Theory Comput. Systems **36**, 71–88 (2003)
DOI: 10.1007/s00224-002-1068-0

Theory of Computing Systems
© 2002 Springer-Verlag New York Inc.

Graph Complexity and Slice Functions*

Satyanarayana V. Lokam

EECS Department, University of Michigan,
Ann Arbor, MI 48109-2122, USA
satyalv@eecs.umich.edu

Abstract. A graph-theoretic approach to study the complexity of Boolean functions was initiated by Pudlák, Rödl, and Savický [PRS] by defining models of computation on graphs. These models generalize well-known models of Boolean complexity such as circuits, branching programs, and two-party communication complexity.

A Boolean function f is called a 2-*slice function* if it evaluates to zero on inputs with less than two 1's and evaluates to one on inputs with more than two 1's. On inputs with exactly two 1's f may be nontrivially defined. There is a natural correspondence between 2-slice functions and graphs. Using the framework of graph complexity, we show that sufficiently strong superlinear monotone lower bounds for the very special class of 2-slice functions would imply superpolynomial lower bounds over a complete basis for certain functions derived from them. We prove, for instance, that a lower bound of $n^{1+\Omega(1)}$ on the (monotone) formula size of an explicit 2-slice function f on n variables would imply a $2^{\Omega(\ell)}$ lower bound on the formula size over a complete basis of another explicit function g on ℓ variables, where $\ell = \Theta(\log n)$.

We also consider lower bound questions for depth-3 bipartite graph complexity. We prove a weak lower bound on this measure using algebraic methods. For instance, our result gives a lower bound of $\Omega((\log n)^3/(\log\log n)^5)$ for bipartite graphs arising from Hadamard matrices, such as the Paley-type bipartite graphs. Lower bounds for depth-3 bipartite graph complexity are motivated by two significant applications: (i) a lower bound of $n^{\Omega(1)}$ on the depth-3 complexity of an explicit n-vertex bipartite

* A preliminary version of this work appeared in *Proc*. 18*th FST* & *TCS* [L]. Results in Section 4 are substantially improved from [L]. This work was partially supported by NSF Grant CCR-9988359. Part of the work was done while the author was at the Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G4, and at the Department of Mathematical and Computer Sciences, Loyola University Chicago, Chicago, IL 60626, USA.

graph would yield superlinear size lower bounds on log-depth Boolean circuits for an explicit function, and (ii) a lower bound of $\exp((\log\log n)^{\omega(1)})$ would give an explicit language outside the class Σ_2^{cc} of the two-party communication complexity as defined by Babai, Frankl, and Simon [BFS].

Our lower bound proof is based on sign-representing polynomials for DNFs and lower bounds on ranks of ± 1 matrices even after being subjected to sign-preserving changes to their entries. For the former, we use a result of Nisan and Szegedy [NS] and an idea from a recent result of Klivans and Servedio [KS]. For the latter, we use a recent remarkable lower bound due to Forster [F1].

1. Introduction

1.1. *Slice Functions and Monotone Complexity*

Proving superlinear (superpolynomial) lower bounds on the circuit size (formula size) of an explicit Boolean function is a major challenge in computational complexity. On the other hand, remarkable results have been proved in restricted models such as constant depth circuits [H1], [R3], [S], monotone circuits [AB], [R2], [HR], and monotone formulas [RW], [KW1], [RM]. In general, techniques from these results on restricted models are considered unlikely to be useful to attack the general question. In fact, there are functions with exponential lower bounds on the monotone complexity, but with polynomial upper bounds on their complexity over a complete basis [R1], [T]. However, for the special class of *slice functions*, monotone complexity and general complexity differ only by a small polynomial amount [B], [V2], [W]. An n-variable Boolean function f is called a *k-slice function* if $f(x) = 1$ whenever $|x| > k$ and $f(x) = 0$ whenever $|x| < k$, where $|x|$ denotes the number of 1's in the input assignment x. When $|x| = k$, $f(x)$ may be nontrivially defined. A function is called a slice function if it is a k-slice function for some k, $1 \le k \le n-1$. There are NP-complete languages whose characteristic functions are slice functions [D]. Hence, it is *conceivable* that superpolynomial lower bounds on the monotone circuit complexity of slice functions can be used to attack the P versus NP question.

For the approach via slice functions described above to yield superpolynomial lower bounds, we must consider a nonconstant k, because a k-slice function has complexity at most $O(n^k)$. However, we will show in this paper that for constant k, in fact for $k = 2$, sufficiently strong superlinear lower bounds on k-slice functions would already imply superpolynomial lower bounds for some *other* functions derived from the 2-slice functions. More specifically, a lower bound of $n^{1+\Omega(1)}$ on the (monotone) formula size of an n-variable 2-slice function f implies a lower bound of $2^{\Omega(\ell)}$ on the formula size over a complete basis of an ℓ-variable function g. We remark that a lower bound of $n^{1+\Omega(1)}$ on the nonmonotone complexity of f already follows from the well-known relation between the monotone and nonmonotone complexity of slice functions [B], [V2], [W]. The magnification to an *exponential* lower bound for a different function g is the new result. We show this using the framework of graph complexity introduced by Pudlák et al. [PRS].

Note that the nontrivial part, namely when $|x| = 2$, of a 2-slice function is essentially a labeled graph. Conversely, with every labeled n-vertex graph G we can associate an n-variable 2-slice function f_G (see Definition 3.1).

1.2. *Models of Computation on Graphs*

The notion of graph complexity [PRS] is a common generalization of Boolean circuit complexity and two-party communication complexity. Measures of graph complexity such as affine dimension and projective dimension of graphs have been proposed and studied in [PR2], [R4], and [PR1] as criteria for lower bounds on formula size and branching program size of Boolean functions. Separation questions about classes of two-party communication complexity [Y], [BFS] can be reformulated as lower bound questions about bipartite graph complexity as noted in [PRS]. The generality of graph complexity is illustrated in more detail in Section 2.

In graph complexity, an n-vertex graph G is constructed or computed as follows. We are given a set of atomic n-vertex graphs called *generators*; these are analogous to the input variables in a circuit or a formula. In each step of the computation, we can perform a set-operation on the sets of edges of graphs we have constructed so far. Starting with the generators and performing the allowed operations on intermediate graphs, we would like to obtain G as the result of the computation. By stipulating the structure of the computation (such as circuits), the set of generators (such as complete bipartite graphs), and the allowed set-operations (such as union and intersection), we get various measures of the cost of the computation and a corresponding definition of the complexity of G. The complexity of G is the minimum cost of such a computation to construct G.

With an arbitrary Boolean function f on 2ℓ variables, we can naturally associate a bipartite graph G_f with color classes $\{0, 1\}^\ell$ such that (x, y) is an edge of G_f iff $f(x, y) = 1$, where x is an assignment to the first ℓ variables and y is an assignment to the last ℓ variables. Let $n = 2^\ell$, so that G_f is an $n \times n$ bipartite graph. In [PRS] it is shown that a lower bound of $\psi(\log n)$ on the circuit size (formula size) complexity of G_f, with complete bipartite graphs as generators and union and intersection as operators, would imply a lower bound of $\psi(\ell)$ on the Boolean circuit size (formula size) of f. Hence superlogarithmic (superpolylogarithmic) lower bounds on the complexity of some explicit bipartite graphs would yield superlinear (superpolynomial) lower bounds on the circuit size (formula size) of explicit Boolean functions—resolving long-standing open questions in computational complexity.

In this paper we make the simple observation that lower bounds on graph complexity are implied by lower bounds on 2-slice functions. We prove that a lower bound of $n \log n \cdot \beta(n)$, for any $\beta(n) = \omega(1)$, on the *monotone* formula size of a 2-slice function implies a lower bound of $\beta(n)$ on the formula complexity of the corresponding graph. Similar results hold for circuit size. Combining this relation with the results from [PRS] we prove that sufficiently strong superlinear lower bounds on the monotone complexity of the very special class of 2-slice functions imply exponential lower bounds on the general complexity of certain other Boolean functions.

1.3. *Lower Bounds on Graph Complexity*

Next, we consider lower bounds on graph complexity. As mentioned above, the model of graph complexity is more general than the models of Boolean circuits and two-party communication complexity. Thus, proving lower bounds on graph complexity is even harder. However, studying the graph-theoretic structure of Boolean functions may provide insights into their complexity. Understanding the properties of graphs that imply high graph complexity may suggest candidate functions with high Boolean complexity. Lower bound arguments for such Boolean functions may in turn exploit tools from the well-studied area of graph theory.

Pudlák et al. [PRS] prove some nontrivial formula size lower bounds in graph complexity. Their lower bounds are on the *star-formula* complexity of graphs. A star-formula for a graph is a formula with union and intersection operators and "stars" as the generators, where a *star* is a complete bipartite graph with a single vertex on one side and the remaining vertices on the other side. In this paper we focus on the *bipartite-formula* complexity of graphs. Here the generators are complete bipartite graphs, and the operators are union and intersection. Results in this more general model translate more readily into the frameworks of Boolean circuit complexity and two-party communication complexity.

In [PRS], a lower bound of $\Omega(n \log n)$ is proved for the star-formula complexity of some explicit bipartite graphs. This immediately implies a lower bound of $\Omega(\log n)$ on the bipartite-formula complexity. Improving this to $\Omega(\log^3 n)$ would yield (see Proposition 2.6) new lower bounds on Boolean formula complexity. Recall that currently the strongest lower bound [H2] on the formula size of an explicit Boolean function of ℓ variables is $\Omega(\ell^{3-o(1)})$. The methods used in [PRS] cannot give bounds beyond $\Omega(\log n)$ on bipartite-formula complexity. Furthermore, Pudlák et al. [PRS] also show that certain Ramsey-type properties of graphs (absence of large cliques and independent sets) *do not* imply strong enough lower bounds on graph complexity to give new results in Boolean function complexity.

Here we consider constant depth computations in graph complexity. We prove a general lower bound on *depth*-3 bipartite-formula complexity of some bipartite graphs. Our result gives a lower bound of $\Omega((\log n)^3/(\log\log n)^5)$ for explicit bipartite graphs arising from Hadamard matrices such as the Paley-type bipartite graphs. In fact, our lower bounds are expressed in terms of the spectrum of the ± 1 incidence matrix associated with the bipartite graph. A preliminary version of this paper [L] gave a lower bound of $\Omega((\log n)^2/(\log\log n)^2)$ using approximating polynomials for the OR function [NS] and lower bounds on ranks of ± 1 matrices under sign-preserving changes of small magnitude [KW2]. In this paper we improve the lower bound to $\Omega((\log n)^3/(\log\log n)^5)$. To obtain the improvement, we additionally use a trick from a recent result of Klivans and Servedio [KS] and a remarkable lower bound by Forster [F1] on ranks of ± 1 matrices under sign-preserving changes of *unbounded magnitude*.

Our lower bound is still too weak to imply any new results in Boolean circuit or communication complexity. We note that depth-3 bipartite-formula complexity is related to depth-3 Boolean formula complexity and the class Σ_2^{cc} of the two-party communication complexity model [BFS]. Lower bounds on depth-3 Boolean complexity have recently been pursued in [HJP], [PPZ], and [PSZ]. One motivation for "strongly" exponential depth-3 lower bounds comes from Valiant's [V1] result that depth-3 lower bounds of $2^{\omega(\ell/\log\log\ell)}$ for an ℓ-variable Boolean function would imply superlinear size lower bounds on log-depth circuits computing that function. Currently, the best known lower bound on depth-3 circuits is $\Omega(\ell^{1/4}2^{\sqrt{\ell}})$ for the parity function [PPZ]. One approach to develop tools for depth-3 lower bounds in Boolean complexity is to understand the corresponding (more general) question in graph complexity. Such an approach might lead to graph-theoretic criteria for depth-3 lower bounds in Boolean complexity.

A lower bound of $n^{\Omega(1)}$ on the depth-3 bipartite-formula complexity of an explicit bipartite graph would give the "strongly" exponential lower bounds mentioned above and hence would imply superlinear size lower bounds on log-depth circuits for an explicit function. However, the current lower bounds on depth-3 bipartite-formulas are too weak even to re-derive the current best lower bound of $2^{\sqrt{\ell}}$ on depth-3 Boolean formulas. In fact, *generalizing* the lower bound of $2^{\sqrt{\ell}}$ in depth-3 Boolean circuit complexity to the framework of graph complexity or even obtaining the weaker bound of $\exp((\log \ell)^{\omega(1)})$ from depth-3 bipartite graph complexity would already resolve a long-standing open question in communication complexity: a lower bound of $\exp((\log\log n)^{\omega(1)})$ on depth-3 bipartite-formula complexity of an explicit $n \times n$ bipartite graph gives an explicit language outside the class Σ_2^{cc} of two-party communication complexity. Such strong bounds on graph complexity, however, remain as interesting open questions.

2. Models of Graph Complexity

The complexity of a graph G measures the difficulty of constructing G using a given collection of primitive graphs, called *generators*, and a given basis of operations on sets of edges. All the graphs involved are assumed to have the same set of vertices, typically $V = \{1, \ldots, n\}$. A set operation on graphs refers to the operation on the corresponding edge sets. For instance, the result of $G_1 \cup G_2$ on graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ is the graph $G = (V, E_1 \cup E_2)$. Models of graph complexity are defined analogous to the standard models of circuits and formulas where the generator graphs play the role of input variables and the set operations play the role of gates.

We now give some formal definitions, most of which are based on [PRS].

Fix a set of generator graphs $\mathcal{G}$ with vertex set V and a basis $\mathcal{O}$ of set operations. A *graph circuit* with generators $\mathcal{G}$ and basis $\mathcal{O}$ is a sequence of equations or *gates* $g_1, \ldots, g_s$ where, for $1 \le i \le s$, g_i computes a graph G_i such that

$$G_i = H \qquad \text{for some} \quad H \in \mathcal{G},$$

or

$$G_i = G_j \circ G_k \qquad \text{where} \quad \circ \in \mathcal{O} \quad \text{and} \quad j, k < i.$$

Here we are assuming for simplicity that $\circ$ is binary; analogous equations can be written for operations of other arities. The graph G_s is the graph computed by the circuit. The *circuit complexity* of a graph G, with respect to generators $\mathcal{G}$ and basis $\mathcal{O}$, is the smallest s for which there exists a circuit that computes G.

As usual, we can imagine a circuit to be a directed acyclic graph (DAG) with the input nodes (of in-degree 0) labeled by the generator graphs and the internal nodes (gates) labeled by set operations from the basis. The target graph appears at the root (of out-degree 0) of this DAG. The *depth of the circuit* is defined to be the length of the longest path in the DAG. Its *size* is the number of nodes in it.

A *graph formula* is a graph circuit in which the out-degree of each gate is at most one. Thus a graph formula can be represented as a tree with the leaves labeled by generator graphs and the internal nodes labeled by operations from the basis. The size of a formula is the number of leaves in its tree. The *formula complexity* of a graph is the smallest size of a formula that computes the graph (with respect to a fixed set of generators and a basis).

We can also define natural restricted models such as *constant depth* graph circuits and formulas. In these models we allow unbounded fan-in and assume that the operations from the basis are naturally extendable to an unbounded number of operands (for example, union, intersection, and symmetric difference of sets have this property). We can similarly generalize other models of Boolean complexity such as decision trees and branching programs to graph complexity.

In the following definitions and results, for concreteness, we concentrate on the formula complexity of graphs. We consider graph complexity with the set operations of $\cup$ (UNION) and $\cap$ (INTERSECTION) only. We naturally want the sets of generators to be *complete* in the sense that every graph should be constructible from these generators using $\cap$ and $\cup$ operators in a circuit or a formula. Definitions 2.1 and 2.2 below give two such sets of generators.

Definition 2.1. Fix a vertex set V, where $|V| = n$. The set of stars is defined to be

$$\mathcal{S} = \left\{ H \subseteq \binom{V}{2} : H \cong K_{1,n-1} \right\}.$$

For a graph $G \subseteq \binom{V}{2}$, the star-formula complexity of G, denoted $L_{\mathcal{S}}(G)$, is the smallest size of a graph formula computing G using the stars $\mathcal{S}$ as the set of generators and $\cup$ and $\cap$ as the basis.

We are especially interested in the complexity of bipartite graphs because of their direct relevance to lower bounds on Boolean circuits and communication complexity.

Definition 2.2. Fix the color classes U and V. Let $\mathcal{B}$ denote the following set of complete bipartite graphs:

$$\mathcal{B} = \{A \times V : A \subseteq U\} \cup \{U \times B : B \subseteq V\}.$$

For a bipartite graph $G \subseteq U \times V$, the bipartite-formula complexity of G is the smallest size of a graph formula computing G using $\mathcal{B}$ as the set of generators and $\cup$ and $\cap$ as the basis. Bipartite-formula complexity of G is denoted by $L_{\mathcal{B}}(G)$.

The following relation holds between bipartite complexity and star complexity:

Proposition 2.3 [PRS]. *Let $G \subseteq U \times V$ be a bipartite graph, $|U| = |V| = n$. Then*

$$L_{\mathcal{S}}(G) \le L_{\mathcal{B}}(G) \cdot n + 2n, \tag{1}$$

where the stars are on the vertex set $U \cup V$.

The next observation of [PRS] is based on the fact that every n-vertex graph is a union of $\lceil \log n \rceil$ bipartite graphs. It can be used to translate a lower bound on the star-complexity of general, i.e., not necessarily bipartite, graphs to a lower bound on bipartite graphs.

Proposition 2.4 [PRS]. *Let G be a graph on W, $|W| = n$, where n is assumed to be even (for simplicity). Then there exists a partition $W = U \cup V$ with $|U| = |V| = n/2$ and a bipartite subgraph $H \subseteq U \times V$ of G such that*

$$L_{\mathcal{S}}(H) \geq \frac{L_{\mathcal{S}}(G)}{\lceil \log_2 n \rceil}. \tag{2}$$

Definition 2.5. Let f be a Boolean function on 2ℓ variables, written as $f\colon \{0,1\}^\ell \times \{0,1\}^\ell \longrightarrow \{0,1\}$. Let $n := 2^\ell$. The $n \times n$ bipartite graph $G_f \subseteq \{0,1\}^\ell \times \{0,1\}^\ell$ is defined by including the edge (x, y) in G_f iff $f(x, y) = 1$, where $x, y \in \{0,1\}^\ell$.

Note that the AND and OR operations on Boolean functions correspond to UNION and INTERSECTION operations on the edge sets of their corresponding graphs. In other words, $G_{f_1 \wedge f_2} = G_{f_1} \cap G_{f_2}$ and $G_{f_1 \vee f_2} = G_{f_1} \cup G_{f_2}$. This suggests a syntactic transformation of a Boolean formula (assuming all negations are pushed to the leaves) into a graph formula. However, what about the input literals of the Boolean formula? The literals are simply the projection functions and the graphs corresponding to projection functions are complete bipartite graphs isomorphic to $K_{n/2,n}$ and $K_{n,n/2}$. For instance, G_{x_i} is the complete bipartite graph $\{x \in \{0,1\}^\ell : x_i = 1\} \times \{0,1\}^\ell$. Thus each literal can be translated into a generator in $\mathcal{B}$. With this transformation of a Boolean formula for f into a bipartite-formula for G_f, it follows that

$$L_{\mathcal{B}}(G_f) \leq L(f), \tag{3}$$

where $L(f)$ is the minimum size of a formula (with tight negations) computing f.

Given an $n \times n$ bipartite graph G, where n is a power of 2, we can clearly define a function f such that $G_f = G$. Thus we get the following criterion for lower bounds on Boolean formula size:

Proposition 2.6 [PRS]. *A lower bound of $L_{\mathcal{B}}(G) \geq \psi(\log n)$ for an explicit $n \times n$ bipartite graph G, where $n = 2^\ell$, would yield an explicit function f on 2ℓ variables with formula size lower bound $L(f) \geq \psi(\ell)$.*

For some explanation of the notion of explicitness, please see Remark 3.4.

Since the proof of this proposition is essentially syntactic, similar relations hold for other models such as circuits, decision trees, and branching programs.

Note, however, that the graph complexity of G_f could be much smaller than the Boolean complexity of f. This is because in a bipartite-formula we have access to an exponential (in n) number of generators $\mathcal{B}$, whereas the transformation above uses only the $2\log n$ "canonical" generators corresponding to the projection functions. In fact, the generators in $\mathcal{B}$, with $U = V = \{0,1\}^\ell$, can be viewed as defining *arbitrary* Boolean functions of either the first ℓ or the last ℓ variables. This interpretation captures the connection between two-party communication complexity and graph complexity.

We conclude this section by indicating the generality and usefulness of graph complexity. In the following remarks, we assume the generators are from $\mathcal{B}$:

- When the model of computation is a decision tree, we get the model of Yao's two-party communication complexity [Y].
- When the model is a Boolean formula of constant depth (polylog(ℓ) depth) and quasi-poly(ℓ) size, we get the definitions of "polynomial hierarchy" PH^{cc} (PSPACE^{cc}, respectively) in the two-party communication complexity model as defined in [BFS].
- In [PR1], graph complexity in the model of branching programs is used to derive criteria for the branching program size of the corresponding Boolean function. In particular, Pudlák and Rödl define the notion of the projective dimension of graphs and show that lower bounds on the projective dimension of graphs imply lower bounds on the branching program size of Boolean functions.
- The formula complexity of graphs, with $\cup$ and $\cap$ operators, is used by Razborov [R4] to derive criteria for lower bounds on the formula size of the corresponding Boolean function. He defines the notion of the affine dimension of graphs and shows that lower bounds on the affine dimension of graphs imply lower bounds on the formula size of Boolean functions. He also shows relations between the affine and projective dimensions of graphs in various cases.
- Using the translation of graph complexity to Boolean function complexity as an intermediate step, we show below that sufficiently strong lower bounds on the *monotone* complexity of the very special class of 2-slice functions imply lower bounds on the complexity over a complete basis of certain Boolean functions.

3. 2-Slice Functions

In this section we relate graph complexity to the Boolean complexity of 2-slice functions.

Definition 3.1. A Boolean function $f\colon \{0,1\}^n \longrightarrow \{0,1\}$ is a 2-slice function if $f(x) = 0$ for all x with $|x| < 2$ and $f(x) = 1$ for all x with $|x| > 2$. On inputs x with $|x| = 2$, f may be nontrivially defined. The inputs x such that $|x| = 2$ and $f(x) = 1$ can be identified with the edges of a graph G with vertex set $\{1, \ldots, n\}$ in an obvious way.

Conversely, every graph G on n vertices gives rise to a 2-slice function f_G on n variables: given a graph $G = ([n], E)$, we define the function f_G by

$$f_G(x) = \begin{cases} 1 & \text{if} \quad |x| > 2, \\ 0 & \text{if} \quad |x| < 2, \\ 1 & \text{if} \quad |x| = 2 \quad \text{and} \quad X \in E(G), \\ 0 & \text{if} \quad |x| = 2 \quad \text{and} \quad X \notin E(G). \end{cases}$$

Here X denotes the set with characteristic vector x, i.e., $X = \{i\colon x_i = 1, 1 \le i \le n\}$ for $x \in \{0,1\}^n$.

For $1 \leq i \leq n$, the star S_i is defined as the graph with vertex set $V(S_i) = \{1, \ldots, n\}$ and edge set $E(S_i) = \{\{i, j\} : j \neq i\}$.

Lemma 3.2. *Let T_g be a star-formula computing the graph $G = ([n], E)$. Let T_b be the Boolean formula obtained from T_g by replacing $\cup$'s by $\vee$'s and $\cap$'s by $\wedge$'s and the star S_i by the variable x_i. Let $f\colon \{0,1\}^n \longrightarrow \{0,1\}$ be the function computed by T_b. Then, for all x such that $|x| = 2$, $f(x) = 1$ iff $X \in E$.*

Proof. By induction on the number of operators in T_g that computes G.

In the base case we have zero operators in T_g and this computes a single star S_i for some i, $1 \leq i \leq n$. The corresponding T_b is x_i. Clearly, the inputs X of size 2 such that $f(x) = 1$ are exactly the edges of S_i.

For the inductive step, first consider the case when the top gate of T_g is $\cup$. Let $T_g = T_{g_1} \cup T_{g_2}$, and let T_{g_i} compute G_i for $i = 1, 2$, so that $G = G_1 \cup G_2$. Correspondingly, we get $T_b = T_{b_1} \vee T_{b_2}$ and $f = f_1 \vee f_2$, where T_{b_i} computes f_i. If $X \in G$, then $X \in G_i$ for $i = 1$ or $i = 2$. By the induction hypothesis $f_i(x) = 1$ and therefore $f(x) = 1$. Conversely, suppose $f(x) = 1$ and $|x| = 2$. Then for at least one $i \in \{1, 2\}$, we have $f_i(x) = 1$. By the induction hypothesis $f_i(x) = 1$ iff $X \in G_i$ when $|x| = 2$. Thus $X \in G_i$ and therefore $X \in G$.

The proof when the top gate of T_g is $\cap$ is similar. $\square$

Theorem 3.3. *Let G be an n-vertex graph and let f_G be the associated 2-slice function on n variables. Let $L_{\text{mon}}(f_G)$ be the* {AND, OR} *(monotone) formula complexity of f_G and let $L_{\mathcal{S}}(G)$ be the star-formula complexity of the graph G. Then*

$$L_{\text{mon}}(f_G) \leq L_{\mathcal{S}}(G) + O(n \log n).$$

Proof. Let f be the function from Lemma 3.2 obtained from an optimal star-formula for G. Note that $f_G \equiv f \wedge Th_2^n \vee Th_3^n$, where Th_k^n denotes the kth threshold function on n variables. Now we use the fact that for constant k there are monotone formulas of size $O(n \log n)$ to compute Th_k^n [F2]. $\square$

Using the connections described so far, we transform a 2-slice function f of sufficiently superlinear (monotone) formula complexity, say $n^{1+\Omega(1)}$, into a function h with exponential formula complexity over a complete basis. For the lower bound to be interesting, we obviously need h to be *explicit*. If we take f to be sufficiently explicit, then we expect h also to be explicit. To clarify this, some explanation of the notion of explicitness of graphs and functions may be in order here.

Remark 3.4 (On Explicitness). Intuitively, we consider an infinite family $\{g_m\}$ of Boolean functions to be explicit if on an input $x \in \{0,1\}^m$, $g_m(x)$ can be computed by a Turing machine within a sufficiently small (but large enough for the lower bound to make sense) complexity class. A standard example is the characteristic function of a language in NP. Analogously, we may define an infinite family of graphs $\{G_n\}$ to be explicit if, given the labels i and j of vertices (of $\log n$ bits each for an n-vertex graph

G_n), a nondeterministic Turing machine can decide if i and j are connected by an edge in G_n in polynomial (in $\log n$) time. For a 2-slice function f, adapting the notion of explicitness from *graphs* seems more appropriate than adapting it from general Boolean functions. Note that every 2-slice function on n variables has a description of length $O(n^2)$, just like an n-vertex graph. Specifically, we may define an infinite family $\{f_n\}$, where f_n is defined on n variables, of 2-slice functions to be explicit if, given i and j of $\log n$ bits each, a nondeterministic Turing machine can decide in polynomial (in $\log n$) time whether $f_n(x) = 1$ where x has a 1 in ith and jth positions and 0 everywhere else.

Theorem 3.5. *Let $f\colon \{0,1\}^n \longrightarrow \{0,1\}$ be a 2-slice function where $n = 2^l$. Suppose that $L_{\text{mon}}(f) \geq n \log n \cdot \beta(\log n)$, for some $\beta(\log n) = \omega(1)$. Then there exists a function $h\colon \{0,1\}^{2l-2} \longrightarrow \{0,1\}$ such that $L(h) = \Omega(\beta(l))$. Moreover, if f is explicit, so is h.*

Proof. Let f be a 2-slice function on n variables where $n = 2^l$, and let G be the associated graph (see Definition 3.1). From Theorem 3.3, $L_{\mathcal{S}}(G) \geq L_{\text{mon}}(f) - O(n \log n)$. Using Proposition 2.4, there exists an $n/2 \times n/2$ bipartite subgraph H of G such that $L_{\mathcal{S}}(H) \geq L_{\mathcal{S}}(G)/\log n$. From Proposition 2.3, $L_{\mathcal{B}}(H) \geq 2\, L_{\mathcal{S}}(H)/n - 2$. Combining the three inequalities, we have $L_{\mathcal{B}}(H) \geq \Omega(L_{\text{mon}}(f)/n \log n)$.

Since H is a bipartite graph with $n/2 = 2^{l-1}$ vertices on each side, we can identify its color classes with $\{0,1\}^{l-1}$ and define a function h on $2l-2$ variables such that the $G_h = H$ (as in Definition 2.5) and from inequality (3) in Section 2, we get $L(h) \geq L_{\mathcal{B}}(H) \geq \Omega\left(L_{\text{mon}}(f)/n \log n\right)$. It follows that a lower bound of $n \log n \cdot \beta(\log n)$ on $L_{\text{mon}}(f)$ would give a lower bound of $\Omega(\beta(l))$ on $L(h)$, since h depends on at most $2l - 2 = O(\log n)$ variables. □

The derived function h in Theorem 3.5 is constructed from the bipartite subgraph H of G whose *existence* was proved by an averaging argument in Proposition 2.4. We can avoid this averaging argument in the transformation above by an easy trick. Proof of Proposition 2.4 gives $O(\log n)$ such bipartite graphs and through Theorem 3.5 we can construct $O(\log n)$ functions, one of which is guaranteed to be hard. We can construct a single hard function from such a small collection by "addressing" into it using an additional argument. We make this simple idea formal in the following:

Corollary 3.6. *An explicit n-variable 2-slice function f with $L_{\text{mon}}(f) = n^{1+\Omega(1)}$ would give an explicit m-variable function g such that $L(g) = 2^{\Omega(m)}$.*

Proof. Let $H_0, \ldots, H_{l-1}$ be the bipartite subgraphs given by Proposition 2.4 of the graph G corresponding to f (see Definition 3.1). Define the functions $h_0, \ldots, h_{l-1}$ on $2l-2$ variables each from these bipartite subgraphs as we defined h from H in the proof of Theorem 3.5. Theorem 3.5 shows that one of these functions is hard: for some j, $0 \leq j \leq l-1$, $L(h_j) = 2^{\Omega(l)}$ since $\beta(\log n) = n^{\Omega(1)}/\log n = \exp(\Omega(\log n))$.

Define a function $g\colon \{0,1\}^{2l-2} \times \{0,1\}^{\log l} \longrightarrow \{0,1\}$ by setting $g(x, j) = h_j(x)$, where the $\log l$- bit second argument is treated as the binary representation of an integer $j \in \{0, \ldots, l-1\}$. Let $m := 2l - 2 + \log l$. Suppose now that $L(g) = 2^{o(m)}$. Clearly, for all j, $L(h_j) = 2^{o(m)}$ since h_j is a restriction of g. Since $m = O(l)$, this gives

$L(h_j) = 2^{o(l)}$ for all j, contradicting the conclusion of Theorem 3.5. Therefore, we must have $L(g) = 2^{\Omega(m)}$. $\square$

Unwinding the definitions, we can describe the function g in terms of the graph G as follows. Label the vertices of G using $\{0, \dots, n-1\}$, where $n = 2^l$. Let $(x, y) \in \{0,1\}^{l-1} \times \{0,1\}^{l-1}$ (given by the $(2l-2)$-bit first argument of g) and $j \in \{0, \dots, l-1\}$ (given by its $\log l$-bit binary representation as the second argument of g). Define the vertex u as the integer whose binary representation has 0 in position j and whose other bits are given by the assignment x. Similarly define the vertex v as the integer whose binary representation has a 1 in positions j and whose other bits are given by the assignment y. Now $g(x, y, j) = 1$ iff (u, v) is an edge of G. It is clear that g is as explicit as the graph G (or equivalently the 2-slice function f).

4. Depth-3 Lower Bounds

In this section we consider lower bounds on depth-3 bipartite formulas computing bipartite graphs $G \subseteq U \times V$, $|U| = |V| = n$. Recall that the leaves of the formula are graphs from $\mathcal{B} = \{A \times V : A \subseteq U\} \cup \{U \times B : B \subseteq V\}$.

We first observe that the bottom gates of a bipartite formula need not have fan-in more than 2. Indeed, an $\cap$ gate at the bottom computes a complete bipartite graph $A \times B$ and a $\cup$ bottom gate computes the complement of a complete bipartite graph $\overline{A \times B}$, where $A \subseteq U$ and $B \subseteq V$. These can be written as intersection and union, respectively, of at most two graphs from $\mathcal{B}$.

Without loss of generality, we consider $\cup\cap\cup$ formulas. By the remark above, we can write such a formula as $G = \cup_i \cap_j G_{ij}$, where G_{ij} is the complement of a complete bipartite graph, i.e., $\overline{A_{ij} \times B_{ij}}$ for some $A_{ij} \subseteq U$ and $B_{ij} \subseteq V$.

Our lower bound proof uses sign-representing polynomials for DNFs and lower bounds on ranks of ± 1 matrices under sign-preserving changes of *unbounded magnitude*. A preliminary version of this paper [L] gave a lower bound of $\Omega((\log n)^2/(\log\log n)^2)$. In this paper we improve the lower bound to $\Omega((\log n)^3/(\log\log n)^5)$.

Nisan and Szegedy [NS] give the following construction of ε-approximating polynomials for the OR function. They assume a constant ε. The refined analysis to bring out the dependence on ε is due to Hayes and Kutin [HK]. We give the proof here for completeness.

Lemma 4.1 [NS], [HK]. *The* OR*-function of n Boolean variables can be ε-approximated by a real polynomial of degree at most $O(\sqrt{n}\log(2/\varepsilon))$. More precisely, for every $0 < \varepsilon < \frac{1}{2}$, there is a real polynomial p of degree at most $O(\sqrt{n}\log(2/\varepsilon))$ such that for every $x \in \{0,1\}^n$, $|\mathrm{OR}(x) - p(x)| \le \varepsilon$.*

Proof. We construct a univariate real polynomial $q(z)$ of degree at most $k := c \cdot \sqrt{n}\log(2/\varepsilon)$ (for a constant c) such that for $0 \le z \le n-1$, $|q(z)| \le \varepsilon$ and $q(n) = 1$. Then defining $p(x_1, \dots, x_n) := 1 - q(n - x_1 - \cdots - x_n)$ proves the theorem.

The polynomial q is essentially a normalized Chebyshev polynomial. The degree-k Chebyshev polynomial T_k is given by

$$T_k(x) = \tfrac{1}{2}[x + \sqrt{x^2 - 1}]^k + \tfrac{1}{2}[x - \sqrt{x^2 - 1}]^k.$$

Define

$$q(z) := \frac{T_k(z/(n-1))}{T_k(n/(n-1))}.$$

Clearly, $q(n) = 1$. We want to select a k such that $T_k(n/(n-1)) \geq 1/\varepsilon$. Then, since for $-1 \leq x \leq 1$, $|T_k(x)| \leq 1$, we will have $|q(z)| \leq \varepsilon$ for every $z \in [0, n-1]$.

In fact, for $x = n/(n-1)$, $x + \sqrt{x^2 - 1} \geq 1 + \sqrt{2/(n-1)}$, and we get

$$T_k\left(\frac{n}{n-1}\right) \geq \frac{1}{2}\left[1 + \sqrt{\frac{2}{n-1}}\right]^k.$$

We will have the right-hand side quantity greater than or equal to $1/\varepsilon$ if

$$k \geq \frac{\log(2/\varepsilon)}{\log(1 + \sqrt{2/(n-1)})}.$$

This last inequality is satisfied if

$$k \geq c \cdot \sqrt{n} \cdot \log(2/\varepsilon),$$

for a suitable constant c. □

For a bipartite graph $G \subseteq U \times V$, we let $G(x, y) = 1$ if $(x, y) \in G$ and $G(x, y) = 0$ if $(x, y) \notin G$.

Lemma 4.2. *Suppose an $n \times n$ bipartite graph $H \subseteq U \times V$ is written as a union of d complete bipartite graphs*:

$$H = \bigcup_{i=1}^{d} (A_i \times B_i), \qquad \textit{where} \quad A_i \subseteq U, \quad B_i \subseteq V.$$

Then, for every ε, where $0 < \varepsilon < \frac{1}{2}$, there is a real matrix M_H such that

- *for all $(x, y) \in U \times V$, $|M_H(x, y) - H(x, y)| \leq \varepsilon$, and*
- $\operatorname{rank}(M_H) \leq \exp(O(\sqrt{d}\log(2/\varepsilon)\log d))$.

Proof. Let R be the incidence matrix of H, and similarly let R_i be the incidence matrices of the complete bipartite graphs $A_i \times B_i$, $1 \leq i \leq d$, covering H. Note that R is simply the entry-wise OR of the R_i. Furthermore, each R_i is of rank one as a real matrix. We obtain M_H from R using the approximating polynomials for the OR-function given by Lemma 4.1.

Suppose $p(z_1, \dots, z_d)$ is an ε-approximating polynomial of degree $k := c \cdot \sqrt{d} \log(2/\varepsilon)$ for the OR-function of d Boolean variables. Syntactically substitute the matrix R_i for z_i in this polynomial, but interpret the product as an entry-wise product of matrices, i.e., a monomial $z_i z_j$ is replaced by $R_i \circ R_j$, where for matrices A and B, $(A \circ B)(x, y) := A(x, y)B(x, y)$. Note that if A and B are rank-1 matrices, then $A \circ B$ is also a rank-1 matrix. Thus, a monomial $z_{i_1} \cdots z_{i_t}$ is replaced by the rank-1 0-1 matrix $R_{i_1} \circ \cdots \circ R_{i_t}$. The matrix obtained by computing the polynomial $p(R_1, \dots, R_d)$ in this way gives us the desired matrix M_H.

It is clear that $M_H(x, y) = p(R_1(x, y), \dots, R_d(x, y))$. From the properties of p, it is easy to see that for all x, y, $|M_H(x, y) - H(x, y)| \le \varepsilon$. Since M_H is a linear combination of rank-1 matrices, one for each monomial, it follows that the rank of M_H is at most the number of monomials in p which is bounded by $\sum_{j=0}^{k} \binom{d}{j} \le \exp(O(k \log d))$. □

Lemma 4.3. *Let $G \subseteq U \times V$ be a bipartite graph. If G is realized by a depth-3 bipartite formula*

$$G = \bigcup_{i=1}^{t} \bigcap_{j=1}^{d_i} \overline{(A_{ij} \times B_{ij})}, \qquad \text{where} \quad A_{ij} \subseteq U, \quad B_{ij} \subseteq V,$$

then there exists a matrix M such that

(i) *if $G(x, y) = 0$, then $M(x, y) \le -\frac{1}{6}$,*
(ii) *if $G(x, y) = 1$, then $M(x, y) \ge +\frac{1}{6}$, and*
(iii) $\operatorname{rank}(M) \le \exp(O(\sqrt{D} \log t \log D))$, *where $D = \max_{i=1}^{t} d_i$.*

Proof. Let $G_1, \dots, G_t$ be the input graphs to the top gate so that $G = \cup_{i=1}^{t} G_i$. Since each G_i is an intersection of complements of complete bipartite graphs, its complement, $\overline{G_i}$, is computed by a union of complete bipartite graphs. Thus we can apply Lemma 4.2 to these complements $\overline{G_i}$. Let M_i be the real matrix given by Lemma 4.2 that ε_i-approximates $\overline{G_i}$, where $\varepsilon_i := 1/3t$. We also have $\operatorname{rank}(M_i) \le \exp(O(\sqrt{d_i} \log(1/\varepsilon_i) \log d_i)) \le \exp(O(\sqrt{D} \log t \log D))$.

Let $M := M_1 + \cdots + M_t - \frac{1}{2} \cdot J$, where J is the $n \times n$ all 1's matrix. Let us see the relation between M and G:

If $G(x, y) = 0$, then $\forall i\ G_i(x, y) = 0$, and hence $\forall i\ |M_i(x, y)| \le \varepsilon_i$. It follows that $M(x, y) \le \sum_{i=1}^{t} \varepsilon_i - \frac{1}{2} \le -\frac{1}{6}$.

If $G(x, y) = 1$, then $\exists i\ G_i(x, y) = 1$ and for this i, $1 - \varepsilon_i \le M_i(x, y) \le 1 + \varepsilon_i$. For $j \ne i$, $-\varepsilon_j \le M_j(x, y) \le 1 + \varepsilon_j$. Hence, we have $1 - \varepsilon_i - \sum_{j \ne i} \varepsilon_j - \frac{1}{2} \le M(x, y) \le \sum_{j=1}^{t} (1 + \varepsilon_j) - \frac{1}{2}$. So, in this case, $\frac{2}{3} - \frac{1}{2} \le M(x, y) \le t + \frac{1}{3} - \frac{1}{2}$, and hence $M(x, y) \ge \frac{1}{6}$.

Moreover, $\operatorname{rank}(M) \le \sum_{i=1}^{t} \operatorname{rank}(M_i) + 1 \le t \exp(O(\sqrt{D} \log D \log t)) + 1 \le \exp(O(\sqrt{D} \log D \log t))$. □

Lemma 4.4. *Let G be an $n \times n$ bipartite graph $G \subseteq U \times V$. If G is realized by a depth-3 bipartite formula*

$$G = \bigcup_{i=1}^{t} \bigcap_{j=1}^{d_i} \overline{(A_{ij} \times B_{ij})}, \qquad \textit{where} \quad A_{ij} \subseteq U, \quad B_{ij} \subseteq V,$$

then there exists a matrix M such that

(i) *if $G(x, y) = 0$, then $M(x, y) \geq 1$,*
(ii) *if $G(x, y) = 1$, then $M(x, y) = 0$, and*
(iii) $\operatorname{rank}(M) \leq \prod_{i=1}^{t} d_i$.

Proof. It will be convenient to consider the complement graph $\bar{G}$ of G. Clearly,

$$\bar{G} = \underbrace{\bigcap_{i=1}^{t} \bigcup_{j=1}^{d_i} \underbrace{(A_{ij} \times B_{ij})}_{G_{ij}}}_{G_i}.$$

Let R_{ij} be the incidence matrix of the complete bipartite graph G_{ij}. Define $M_i = \sum_{j=1}^{d_i} R_{ij}$. Finally, let $M = M_1 \circ M_2 \circ \cdots \circ M_t$. Hence, we have

$$M(x, y) = \prod_{i=1}^{t} \sum_{j=1}^{d_i} R_{ij}(x, y).$$

Note that if $(x, y) \in G_i$, then $M_i(x, y) = \sum_{j=1}^{d_i} R_{ij}(x, y) \geq 1$ and if $(x, y) \notin G_i$, then $M_i(x, y) = 0$. Hence, we have

$$(x, y) \in \bar{G} \implies M(x, y) \geq 1,$$
$$(x, y) \notin \bar{G} \implies M(x, y) = 0.$$

From this (i) and (ii) follow.

To see the bound on $\operatorname{rank}(M)$, note that rank is submultiplicative under $\circ$: $\operatorname{rank}(A \circ B) \leq \operatorname{rank}(A) \cdot \operatorname{rank}(B)$. Since $\operatorname{rank}(R_{ij}) = 1$, we have $\operatorname{rank}(M_i) \leq d_i$. Hence $\operatorname{rank}(M) \leq \prod_{i=1}^{t} \operatorname{rank}(M_i) \leq \prod_{i=1}^{t} d_i$. This gives (iii). $\square$

Theorem 4.5. *Let G be an $n \times n$ bipartite graph $G \subseteq U \times V$. If G is realized by a depth-3 bipartite formula*

$$G = \bigcup_{i=1}^{t} \bigcap_{j=1}^{d_i} \overline{(A_{ij} \times B_{ij})}, \qquad \textit{where} \quad A_{ij} \subseteq U, \quad B_{ij} \subseteq V,$$

then there exists a matrix M such that

(i) *if* $G(x, y) = 0$, *then* $M(x, y) \le -\frac{1}{12}$,
(ii) *if* $G(x, y) = 1$, *then* $M(x, y) \ge +\frac{1}{12}$, *and*
(iii) $\operatorname{rank}(M) \le \exp(O(L^{1/3} \log^{5/3} L))$, *where* $L = \sum_{i=1}^{t} d_i$.

Proof. Let D^* be a parameter to be fixed later. We separate the formula into two parts depending on whether the middle fan-in d_i is smaller or larger than D^*. This idea of splitting the formula based on middle fan-in is due to Klivans and Servedio [KS].

Specifically, let G_s be the subgraph of G realized by the subformula given by union of middle gates of fan-in at most D^* and let G_l be the subgraph of G similarly given by middle gates of fan-in larger than D^*:

$$G_s = \bigcup_{d_i \le D^*} \bigcap_{j=1}^{d_i} \overline{(A_{ij} \times B_{ij})},$$

$$G_l = \bigcup_{d_i > D^*} \bigcap_{j=1}^{d_i} \overline{(A_{ij} \times B_{ij})}.$$

We apply Lemma 4.3 to G_s to get a matrix M_s such that $M_s(x, y) \ge \frac{1}{6}$ if $(x, y) \in G_s$ and $M_s(x, y) \le -\frac{1}{6}$ if $(x, y) \notin G_s$. Furthermore, we have $\operatorname{rank}(M_s) \le \exp(O(\sqrt{D^*} \log D^* \log t))$ since all middle fan-in's of the formula for G_s are at most D^* and the top fan-in is at most t.

We apply Lemma 4.4 to G_l and get a matrix M_l such that $M_l(x, y) \ge 1$ if $(x, y) \notin G_l$ and $M_l(x, y) = 0$ if $(x, y) \in G_l$. Since all middle fan-in's of the formula for G_l are at least D^*, the top fan-in t_l of this formula is at most $t_l \le \sum_{i=1}^{t} d_i / D^* \le L/D^*$. We bound the rank of M_l as follows:

$$\begin{aligned}
\operatorname{rank}(M_l) &\le \prod_{d_i > D^*,\ 1 \le i \le t} d_i \\
&\le \left(\sum_{d_i > D^*} \frac{d_i}{t_l} \right)^{t_l} \\
&\le \exp(O(t_l \log D)) \qquad (\text{where } D = \textstyle\sum_{d_i > D^*} d_i / t_l \le L) \\
&\le \exp\left(O\left(\frac{L}{D^*} \log D \right) \right).
\end{aligned}$$

Define $M = M_s \circ M_l + \frac{1}{12} J$. If $(x, y) \in G_l$, then $M(x, y) = \frac{1}{12}$, and if $(x, y) \in G_s \setminus G_l$, then $M(x, y) \ge \frac{1}{6} + \frac{1}{12}$. If $(x, y) \notin G_s \cup G_l$, then $M(x, y) \le -1/6 + 1/12 \le -\frac{1}{12}$. Hence (i) and (ii) are satisfied by M.

By submultiplicativity of rank under $\circ$, we get the upper bound on the rank of M:

$$\operatorname{rank}(M) \le \exp\left(O(\sqrt{D^*} \log D^* \log t) + O\left(\frac{L}{D^*} \log D \right) \right) + 1.$$

We set $D^* = \Theta((L/\log L)^{2/3})$. Using the trivial upper bound of $O(\log L)$ on $\log D^*$, $\log D$, and $\log t$, we get that $\operatorname{rank}(M) = \exp(O(L^{1/3}(\log L)^{5/3}))$, verifying (iii). $\square$

Using a recent remarkable result by Forster [F1], we now show that for some "interesting" graphs G any matrix satisfying (i) and (ii) of Theorem 4.5 must have a large rank and hence conclude a lower bound on the depth-3 complexity of G using (iii).

Theorem 4.6 [F1]. *Let A be an $n \times n$ ± 1 matrix and let B be a real matrix such that $|b_{ij}| \geq 1$ and $\operatorname{sign}(a_{ij}) = \operatorname{sign}(b_{ij})$ for all i, j. Then $\operatorname{rank}(B) \geq n/\|A\|$. Here $\|A\|$ denotes the operator norm of matrix A defined as $\|A\| := \max_{\|x\|=1} \|Ax\|$, where the vector norm denotes the Euclidean norm. Recall also that $\|A\|$ is the largest singular value of A or equivalently $\|A\|^2$ is the largest eigenvalue of AA^* where A^* is the conjugate transpose of A.*

Theorem 4.7. *Let G be an $n \times n$ bipartite graph and let A_G be its ± 1 incidence matrix, i.e., $A_G(x, y) = 1$ if (x, y) is an edge of G and $A_G(x, y) = -1$ if (x, y) is not an edge of G. Then any depth-3 bipartite formula for G must have size at least*

$$\Omega\left(\frac{\log^3(n/\|A_G\|)}{\log\log^5(n/\|A_G\|)}\right).$$

Proof. Given a depth-3 formula for G of size L, let M be the matrix given by Theorem 4.5. Note that if $(x, y) \notin G$, then $M(x, y) \leq -\frac{1}{12}$, and if $(x, y) \in G$, then $M(x, y) \geq \frac{1}{12}$. Hence $12M$ is a sign-preserving variation of A_G and we can apply Theorem 4.6: $\operatorname{rank}(M) = \operatorname{rank}(12M) = \Omega(n/\|A_G\|)$. On the other hand, from Theorem 4.5 (iii) we get that $\operatorname{rank}(M) \leq \exp(O(L^{1/3}\log^{5/3} L))$. Combining the two estimates on $\operatorname{rank}(M)$:

$$\exp(O(L^{1/3}\log^{5/3} L)) = \frac{n}{\|A_G\|}.$$

Solving for L proves the theorem. $\square$

An $n \times n$ Hadamard matrix is a ± 1 matrix such that $HH^\top = nI$. It is obvious that $\|H\| = \sqrt{n}$.

Corollary 4.8. *For any graph G such that A_G is a Hadamard matrix, the depth-3 bipartite formula complexity of G is at least $\Omega((\log n)^3/(\log\log n)^5)$. An example of such a graph is the Paley-type bipartite graph.*

Acknowledgments

I am grateful to the anonymous referees for several valuable suggestions.

References

[AB] Alon, N., Boppana, R.: The Monotone Circuit Complexity of Boolean Functions, *Combinatorica*, **7**(1) (1987), 1–22.

[B] Berkowitz, S.: On Some Relationships Between Monotone and Non-Monotone Circuit Complexity, Technical Report, University of Toronto, 1982.

[BFS] Babai, L., Frankl, P., Simon, J.: Complexity Classes in Communication Complexity Theory, *Proc*. 26*th IEEE FOCS*, 1986, pp. 337–347.

[D] Dunne, P.: The Complexity of Central Slice Functions, *Theoret*. *Comput*. *Sci*., **44** (1986), 247–257.

[F1] Forster, J.: A Linear Lower Bound on the Unbounded Error Probabilistic Communication Complexity, *Proc*. 16*th IEEE Conference on Computational Complexity* (*CCC*), 2001, pp. 100–106.

[F2] Friedman, J.: Constructing $O(n \log n)$ Size Monotone Formulae for the kth Elementary Symmetric Polynomial of n Boolean Variables, *SIAM J. Comput*., **15**(3) (1986), 641–654.

[H1] Håstad, J.: Almost Optimal Lower Bounds for Small Depth Circuits, in S. Micali (ed.), *Advances in Computer Research*, *Vol* 5: *Randomness and Computation*, JAI Press, Greenwich, CT, 1989.

[H2] Håstad, J.: The Shrinkage Exponent of de Morgan Formulas Is 2. *SIAM J. Comput*., **27**(1) (1998), 48–64.

[HJP] Håstad, J., Jukna, S., Pudlák, P.: Top-Down Lower Bounds for Depth-3 Circuits, *Proc*. 34*th IEEE FOCS*, pp. 124–129.

[HK] Hayes, T., Kutin, S.: Personal communication.

[HR] Harnik, D., Raz, R.: Higher Lower Bounds on Monotone Size. *Proc. ACM STOC*, 2000, pp. 378–387.

[KS] Klivans, A., Servedio, R.: Learning DNF in Time $2^{O(n^{1/3})}$, *Proc. ACM STOC*, 2001, pp. 258–265.

[KW1] Karchmer, M., Wigderson, A.: Monotone Circuits for Connectivity Require Sper-Logarithmic Depth, *SIAM J. Discrete Math*. **3**(2) (1990), 255–265.

[KW2] Krause, M., Waack, S.: Variation Ranks of Communication Matrices and Lower Bounds for Depth-Two Circuits Having Symmetric Gates with Unbounded Fan-In, *Proc*. 32*nd IEEE FOCS*, 1991, pp. 777–782.

[L] Lokam, S. V.: Remarks on Graph Complexity, *Proc*. 18*th FST & TCS*, 1998, pp. 307–318.

[NS] Nisan, N., Szegedy, M.: On the Degree of Boolean Functions as Real Polynomials, *Proc*. 24*th ACM STOC*, 1991, pp. 462–467.

[PPZ] Paturi, R., Pudlák, P., Zane, F.: Satisfiability Coding Lemma, *Proc*. 38*th IEEE FOCS*, 1997, pp. 566–574.

[PR1] Pudlák, P., Rödl, V.: A Combinatorial Approach to Complexity, *Combinatorica*, **14** (1992), 221–226.

[PR2] Pudlák, P., Rödl, V.: Some Combinatorial-Algebraic Problems from Complexity Theory, *Discrete Math*., **136** (1994), 253–279.

[PRS] Pudlák, P., Rödl, V., Savický, P.: Graph Complexity, *Acta Informatica*, **25** (1988), 515–535.

[PSZ] Paturi, R., Saks, M., Zane, F.: Exponential Lower Bounds on Depth 3 Boolean Circuits, *Proc*. 29*th ACM STOC*, 1997, pp. 86–91.

[R1] Razborov, A. A.: A Lower Bound on the Monotone Network Complexity of the Logical Permanent, *Mat*. *Zametki* **37**(6) (1985), 887–900 (in Russian), English translation in: *Math*. *Notes* **37**(6) (1985), 485–493.

[R2] Razborov, A. A.: Lower Bounds on the Monotone Complexity of Some Boolean Functions, *Dokl*. *Akad*. *Nauk SSSR* **281**(4) (1985), 798–801 (in Russian), English translation in: *Soviet Math*. *Dokl*. **31** (1985), 354–357.

[R3] Razborov, A. A.: Lower Bounds on the Size of Bounded Depth Networks Over a Complete Basis with Logical Addition, *Mat*. *Zametki* **41**(4) (1987), 598–607 (in Russian), English translation in: *Math*. *Notes* **41**(4) (1987), 333–338.

[R4] Razborov, A. A.: Applications of Matrix Methods for the Theory of Lower Bounds in Computational Complexity, *Combinatorica*, **10** (1990), 81–93.

[RM] Raz, R., McKenzie, P.: Separation of the Monotone NC Hierarchy, *Proc*. 38*th IEEE FOCS*, 1997, pp. 234–243.

[RW] Raz, R., Wigderson, A.: Monotone Circuits for Matching Require Linear Depth, *J. Assoc*. *Comput*. *Mach*., **39**(3) (1992), 736–744.

[S] Smolensky, R.: Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity, *Proc*. 19*th STOC*, 1987, pp. 77–82.

[T] Tardos, É.: The Gap Between Monotone and Non-Monotone Circuit Complexity Is Exponential, *Combinatorica*, **8**(1) (1988), 141–142.

[V1] Valiant, L.: Graph-Theoretical Methods in Low-level Complexity, *Proc*. 6*th MFCS*, LNCS vol. 53, Springer-Verlag, Berlin, 1977, pp. 162–176.

[V2] Valiant, L.: Negation Is Powerless for Boolean Slice Functions, *SIAM J. Comput*., **15** (1986), 531–535.

[W] Wegener, I.: *The Complexity of Boolean Functions*, Wiley–Teubner Series in Computer Science, Teubner, Stuttgart/Wiley, Chichester, 1987.

[Y] Yao, A.: Some Complexity Questions Related to Distributive Computing, *Proc*. 11*th ACM STOC*, 1979, pp. 209–213.

Received January 30, 2002, *and in revised form July* 3, 2002. *Online publication November* 15, 2002.