Electronic Colloquium on Computational Complexity, Report No. 64 (2016)

Exponential Lower Bounds for Monotone Span Programs

Robert Robere*
University of Toronto
robere@cs.toronto.edu

Toniann Pitassi*
University of Toronto
toni@cs.toronto.edu

Benjamin Rossman†
NII, University of Toronto
rossman@cs.toronto.edu

Stephen A. Cook*
University of Toronto
sacook@cs.toronto.edu

April 19, 2016

Abstract

Monotone span programs are a linear-algebraic model of computation which were introduced by Karchmer and Wigderson in 1993 [30]. They are known to be equivalent to linear secret sharing schemes, and have various applications in complexity theory and cryptography. Lower bounds for monotone span programs have been difficult to obtain because they use non-monotone operations to compute monotone functions; in fact, the best known lower bounds are quasipolynomial for a function in (nonmonotone) P [9]. A fundamental open problem is to prove exponential lower bounds on monotone span program size for *any* explicit function.

We resolve this open problem by giving exponential lower bounds on monotone span program size for a function in monotone P. This also implies the first exponential lower bounds for linear secret sharing schemes. Our result is obtained by proving exponential lower bounds using Razborov's rank method [42], a measure that is strong enough to prove lower bounds for many monotone models. The best prior results on the rank measure were quasipolynomial lower bounds for a function in NP. As corollaries we obtain new proofs of exponential lower bounds for monotone formula size, monotone switching network size, and the first lower bounds for monotone comparator circuit size for a function in monotone P. We also obtain new polynomial degree lower bounds for Nullstellensatz refutations using an interpolation theorem of Pudlak and Sgall [37]. Finally, we obtain quasipolynomial lower bounds on the rank measure for the st-connectivity function, implying tight bounds for st-connectivity in all of the computational models mentioned above.

1 Introduction

Razborov [42] introduced a simple matrix-theoretic technique (which we will call the *rank method*) to study lower bounds on formula size for boolean functions, and using this method he was able to give a simple proof that any monotone formula computing a certain monotone function in NP must have size at least $n^{\Omega(\log n)}$. While not the strongest lower bound known against monotone formula size — similar bounds were already known for st-connectivity, and stronger lower bounds are known for other functions — Razborov's method is exceptionally elegant, and applies to models of computation

*Research supported by NSERC.
†Supported by the JST ERATO Kawarabayashi Large Graph Project.

ISSN 1433-8092

that seem to be out of reach of standard techniques. Two examples of such models are *monotone span programs* and *monotone switching networks* [20, 30]: monotone span programs use non-monotone (algebraic) operations to compute monotone functions, which makes them remarkably powerful and technically difficult to lower bound [3, 4, 8, 9, 20, 21, 30]; monotone switching networks are a classic model which resisted strong lower bounds for directed st-connectivity until Potechin [36] gave an ingenious Fourier-analytic argument.

Despite its elegance and connections with other models, very little is known about Razborov's rank method. In fact, Razborov's original argument is the only known lower bound for the rank measure, giving a quasipolynomial lower bound for a function in NP. This suggests several natural questions: First, is it possible to use Razborov's rank method to give nontrivial lower bounds for a function in monotone P or even in P? Secondly, can the rank method be used to prove exponential size lower bounds?

In this paper we resolve both of these problems. First, we prove $n^{\Omega(\log n)}$ lower bounds for directed st-connectivity using the rank method. Directed st-connectivity is one of the most basic functions: it is the canonical NL-complete problem and can be computed by polynomial-size, $O(\log^2 n)$-depth monotone circuits. Thus, our proof gives new (and arguably simpler) proofs of some celebrated results: it implies both Potechin's lower bound for monotone switching networks [36], as well as the classic Karchmer-Wigderson lower bound for monotone formulas [29].

Second, we prove exponential size lower bounds using the rank method against the GEN function, which is computable in monotone P. As well as being the first exponential lower bounds using the rank method, this implies both exponential lower bounds on monotone span program size for a function in monotone P (solving a well-known open problem), as well as the first exponential lower bounds for linear secret sharing schemes.

In addition, we show how to apply the rank method to *monotone comparator circuits*, which allows us to prove the first nontrivial lower bounds for any family of these circuits computing a function in monotone P. Before this, no size lower bounds were known for monotone comparator circuits except those implied by the classic lower bounds for computing clique and perfect matching [41].

It is known that the directed connectivity problem is computable by non-monotone span programs [30, 50], as well as non-monotone comparator circuits [17, 49], and so our argument also gives new examples of separations between monotone and non-monotone complexity classes.

Finally, using the interpolation argument by Pudlak and Sgall [37], we can apply our monotone span program lower bound to obtain degree lower bounds for the *Nullstellensatz* proof system, which is a natural algebraic proof system based on Hilbert's Nullstellensatz.

1.1 Monotone Span Programs and Related Models

Let $\mathbb{F}$ be any field. A span program is a matrix A over $\mathbb{F}$ with rows labelled by input literals. Given an assignment x to the input literals, the span program accepts x if and only if the all-1 vector is in the linear span of the rows of the matrix whose labels are satisfied by the input; a span program computes a boolean function $f : \{0,1\}^n \to \{0,1\}$ if it accepts exactly the 1-inputs of the function. The *size* of a span program is the number of rows of the matrix, and a span program is *monotone* if all rows are labelled with positive literals. Span programs were introduced by Karchmer and Wigderson [30], where they showed that non-monotone span programs capture logspace counting classes such as $\oplus$L and Mod_pL; monotone span programs are also known to characterize a subclass of secret sharing schemes known as *linear secret sharing schemes* [7, 46].

There is a fairly long history of lower bounds for monotone span programs. The first lower bounds

for monotone span programs, due to Karchmer and Wigderson [30], showed that all threshold functions over $\mathrm{GF}(2)$ require monotone span programs of size $\Omega(n \log n)$, which was quickly improved by Csirmaz [18] to an $\Omega(n^2/\log n)$ lower bound. Beimel et al. [8] gave a lower bound of $n^{5/2}$, and then Babai et al. [4] proved the first superpolynomial lower bound on the order of $n^{\Omega(\log n/\log\log n)}$. Each of these results were obtained by direct combinatorial arguments, which were simplified and improved by Gál to $n^{\Omega(\log n)}$ [20]. In the same paper, Gál observed the connection between monotone span programs and the rank method, and this connection was further investigated by Gál and Pudlák [21]. The superpolynomial lower bounds cited above only applied to functions computable in NP, so, to improve this Beimel and Weinreb [9] gave quasipolynomial lower bounds $n^{\Omega(\sqrt{\log n})}$ for a function in uniform NC^2 (therefore for a function in P), proving that monotone span programs can be weaker than polynomial time.

An interesting feature of monotone span programs is that they are not "really" monotone – monotone span programs use non-monotone operations to compute monotone functions. Largely because of this, the relationship between monotone span programs and monotone circuits has been unresolved. It is known that monotone span programs can be much more powerful than monotone circuits: Babai et al. [4] exhibited a function with linear size monotone span programs that requires superpolynomial-size monotone circuits and exponential-size monotone formulas. This immediately implies that the size and depth lower bound methods for monotone circuits cannot be used to prove lower bounds for monotone span programs.

Due to their strength there are several open problems concerning monotone span programs. First, it is open to prove exponential lower bounds on monotone span program size for any explicit function. Second, it is open to show whether there are functions in monotone P that require monotone span programs of superpolynomial size. Third, it is open to give an example of a function with small (non-monotone) span programs but requiring large monotone span programs.

Our main result for span programs is the following theorem.

Theorem 1.1. *The st-connectivity function requires $n^{\Omega(\log n)}$ size monotone span programs over $\mathbb{R}$. The GEN function requires $\exp(n^{\Omega(1)})$ size monotone span programs over $\mathbb{R}$.*

This resolves all of the open problems mentioned above. First, our lower bound for the GEN function is the first lower bound on monotone span program size greater than $n^{\Omega(\log n)}$ for any explicit function. Since GEN is in monotone P, this implies an exponential separation between monotone P and monotone span program size, resolving the second open problem mentioned above. Furthermore, our lower bound for st-connectivity implies a quasipolynomial separation between mNC^2 and monotone span programs, since st-connectivity is well-known to be computable by polynomial-size, $O(\log^2 n)$ depth monotone circuits. Finally, Karchmer and Wigderson showed that non-uniform polynomial-size span programs over $\mathrm{GF}(2)$ compute exactly those functions in $\oplus\mathsf{L}/\mathsf{poly}$. Wigderson [50] showed that $\mathsf{NL}/\mathsf{poly} \subseteq \oplus\mathsf{L}/\mathsf{poly}$, and since directed st-connectivity is in NL it follows it is computable by (non-uniform) polynomial-size span programs. Thus, we exhibit a function with small non-monotone span programs but requiring large monotone span program size.

Secret Sharing Schemes. A *secret sharing scheme* is a cryptographic tool where a dealer shares a secret among a set of participants such that only the "authorized" subsets of participants are able to reconstruct the secret [46]. The subsets correspond to a monotone boolean function f on n bits, where n is the number of participants. Monotone span program size measures the amount of information that has to be given to the participants in so-called *linear* secret sharing schemes; thus lower bounds on monotone span program size imply lower bounds on the length of the shares in linear secret sharing schemes [30]. Our result for GEN gives the first exponential lower bounds on the size of linear secret

sharing schemes. For s-t connectivity, our quasipolynomial lower bound is especially striking due to the known polynomial upper bounds for the access structure corresponding to *undirected* s-t connectivity.

Nullstellensatz. Nullstellensatz (NS) refutations are a natural algebraic proof system for proving unsolvability of systems of polynomial equations based on Hilbert's Nullstellensatz [5]. Given a set of polynomial equations $p_1 = 0, \ldots, p_m = 0$, an NS refutation is given by a sequence of polynomials $q_1, \ldots, q_m$ such that $\sum p_i q_i = 1$. The degree of the refutation is $d = \max_i \deg(q_i p_i)$, and the size is the total number of monomials in all of the polynomials.

Pudlak and Sgall [37] proved a strong connection between Nullstellsatz refutations and span programs. In particular, they proved that interpolants for degree-d refutations are exactly characterized by size $n^{O(d)}$ span programs, and the characterization also holds in the monotone setting. By this characterization, our lower bounds for monotone span programs imply strong (i.e. polynomial) lower bounds on the degree of NS refutations. While these are not the first strong lower bounds known for NS refutations, they are the first strong lower bounds proven via the interpolation method. Previous lower bounds achieving superlogarithmic degree bypassed the interpolation method, and were obtained either by studying combinatorial properties of the dual system of equations, or by explicitly constructing a Groebner basis for the original polynomials [11, 15, 43]. Thus the corresponding lower bounds are "instance-specific", as the argument depends on the specific combinatorial properties of the initial system of equations. In contrast, our approach yields a general methodology for proving NS lower bounds for a broad family of unsolvable equations.

1.2 Monotone Switching Networks

Switching networks are a non-uniform model of computation used to study space complexity. A switching network is specified by an undirected graph with two special nodes s, t and with an input literal labelled on each edge. Given an assignment x to the literals, the switching network *accepts* x if there is a path from s to t using literals consistent with the input assignment; the network then computes a boolean function $f : \{0,1\}^n \to \{0,1\}$ in the natural way. The size of a switching network is the number of nodes, and a switching network is *monotone* if every input literal labelled on an edge is positive. Polynomial-size, uniform switching networks compute exactly the languages in L [45], and so we use mL to denote the class of problems computable by polynomial-size, monotone switching networks.

It is well known that switching network size is closely related to circuit depth [10]: specifically, anything computable by a switching network with size s can be computed by a circuit with depth $O(\log^2 s)$. (A stronger converse is not hard to see: any function computable by depth-d circuits can be computed by a switching network with size $O(2^d)$.) This simulation is known to be tight as *undirected* st-connectivity is computable by linear-size switching networks, but any polynomial-size monotone circuit computing undirected st-connectivity requires depth $\Omega(\log^2 n)$ [27].

It is known that if a function can be computed by a switching network of size S, then it can also be computed by a span program of size S over any field, and the same holds for their monotone versions (see [30] for a proof). Thus polynomial-size monotone span programs (mSP) are at least as strong as both polynomial-size monotone formulas (mNC_1) and polynomial-size monotone switching networks (mL).

It was long conjectured that *directed* st-connectivity required quasi-polynomial size monotone switching networks, which was resolved in the affirmative by Potechin [36]. Potechin's argument is direct, and an impressive — albeit extremely technical — application of Fourier analytic techniques. Extending Potechin's techniques, Chan and Potechin [13] recently proved asymptotically tight $n^{\Omega(h)}$ lower bounds for the GEN function on pyramids of height h. (Weaker size lower bounds were previ-

ously implied by the work of Raz and McKenzie [39].) Thanks to the known simulation of monotone switching networks by monotone span programs we get alternative proofs of both of these theorems.

Theorem 1.2. *The st-connectivity function requires $n^{\Omega(\log n)}$ size monotone switching networks. The GEN function requires $\exp(n^{\Omega(1)})$ size monotone switching networks.*

1.3 Monotone Comparator Circuits

A *sorting network* is a model of a sorting algorithm which is input-oblivious. The model is quite simple: the network receives as input n integers on n parallel wires travelling from left to right, with a sequence of *comparator gates* connecting pairs of wires that map $(x, y) \mapsto (\min\{x, y\}, \max\{x, y\})$. The goal is to sort all n integer inputs using the fewest number of gates (or, alternatively, with the smallest depth). Shallow sorting networks have many applications in theoretical computer science, and explicit constructions have been extensively studied in the literature (see [31] for an extensive survey). The famous AKS construction gives $O(\log n)$ depth sorting networks [2]; an alternative construction was recently given by Goodrich [23].

When the inputs are restricted to be boolean, a sorting network is called a *comparator circuit*. Over the boolean domain the comparator gates become joint $(\wedge, \vee)$-gates, and it is not hard to see that circuits built of these gates are incapable of copying bits. Because of this, comparator circuits have been used as a method of studying *fanout* past NC^1 — this line of research was first followed by Subramanian [49]. The class of problems computable by polynomial-size uniform comparator circuits is called CC, and the structural complexity of this class was intensively studied by Cook, Filmus and Le [17]. Despite their inability to copy, polynomial-size comparator circuits are surprisingly powerful: they can compute everything in NL and appear to be incomparable with NC:

$$\mathsf{NL} \subseteq \mathsf{CC} \subseteq \mathsf{P}.$$

There are many interesting complete problems for CC, including: the stable marriage problem [17, 49], predicting internal diffusion-limited aggregation clusters in theoretical physics [34], the telephone connection problem [38], and, most recently, simulating the "Digi-Comp II", a wooden mechanical computer [1]. Since all known algorithms for the stable matching problem (which is complete for CC) are inherently sequential, Subramanian (and separately, Cook et al) conjectured that CC is not contained in NC [17, 49]. This conjecture has been supported by oracle separations [17].

Monotone comparator circuits are a natural restriction of comparator circuits where the input bits are either constants or positive literals; we let mCC denote the class of languages computable by polynomial-size monotone comparator circuits. This model is perhaps the most natural model for sorting (the famous AKS sorting network is a monotone comparator circuit [2]). However, when it comes to computing arbitrary monotone boolean functions essentially nothing is known. It is easy to see that monotone comparator circuits can simulate monotone formulas, and in turn can be simulated by (unrestricted) monotone circuits, but this is essentially it:

$$\mathsf{mNC}_1 \subseteq \mathsf{mCC} \subseteq \mathsf{mP}.$$

In particular, the relationship between mCC and mNC was open. We are able to show that the rank method applies to monotone comparator circuits, and so our main result for comparator circuits is the following.

Theorem 1.3. *The directed st-connectivity function requires $n^{\Omega(\log n)}$ size monotone comparator circuits. The GEN function requires $\exp(n^{\Omega(1)})$ size monotone comparator circuits.*

Since st-connectivity is in mNC^2, we establish that mCC is not contained in mNC^2, supporting the conjecture that CC is not contained in NC. Since directed st-connectivity is computable in NL, and $\mathsf{NL} \subseteq \mathsf{CC}$ [17, 49], our theorem also shows that non-monotone comparator circuits are more powerful than monotone comparator circuits even when computing monotone functions. Finally, since GEN is in mP, we establish an exponential separation between mP and mCC.

1.4 Overview of Proof

We will explain the main ideas in the context of the st-connectivity function, although the argument easily generalizes to any GEN function. (We will see that the st-connectivity function is a special instance of the GEN function where the underlying graph G is a path graph.) Let $f : \{0,1\}^n \to \{0,1\}$ be the *layered st-connectivity function* $\mathrm{STCONN} = \mathrm{STCONN}_{h,w}$, for which the input is a list of edges encoding a subgraph of the layered, directed graph with w layers and h nodes per layer. Let $U \subseteq \mathrm{STCONN}^{-1}(1)$ be a subset of the 1-inputs of STCONN (i.e. subgraphs containing an s-t path), and let $V \subseteq \mathrm{STCONN}^{-1}(0)$ be a subset of the 0-inputs of (i.e. subgraphs with an s-t cut). Let A be a $|U| \times |V|$ matrix over $\mathbb{R}$ with rows labelled by $u \in U$ and columns labelled by $v \in V$. For each underlying input variable x_e of STCONN, define the subrectangle R_e to be the set of pairs $(u, v) \in U \times V$ such that $u_e = 1$ and $v_e = 0$. Let $\mathcal{R}_{\mathrm{STCONN}}(U, V)$ denote the collection of all of these rectangles.

The *rank measure* of A is defined to be the ratio of the rank of A and the maximum rank of the submatrix of A indexed by any of these rectangles

$$\mu_A(\mathrm{STCONN}) = \frac{\operatorname{rank} A}{\max\limits_{R \in \mathcal{R}_{\mathrm{STCONN}}(U,V)} \operatorname{rank} A|_R}.$$

This measure was originally introduced by Razborov [42], and for any A the measure $\mu_A(\mathrm{STCONN})$ is a lower bound on each of the monotone computation models we have discussed above. Thus, our overall goal is to find a family of matrices $\{A_n\}$ for the STCONN function for which the rank measure is $n^{\Omega(\log n)}$.

At a high-level our argument is a reduction from the rank measure to reversible pebbling number. Essentially, the proof shows that the optimal monotone algorithm for st-connectivity in any computational model bounded by the rank measure simulates the *reversible pebbling game*, which is a well known combinatorial game used for measuring space complexity [13, 16, 36]. Previously, reductions to pebbling have been used to prove lower bounds on circuit depth [39], and also switching network size [13, 36]. Both of these arguments use a kind of "lifting" from a simple complexity measure to a harder complexity measure, and the works by Potechin and Chan/Potechin [13,36] use Fourier analysis in an intricate way. Our proof generalizes both of these arguments and is arguably simpler.

Our lower bound argument proceeds in two steps. First, we prove a "lifting theorem" connecting the rank measure to a new algebraic complexity measure on boolean functions that we call the *algebraic gap complexity*, which may be of independent interest. This lifting theorem uses the well-known *Pattern Matrix Method*, and is morally similar to the many query-to-communication complexity lifts in the literature [14, 26, 28, 32, 33, 39, 47, 48]. The second step is to actually prove a lower bound on the gap complexity.

Step 1: The Pattern Matrix Lift. Sherstov [48] gave a general method to construct a "pattern matrix" A_p from a boolean function $p : \{0,1\}^m \to \mathbb{R}$ such that the analytic properties of A_p are related to the Fourier analytic properties of the function p. The matrix is constructed as follows: the rows of A_p are indexed by strings $y \in \{0,1\}^n$ for some $n > m$, the columns of A_p are indexed by pairs

(x, w) where $x \in [n/m]^m$ is a string of "pointers" to indices in y, $w \in \{0,1\}^m$, and then for each pair $(y, (x, w))$ the value $A_p[y, (x, w)]$ is $p(y|_x \oplus w)$.

The main idea is to use a pattern matrix A_p (for a suitably chosen p) to certify a lower bound on the rank measure, using a theorem of Sherstov [48] showing that the rank of pattern matrices can be directly calculated from the Fourier spectrum of the function p used to generate the matrix. With this in mind, we show that the rows of the pattern matrix A_p can be interpreted as rejecting instances of the st-connectivity function (specifically a collection of s-t cuts) and the columns of A_p can be interpreted as accepting instances of the st-connectivity function (specifically a collection of s-t paths with length $m+1$). Using Sherstov's rank theorem we then calculate the rank of A_p directly from p, as well as the rank of each "rectangle submatrix" of A_p from $p|_e$, where $p|_e$ is a restriction of the function p obtained naturally from the edge e underlying the rectangle R_e. This implies that the matrix A_p will certify a large rank measure if the function p exhibits a large *algebraic gap*, in that the Fourier degree of p is large, but the Fourier degree of each of the restrictions $p|_e$ is small.

Step 2: Exhibiting Large Algebraic Gaps. The second step of our argument is to actually construct a function p exhibiting large algebraic gaps. We first show that for each positive integer m, the problem of constructing a boolean function $p : \{0,1\}^m \to \mathbb{R}$ with large algebraic gap is equivalent to the satisfiability of an (exponentially large) system of linear equations. To show this system is satisfiable, we introduce a new proof system that can be viewed as a depth-restricted form of resolution. Our main technical argument is a completeness theorem, showing that this system of linear equations is satisfiable if the (depth-restricted) proof system cannot refute the unsatisfiable CNF formula associated with st-connectivity (which happens to be the induction principle). Thus we reduce the problem of proving a large algebraic gap for st-connectivity to resolution depth of the induction principle. Since resolution-depth is equivalent to the decision tree-complexity of the corresponding search problem, our lower bound follows from the known $\Omega(\log m)$ lower bound on the reversible pebbling number of the m-node path graph. More generally, we prove that if we start with the a GEN function with minterms isomorphic to some template graph G, then algebraic gaps for the associated search problem can be obtained from lower bounds on the reversible pebbling number of G.

1.5 Related Work

There is an extensive literature on lower bounds for the size and depth of monotone computational models; we have reviewed many of the relevant results above. Here we examine other relevant results.

Razborov [42] introduced the rank measure and proved $n^{\Omega(\log n)}$ lower bounds for a function in NP by using the disjointness matrix; in a later work [40] he showed that the rank measure cannot give superlinear lower bounds in non-monotone models of computation. Razborov's lower bound on the rank measure was studied by Gál and Pudlak [21], where it was shown to be related to the method of avoiding families used in monotone span program lower bounds [4, 20].

Karchmer and Wigderson [30] showed that monotone span program size upper bounds the size of *linear secret sharing schemes*; Beimel showed that they give an exact characterization [6]. See the comprehensive survey of Beimel for more on secret sharing schemes [7]. Span programs have also been connected to quantum algorithms [44].

The idea of "lifting" lower bounds on a simple complexity measure from weak to strong computation models has appeared in many forms, and has been enormously successful for proving lower bounds for a variety of models. The basic idea is to start with an "outer" function f for which we have given a lower bound in a weak model of computation, and "lift" f by composing f with an "inner" function g to get a new function, $f \circ g^n$ that is provably hard in a stronger model of computation.

The *Pattern Matrix Method* is a particular instantiation of this due to Sherstov [48]; the crux of the method is to leverage the Fourier-analytic properties of the outer function f to prove lower bounds on linear algebraic properties of the matrix associated with the lifted function $f \circ g^n$. This method has led to many lower bounds in communication complexity, in classical, quantum, and number-on-forehead models [14, 24, 33, 48]. In circuit complexity similar approaches have led to strong lower bounds on monotone circuit depth [26,39], and similarly for lower bounds in proof complexity [26,28]. Other lifting techniques have given strong lower bounds against extended formulations of linear programs [25, 32].

2 Definitions

A real-valued boolean function is any function $p : \{0,1\}^n \to \mathbb{R}$. If A is any set and $x \in A^n$ we let x_i denote the ith component of x. If $x, y \in \{0,1\}^n$ we let $x \oplus y \in \{0,1\}^n$ denote the string obtained by taking the bitwise XOR of x and y.

For any n, the collection of all n-ary real-valued boolean functions $\{p : \{0,1\}^n \to \mathbb{R}\}$ forms a vector space under pointwise addition and scalar multiplication. For any $C \subseteq [n]$, the *Fourier character* at C is the function $\chi_C : \{0,1\}^n \to \{-1,1\}$ defined by

$$\chi_C(x) = (-1)^{\sum_{i \in C} x_i}.$$

The collection of characters $\{\chi_C\}_{C \subseteq [n]}$ form an orthonormal basis for the vector space of real-valued boolean functions known as the *Fourier basis*, where the vector space is equipped with the inner product

$$\langle p, q \rangle = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} p(x)q(x).$$

Since this basis is orthonormal, given any function $p : \{0,1\}^n \to \mathbb{R}$, we can represent p in the Fourier basis as

$$p(x) = \sum_{C \subseteq [n]} \langle p, \chi_C \rangle \chi_C(x).$$

This representation is called the *Fourier transform* of p.

We let $\hat{p}(C) = \langle p, \chi_C \rangle$ denote the coefficient of χ_C of p in the Fourier basis — this is the *Fourier coefficient* of p at C. The collection of non-zero Fourier coefficients of p is called the *Fourier spectrum* of p. The *Fourier degree* is the size of the largest non-zero Fourier coefficient of p:

$$\deg p = \max_{S \subseteq [m]} \{|S| \mid \hat{p}(S) \neq 0\},$$

which, equivalently, is the degree of the unique representation of p as a multilinear polynomial over the real numbers. See [35] for a comprehensive survey of boolean function analysis.

If $x, y \in \{0,1\}^n$ then we write $x \leq y$ if $x_i \leq y_i$ for all i. A function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if $f(x) \leq f(y)$ whenever $x \leq y$. If f is monotone then an input $x \in \{0,1\}^n$ is a *maxterm* of f if $f(x) = 0$ but $f(x') = 1$ for any x' obtained from x by flipping a single bit from 0 to 1; dually, x is a *minterm* if $f(x) = 1$ but $f(x') = 0$ for any x' obtained by flipping a single bit of x from 1 to 0. More generally, if $f(x) = 1$ we call x an *accepting instance* or a *yes instance*, while if $f(x) = 0$ then we call x a *rejecting instance* or a *no instance*. If x is any yes instance of f and y is any no instance of f then there exists an index $i \in [n]$ such that $x_i = 1, y_i = 0$, as otherwise we would have $x \leq y$, contradicting the fact that f is monotone.

Suppose that $U, V \subseteq \{0,1\}^n$ are any sets satisfying $f(U) = 1, f(V) = 0$. A set $R \subseteq U \times V$ is called a *rectangle* if there are sets $U_0 \subseteq U, V_0 \subseteq V$ such that $R = U_0 \times V_0$. For each $i \in [n]$ let

$$X_i = \{x \in \{0,1\}^n \mid x_i = 1\} \times \{x \in \{0,1\}^n \mid x_i = 0\},$$

and let $R_i = X_i \cap (U \times V)$. Denote by $\mathcal{R}_f(U,V)$ the collection of rectangles

$$\mathcal{R}_f(U,V) = \{R_i \mid i = 1, 2, \ldots, n\}.$$

Since f is a monotone function there is an index i such that $u_i = 1, v_i = 0$ for all $u \in U, v \in V$, and so every entry of $U \times V$ is covered by some rectangle in $R_f(U,V)$. Let A be any $|U| \times |V|$ matrix with rows labelled by entries of U and columns labelled by entries of V, and if $S \subseteq U \times V$ is any subset of $U \times V$ let $A|_S$ be the submatrix indexed by S.

Definition 2.1. Let $f : \{0,1\}^n \to \{0,1\}$ and let $U \subseteq f^{-1}(1), V \subseteq f^{-1}(0)$. Let A be any $|U| \times |V|$ matrix over $\mathbb{R}$[1]. The *rank measure* of f with respect to A is

$$\mu_A(f) := \frac{\operatorname{rank}(A)}{\max\limits_{R \in \mathcal{R}_f(U,V)} \operatorname{rank}(A|_R)}.$$

The rank measure was introduced by Razborov [42] to give simple superpolynomial lower bounds on the size of monotone boolean formulas. He did so by showing that the rank measure lower bounded monotone formula size, and then constructed a family of monotone functions f_n (computable in NP) and a matrix A_n such that $\mu_{A_n}(f_n) \geq n^{\Omega(\log n)}$. In this paper we give a similar result for variants of the GEN problem.

Definition 2.2. Let n be a positive integer, and let $L \subseteq [n]^3$ be a collection of triples on $[n]$. For any 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 triples of vertices L and two distinguished points $s, t \in [n]$, decide if t generated from $\{s\}$.

Formally, an instance of GEN is given by two nodes $s, t \in [n]$ and n^3 boolean values coding the set $L \subseteq [n]^3$. For definiteness, in the remainder of the paper assume s, t are arbitrary fixed points in $[n]$, and we let GEN denote the corresponding monotone function.

We can naturally associate GEN instances with some graphs.

Definition 2.3. A DAG $G = (V, E)$ is *good* if it is connected, has maximum in-degree 2, and has a unique sink node.

If G is a good DAG then we can form an instance of GEN from G by

1. Adding triples connecting the source point s to the sources of G,
2. Adding a triple connecting the sink node of G to the target t, and

[1] This definition makes sense with respect to any field, but we will work exclusively in the reals.

3. For each internal node z, if z has in-degree 2 with distinct in-edges $(x, z), (y, z)$, then add a triple (x, y, z). Otherwise, if z has in-degree 1 with an in-edge (x, z), then add the triple (x, x, z).

We say input triples obtained in this way are *legal*. The lower bounds in this paper are proven for sub-problems of GEN obtained by "lifting" good graphs G.

Definition 2.4. Let G be a good DAG, let t be the sink node of G, and let o be a positive integer. The *o-lifted graph* $G^{\uparrow o}$ is obtained by taking the tensor product of G with the complete directed graph on o vertices, and then adding a "super-source" node $\mathbf{s}$ and a "super-target" node $\mathbf{t}$. Explicitly, $G^{\uparrow o}$ is obtained from G as follows: we replace each node $u \in G$ with o copies $\left\{u^{(1)}, u^{(2)}, \ldots, u^{(o)}\right\}$. For each edge (u, v) add o^2 edges $(u^{(i)}, v^{(j)})$ for all $i, j \in [o]$. Finally, add a new source node $\mathbf{s}$ and a new target node $\mathbf{t}$ and add edges connecting $\mathbf{s}$ to the lifted source nodes $u^{(i)}$, as well as edges connecting the lifted sink nodes $t^{(i)}$ to the target node $\mathbf{t}$.

Given a node $u^{(i)} \in G^{\uparrow o}$ let $\pi(u^{(i)}) = u$ be the underlying node in the graph G. The $G^{\uparrow o}$-GEN *problem* is a subproblem of GEN obtained by restricting the allowed input triples to those triples of vertices $(u, v, w) \in G^{\uparrow o}$ such that $(\pi(u), \pi(v), \pi(w))$ is a legal triple of the underlying graph G.

The following proposition connects GEN and st-connectivity.

Proposition 2.5. *Let m, o be positive integers. Let P_m be the directed path graph with m nodes and let $\mathrm{STCONN}_{o,m}$ be the st-connectivity function on the graph $P_m^{\uparrow o}$. Then $\mathrm{STCONN}_{o,m} = P_m^{\uparrow o}$-GEN. (See Figure 1.)*

Proof. As alluded to in Definition 2.2, the legal triples in the path graph P_m are each of the form (u, u, v) for each pair of nodes (u, v) connected by an edge in P_m. It follows that asking if the node $\mathbf{t}$ can be generated from $\mathbf{s}$ is equivalent to asking whether or not there is a path from $\mathbf{s}$ to $\mathbf{t}$ using the triples in the input that are set to 1, which is exactly the st-connectivity problem. □

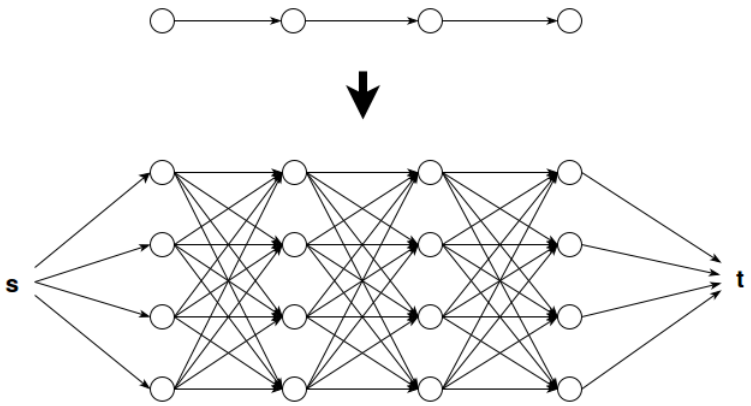

Figure 1: A path P_4 and the lifted graph $P_4^{\uparrow 4}$ (we have added a new source node $\mathbf{s}$ and a new target node $\mathbf{t}$ connected to the lifted source and target nodes). The function $P_4^{\uparrow 4}$-GEN is exactly the layered s-t connectivity problem $\mathrm{STCONN}_{4,4}$.

As usual in monotone circuit complexity, we will be interested in a particular collection of accepting and rejecting instances of $G^{\uparrow o}$-GEN. The accepting instances $\mathcal{Y}$ will be exactly those sets of triples T such that the graph underlying T obtained by applying the projection π to each node in T is isomorphic

to G. (For example, in $P_4^{\uparrow 4}$-GEN depicted in Figure 1, the accepting instances $\mathcal{Y}$ are just the s-t paths.) The rejecting instances $\mathcal{N}$ are called *cut instances*; they are obtained by choosing a subset C vertices of $G^{\uparrow o}$ that contains s, and then adding all triples **except** those that "cross the cut", in the sense that $u, v \in C$ and $w \notin C$. Clearly each cut instance is a rejecting instance of $G^{\uparrow o}$-GEN.

We also need a variant of the well-known black pebbling game on DAGs [13, 36].

Definition 2.6. Let $G = (V, E)$ be a good DAG with sources R and a unique sink t, and 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 the in-neighbours of x are pebbled, a *legal pebbling move* consists of either pebbling or unpebbling x, see Figure 2. Since the source nodes $s \in R$ do not have any in-neighbours they can always be pebbled or unpebbled.

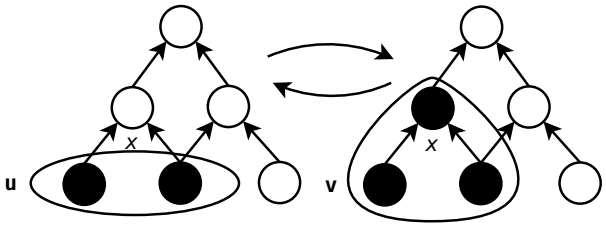

Figure 2: Legal pebbling moves involving x; the corresponding pebbling configurations are $\mathbf{u}$ and $\mathbf{v}$

The goal of the reversible pebbling game is as follows: starting with the empty configuration, place a pebble on t using only legal pebbling moves such that the maximum number of pebbles in any pebbling configuration is minimized. Formally, we want to find a sequence of pebbling configurations $\emptyset = S_0, S_1, \ldots, S_n$ 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 *pebbling sequence* for G. The *cost* of the sequence is $\max_i |S_i|$. The *reversible pebbling number* of a DAG G, denoted $\mathsf{rpeb}(G)$, is the minimum cost of a reversible pebbling sequence for G.

3 Rank Measure Lower Bounds

The main result in this paper is a lower bound on the rank measure of $G^{\uparrow o}$-GEN in terms of the reversible pebbling number of the underlying graph G.

Theorem 3.1. *Let G be any good DAG with m vertices. There is a real matrix A such that*

$$\mu_A(G^{\uparrow 2m^2}\text{-GEN}) \geq \Omega(m^{\mathsf{rpeb}(G)}).$$

This theorem implies a number of lower bounds in monotone complexity theory, both old and new. In the remainder of this section we use Theorem 3.1 to give proofs of each of these lower bounds. We begin by introducing the graphs which are used to prove our exponential lower bounds.

Definition 3.2. A *pyramid graph* with h levels is defined as follows. Introduce $h(h-1)/2$ vertices V, partitioned into h sets $V_1, V_2, \ldots, V_h$ where V_i has i vertices. Order V_i as $v_{i,1}, v_{i,2}, \ldots, v_{i,i}$; then for each $i = 2, 3, \ldots, h$, if $v_{i,j}$ and $v_{i,j+1}$ are adjacent vertices in V_i add two edges $(v_{i,j}, v_{i-1,j})$ and $(v_{i,j+1}, v_{i-1,j})$. (See Figure 2 for height-3 pyramid graphs.)

Theorem 3.3. *Let h be a positive integer, let Δ_h be the pyramid graph with h levels, and let $m = \binom{h}{2}$ be the number of nodes in Δ_h. Let $N = O(m^7)$ be the number of input triples to $\Delta_h^{\uparrow 2m^2}$-GEN. Let $\varepsilon = 1/14$. Then there is a real matrix A such that*

$$\mu_A(\Delta_h^{\uparrow 2m^2}\text{-GEN}) \geq 2^{\Omega(N^\varepsilon \log N)}.$$

Proof. Cook [16] showed that the black pebbling number of pyramids is h, where the black pebbling game is the same as the reversible pebbling game except you are always allowed to remove pebbles. Clearly every reversible pebbling strategy is a black pebbling strategy, thus $\mathsf{rpeb}(\Delta_h) \geq h$. Since $h = \Omega(\sqrt{m})$ and $m = \Omega(N^{1/7})$, setting $\varepsilon = 1/14$ and applying Theorem 3.1 yields

$$\mu_A(\Delta_h^{\uparrow 2m^2}\text{-GEN}) \geq m^{\Omega(h)} \geq N^{\Omega(N^\varepsilon)} \geq 2^{\Omega(N^\varepsilon \log N)}. \qquad \square$$

We remark that Gilbert and Tarjan [22] constructed a family of good DAGs with reversible pebbling number $\Omega(n/\log n)$, which is known to be tight due to an upper bound by Dymond and Tompa [19]. One could use these graphs to obtain lower bounds, but essentially this would just improve the value of ε in the previous theorem from $1/14$ to $1/7$.

We also use a lower bound by Potechin [36] on the reversible pebbling number of path graphs to give a lower bound for st-connectivity.

Theorem 3.4. *Let m be a positive integer, let P_m be the directed path with m nodes, and consider the lifted path graph $P_m^{\uparrow 2m^2}$. Let $N = O(m^5)$ be the number of input triples to $\mathrm{STCONN}_{2m^2,m}$. Then there is a real matrix A such that*

$$\mu_A(\mathrm{STCONN}_{2m^2,m}) \geq N^{\Omega(\log N)}.$$

Proof. Proposition 2.5 shows that $P_m^{\uparrow 2m^2}$-GEN is exactly $\mathrm{STCONN}_{2m^2,m}$. Potechin [36] proved that the reversible pebbling number of the path graph P_m is $\Omega(\log m)$, so using the fact that $m = \Omega(N^{1/5})$ and applying Theorem 3.1 implies

$$\mu_A(\mathrm{STCONN}_{2m^2,m}) = \mu_A(P_h^{\uparrow 2m^2}\text{-GEN}) \geq m^{\Omega(\log m)} \geq N^{\Omega(\log N)}. \qquad \square$$

3.1 Span Program Lower Bounds and Corollaries

Let $\mathbb{F}$ be any field, and let $\vec{1}$ be the all-1s vector. A *monotone span program* over $\mathbb{F}$ is a matrix M with its rows labelled by boolean variables $x_1, \ldots, x_n$. On an input $x \in \{0,1\}^n$, let M_x denote the submatrix of M containing all rows labelled with variables set to 1 by x. The program *accepts* the input if $\vec{1}$ lies in the linear span of the rows of M_x. Note that any monotone span program computes a monotone function since the linear span of a set of vectors is monotone nondecreasing. Let $\mathsf{mSP}_\mathbb{F}$ denote the set of all monotone functions computable by polynomial-size monotone span programs over $\mathbb{F}$, and let $\mathsf{mSP}_\mathbb{F}(f)$ denote the size of the smallest monotone span program over $\mathbb{F}$ computing f.

It is known that monotone span programs can simulate monotone switching networks and so they can compute any monotone function [30]. In fact, it has been shown that there is a function computable by monotone span programs that is *not* computable by polynomial-size monotone circuits [4]. In this section, we use Theorems 3.3 and 3.4 to show that monotone span programs and monotone circuits are *incomparable*: there exists a function computable by polynomial-size monotone circuits which requires exponential-size monotone span programs, and there is also a function computable by polynomial-size, $O(\log^2 n)$ depth monotone circuits requiring superpolynomial-size monotone span programs.

Gál [20] showed that the rank measure is a lower bound on monotone span program size.

Theorem 3.5 (Lemma 3.2 and Theorem 3.4 in [20])**.** *Let $\mathbb{F}$ be any field, let f be any monotone boolean function, and let $U \subseteq f^{-1}(1)$ and $V \subseteq f^{-1}(0)$. Then for any $|U| \times |V|$ matrix A over $\mathbb{F}$*

$$\mu_A(f) = \frac{\operatorname{rank}(A)}{\max\limits_{R \in \mathcal{R}_f(U,V)} \operatorname{rank}(A|_R)} \leq \mathsf{mSP}_{\mathbb{F}}(f).$$

Theorem 3.3 therefore gives exponential lower bounds on the size of *real* monotone span programs computing $\Delta_h^{\uparrow 2m^2}$-GEN, showing $\mathsf{mP} \not\subseteq \mathsf{mSP}$.

Theorem 3.6. *Let h be a positive integer, let m be the number of nodes in the height-h pyramid graph Δ_h, and let N be the number of input variables to the function $\Delta_h^{\uparrow 2m^2}$-GEN. Let $\varepsilon = 1/14$. Then*

$$\mathsf{mSP}_{\mathbb{R}}(\Delta_h^{\uparrow 2m^2}\text{-GEN}) \geq 2^{\Omega(N^{\varepsilon} \log N)}.$$

Theorem 3.4 gives (tight) superpolynomial lower bounds the size of monotone span programs computing $\mathrm{STCONN}_{2m^2,m}$, showing $\mathsf{mNC}^2 \not\subseteq \mathsf{mSP}$.

Theorem 3.7. *Let m be a positive integer, and let N be the number of input variables to the function $\mathrm{STCONN}_{2m^2,m}$. Then*

$$\mathsf{mSP}_{\mathbb{R}}(\mathrm{STCONN}_{2m^2,m}) \geq N^{\Omega(\log N)}.$$

It is known that polynomial-size, non-monotone span programs can compute STCONN if the span programs are allowed to be non-uniform [50], and so the previous theorem also separates monotone span programs from non-monotone span programs. Furthermore, since monotone span programs can simulate monotone switching networks, the above two theorems give alternative proofs of the recent results by Potechin [36] and Chan-Potechin [13] that STCONN requires superpolynomial-size monotone switching networks and $\Delta_h^{\uparrow 2m^2}$-GEN requires exponential-size monotone switching networks.

Corollary 3.8. *Any monotone switching network computing $\Delta_h^{\uparrow 2m^2}$-GEN requires $2^{\Omega(N^{\varepsilon} \log N)}$ states, where $\varepsilon = 1/14$. Any monotone switching network computing $\mathrm{STCONN}_{2m^2,m}$ requires $N^{\Omega(\log N)}$ states.*

Secret Sharing Schemes. A *secret sharing scheme* is a basic cryptographic tool roughly defined as follows (we follow the presentation in [7]; we refer the interested reader there for formal definitions and numerous applications). We have a *dealer* who has a "secret" (say, an element of some field $\mathbb{F}$), a collection of n parties, and a collection $\mathcal{A} \subseteq 2^{[n]}$ of subsets of the n parties which we call an *access structure*. A secret sharing scheme for $\mathcal{A}$ is a method of sharing information with the n parties such that any set of parties in $\mathcal{A}$ can reconstruct the dealer's secret, while any subset of parties not contained in $\mathcal{A}$ cannot reconstruct the dealer's secret. (As a result of the above definition we assume that $\mathcal{A}$ is upward-closed — if $A \in \mathcal{A}$ and $A \subseteq B$ then $B \in \mathcal{A}$.) In a *linear* secret sharing scheme the shares of information given to the parties are vectors in some vector space over $\mathbb{F}$, wherein for each subset of parties in the access structure $\mathcal{A}$ the span of the vectors given contains a certain target vector.

Beimel [6] showed that the size of the smallest monotone span program tightly characterizes the amount of information required to be shared in linear secret sharing schemes. Before our results, the best known lower bounds against any linear secret sharing scheme were quasipolynomial.

Corollary 3.9. *There is an explicitly defined access structure $\mathcal{A}_{\Delta\text{-GEN}}$ such that any linear secret sharing scheme for $\mathcal{A}_{\Delta\text{-GEN}}$ has information ratio $2^{\Omega(N^{\varepsilon} \log N)}$ for $\varepsilon = 1/14$.*

Nullstellensatz. Recall from the introduction that a *Nullstellensatz refutation* of a set of polynomial equations $p_1 = 0, \ldots, p_m = 0$ is given by a system of polynomials $q_1, \ldots, q_m$ such that $\sum p_i q_i = 1$. The degree of the refutation is $d = \max_i \deg(q_i p_i)$, and the size is the total number of monomials in all of the polynomials.

Let $P = \{p_1(\vec{x}, \vec{y}) = 0, \ldots, p_m(\vec{x}, \vec{y}) = 0\}$, $Q = \{q_1(\vec{x}, \vec{z}) = 0, \ldots, q_m(\vec{x}, \vec{z}) = 0\}$ be a system of unsolvable equations in variables $\vec{x}, \vec{y}, \vec{z}$, where $\vec{y}$, $\vec{z}$ are disjoint sets of variables. The system (P, Q) is *monotone* if all $\vec{x}$ variables occurring in P are negative. An *interpolant* for (P, Q) is a boolean function $f(\vec{x})$ with the property that for every assignment $\vec{u}$ to $\vec{x}$, $f(\vec{u}) = 0$ implies that $\{p_1(\vec{u}, \vec{y}) = 0, \ldots, p_m(\vec{u}, \vec{y}) = 0\}$ is unsolvable, and $f(\vec{u}) = 1$ implies that $\{q_1(\vec{u}, \vec{z}) = 0, \ldots, q_m(\vec{u}, \vec{z}) = 0\}$ is unsolvable. If (P, Q) is monotone then an interpolant for (P, Q) is a monotone function [37].

Pudlak and Sgall [37] proved that Nullstellensatz refutations of monotone systems (P, Q) have interpolants computed by monotone span programs:

Theorem 3.10. *Let (P, Q) be a monotone system of unsolvable polynomial equations over a field $\mathbb{F}$, and suppose that (P, Q) has a degree-d NS refutation. Then there is a size $n^{O(d)}$ monotone span program computing an interpolant for the system over $\mathbb{F}$.*

We apply this theorem, using our monotone span program lower bounds, to obtain lower bounds on the degree of NS refutations for certain unsatisfiable polynomial systems. Let f be any function computable by a polynomial-size monotone circuit C_f. We describe a corresponding set of monotone unsolvable polynomial equations, $(P_f(\vec{x}, \vec{y}), Q_f(\vec{x}, \vec{z}))$ associated with f. The $\vec{x}$ variables of $P_f(\vec{x}, \vec{y})$ and $Q_f(\vec{x}, \vec{z})$ are inputs to C_f; the $\vec{y}$ variables of P_f (respectively the $\vec{z}$ variables of Q_f) describe the values assigned to each of the gates in C_f on input $\vec{x}$. Roughly speaking, the equations in P_f say that $\vec{x}$ is a 1-input of f, and the equations in Q_f say that $\vec{x}$ is a 0-input of f. We do this gate-by-gate: for each $\wedge$ gate w with input gates u, v, the P_f equation corresponding to w says that if both y_u and y_v are 1, then y_w is 1, and the Q_f equation corresponding to w says that if either z_u or z_v are 0, then z_w is 0. Similarly if w is an $\vee$ gate with inputs u, v then the P_f equation for w says that if either y_u or y_v is 0, then y_w is 1, and the Q_f equation says that if z_u and z_v are 0, then z_w is 0. For the output gate we add the equation $y_{out} = 1$ in P_f and $z_{out} = 0$ in Q_f. Clearly these equations are unsatisfiable, monotone, and the *unique* monotone interpolant for (P_f, Q_f) is f. Note that the number of variables in P, Q is $\mathsf{poly}(n)$ where n is the number of variables of f, and the degree of P, Q is at most 3.

Applying Theorem 3.4 and Theorem 3.3 we obtain the following lower bounds on Nullstellensatz.

Theorem 3.11. *Every real Nullstellensatz refutation of the system $(P_{\mathrm{STCONN}_{2m^2,2m}}, Q_{\mathrm{STCONN}_{2m^2,2m}})$ has degree $\Omega(\log n)$. Every real Nullstellensatz refutation of the system $(P_{\Delta_h^{\uparrow 2m^2\text{-GEN}}}, Q_{\Delta_h^{\uparrow 2m^2\text{-GEN}}})$ requires Nullstellensatz refutations with degree $\Omega(N^{\varepsilon})$ for some $\varepsilon > 0$.*

Proof. As we have argued above, $\mathrm{STCONN}_{2m^2,m}$ is the interpolant for the system

$$(P_{\mathrm{STCONN}_{2m^2,2m}}, Q_{\mathrm{STCONN}_{2m^2,2m}}).$$

Combining Theorem 3.10 and Theorem 3.4 yields the first part of the theorem. The second part of the theorem is obtained similarly using Theorem 3.3. □

3.2 Comparator Circuit Lower Bounds

We recall the definition of comparator circuits. A *comparator gate* is the function mapping a pair of input bits $(x, y) \mapsto (x \wedge y, x \vee y)$; it is natural to think of a comparator gate as "sorting" the input

(x, y), since the smaller input goes to the first coordinate and the larger input goes to the second. A *comparator circuit* consists of m wires and a sequence $(i_1, j_1), (i_2, j_2), \ldots, (i_s, j_s)$ of comparator gates, each connecting a pair of wires (in this notation, the $\wedge$ output of the comparator gate is attached to the first wire, and the $\vee$ output of the comparator gate is attaced to the second wire). Each of the m wires is labelled with either a constant $0, 1$, some input variable x or its negation $\overline{x}$.

We will be interested in comparator circuits which compute boolean functions $f : \{0,1\}^n \to \{0,1\}$, where n is possibly less than the number of wires. To do this we designate one of the wires as the output wire and allow the labelling of distinct wires with the same input variable. A comparator circuit C is *monotone* if no input wire of C is labelled with the negation of an input variable, and it is clear from the monotonicity of comparator gates that monotone comparator circuits compute only monotone functions.

Figure 3 shows a simple comparator circuit. In this comparator circuit each gate is drawn as an

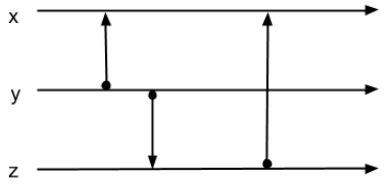

Figure 3: A simple comparator circuit

arrow with a circle on one end. The arrow represents the $\vee$ gate and the circle represents the $\wedge$ gate, so we think of any 1 in the input to a gate as being "pushed" towards the arrow.

If C is a comparator circuit then the *size* of C is the number of wires[2] in C. Let mCC denote the class of languages computable by polynomial-size monotone comparator circuits, and if f is a monotone boolean function let $\mathsf{mCC}(f)$ denote the minimum size of any comparator circuit computing f.

Practically nothing is known about monotone comparator circuits other than trivial simulations. It is easy to see that monotone comparator circuits can directly simulate monotone formulas, and can in turn be directly simulated by monotone circuits; it therefore follows from Razborov's lower bound [41] against polynomial-size monotone circuits that the k-Clique problem cannot be computed by small monotone comparator circuits. In this section we use our rank-measure results to lower bound the size of monotone comparator circuits computing functions in monotone P.

Recall that a *monotone complexity measure* is any function μ mapping monotone boolean functions to $\mathbb{R}$ satisfying the following axioms, for all boolean functions f, g and all coordinate functions x_i:

$$\max\{\mu(f \wedge g), \mu(f \vee g)\} \leq \mu(f) + \mu(g)$$
$$\max\{\mu(x_i)\} \leq 1.$$

If μ satisfies the stronger axiom

$$\mu(f) + \mu(g) \geq \mu(f \wedge g) + \mu(f \vee g)$$

then we say that μ is a *submodular complexity measure*. Razborov proved that the function μ_A is a submodular complexity measure.

[2]It is not hard to show that the number of wires and the number of gates in any comparator circuit without redundant gates are separated by at most a quadratic factor.

Theorem 3.12 (Theorem 1, [40])**.** *For any matrix A the function $\mu_A(\cdot)$ is a submodular complexity measure.*

We obtain our lower bounds by showing that submodular complexity measures lower bound monotone comparator circuit size. This is achieved by first introducing a generalized complexity measure (a *multicomplexity measure*) that lower bounds monotone comparator circuit size, and then showing that submodular complexity measures are multicomplexity measures.

To be more precise, a function ρ is a *multicomplexity measure* if ρ maps *sequences* of Boolean functions to $\mathbb{R}$ such that the following inequalities hold (note that all inequalities below are quantified over all sequences of boolean functions and all t, when necessary):

$$\forall i,j : \rho(f_1, f_2, \ldots, f_i, \ldots, f_j, \ldots, f_t) \geq \rho(f_1, f_2, \ldots, f_i \wedge f_j, \ldots, f_i \vee f_j, \ldots, f_t) \quad (1)$$

$$\rho(f_1, f_2, \ldots, f_t) + \rho(f_{t+1}, \ldots, f_m) \geq \rho(f_1, f_2, \ldots, f_t, f_{t+1}, \ldots, f_m) \quad (2)$$

$$1 \geq \rho(x_i) \quad (3)$$

$$\rho(f_1, f_2, \ldots, f_{t-1}, f_t) \geq \rho(f_1, f_2, \ldots, f_{t-1}) \quad (4)$$

$$\forall \pi \in \text{Permutation}(t), \rho(\pi(f_1, f_2, \ldots, f_t)) = \rho(f_1, f_2, \ldots, f_t) \quad (5)$$

For any sequence of monotone boolean functions $f_1, f_2, \ldots, f_t$ let $\mathsf{mCC}(f_1, f_2, \ldots, f_t)$ denote the number of wires in the smallest monotone comparator circuit computing $f_1, f_2, \ldots, f_t$ among its outputs.

Proposition 3.13. *Let $f_1, f_2, \ldots, f_t$ be any sequence of monotone boolean functions and let ρ be a multicomplexity measure. Then*

$$\rho(f_1, f_2, \ldots, f_t) \leq \mathsf{mCC}(f_1, f_2, \ldots, f_t).$$

Proof. Let t be an arbitrary positive integer, and we prove the proposition by induction on $\mathsf{mCC}(f_1, f_2, \ldots, f_t)$. If $\mathsf{mCC}(f_1, \ldots, f_t) = 1$ then $t = 1$ and f_1 is a variable, and so the inequality follows from (3). So, suppose $s = \mathsf{mCC}(f_1, \ldots, f_t) > 1$ and let C be a comparator circuit witnessing $\mathsf{mCC}(f_1, \ldots, f_t)$. We may assume that C has some nonzero number of comparator gates, for if C has no comparator gates then each function in $\{f_1, \ldots, f_t\}$ is a variable and the proposition follows from the inductive hypothesis and repeated applications of (2) and (3). Let C' be the circuit obtained from C by removing (starting from the output) the minimum number of comparator gates $c_1, c_2, \ldots, c_s \in [s]^2$ such that C' can be partitioned into two disjoint comparator circuits C_1, C_2 with no comparator gates connecting C_1 and C_2. Clearly $s = |C_1| + |C_2|$, and let $g_1, g_2, \ldots, g_i$ be the functions output by C_1 and $g_{i+1}, \ldots, g_s$ the functions output by C_2. Applying the inductive hypothesis we have

$$s = |C_1| + |C_2| \geq \rho(g_1, g_2, \ldots, g_i) + \rho(g_{i+1}, \ldots, g_s) \geq \rho(g_1, \ldots, g_s),$$

where we have applied (2). Now, apply rule (1) to the pairs of wires dictated by the sequence of comparator gates $c_s, c_{s-1}, \ldots, c_1$, obtaining

$$s \geq \rho(g'_1, g'_2, \ldots, g'_s),$$

and note that $\{f_1, \ldots, f_t\} \subseteq \{g'_1, g'_2, \ldots, g'_s\}$. Applying (4) finishes the proof. □

Now we show that submodular complexity measures lower bound monotone comparator circuit size.

Proposition 3.14. *Let μ be a non-negative, submodular complexity measure. For any monotone function f*

$$\mu(f) \leq \mathsf{mCC}(f).$$

Proof. We show that $\rho(f_1, f_2, \ldots, f_t) = \sum_i \mu(f_i)$ is a multicomplexity measure, and the proposition immediately follows. Observe that equations (4), (5), (6), and (7) easily follow from the definition of ρ and the non-negativity of μ. To see that (3) holds we apply submodularity:

$$\begin{aligned}\rho(f_1, f_2, \ldots, f_t) &= \mu(f_1) + \cdots + \mu(f_i) + \cdots + \mu(f_j) + \cdots + \mu(f_t) \\ &\geq \mu(f_1) + \cdots + \mu(f_i \wedge f_j) + \cdots + \mu(f_i \vee f_j) + \cdots + \mu(f_t) \\ &= \rho(f_1, f_2, \ldots, f_i \wedge f_j, \ldots f_i \vee f_j, \ldots f_t).\end{aligned}$$

□

By combining this proposition with Theorem 3.3 we obtain exponential lower bounds on the size of monotone comparator circuits computing $\Delta_h^{\uparrow 2m^2}$-GEN, separating mCC and mP.

Theorem 3.15. *Let h be a positive integer, let m be the number of nodes in the height-h pyramid graph Δ_h, and let N be the number of input variables to the function $\Delta_h^{\uparrow 2m^2}$-GEN. Let $\varepsilon = 1/14$. Then*

$$\mathsf{mCC}(\Delta_h^{\uparrow 2m^2}\text{-GEN}) \geq 2^{\Omega(N^{\varepsilon} \log N)}.$$

Proof. By Theorem 3.12, the rank measure $\mu_A(\cdot)$ is a submodular complexity measure. Combining Theorem 3.3 with Proposition 3.14 yields the result. □

Using Theorem 3.4, we obtain superpolynomial lower bounds on monotone comparator circuits computing STCONN, separating mCC from mNC^2 and also exhibiting a problem computable by polynomial-size, *non*-monotone comparator circuits that is not computable by monotone comparator circuits.

Theorem 3.16. *Let m be a positive integer, and let N be the number of input variables to the function* $\mathrm{STCONN}_{2m^2,m}$. *Then*

$$\mathsf{mCC}(\mathrm{STCONN}_{2m^2,m}) \geq N^{\Omega(\log N)}.$$

Proof. Identical to the proof of the previous theorem, using Theorem 3.4 in place of Theorem 3.3. □

4 Reducing the Rank Measure to Algebraic Gaps

The rest of the paper is devoted to the proof of Theorem 3.1. In this section we show how to reduce the problem of constructing a matrix A witnessing Theorem 3.1 to the construction of a boolean function satisfying certain Fourier-analytic properties. Our reduction is general, and naturally phrased using the canonical search problem associated with unsatisfiable CNFs.

Definition 4.1. Let k be a positive integer and let $\mathcal{C} = C_1 \wedge C_2 \wedge \cdots \wedge C_q$ be an unsatisfiable k-CNF on variables $z_1, z_2, \ldots, z_m$. Associated with $\mathcal{C}$ is the following search problem $\mathsf{Search}(\mathcal{C})$: given an assignment $z \in \{0,1\}^m$ to the variables of $\mathcal{C}$, output a falsified clause C_i.

Each clause C has a unique falsifying assignment, and given a boolean function $p : \{0,1\}^m \to \mathbb{R}$ on the same variables as $\mathcal{C}$ we let $p\restriction_C$ denote the restriction of the function p by this assignment. For example, the single falsifying assignment for $C = \neg z_u \vee \neg z_v \vee z_w$ sets $z_u = 1, z_v = 1, z_w = 0$, so the corresponding function $p\restriction_C$ has the corresponding variables restricted accordingly.

The reduction roughly proceeds as follows. We start with the search problem $\mathsf{Search}(\mathcal{C})$ and introduce a new algebraic complexity measure on such search problems that we call the *algebraic gap complexity*. We give a generic method to convert the search problem $\mathsf{Search}(\mathcal{C})$ into a monotone boolean function $f_{\mathcal{C}}$ by a variant of the construction introduced by Raz and Mckenzie [39]. (In particular, if we begin with the so-called *pebbling contradictions*, then the resulting lifted function $f_{\mathcal{C}}$ is exactly GEN.) The main theorem of this section (Theorem 4.7) gives a method to lift lower bounds on the algebraic gap complexity of $\mathsf{Search}(\mathcal{C})$ to lower bounds on the rank measure of the function $f_{\mathcal{C}}$.

We now introduce our new algebraic complexity measure. Recall that $\deg p$ is the size of the largest non-zero Fourier coefficient of p.

Definition 4.2. Let $\mathcal{C}$ be an unsatisfiable CNF on m variables. The *algebraic gap complexity* of $\mathsf{Search}(\mathcal{C})$ is the largest integer k for which there is a boolean function $p : \{0,1\}^m \to \mathbb{R}$ such that

$$\deg p = m \quad \text{and} \quad \deg p\restriction_C \leq m - k$$

for all clauses C in $\mathcal{C}$.

Next we review *pattern matrices*, which are used to convert the search problem $\mathsf{Search}(\mathcal{C})$ into a monotone boolean function $f_{\mathcal{C}}$ such that algebraic gap lower bounds on $\mathsf{Search}(\mathcal{C})$ imply rank-measure lower bounds for $f_{\mathcal{C}}$. Let o, m be positive integers, let $n = om$, and let

$$V(o,m) = [o]^m \times \{0,1\}^m .$$

Given $x \in [o]^m$ and $y \in \{0,1\}^n$ construct a string $y\restriction_x$ as follows: first, partition $[n]$ into m blocks of size o and write $y = (y_{i,j})$ for $i \in [m]$, $j \in [o]$; then for each $i \in [m]$ set the ith value of $y\restriction_x$ to y_{i,x_i}.

Definition 4.3. Let m, o be positive integers, let $n = om$, and let $p : \{0,1\}^m \to \mathbb{R}$. The (m,o,p)-*pattern matrix* is the real matrix A given by

$$A = [p(y\restriction_x \oplus w)]_{y \in \{0,1\}^n, (x,w) \in V(o,m)} .$$

Note that the rows of the (m,o,p)-pattern matrix are indexed by strings $y \in \{0,1\}^n$, and the columns are indexed by pairs $(x,w) \in V(o,m)$.

Observe that pattern matrices convert real-valued boolean functions $p : \{0,1\}^m \to \mathbb{R}$ into matrices. While they were originally introduced to prove lower bounds on communication complexity [48], here they will be useful as we can easily calculate their rank from the Fourier spectrum of the underlying function p.

Lemma 4.4. *Let $p : \{0,1\}^m \to \mathbb{R}$ be given and let A be the (m,o,p)-pattern matrix. The rank of A is*

$$\operatorname{rank} A = \sum_{S : \hat{p}(S) \neq 0} o^{|S|}.$$

Proof. Essentially Theorem 4.3 in [48]. □

The Pattern Matrix Lift. Let m, o be positive integers, let $n = om$, and consider any unsatisfiable d-CNF $\mathcal{C}$ on m variables $z_1, z_2, \ldots, z_m$. We show how to use a pattern matrix to "lift" the search problem $\mathsf{Search}(\mathcal{C})$ to a monotone boolean function $f_{\mathcal{C}}$ such that algebraic gap lower bounds for $\mathsf{Search}(\mathcal{C})$ translate to rank measure lower bounds for $f_{\mathcal{C}}$. Our lifted function is a variant of the transformation first given by Raz and Mckenzie [39] and then further explored by Göös and Pitassi [26].

Definition 4.5. Let d, m, o be positive integers and let $n = om$. Let $\mathcal{C}$ be an unsatisfiable d-CNF on m variables $z_1, z_2, \ldots, z_m$. The $(\mathcal{C}, o)$-*lifted function* $f_{\mathcal{C}}$ is the $\{0,1\}$-valued monotone boolean function defined as follows. The variables of $f_{\mathcal{C}}$ are indexed by pairs $(C, (a, b))$, where C is a clause in $\mathcal{C}$ and

$$(a, b) \in [o]^{\mathsf{vars}(C)} \times \{0,1\}^{\mathsf{vars}(C)}.$$

Given a $\{0,1\}$-assignment to the variables of $f_{\mathcal{C}}$, the function outputs 1 if there is an $(x, w) \in V(o, m)$ such that for each clause C, the variable $(C, (x_{\mathsf{vars}(C)}, w_{\mathsf{vars}(C)}))$ is set to 1. Clearly the function is monotone, and observe that $f_{\mathcal{C}}$ has $|\mathcal{C}|(2o)^d$ input variables.

Define the following set of accepting and rejecting instances of $f_{\mathcal{C}}$:

Accepting Instances $\mathcal{Y}$. For any $(x, w) \in V(o, m)$ let $Y(x, w)$ be the accepting instance obtained by setting all variables of the form $(C, (x_{\mathsf{vars}(C)}, w_{\mathsf{vars}(C)}))$ to 1 for each clause C in $\mathcal{C}$.

Rejecting Instances $\mathcal{N}$. For any $y \in \{0,1\}^n$ let $N(y)$ be the rejecting instance obtained by setting the variable $(C, (x_{\mathsf{vars}(C)}, w_{\mathsf{vars}(C)}))$ to 1 iff the clause C is satisfied when evaluated on the string $y\restriction_{x_{\mathsf{vars}(C)}} \oplus w_{\mathsf{vars}(C)}$. Observe that this is a 0-input of $f_{\mathcal{C}}$ since the formula $\mathcal{C}$ is unsatisfiable.

Let A be the (m, o, p)-pattern matrix for some function p. There is an obvious bijective map from the rows $y \in \{0,1\}^{om}$ of A to the rejecting instances $N(y)$ of the $(\mathcal{C}, o)$-lifted function $f_{\mathcal{C}}$, and similarly from the columns $(x, w) \in V(o, m)$ of A and the accepting instances $Y(x, w)$ of $f_{\mathcal{C}}$. We therefore consider the rank measure $\mu_A(f_{\mathcal{C}})$ of $f_{\mathcal{C}}$ with respect to A. Since $f_{\mathcal{C}}$ is monotone, for each accepting instance $Y(x, w)$ and each rejecting instance $N(y)$ there is an input variable $(C, (a, b))$ of $f_{\mathcal{C}}$ set to 1 in $Y(x, w)$ and set to 0 in $N(y)$. If we consider, for each input variable $(C, (a, b))$, the rectangle $R \subseteq \mathcal{Y} \times \mathcal{N}$ of all inputs"intersecting" in $(C, (a, b))$ in this sense, then in the rank measure $\mu_A(f_{\mathcal{C}})$ we must analyze the rank of the submatrices of A corresponding to such inputs. It is therefore important to understand the structure of $A\restriction_R$ for each rectangle $R \in \mathcal{R}_{f_{\mathcal{C}}}(\mathcal{Y}, \mathcal{N})$.

Given how pattern matrices are defined it is reasonable to suspect that the submatrices $A\restriction_R$ are related to restrictions of the underlying function p by the variable $(C, (a, b))$ corresponding to R. It turns out that this is true, and in fact the submatrix $A\restriction_R$ is (essentially) a pattern matrix generated by the restricted function $p\restriction_C$. Using this fact we can apply the rank lemma for pattern matrices (Lemma 4.4) to calculate the rank of the submatrix $A\restriction_R$ by examining the Fourier spectrum of the restricted function $p\restriction_C$.

Lemma 4.6. *Let o, m be positive integers and let $n = om$. Let $\mathcal{C}$ be an unsatisfiable d-CNF defined on m variables $z_1, z_2, \ldots, z_m$. Let $p : \{0,1\}^m \to \mathbb{R}$, and let A be the (m, o, p)-pattern matrix. Let $(C, (a, b))$ be any input variable of $f_{\mathcal{C}}$, and let R be the rectangle corresponding to $(C, (a, b))$ in $\mathcal{R}_{f_{\mathcal{C}}}(\mathcal{Y}, \mathcal{N})$. Then*

$$\operatorname{rank}(A\restriction_R) = \sum_{S : \widehat{p\restriction_C}(S) \neq 0} o^{|S|}.$$

Note that the above sum is taken over all subsets S of the unrestricted variables of $p\restriction_C$.

Proof. Let d be the arity of the clause C. Let A' denote the $(o, m-d, p\restriction_C)$-pattern matrix. We claim that the matrix $A\restriction_R$ is row equivalent to the matrix

$$\begin{pmatrix} A' \\ A' \\ \vdots \\ A' \end{pmatrix}$$

for some number of copies of A'. To see the claim gives the lemma, observe that that the claim implies that $\operatorname{rank} A\restriction_R = \operatorname{rank} A'$. Since A' is the $(o, m-d, p\restriction_C)$-pattern matrix, Lemma 4.4 implies that

$$\operatorname{rank} A\restriction_R = \operatorname{rank} A' = \sum_{S:\widehat{p\restriction_C}(S)\neq 0} o^{|S|},$$

and the lemma follows.

Let us prove the claim. Let $(x, w) \in V(o, m)$ and $y \in \{0,1\}^n$ be any pair appearing in the rectangle R, and note that the corresponding entry of the matrix $A\restriction_R$ is $p(y\restriction_x \oplus w)$. By definition of R, the variable $(C, (a, b))$ is set to 1 in $Y(x, w)$ and set to 0 in $N(y)$. This means that $a = x_{\mathsf{vars}(C)}$ and $b = w_{\mathsf{vars}(C)}$, so write the variable as $(C, (x_{\mathsf{vars}(C)}, w_{\mathsf{vars}(C)}))$ accordingly. By definition of $N(y)$, the variable $(C, (x_{\mathsf{vars}(C)}, w_{\mathsf{vars}(C)}))$ is set to 0 in $N(y)$ if and only if the clause C is *not* satisfied by the assignment

$$v = y\restriction_{x_{\mathsf{vars}(C)}} \oplus w_{\mathsf{vars}(C)}.$$

It follows that $(y\restriction_x \oplus w)\restriction_{\mathsf{vars}(C)} = v$ for each pair $((x, w), y)$ in the rectangle R.

This implies that each entry of $A\restriction_R$ does not depend on the values of $y_{i,j}$ for any i indexing a variable in $\mathsf{vars}(C)$ and any $j \neq x_i$. Thus, let us restrict $y_{i,j} = 0$ for all such $y_{i,j}$ and let A'' be the submatrix of $A\restriction_R$ obtained by this further restriction on y. We show that $A'' = A'$, and the claim follows from letting the further restricted values $y_{i,j}$ range over $\{0, 1\}$.

Since $(C, (a, b))$ must lie in the accepting instance $Y(x, w)$, it follows that $x_{\mathsf{vars}(C)} = a$ and $w_{\mathsf{vars}(C)} = b$, as stated above, while the rest of x and w can range arbitrarily over $V(o, m-d) = [o]^{m-d} \times \{0,1\}^{m-d}$. Similarly, since $(C, (a, b))$ is not in the rejecting instance $N(y)$, it follows that $(y\restriction_x \oplus w)\restriction_{\mathsf{vars}(C)}$ is the unique falsifying assignment to the variables of the clause C. We have restricted the values $y_{i,j} = 0$ for all i indexing variables in $\mathsf{vars}(C)$ and all $j \neq x_i$, and the remaining values of y can range over $\{0,1\}^{n-od}$ arbitrarily. Collecting it all together, we can write the entry of A'' indexed by $((x, w), y)$ as

$$\begin{aligned} &p((y\restriction_x \oplus w)_{\mathsf{vars}(C)}, (y\restriction_x \oplus w)_{d+1}, \ldots, (y\restriction_x \oplus w)_m) \\ &= p\restriction_C((y\restriction_x \oplus w)_{d+1}, \ldots, (y\restriction_x \oplus w)_m) \end{aligned}$$

where we have assumed that the first d variables of p are $\mathsf{vars}(C)$ without loss of generality. Letting y range over $\{0,1\}^{n-od}$ and (x, w) range over $V(o, m-d)$ shows $A'' = A'$, finishing the proof of the claim and the lemma. □

Now we have that the rank of the pattern matrix A is related to the degree of p, while the rank of the submatrix $A\restriction_R$ is related to the degree of $p\restriction_C$, where $(C, (a, b))$ is the input variable of $f_{\mathcal{C}}$ corresponding to the rectangle R. It follows that to maximize the rank measure we need a function p which maximizes the difference between these two degrees (thus explaining the definition of the algebraic gap complexity). This is formalized in the next theorem, which is the main result of this section.

Theorem 4.7. *Let m, d be positive integers and let $\mathcal{C}$ be an unsatisfiable d-CNF on m variables. Let k be the algebraic gap complexity of $\mathsf{Search}(\mathcal{C})$, and let $f_{\mathcal{C}}$ be the $(\mathcal{C}, m^2)$-lifted function. There is a matrix A such that $\mu_A(f_{\mathcal{C}}) \geq cm^k$ for some universal constant c.*

Proof. Let $o = m^2$, let $p : \{0,1\}^m \to \mathbb{R}$ be the function witnessing the algebraic gap complexity of $\mathsf{Search}(\mathcal{C})$, and let A be the (m, o, p)-pattern matrix. We lower bound

$$\mu_A(f_{\mathcal{C}}) = \frac{\operatorname{rank} A}{\max\limits_{R \in \mathcal{R}_{f_{\mathcal{C}}}(\mathcal{Y},\mathcal{N})} \operatorname{rank} A\restriction_R}.$$

By Lemma 4.4,

$$\operatorname{rank} A = \sum_{S:\hat{p}(S)\neq 0} o^{|S|} \geq m^{2m}$$

since $\hat{p}([m]) \neq 0$ and $o = m^2$. Let $R \in \mathcal{R}_{f_{\mathcal{C}}}(\mathcal{Y},\mathcal{N})$ be chosen arbitrarily, let $(C,(a,b))$ be the input variable of $f_{\mathcal{C}}$ corresponding to R. Note that we may assume that $\hat{p}(S) = 0$ for all $S \subseteq [m]$ with $|S| < m-k$ w.l.o.g. since this does not affect the algebraic gap exhibited by p. Since $\deg p\restriction_C \leq m-k$, it follows that all of the non-zero Fourier coefficients $\widehat{p\restriction_C}(S')$ are obtained as a linear combination of non-zero Fourier coefficients of $\hat{p}(S)$ where $|S| \leq |S'| + d$. Applying Lemma 4.6 and using these two facts, we have

$$\begin{aligned}
\operatorname{rank} A\restriction_R = \sum_{S:\widehat{p\restriction_C}(S)\neq 0} o^{|S|} &\leq \sum_{i=0}^{d} \binom{m}{k-i} m^{2(m-k-i)} \\
&\leq \sum_{i=0}^{d} \left(\frac{em}{k-i}\right)^{k-i} m^{2(m-k-i)} \\
&\leq \sum_{i=0}^{d} \left(\frac{e}{k-i}\right)^{k-i} m^{2m-k-i} \\
&\leq m^{2m-k} \sum_{i=0}^{d} \left(\frac{e}{k-i}\right)^{k-i} \leq 6m^{2m-k},
\end{aligned}$$

since $e + (e/2)^2 + (e/3)^3 + \cdots \leq 6$. Putting it all together we get

$$\mu_A(f_{\mathcal{C}}) = \frac{\operatorname{rank} A}{\max\limits_{R \in \mathcal{R}_{f_{\mathcal{C}}}(\mathcal{Y},\mathcal{N})} \operatorname{rank} A\restriction_R} \geq \frac{m^{2m}}{6m^{2m-k}} \geq cm^k$$

where $c = 1/6$. □

5 Reducing Algebraic Gaps to Reversible Pebbling for GEN

Recall that a DAG G is *good* if it is connected, has maximum in-degree 2, and has a unique sink node t. In this section we complete the proof of Theorem 3.1 by using *pebbling contradictions*.

Definition 5.1. Let $G = (V, E)$ be any good DAG with sources R and target t. Let Peb_G denote the following unsatisfiable CNF formula. There is one variable z_v for each vertex $v \in V$, and we add the following clauses:

1. The *target clause* $(\neg z_t)$.
2. For each source vertex $u \in R$ add the *source clause* (z_u).

3. For each internal vertex w with in-neighbours $W \subseteq V$ add the *edge clause* $\left(z_w \vee \bigvee_{v \in W} \neg z_v\right)$.

By Theorem 4.7 proved at the end of the previous section, proving lower bounds against the algebraic gap complexity of $\mathsf{Search}(\mathsf{Peb}_G)$ will imply rank measure lower bounds for f_{Peb_G}. The main theorem of this section is that the algebraic gap complexity of $\mathsf{Search}(\mathsf{Peb}_G)$ is at least the reversible pebbling number of G.

Theorem 5.2. *For any good DAG G the algebraic gap complexity of $\mathsf{Search}(\mathsf{Peb}_G)$ is at least $\mathsf{rpeb}(G)$.*

We leave the proof of Theorem 5.2 to the next section, and use it to prove Theorem 3.1. First we prove a lemma showing that f_{Peb_G} is a restriction of the function $G^{\uparrow 2o}$-GEN.

Lemma 5.3. *Let G be a good DAG, let o be a positive integer, and let f_{Peb_G} be the (Peb_G, o)-lifted function. Then f_{Peb_G} can be obtained from $G^{\uparrow 2o}$-GEN by restricting some input variables to 0.*

Proof. Suppose $(C, (a, b))$ is an input variable of f_{Peb_G}, where C is a clause of Peb_G and $(a, b) \in [o]^{\mathsf{vars}(C)} \times \{0,1\}^{\mathsf{vars}(C)}$. Consider the following indexing scheme for the nodes of $G^{\uparrow 2o}$: if u is a node of G then let the lifted nodes of u in $G^{\uparrow 2o}$ be represented by $\left\{u^{(i,b)} \mid i \in [o], b \in \{0,1\}\right\}$. Define an inductive mapping ρ from the variables of f_{Peb_G} to the variables of $G^{\uparrow 2o}$-GEN as follows, depending on the type of the clause C.

1. If C is the target clause $\neg z_t$, then $(a, b) \in [o] \times \{0,1\}$. Set $\rho(C, (a, b)) = (t^{(a,b)}, t^{(a,b)}, \mathbf{t})$.
2. If C is a source clause z_v for some source of G, then again $(a, b) \in [o] \times \{0,1\}$. Set $\rho(C, (a, b)) = (\mathbf{s}, \mathbf{s}, s^{(a,b)})$.
3. If C is an edge clause corresponding to the node w with in-neighbours u, v then write $a = a_w a_u a_v$ and $b = b_w b_u b_v$. Set $\rho(C, (a, b)) = (u^{(a_u, b_u)}, v^{(a_v, b_v)}, w^{(a_w, b_w)})$. If w has a single in-neighbour u, then set $\rho(C, (a, b)) = (u^{(a_u, b_u)}, u^{(a_u, b_u)}, w^{(a_w, b_w)})$.

Set all input variables of $G^{\uparrow 2o}$-GEN not mapped to by ρ to 0. Suppose that z is an assignment to the variables of f_{Peb_G} such that $f_{\mathsf{Peb}_G}(z) = 1$, and we argue that $G^{\uparrow 2o}\text{-GEN}(\rho(z)) = 1$. Recall that $f_{\mathsf{Peb}_G}(z) = 1$ if and only if there is an $(x, w) \in V(o, m)$ such that for each clause C the variable $(C, (x_{\mathsf{vars}(C)}, w_{\mathsf{vars}(C)})) = 1$. Given $(x, w) \in V(o, m)$, consider the induced subgraph of $G^{\uparrow 2o}$ obtained by adding the vertex $u^{(x_u, w_u)}$ for all nodes u in G. This subgraph is isomorphic to G, and by the definition of the mapping ρ, all input triples connecting the nodes in this subgraph are set to 1 in the input $\rho(z)$. Thus $G^{\uparrow 2o}\text{-GEN}(\rho(z)) = 1$.

Conversely, suppose that $G^{\uparrow 2o}\text{-GEN}(z) = 1$. Then the input z must contain a subgraph of $G^{\uparrow 2o}$ isomorphic to G by the definition of $G^{\uparrow 2o}$-GEN and by the restriction ρ. By inverting the function ρ, a similar argument as above shows that it is easy to map this subgraph into a pair (x, w) certifying that $f_{\mathsf{Peb}_G}(\rho^{-1}(z)) = 1$. □

Theorem 3.1. *Let G be any good DAG with m vertices. There is a real matrix A such that*

$$\mu_A(G^{\uparrow 2m^2}\text{-GEN}) \geq \Omega(m^{\mathsf{rpeb}(G)}).$$

Proof. Let G be any good DAG with m vertices. By Theorem 5.2 we have that the algebraic gap complexity of $\mathsf{Search}(\mathsf{Peb}_G)$ is at least $\mathsf{rpeb}(G)$. Let f_{Peb_G} be the (Peb_G, m^2)-lifted function. Theorem 4.7 implies that there is a matrix A such that

$$\mu_A(f_{\mathsf{Peb}_G}) \geq \Omega(m^{\mathsf{rpeb}(G)}).$$

Finally, Lemma 5.3 shows that f_{Peb_G} is a restriction of $G^{\uparrow 2m^2}$-GEN, and so

$$\mu_A(G^{\uparrow 2m^2}\text{-GEN}) = \mu_A(f_{\mathsf{Peb}_G}) \geq \Omega(m^{\mathsf{rpeb}(G)}).$$ □

5.1 Proving Algebraic Gap Lower Bounds for $\mathsf{Search}(\mathsf{Peb}_G)$

Theorem 5.2 is proved by a series of reductions. First we show that the algebraic gap complexity of $\mathsf{Search}(\mathsf{Peb}_G)$ is at least k if a certain system of linear equations is satisfied. By considering the dual of this system, we can construct a natural proof system such that depth lower bounds on the proof system imply that the original system of linear equations is satisfiable (this can be viewed as a completeness result for the system). Finally, we show how to simulate this proof system by resolution, from which depth lower bounds are known to follow from reversible pebbling [12].

We begin by constructing the system of equations equivalent to exhibiting algebraic gap complexity. In this section we will abuse notation when writing Fourier coefficients and often write $\hat{p}(S, T, x)$ to mean $\hat{p}(S \cup T \cup \{x\})$.

Lemma 5.4. *Let k be a positive integer and let G be any good DAG with sources R and sink t. Let $m = |V|$. The search problem Peb_G has algebraic gap complexity at least k if there is a boolean function $p : \{0,1\}^V \to \mathbb{R}$ such that $\deg p = m$ and the following holds: for any clause C of Peb_G and for any $S \subseteq V$ such that $|S| \geq m - k + 1$ and S does not contain any vertices with variables in C we have*

$$\hat{p}(S) = \begin{cases} -\hat{p}(S, u) & C \text{ is a source clause for some } u \in R, \\ \hat{p}(S, t) & C \text{ is the target clause.} \\ -\sum_{T \subseteq W \cup \{w\}} (-1)^{|T \cap W|} \hat{p}(S, T) & C \text{ is the edge clause } z_w \vee \bigvee_{v \in W} \neg z_v \text{ for } w. \end{cases}$$

Proof. This is an immediate consequence of Proposition 3.21 in [35], although we sketch a proof for completeness.

Suppose that $p : \{0,1\}^V \to \mathbb{R}$ is a function satisfying the requirements of the lemma, and we show that the fork game has algebraic gap complexity at least k. Since $\deg p = m$ by assumption, we just need to show that $\deg p\restriction_C \leq m - k$ for each clause C.

Assume that C is the source clause for $u \in S$, and the other cases follow symmetrically. Recall that the restriction corresponding to C sets $x_u = 0$. To show $\deg p\restriction_C \leq m - k$ we must show that $\widehat{p\restriction_C}(S) = 0$ for all $S \subseteq V \setminus \{u\}$ with $|S| > m - k$. We claim that

$$\widehat{p\restriction_C}(S) = \hat{p}(S) + \hat{p}(S, u).$$

This is easy to see if we change basis from $\{0,1\}$ to $\{-1,1\}$ by the mapping $x \mapsto 1 - 2x$. It is well-known that the Fourier transform of any function $q : \{-1,1\}^m \to \mathbb{R}$ is simply the representation of q as a real multilinear polynomial. So, applying the change of basis and thinking of $p : \{-1,1\}^m \to \mathbb{R}$ as a multilinear polynomial, the terms corresponding to $\hat{p}(S)$ and $\hat{p}(S, u)$ in the Fourier transform of p are

$$\hat{p}(S) \prod_{v \in S} x_v + \hat{p}(S, u) x_u \prod_{v \in S} x_v.$$

Under the mapping $x \mapsto 1 - 2x$, the restriction $x_u = 0$ becomes $x_u = 1$, so the coefficient $\widehat{p\restriction_C}(S)$ of $\prod_{v \in S} x_v$ is $\hat{p}(S) + \hat{p}(S, u)$. By assumption, $\hat{p}(S) = -\hat{p}(S, u)$, and so $\widehat{p\restriction_C}(S) = 0$. □

For any good DAG $G = (V, E)$ with m vertices, target node t, and any $k \leq m$, the previous lemma gives an algebraic gap complexity lower bound of $k + 1$ whenever the following system $\mathcal{G}(G, k)$ is satisfiable.

System of Equations $\mathcal{G}(G,k)$.

1. *Degree Equation.* $\hat{p}(V) = 1$.
2. *Root Equations.* For each $S \subseteq V$ with $|S| \geq m - k$ and for which the target $t \notin S$ we have
$$\hat{p}(S) - \hat{p}(S,t) = 0.$$
3. *Source Equations.* For any source vertex u of G and for each $S \subseteq V$ with $|S| \geq m - k$ and $u \notin S$ we have
$$\hat{p}(S) + \hat{p}(S,u) = 0.$$
4. *Edge Equations.* For each vertex $w \in V$ with in-neighbours W and each $S \subseteq V$ with $|S| \geq m-k$ and $(W \cup \{w\}) \cap S = \emptyset$ we have
$$\sum_{T \subseteq W \cup \{w\}} (-1)^{|T \cap W|} \hat{p}(S,T) = 0.$$

Corollary 5.5. *If $\mathcal{G}(G,k)$ has a solution then Peb_G has algebraic gap complexity $k+1$.*

We introduce some general notation to represent the equations appearing in $\mathcal{G}(G,k)$. For any $A, S, U \subseteq V$ with $|U| \leq 1$ and $U \cap (A \cup S) = \emptyset$ let $\mathcal{E}(A,S,U)$ denote the equation
$$0 = \sum_{T \subseteq A \cup U} (-1)^{|T \cap A|} \hat{p}(S,T).$$

With this notation the root equations are represented by $\mathcal{E}(\{r\}, S, \emptyset)$, the source equations are represented by $\mathcal{E}(\emptyset, S, \{u\})$, and the edge equations are represented by $\mathcal{E}(W, S, \{w\})$.

We now introduce derivation rules on these equations. First, for any $A, S, U \subseteq V$ with $U \cap (A \cup S) = \emptyset$ and any $y \notin U$ add the "weakening" rule
$$\mathcal{E}(A,S,U) \vdash \mathcal{E}(A, S \cup \{y\}, U). \tag{6}$$

For two equations $\mathcal{E}(A,S,\{x\})$ and $\mathcal{E}(B,S,U)$ where $U \cap A = \emptyset$ and $x \in B$ we introduce a *cut rule*
$$\mathcal{E}(A,S,\{x\}), \mathcal{E}(B,S,U) \vdash \mathcal{E}(A \cup B \setminus \{x\}, S \cup \{x\}, U). \tag{7}$$

Let $\mathcal{G}^*(G,k)$ denote the system of equations obtained by taking the closure of the equations in $\mathcal{G}(G,k)$ under the derivation rules.

The following proposition will be useful later.

Proposition 5.6. *For any equation in $\mathcal{G}^*(G,k)$ of the form $\mathcal{E}(A,S,U)$, if $U \neq \emptyset$ then for any $a \in A, u \in U$ there is a path in G connecting a to u.*

Proof. We prove the proposition by structural induction on $\mathcal{E}$-proofs. Each equation $\mathcal{E}(A,S,U)$ is given by a proof from the axioms in $\mathcal{G}(G,k)$. For the source axioms $\mathcal{E}(\emptyset,S,\{u\})$ and the target axiom $\mathcal{E}(\{t\},S,\emptyset)$ the proposition holds vacuously. Similarly, for any edge axiom $\mathcal{E}(W,S,w)$, each node in W is connected by an edge to w, and so the proposition also follows.

If the equation $\mathcal{E}(A,S,U)$ is obtained by applying the weakening rule (6) to $\mathcal{E}(A, S \setminus \{y\}, U)$, then the sets A and U are not changed and so the proposition follows from the inductive hypothesis.

On the other hand, if we have two equations $\mathcal{E}(A, S, \{x\})$ and $\mathcal{E}(B, S, U)$ with $x \in B$ and U disjoint from $A \cup S$, then by applying the cut rule (7) we obtain the equation $\mathcal{E}(A \cup (B \setminus \{x\}), S \cup \{x\}, U)$. Every node in $B \setminus \{x\}$ has a path to each node in U by the inductive hypothesis. Similarly, each node $a \in A$ has a path to each node $u \in U$ by concatenating the path connecting a to x (guaranteed by $\mathcal{E}(A, S, \{x\})$) and the path connecting x to u (guaranteed by $\mathcal{E}(B, S, U)$ since $x \in B$). □

Observe that the equation $\mathcal{E}(\emptyset, V, \emptyset)$ is exactly $\hat{p}(V) = 0$, which contradicts the degree equation $\hat{p}(V) = 1$. It follows that if $\mathcal{G}^*(G, k)$ contains this equation then the system is unsatisfiable. In the next lemma we show that this is the *only* obstruction to the system's satisfiability: if $\mathcal{G}^*(G, k)$ does not contain $\mathcal{E}(\emptyset, V, \emptyset)$, then it is actually satisfiable. (Observe that this essentially a "completeness" result for the proof system.)

The proof of the lemma is straightforward, if technical. An obvious method to construct a solution to the system $\mathcal{G}^*(G, k)$ is as follows. Begin by setting $\hat{p}(V) = 1$, and then proceed to set the values $\hat{p}(S)$ for $S \subsetneq V$ using the value induced by equations $\mathcal{E}(A, S, U)$ appearing in $\mathcal{G}^*(G, k)$. To show that this actually gives a consistent assignment we must show that the value induced on $\hat{p}(S)$ by any pair of equations $\mathcal{E}(A, S, X)$, $\mathcal{E}(B, S, Y)$ is the same. If $S = V$, then this is true iff $\mathcal{E}(\emptyset, V, \emptyset)$ is not in $\mathcal{G}^*(G, k)$; for smaller sets we prove this directly using downward induction.

The next definition formalizes what we mean for an equation $\mathcal{E}(A, S, U)$ to induce a value on $\hat{p}(S)$.

Definition 5.7. For $S \subseteq V$ an *S-equation* is any equation $\mathcal{E}$ in $\mathcal{G}^*(G, k)$ of the form $\mathcal{E}(A, S, U)$ for some (possibly empty) A, U, or the equation $\hat{p}(V) = 1$ if $S = V$. For any equation $\mathcal{E} \in \mathcal{G}^*(G, k)$ define the *value induced by $\mathcal{E}$* to be

- $-\sum_{\emptyset \neq T \subseteq A \cup U} (-1)^{|A \cap T|} \hat{p}(S, T)$ if $\mathcal{E} = \mathcal{E}(A, S, U)$ for some A, U.
- 1 if $\mathcal{E}$ is the equation $\hat{p}(V) = 1$.

Lemma 5.8. *If $\mathcal{G}^*(G, k)$ does not contain the equation $\mathcal{E}(\emptyset, V, \emptyset)$ then it has a solution.*

Proof. Suppose that $\mathcal{G}^*(G, k)$ does not contain $\mathcal{E}(\emptyset, V, \emptyset)$, and we construct an explicit solution $\hat{p}$ for $\mathcal{G}^*(G, k)$. Set $\hat{p}(V) = 1$. Then, suppose that for each $1 \leq t \leq k$ we have defined $\hat{p}(S)$ for all $|S| \geq m - t + 1$, and we show how to define the value of $\hat{p}(S)$ for sets of size $|S| = m - t$. Let $S \subseteq V$ be any set with $|S| = m - t$. If there is no S-equation in $\mathcal{G}^*(G, k)$ then set $\hat{p}(S) = 0$; otherwise, choose an S-equation $\mathcal{E}(A, S, U)$ arbitrarily in $\mathcal{G}^*(G, k)$ and set $\hat{p}(S)$ to the value induced by $\mathcal{E}(A, S, U)$:

$$\hat{p}(S) = -\sum_{\emptyset \neq T \subseteq A \cup U} (-1)^{|A \cap T|} \hat{p}(S, T).$$

The lemma follows from the next claim.

Claim. Let $\hat{p}$ be the function constructed above, and let $S \subseteq V$ with $m - k \leq |S| \leq m$ be any set for which there is an S-equation in $\mathcal{G}^*(G, k)$. For any two S-equations $\mathcal{E}_1, \mathcal{E}_2$, in $\mathcal{G}^*(G, k)$ the value induced by both equations is the same.

Proof of Claim. We prove the claim by downward induction on $|S|$. First, suppose that $S = V$. All V-equations are of the form $\hat{p}(V) = 1$ or $\mathcal{E}(A, V, U)$ for some A, U. In fact, we can take U to be $\emptyset$, since for any equation $\mathcal{E}(A, V, U)$ we must have $\emptyset = U \cap (V \cup A) = U \cap V$. It follows that all V-equations

other than $\hat{p}(V) = 1$ are of the form $\mathcal{E}(A, V, \emptyset)$ for some $A \subseteq V$. If $A = \emptyset$ then the equation $\mathcal{E}(\emptyset, V, \emptyset)$ is not in $\mathcal{G}^*(G, k)$ by assumption. Otherwise, suppose $A \neq \emptyset$. Then the value induced by $\mathcal{E}(A, V, \emptyset)$ is

$$\hat{p}(V) = -\sum_{\emptyset \neq T \subseteq A} (-1)^{|T \cap A|} \hat{p}(V, A) = \sum_{\emptyset \neq T \subseteq A} (-1)^{|T|+1} \hat{p}(V) = \hat{p}(V),$$

which is trivially satisfied.

By way of induction, let $1 \leq t \leq k$, and suppose that the claim is true for all sets S with $|S| \geq m - t + 1$. We prove the claim when $|S| = m - t$. Observe that the inductive hypothesis implies that $\hat{p}$ satisfies all S-equations in $\mathcal{G}^*(G, k)$ when $|S| \geq m - t + 1$. Let $\mathcal{E}(A, S, X)$ and $\mathcal{E}(B, S, Y)$ be two distinct S-equations. Then the value induced by $\mathcal{E}(A, S, X)$ and $\mathcal{E}(B, S, Y)$ is the same iff

$$\sum_{\emptyset \neq T \subseteq A \cup X} (-1)^{|A \cap T|} \hat{p}(S, T) = \sum_{\emptyset \neq U \subseteq B \cup Y} (-1)^{|B \cap U|} \hat{p}(S, U). \tag{8}$$

Let L denote the LHS and let R denote the RHS of Equation 8. By Proposition 5.6, since the graph G is acyclic we cannot have both $X \cap B \neq \emptyset$ and $Y \cap A \neq \emptyset$. Without loss of generality, assume $X = \{x\}$ and $Y = \{y\}$ (the proof will hold symmetrically if either X or Y is empty). To reduce clutter we write $A_x = A \cup \{x\}$ and $B_y = B \cup \{y\}$. We have two cases.

Case 1. $x \notin B$ and $y \notin A$.

First suppose that $x \neq y$. Since $X \cap B = \emptyset$ and $X \neq Y$, for every $T \subseteq A \cup X$ the system $\mathcal{G}^*(G, k)$ contains the equation $\mathcal{E}(B, S \cup T, Y)$ by the weakening rule; similarly, since $Y \cap A = \emptyset$, for every $U \subseteq B \cup Y$ the system contains the equation $\mathcal{E}(A, S \cup U, X)$. We therefore have

$$\begin{aligned} L = \sum_{\emptyset \neq T \subseteq A_x} (-1)^{|T \cap A|} \hat{p}(S, T) &= \sum_{\emptyset \neq T \subseteq A_x} \sum_{\emptyset \neq U \subseteq B_y} (-1)^{|T \cap A| + |U \cap B| + 1} \hat{p}(S, T, U) \\ &= \sum_{\emptyset \neq U \subseteq B_y} \sum_{\emptyset \neq T \subseteq A_x} (-1)^{|T \cap A| + |U \cap B| + 1} \hat{p}(S, U, T) \\ &= \sum_{\emptyset \neq U \subseteq B_y} (-1)^{|U \cap B|} \hat{p}(S, U) = R, \end{aligned}$$

where we have expanded each term $\hat{p}(S, T)$ using the value induced by the equation $\mathcal{E}(B, S \cup T, Y)$, and then contracted using the equation $\mathcal{E}(A, S \cup U, X)$.

Now suppose $X = Y = \{x\}$. The proof is essentially the same as before, except now we partition the sum into those terms containing x and those terms which do not. Consider

$$L = \hat{p}(S, x) + \sum_{\emptyset \neq T \subseteq A} (-1)^{|T \cap A|} \hat{p}(S, T, x) + (-1)^{|T \cap A|} \hat{p}(S, T).$$

For each $T \subseteq A$ we have $T \cap X = \emptyset$ and so the system $\mathcal{G}^*(G, k)$ contains the equation $\mathcal{E}(B, S \cup T, Y) = \mathcal{E}(B, S \cup T, \{x\})$. Expanding each term with the value induced by the corresponding equation we get

$$\begin{aligned} L &= \hat{p}(S, x) + \sum_{\emptyset \neq T \subseteq A} (-1)^{|T \cap A|} \hat{p}(S, T, x) + (-1)^{|T \cap A|} \hat{p}(S, T) \\ &= \hat{p}(S, x) + \sum_{\emptyset \neq T \subseteq A} \left[(-1)^{|T \cap A|} \hat{p}(S, T, x) + \sum_{\emptyset \neq U \subseteq B \cup \{x\}} (-1)^{|T \cap A| + |U \cap B| + 1} \hat{p}(S, T, U) \right] \\ &= \hat{p}(S, x) + \sum_{\emptyset \neq T \subseteq A} \sum_{\emptyset \neq U \subseteq B} \sum_{W \subseteq \{x\}} (-1)^{|T \cap A| + |U \cap B| + 1} \hat{p}(S, T, U, W), \end{aligned}$$

where the third equation follows since if $U = \{x\}$ in the last sum the term

$$(-1)^{|T\cap A|+|U\cap B|+1}\hat{p}(S,T,x) = (-1)^{|T\cap A|+1}\hat{p}(S,T,x)$$

cancels with the term $(-1)^{|T\cap A|}\hat{p}(S,T,x)$ in the first sum. Arguing analogously for R we get

$$\begin{aligned}
R &= \hat{p}(S,x) + \sum_{\emptyset\neq U\subseteq B} (-1)^{|U\cap B|}\hat{p}(S,U,x) + (-1)^{|U\cap B|}\hat{p}(S,U) \\
&= \hat{p}(S,x) + \sum_{\emptyset\neq U\subseteq B} \left[(-1)^{|U\cap B|}\hat{p}(S,U,x) + \sum_{\emptyset\neq T\subseteq A\cup\{x\}} (-1)^{|U\cap B|+|T\cap A|+1}\hat{p}(S,U,T) \right] \\
&= \hat{p}(S,x) + \sum_{\emptyset\neq U\subseteq B}\sum_{\emptyset\neq T\subseteq A}\sum_{W\subseteq\{x\}} (-1)^{|T\cap A|+|U\cap B|+1}\hat{p}(S,U,T,W),
\end{aligned}$$

and thus $L = R$, as desired.

Case 2. Either $x \in B$ or $y \in A$.

By Proposition 5.6 we may assume that it is not the case that both $x \in A$ and $y \in B$. So, assume without loss of generality that $x \in B$ and $y \notin A$, and recall that

$$\begin{aligned}
L &= \sum_{\emptyset\neq T\subseteq A_x} (-1)^{|T\cap A|}\hat{p}(S,T) \\
R &= \sum_{\emptyset\neq U\subseteq B_y} (-1)^{|U\cap B|}\hat{p}(S,U).
\end{aligned}$$

We show that $L = R$ if $\mathcal{E}(A \cup (B \setminus \{x\}), S \cup \{x\}, y)$ is satisfied, which is true by the inductive hypothesis. Overall the proof is similar to the previous case: we break the sums up according to x and apply weakening. Let us begin by considering L:

$$L = \hat{p}(S,x) + \sum_{\emptyset\neq T\subseteq A} (-1)^{|T\cap A|}\hat{p}(S,T,x) + (-1)^{|T\cap A|}\hat{p}(S,T).$$

As before, if $T \subseteq A$ then $T \cap \{y\} = \emptyset$ by assumption, and so by the weakening rule the system $\mathcal{G}^*(G,k)$ contains the equation $\mathcal{E}(B, S \cup T, \{y\})$. As usual substitute in the value induced by these equations:

$$\begin{aligned}
L &= \hat{p}(S,x) + \sum_{\emptyset\neq T\subseteq A} (-1)^{|T\cap A|}\hat{p}(S,T,x) + (-1)^{|T\cap A|}\hat{p}(S,T) \\
&= \hat{p}(S,x) + \sum_{\emptyset\neq T\subseteq A} \left[(-1)^{|T\cap A|}\hat{p}(S,T,x) + \sum_{\emptyset\neq U\subseteq B_y} (-1)^{|T\cap A|+|U\cap B|+1}\hat{p}(S,T,U) \right].
\end{aligned}$$

Partition the last sum into those terms which contain x and those terms which do not:

$$\begin{aligned}
&\sum_{\emptyset\neq U\subseteq B_y} (-1)^{|T\cap A|+|U\cap B|+1}\hat{p}(S,T,U) = \\
&\qquad (-1)^{|T\cap A|}\hat{p}(S,T,x) + \sum_{\emptyset\neq U\subseteq B_y\setminus\{x\}} (-1)^{|T\cap A|+|U\cap B|+1}(\hat{p}(S,T,U) - \hat{p}(S,T,U,x)).
\end{aligned}$$

Collecting terms, we get

$$L = \hat{p}(S,x) + \sum_{\emptyset \neq T \subseteq A} \left[2(-1)^{|T \cap A|} \hat{p}(S,T,x) + \sum_{\emptyset \neq U \subseteq B_y \setminus \{x\}} (-1)^{|T \cap A| + |U \cap B| + 1} (\hat{p}(S,T,U) - \hat{p}(S,T,U,x)) \right]. \tag{9}$$

Now we attack R. Again, begin by partitioning the terms of R depending on x

$$R = -\hat{p}(S,x) + \sum_{\emptyset \neq U \subseteq B_y \setminus \{x\}} (-1)^{|U \cap B| + 1} \hat{p}(S,U,x) + (-1)^{|U \cap B|} \hat{p}(S,U).$$

Expanding the terms $\hat{p}(S,U)$ using the value induced by $\mathcal{E}(A, S \cup U, \{x\})$ we get

$$\begin{aligned} R &= -\hat{p}(S,x) + \sum_{\emptyset \neq U \subseteq B_y \setminus \{x\}} (-1)^{|U \cap B| + 1} \hat{p}(S,U,x) + (-1)^{|U \cap B|} \hat{p}(S,U) \\ &= -\hat{p}(S,x) + \sum_{\emptyset \neq U \subseteq B_y \setminus \{x\}} \left[(-1)^{|U \cap B| + 1} \hat{p}(S,U,x) + \sum_{\emptyset \neq T \subseteq A_x} (-1)^{|U \cap B| + |T \cap A| + 1} \hat{p}(S,U,T) \right]. \end{aligned}$$

Once again partition the last sum with respect to x

$$\begin{aligned} &\sum_{\emptyset \neq T \subseteq A_x} (-1)^{|T \cap A| + |U \cap B| + 1} \hat{p}(S,U,T) = \\ &\qquad (-1)^{|U \cap B| + 1} \hat{p}(S,U,x) + \sum_{\emptyset \neq T \subseteq A} (-1)^{|U \cap B| + |T \cap A| + 1} (\hat{p}(S,T,U) + \hat{p}(S,T,U,x)). \end{aligned}$$

Collecting terms we get

$$R = -\hat{p}(S,x) + \sum_{\emptyset \neq U \subseteq B_y \setminus \{x\}} \left[2(-1)^{|U \cap B| + 1} \hat{p}(S,U,x) + \sum_{\emptyset \neq T \subseteq A} (-1)^{|T \cap A| + |U \cap B| + 1} (\hat{p}(S,T,U) + \hat{p}(S,T,U,x)) \right]. \tag{10}$$

Using Equations 9 and 10 it follows that proving $L = R$ is equivalent to showing

$$\begin{aligned} &\hat{p}(S,x) + \sum_{\emptyset \neq T \subseteq A} \left[2(-1)^{|T \cap A|} \hat{p}(S,T,x) + \sum_{\emptyset \neq U \subseteq B_y \setminus \{x\}} (-1)^{|T \cap A| + |U \cap B| + 1} (\hat{p}(S,T,U) - \hat{p}(S,T,U,x)) \right] \\ &= -\hat{p}(S,x) + \sum_{\emptyset \neq U \subseteq B_y \setminus \{x\}} \left[2(-1)^{|U \cap B| + 1} \hat{p}(S,U,x) + \sum_{\emptyset \neq T \subseteq A} (-1)^{|T \cap A| + |U \cap B| + 1} (\hat{p}(S,T,U) + \hat{p}(S,T,U,x)) \right]. \end{aligned}$$

We claim that this equation is equivalent to $\mathcal{E}(A \cup B \setminus \{x\}, S \cup \{x\}, \{y\})$. Rearranging and cancelling like terms we get

$$\begin{aligned} 0 = 2\hat{p}(S,x) &+ \sum_{\emptyset \neq T \subseteq A} 2(-1)^{|T \cap A|} \hat{p}(S,T,x) + \sum_{\emptyset \neq U \subseteq B_y \setminus \{x\}} 2(-1)^{|U \cap B|} \hat{p}(S,U,x) \\ &+ \sum_{\emptyset \neq U \subseteq B_y \setminus \{x\}} \sum_{\emptyset \neq T \subseteq A} 2(-1)^{|T \cap A| + |U \cap B| + 2} \hat{p}(S,T,U,x), \end{aligned}$$

which is the same as

$$0 = 2 \sum_{T \subseteq A \cup (B_y \setminus \{x\})} (-1)^{|T \cap (A \cup B \setminus \{x\})|} \hat{p}(S, T, x).$$

Dividing by 2 yields $\mathcal{E}(A \cup B \setminus \{x\}, S \cup \{x\}, \{y\})$.

□

Finally, we use Lemma 5.8 to prove Theorem 5.2. The following theorem by Chan [12] about the depth of resolution refutations of Peb_G will be helpful.

Theorem 5.9. *[Theorem 1 and Theorem 3 in [12]] Any resolution refutation of* Peb_G *has height at least* $\mathsf{rpeb}(G)$.

Proof of Theorem 5.2. Let k be a positive integer and let G be a good DAG. We show that if $\mathcal{G}^*(G, k)$ contains $\mathcal{E}(\emptyset, V, \emptyset)$, then there is a resolution refutation of Peb_G with height k. Recall that the formula Peb_G has a single variable z_v for each vertex $v \in G$. We associate with each equation $\mathcal{E}(A, S, U)$ the clause

$$\bigvee_{u \in U} x_u \vee \bigvee_{a \in A} \neg x_a.$$

With this identification, the source axioms $\mathcal{E}(\emptyset, S, \{u\})$ become clauses z_u, the target axioms $\mathcal{E}(\{t\}, S, \emptyset)$ become clauses $\neg z_t$, and the edge axioms $\mathcal{E}(W, S, \{w\})$ become clauses $\bigvee_{u \in W} \neg z_u \vee z_w$. (These are precisely the clauses appearing in Peb_G.) The cut rule for $\mathcal{E}$-equations

$$\mathcal{E}(A, S, \{x\}), \mathcal{E}(B, S, U) \vdash \mathcal{E}(A \cup B \setminus \{x\}, S \cup \{x\}, U)$$

just becomes the cut rule in resolution

$$\left(\bigvee_{a \in A} \neg z_a \vee z_x \right), \left(\bigvee_{b \in B} \neg z_b \vee \bigvee_{u \in U} z_u \right) \vdash \bigvee_{a \in A \cup (B \setminus \{x\})} \neg z_a \vee \bigvee_{u \in U} z_u.$$

(The weakening rule is idempotent — it does not change the corresponding resolution clause). Since the equation $\mathcal{E}(\emptyset, V, \emptyset)$ maps to the empty clause, it follows that the resulting resolution proof is a refutation of Peb_G.

It remains to argue that the refutation has height k. The key observation here is that whenever we apply the cut rule on two $\mathcal{E}$ equations $\mathcal{E}(A, S, \{x\}), \mathcal{E}(B, S, U)$ with $x \in B$, the new equation contains x in its "middle" set. This implies that we can never use this equation on a cut on x again since the three sets must be mutually disjoint. So, if for every axiom we have $|S| \geq m - k$, it follows that the total number of cuts on any path from each axiom to the root is at most k. Therefore the corresponding resolution proof has the same height, since the weakening rule in the $\mathcal{E}$-equations does no change resolution clauses.

By applying Corollary 5.5 we have that if $\mathcal{G}(G, k)$ has a solution then $\mathsf{Search}(\mathsf{Peb}_G)$ has algebraic gap complexity at least $k + 1$. Lemma 5.8 shows that $\mathcal{G}^*(G, k) \supseteq \mathcal{G}(G, k)$ has a solution unless it contains the equation $\mathcal{E}(\emptyset, V, \emptyset)$, and finally we have just shown that if $\mathcal{G}^*(G, k)$ contains $\mathcal{E}(\emptyset, V, \emptyset)$ then there is a height k resolution refutation of Peb_G. Since every resolution refutation of Peb_G has height at least $\mathsf{rpeb}(G)$ by Theorem 5.9, it follows that $\mathcal{G}^*(G, \mathsf{rpeb}(G) - 1)$ does not contain the equation $\mathcal{E}(\emptyset, V, \emptyset)$. Thus $\mathcal{G}^*(G, \mathsf{rpeb}(G) - 1)$ (and also $\mathcal{G}(G, \mathsf{rpeb}(G) - 1)$) has a solution, and so $\mathsf{Search}(\mathsf{Peb}_G)$ has algebraic gap complexity at least $\mathsf{rpeb}(G)$. □

Acknowledgements

The authors thank Mika Göös and Aaron Potechin for helpful conversations.

References

[1] Scott Aaronson. The power of the digi-comp ii: My first conscious paperlet. `http://www.scottaaronson.com/blog/?p=1902`. Accessed: 2016-03-17.

[2] M. Ajtai, J. Komlos, and E. Szemeredi. An o(n log n) sorting network. *Proceedings of STOC 1983*, pages 1–9, 1983.

[3] László Babai, Anna Gál, János Kollár, Lajos Rónyai, Tibor Szabó, and Avi Wigderson. Extremal bipartite graphs and superpolynomial lower bounds for monotone span programs. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 603–611, 1996.

[4] László Babai, Anna Gál, and Avi Wigderson. Superpolynomial lower bounds for monotone span programs. *Combinatorica*, 19(3):301–319, 1999.

[5] Paul Beame, Russell Impagliazzo, Jan Krajícek, Toniann Pitassi, and Pavel Pudlák. Lower bound on hilbert's nullstellensatz and propositional proofs. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 794–806, 1994.

[6] Amos Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis, Technion, 1996.

[7] Amos Beimel. *Coding and Cryptology: Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, chapter Secret-Sharing Schemes: A Survey, pages 11–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[8] Amos Beimel, Anna Gál, and Mike Paterson. Lower bounds for monotone span programs. *Computational Complexity*, 6(1):29–45, 1997.

[9] Amos Beimel and Enav Weinreb. Separating the power of monotone span programs over different fields. *SIAM J. Comput.*, 34(5):1196–1215, 2005.

[10] Allan Borodin. On relating time and space to size and depth. *SIAM J. Comput.*, 6(4):733–744, 1977.

[11] Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *J. Comput. Syst. Sci.*, 62(2):267–289, 2001.

[12] Siu Man Chan. Just a pebble game. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 133–143, 2013.

[13] Siu Man Chan and Aaron Potechin. Tight bounds for monotone switching networks via fourier analysis. *Theory of Computing*, 10:389–419, 2014.

[14] Arkadev Chattopadhyay and Anil Ada. Multiparty communication complexity of disjointness. *Electronic Colloquium on Computational Complexity (ECCC)*, 15(002), 2008.

[15] Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 174–183, 1996.

[16] Stephen A. Cook. An observation on time-storage trade off. *J. Comput. Syst. Sci.*, 9(3):308–316, 1974.

[17] Stephen A. Cook, Yuval Filmus, and Dai Tri Man Le. The complexity of the comparator circuit value problem. *TOCT*, 6(4):15:1–15:44, 2014.

[18] László Csirmaz. The dealer's random bits in perfect secret sharing schemes. *Studia Sci. Math. Hungary*, 32(3-4):429–437, 1996.

[19] Patrick W. Dymond and Martin Tompa. Speedups of deterministic machines by synchronous parallel machines. *J. Comput. Syst. Sci.*, 30(2):149–161, 1985.

[20] Anna Gál. A characterization of span program size and improved lower bounds for monotone span programs. *Computational Complexity*, 10(4):277–296, 2001.

[21] Anna Gál and Pavel Pudlák. A note on monotone complexity and the rank of matrices. *Inf. Process. Lett.*, 87(6):321–326, 2003.

[22] John R Gilbert and Robert E Tarjan. Variations of a pebble game on graphs. Technical report, Stanford University, 1978.

[23] Michael T. Goodrich. Zig-zag sort: a simple deterministic data-oblivious sorting algorithm running in o(n log n) time. *Proceedings of STOC 2014*, pages 684–693, 2014.

[24] Mika Göös. Lower bounds for clique vs. independent set. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1066–1076, 2015.

[25] Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 257–266, 2015.

[26] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 847–856, 2014.

[27] Michelangelo Grigni and Michael Sipser. Monotone separation of logspace from NC. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991*, pages 294–298, 1991.

[28] Trinh Huynh and Jakob Nordström. On the virtue of succinct proofs: amplifying communication complexity hardness to time-space trade-offs in proof complexity. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 233–248, 2012.

[29] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. Discrete Math.*, 3(2):255–265, 1990.

[30] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111, 1993.

[31] D. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 1997.

[32] James R. Lee, Prasad Raghavendra, and David Steurer. Lower bounds on the size of semidefinite programming relaxations. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 567–576, 2015.

[33] Troy Lee and Adi Shraibman. Disjointness is hard in the multiparty number-on-the-forehead model. *Computational Complexity*, 18(2):309–336, 2009.

[34] Cristopher Moore and Jonathan Machta. Internal diffusion-limited aggregation: parallel algorithms and complexity. *Journal of Statistical Physics*, 99(3-4):661–690, 1999.

[35] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[36] Aaron Potechin. Bounds on monotone switching networks for directed connectivity. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 553–562, 2010.

[37] Pavel Pudlák and Jirí Sgall. Algebraic models of computation and interpolation for algebraic proof systems. In *Proof Complexity and Feasible Arithmetics, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, April 21-24, 1996*, pages 279–296, 1996.

[38] Vijaya Ramachandran and Li-Chung Wang. Parallel algorithm and complexity results for telephone link simulation. In *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing, SPDP 1991, 2-5 December 1991, Dallas, Texas, USA*, pages 378–385, 1991.

[39] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999.

[40] A. A. Razborov. On submodular complexity measures. In *Proceedings of the London Mathematical Society Symposium on Boolean Function Complexity*, pages 76–83, New York, NY, USA, 1992. Cambridge University Press.

[41] Alexander A. Razborov. Lower bounds for the monotone complexity of some boolean functions. *Soviet Math. Dokl.*, 31:354–357, 1985.

[42] Alexander A. Razborov. Applications of matrix methods to the theory of lower bounds in computational complexity. *Combinatorica*, 10(1):81–93, 1990.

[43] Alexander A. Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324, 1998.

[44] Ben W. Reichardt. Span programs are equivalent to quantum query algorithms. *SIAM J. Comput.*, 43(3):1206–1219, 2014.

[45] Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.

[46] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[47] Alexander A. Sherstov. Communication lower bounds using dual polynomials. *Bulletin of the EATCS*, 95:59–93, 2008.

[48] Alexander A. Sherstov. The pattern matrix method. *SIAM J. Comput.*, 40(6):1969–2000, 2011.

[49] A. Subramanian. *The computational complexity of the circuit value and network stability problems*. PhD thesis, Stanford University, 1990.

[50] Avi Wigderson. $\oplus$ l/ poly = nl / poly. http://www.math.ias.edu/ avi/PUBLICATIONS/MYPAPERS/W94/proc.pdf. Accessed: 2016-03-18.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de