_{\wp(D)} \otimes g_{\wp(D)}\rangle] = \binom{N}{m} \mathop{\mathrm{E}}_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell}\rangle^2].$$

We are able to directly upper bound this expectation, although we must leave the proof to the extended version [11] due to space considerations. Then applying Markov's inequality yields an immediate corollary.

Lemma VI.5. *Suppose $N \geq 162m^2$. For a random pointed instance (P, ℓ),*

$$\mathop{\mathrm{E}}_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell}\rangle^2] \leq \epsilon_2 \left(1 + \frac{324m^2}{N}\right).$$

Corollary VI.6. *Suppose $N \geq 162m^2$. Let (P, ℓ) be a random pointed instance. For any $\delta > 0$,*

$$\Pr_{(P,\ell)}[\langle R_{\mathbf{t}}, g_{P,\ell}\rangle \geq \delta] \leq \delta^{-2} \left(1 + \frac{324m^2}{N}\right) \epsilon_2.$$

A. Exponential Lower Bounds

We are now ready to prove our main theorem (Theorem III.3), which gives an exponential lower bound on the size of monotone switching networks computing GEN with error close to $1/2$.

Proof of Theorem III.3: Lemma V.6 gives a design $\{Q_1, \ldots, Q_q\}$ of size $q = ((h-2)N/e(m-1)^2)^{h-2}$ in which $|Q_i| = m - 1$ and $|Q_i \cap Q_j| \leq h - 2$ for all $i \neq j$. Since $m \geq h \geq 3$ we note that $q \geq (hN/4em^2)^{h-2}$. Let π be a random permutation on $[N]$, and let $\wp$ be a random pointed instance function. For each Q_i, $\wp(\pi(Q_i))$ is a random pointed instance, and hence for any δ in the range $0 < \delta < 1$,

$$\Pr_{\pi,\wp}[\langle R_{\mathbf{t}}, g_{\wp(\pi(Q_i))}\rangle \geq \delta] \leq \delta^{-2} \left(1 + \frac{324m^2}{N}\right) \epsilon_2.$$

Moreover, the probability that $\wp(\pi(Q_i))$ is rejected by $\mathbf{M}$ is ϵ_1. The probability that either of these bad events happen is at most

$$\tau := \epsilon_1 + \delta^{-2} \left(1 + \frac{324m^2}{N}\right) \epsilon_2 \leq \delta^{-2} \left(1 + \frac{324m^2}{N}\right) 2\epsilon,$$

since $2\epsilon = \epsilon_1 + \epsilon_2$. The technique of Lemma VI.2 shows that for some π and $\wp$, the number of Q_i for which either $\langle R_{\mathbf{t}}, g_{\wp(\pi(Q_i))}\rangle \geq \delta$ or $\wp(\pi(Q_i))$ is rejected by $\mathbf{M}$ is at most τq. In other words, there exists a set $(P_1, \ell_1), \ldots, (P_{q^*}, \ell_{q^*})$ of $q^* := (1-\tau)q$ pointed instances with underlying sets $Q^*_1, \ldots, Q^*_{q^*}$ such that $|Q^*_i \cap Q^*_j| \leq h - 2$ for all $i \neq j$.

Apply Theorem V.5 to each instance P_i to get a collection of vectors $\{g_{P_i,\ell}\}_{\ell \in P_i}$, and then apply the generalized gap lemma (Lemma V.4) to each such collection of vectors, which yields a set of vectors $\{g_{P_i}\}_{i=1}^{q^*}$ and a set of states $\{\mathbf{u}_i\}_{i=1}^{q^*}$ in $\mathbf{M}$ satisfying $\langle g_{P_i}, R_{\mathbf{u}_i}\rangle \geq (1-\delta)/2n$ (recall n is the size of $\mathbf{M}$). This collection of vectors is orthogonal, and each vector satisfies the upper bound $\|g_{P_i}\|^2 \leq 4(9(m-1))^{h+1} \leq 4(9m)^{h+1}$. We repeat the calculation of Theorem

III.1 and get

$$n \geq \sum_{\mathbf{u}\in\mathbf{M}} \sum_{i=1}^{q^*} \left(\frac{\langle R_{\mathbf{u}}, g_{P_i} \rangle}{\|g_{P_i}\|} \right)^2 \geq q^* \cdot \frac{(1-\delta)^2}{4n^2} \frac{1}{4(9m)^{h+1}}$$
$$\geq (1-\tau)(1-\delta)^2 \left(\frac{hN}{36em^3} \right)^{h-2} (27m)^{-3}.$$

An auspicious choice for δ is $\delta = 1-1/N$. Since $\epsilon \leq 1/2 - 1/N^{1-\alpha}$ and $324m^2/N \leq N^{\alpha-1}$ we can upper bound τ by $\tau \leq 1 - 1/(N^{1-\alpha})$. Substituting δ and τ and simplifying yields

$$n^3 \geq \left(\frac{hN}{36em^3} \right)^{h-2} (27mN)^{-3}.$$

■

REFERENCES

[1] N. Alon and R. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[2] A. Andreev. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Soviet Math. Dokl*, 31(3):530–534, 1985.

[3] P. Beame, T. Pitassi, and N. Segerlind. Lower bounds for Lovász Schrijver from multiparty communication complexity. In *SIAM J. Computing*, 2007.

[4] C. Berg and S. Ulfberg. Symmetric approximation arguments for monotone lower bounds without sunflowers. *Computational Complexity*, 8:1–20, 1999.

[5] A. Bogdanov and L. Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

[6] Allan Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6(4):733–744, December 1977.

[7] Siu Man Chan. Just a pebble game. In *CCC*, 2013.

[8] Siu Man Chan and Aaron Potechin. Tight bounds for monotone switching networks via Fourier analysis. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 495–504. ACM, 2012.

[9] Stephen A. Cook. An observation on time-storage trade off. In *Proceedings of the fifth annual ACM Symposium on Theory of computing*, STOC '73, pages 29–33, New York, NY, USA, 1973. ACM.

[10] Patrick W. Dymond and Martin Tompa. Speedups of deterministic machines by synchronous parallel machines. *Journal of Computer and System Sciences*, 30(2):149–161, April 1985.

[11] Yuval Filmus, Toniann Pitassi, Robert Robere, and Stephen A. Cook. Average case lower bounds for monotone switching networks. *ECCC TR13-054*, 2013.

[12] John R. Gilbert and Robert Endre Tarjan. Variations on a pebble game on graphs. Technical Report STAN-CS-78-661, Stanford University, 1978.

[13] M. Grigni and M. Sipser. Monotone separation of logarithmic space from logarithmic depth. *JCSS*, 50:433–437, 1995.

[14] A. Haken. Counting bottlenecks to show monotone $P \neq NP$. In *FOCS*, pages 36–40, 1995.

[15] D. Harnik and R. Raz. Higher lower bounds on monotone size. In *STOC*, pages 378–387, 2000.

[16] T. Huynh and J. Nordstrom. On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space tradeoffs in proof complexity. In *44th STOC*, pages 233–248, 2012.

[17] G. Karakostas, J. Kinne, and D. van Melkebeek. On derandomization and average-case complexity of monotone functions. *Theoretical Computer Science*, 434:35–44, 2012.

[18] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 539–550, Chicago, IL, May 1988.

[19] Jakob Nordström. New wine into old wineskins: A survey of some pebbling classics with supplemental results. *Logical Methods in Computer Science*.

[20] R. O'Donnell and K. Wimmer. KKL, Kruskal-Katona, and monotone nets. In *FOCS*, pages 725–734, 2009.

[21] A. Potechin. Bounds on monotone switching networks for directed connectivity. In *FOCS*, pages 553–562, 2010.

[22] R. Raz and P. McKenzie. Separation of the monotone NC hierarchy. In *Proceedings of 38th IEEE Foundations of Computer Science*, 1997.

[23] Ran Raz and Avi Wigderson. Probabilistic communication complexity of Boolean relations. In *30th Annual Symposium on Foundations of Computer Science*, pages 562–567, Research Triangle Park, NC, October 1989. IEEE. Full version.

[24] A. A. Razborov. Lower bounds on the monotone complexity of some Boolean function. *Soviet Math. Dokl.*, 31:354–357, 1985.

[25] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[26] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science*, pages 222–227. IEEE, 1977.

APPENDIX

In this appendix we show that a monotone circuit of depth d can be simulated by a monotone switching network of size 2^d, and a monotone switching network of size s can be simulated by a monotone circuit of depth $O(\log^2 s)$. The corresponding results for the nonmonotone case (with

switching networks replaced by branching programs) were proved by Borodin [6].

The second result is easy: Given a monotone switching network of size s together with its input bits $x_1, \ldots, x_n$, we construct an $s \times s$ boolean matrix A, where $A_{ij} = 1$ iff $i = j$ or there is a wire between state i and state j with a label $x_k = 1$. Now the circuit computes A^s by squaring the matrix $\log s$ times. The network accepts the input iff $A_{ab} = 1$, where a is the initial state and b is the accepting state.

Now we show how a monotone switching network can evaluate a monotone circuit C of depth d. We may assume that C is a balanced binary tree where the root is the output gate (at level 1) and the nodes at levels 1 to d are gates, each labelled either $\wedge$ or $\vee$, and each leaf at level $d+1$ is labelled with one of the input variables x_i. The idea is to simulate an algorithm which uses a depth first search of the circuit to evaluate its gates.

We assume that the circuit is laid out on the plane with the output gate at the bottom and the input nodes at the top, ordered from left to right. A *full path* is a path in the circuit from the output gate to some input node.

The states of the network consist of a start state $\mathbf{s}$, a target state $\mathbf{t}$, and a state $\mathbf{u_p}$ for each full path p in the circuit (so there are 2^d such states).

Definition A.1. We say that a path p from a gate g in C to some input node is *initial* if p leaves every $\wedge$-gate g on the path via the left input to g. We say that p is *final* if p leaves every $\wedge$-gate via the right input to g. If p is a full path, then the state $\mathbf{u_p}$ is *initial* if p is initial, and $\mathbf{u_p}$ is *final* if p is final.

We say that a state $\mathbf{u_p}$ is *sound* (for a given setting of the input bits $\vec{x}$) if for each $\wedge$-gate g on the path p, if the path leaves g via its right input, then the left input to g has value 1.

We will construct our network so that following holds.

Claim A.1. For each input $\vec{x}$, a state $\mathbf{u_p}$ is reachable iff $\mathbf{u_p}$ is sound.

Now we define the wires and their labels in the network.

The start state $\mathbf{s}$ is connected via a wire labelled 1 (that is, this wire is always alive) to every initial state $\mathbf{u_p}$. Note that every initial state is (vacuously) sound, as required by the claim.

For every nonfinal full path p to some input x_i, $\mathbf{u_p}$ is connected by a wire labelled x_i to every state $\mathbf{u_{p'}}$ such that p' is a full path which follows p up to the last $\wedge$-gate g such that p leaves g via its left input, and then p' leaves g via its right input and continues along any initial path to a circuit input.

The following is easy to verify:

Claim A.2. It p, p' and x_i are as above, and $x_i = 1$, then $\mathbf{u_p}$ is sound iff $\mathbf{u_{p'}}$ is sound.

Claim A.1 follows from Claim A.2 and the facts that all initial states are sound, and every noninitial sound state $\mathbf{u_{q'}}$ is connected by a wire labelled 1 to a sound state $\mathbf{u_q}$ where the path q is to the left of path q'.

For every final full path p to some input x_i, the state $\mathbf{u_p}$ is connected to the target $\mathbf{t}$ by a wire labelled x_i. Note that if $\mathbf{u_p}$ is reachable and sound, and $x_i = 1$, then every gate along p has value 1, including the output gate. This and Claim A.1 shows that the network is sound (the circuit output is 1 if the target $\mathbf{t}$ in the network is reachable).

Conversely, if the circuit outputs 1, then there is a sound final full path p which witnesses it. Claim A.1 shows that $\mathbf{u_p}$ is reachable, and so $\mathbf{t}$ is reachable. We conclude that the network is complete.