Blackbox Identity Testing for Simple Depth 3 Circuits

A thesis submitted in fulfilment of the requirements

for the degree of Master of Technology

by

Rishabh Vaid

13111003

under the guidance of

Dr. Manindra Agrawal and Dr. Nitin Saxena

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur

December 2015

Abstract

The Polynomial Identity Testing problem (PIT) requires one to determine whether a given polynomial is identically equal to the zero polynomial. In the blackbox version we are promised that the polynomial belongs to a certain class, and are only provided input/output access to it. There is a strong connection between efficent blackbox PIT algorithms for a class of polynomials, and lower bounds against that class of polynomials. Coupled with the recently proved 'Chasm at Depth 3', this can used to show that polynomial time blackbox PIT for depth-3 circuits is enough for explicit polynomials with subexponential lower bounds. This has motivated inquiry into blackbox PIT algorithms for restricted classes of polynomials.

In this thesis, we provide alternate quasi-polynomial time algorithms for two well studied classes – the Diagonal depth-3 model, and the Basic Set Multilinear model. In the case of Diagonal depth-3, we design a map that reduces the number of variables, while preserving the non-zeroness of such a circuit. The running time of our algorithm is $n^{O(\log n)}$.

We approach the Basic Set Multilinear model from the tensorial point of view. We show that the correctness of our PIT algorithm is equivalent to tensor rank lower bounds for a class of tensors we call *simplicial tensors*. These tensors are a simple generalization of triangular matrices, and rank lower bounds for them may be of independent interest – We direct most of our efforts here to proving these bounds. The running time of our algorithm is $n^{O(\log n \log\log n)}$.

Acknowledgements

I will always be grateful to my guides Professor Nitin Saxena and Professor Manindra Agrawal for giving me a chance to work with them.

Sitting in Professor Agrawal's lectures, I was amazed by his fearless attitude and raw speed when it came to mathematics – If he forgot a proof, he would simply reconstruct it from base principles, and he made it seem incredibly easy. As an advisor, he gave me absolute freedom to choose the direction of my work and despite his busy schedule, he was always available when I needed him.

Working with Professor Saxena has been delightful. I did not have to use the internet much while working on my thesis, because of his encyclopedic knowledge of the field. I will never cease to be amazed by the clarity and precision with which he approached mathematics. Trying to emulate this clarity provided me with insight into various meta-principles that helped guide my research. In addition to his technical competence, Professor Saxena is also an extremely understanding and patient boss. He tolerated both, my stubborn refusal to run simulations and my endless procrastination at various stages in this thesis. He afforded me total freedom in my research, while providing structure and guidance in the form of regular meetings every week. I spent several months running around in circles, but my enthusiasm never flagged, owing to his support and encouragement.

I would also like to thank all the professors that taught me, for an enriching and inspiring education. I thoroughly enjoyed every lecture of every course that I took in IITK. The enthusiastic, genial, and above all else, logical nature of our professors is something I will always be grateful for. Despite their stratospheric intelligence, I have never seen a professor condescend to a student

for asking too simple a doubt[1]. Conversely, I also found my professors to have a strong awareness of the limits of their knowledge – Academic arguments[2] were always settled based on who was right, without appeals to authority. In my experience, this places the IITK CS department head and shoulders above any other academic institution I have been a part of.

I would like to thank Professor Ameya Karkare for agreeing to teach ESC 101 for an extra semester, enabling Professor Saxena to teach Advanced Complexity in my last semester. Chapter 3 of this thesis would have been much shorter if not for the ideas I was exposed to in that course.

I would like to thank Scott Aaronson for introducing me to the magic of theoretical computer science. I would like to thank my parents for giving birth to me and for their unconditional love and support ever since. I would like to thank my academic siblings Arpita, Rohit, Amit, Shubham, Anurag and Sumanta for the conversations and SIGTACS talks. I would also like to thank my friend Pratik for the many hours of intellectual jousting and problem solving fun.

[1] In particularly bad situations, they might express concern at the student's lack of understanding, but that is different from condescension.

[2] I have never had a non-academic argument with any professor in IIT Kanpur.

Contents

Dedicated to
my parents

Chapter 1

Polynomial Identity Testing

Polynomial Identity Testing is one of the problems that sits at the heart of theoretical computer science. The problem seems innocent enough; To determine whether a given polynomial is identically equal to the zero polynomial. A little thought reveals that this is not surprising for two reasons – Polynomials are ubiquitous in mathematics and theoretical computer science; Primarily, this is because they are both, expressive [1] and tractable. Secondly, when studying polynomials, *identity* is pretty much the most basic question about a polynomial that one can ask; Essentially, whether a given polynomial does something or nothing. Taken together, these points explain the wide ranging implications of efficient PIT algorithms in areas of theory as diverse as graph theory [39], number theory [3], the PCP theorem [7] and circuit lower bounds.

Another reason that PIT has a special place in the hearts of computer scientists is that it beautifully captures the power of randomization – The Schwartz-Zippel lemma tells us that the most obvious randomized algorithm of evaluating at a bunch of random inputs works with high probability [35].

[1] By expressive, we mean that algorithmic questions in many diverse fields can be rephrased, either exactly or approximately, as questions about polynomials.

Moreover, this is a blackbox test; All it requires is a guarantee on the size of the polynomial [2]. Derandomizing blackbox PIT is one of the holy grails of our field – Not least of all because even a sub-exponential time algorithm implies circuit lower bounds, as demonstrated by [18, 1].

1.1 Current Status

Recently, depth reduction results starting with [5] and ending at [16] showed that polynomial time blackbox PIT algorithms for depth-3 arithmetic circuits would lead to quasi-polynomial PIT algorithms for general circuits (with low degree) which is already enough to prove significant lower bounds. These results showed that even very simple circuits capture a large amount of the complexity of general circuits and motivated the community to examine the low depth regime of arithmetic circuits in higher resolution. These ideas motivated a long line of results, all examining PIT in different regimes of general depth-3 circuits such as constant top fan-in depth-3 circuits [9, 21, 20, 19, 33, 34], set-multilinear circuits [28, 12, 4] and Read-once Oblivious Algebraic Branching Programs (ROABPs) [28, 14, 11, 2].

1.2 Contributions of this Thesis

For any class of ploynomials, it is easy to see that a deterministic blackbox identity test is equivalent to a *hitting set* – A set of evaluation points such that any non-zero polynomial in the class has a non-zero evaluation on at least one

[2] The size of a polynomial is typically given by it's degree and the number of variables. When talking about deterministic blackbox PIT, we also introduce other size parameters, depending on the subclass of polynomial we consider. All these size parameters are assumed to be polynomially related, since this is the regime where we can use Schwartz Zippel to prove the *existence* of deterministic polynomial time blackbox PIT algorithms.

of them. In this thesis, we give quasi-polynomial size hitting sets for two related classes of polynomials. The polynomials we consider are depth-3 circuits of the form: $P(\boldsymbol{x}) = \sum_{j=1}^{k} E_j$, where the E_j's are elementary polynomials of some kind, and $k = \mathsf{poly}(\mathrm{n})$. In Chapter 2, we consider the Diagonal depth-3 model first proposed by Saxena, wherein the E_j's are of the form of a linear function raised to a power, i.e. $E(\boldsymbol{x}) = (\sum_{i=1}^{n} a_i x_i)^d$. The current best algorithm for this model was given by [11] and requires $n^{O(\log \log n)}$ evaluations. In this thesis, we provide an alternate hitting set for this model, with the same size. While our proof is simpler, the result from [11] is much more general and applies to all polynomials with a low dimensional space of partial derivatives. Our strategy is to use a simple hash function that makes the number of variables logarithmic and then use Saxena's Duality Trick. This is a very useful lemma from [30] whereby each linear power is further simplified to a sum of products of univariates. In terms of the above notation, we rewrite $P(\boldsymbol{x})$ as a sum of $K = k(nd+d+1)$ elementary polynomials, with each one of the form $E_i(\boldsymbol{x}) = \Pi_{i=1}^{n}(a_{i0}+a_{i1}x_i+\cdots+a_{id}x_i^d)$. Such polynomials are referred to as Commutative read-once Oblivious Algebraic Branching Programs, or Commutative ROABP in short, and are widely studied. There are several elegant hitting sets for this model of size $k^{O(\log n)}$. Any of these can be used to test the log-variate ROABP we reduce the Diagonal depth-3 circuit to, giving a total size of $n^{O(\log \log n)}$. Commutative ROABPs also bring us neatly to Chapter 3, where we develop another approach to this model, by relying on it's connection with the theory of tensors.

For the uninitiated, tensors are basically higher dimensional analogs of matrices – An n-dmensional array of field elements. Clearly, given an n-variate polynomial, we can associate it with an n-dimensional tensor – Place the coefficient of the monomial $x_1^{i_i} x_2^{i_2} \cdots x_n^{i_n}$ in the $(i_1, i_2, \cdots, i_n)^{\text{th}}$ index of the tensor. Like matrices, tensors also come equipped with an analogous notion of rank,

which is essentially the minimal number of rank one tensors that must be summed to obtain the given tensor. We will see the details later, but for now we just state that the tensor associated with a polynomial of the form $E(\boldsymbol{x})$ (A product of univariates) has tensor rank one, and that hence Commutative ROABPs correspond to tensors with rank $\leq k$.

In Chapter 3, we study a simplification of Commutative ROABPs called the Basic Set Multilinear model where the elementary polynomials are now of the form $E(\boldsymbol{x}) = \Pi_{i=1}^{n}(1+a_i x_i)$, i.e. the individual degree of each univariate is one. In this case, we exploit the tensorial connection to develop a blackbox PIT. To do this, we define a class of tensors we call *simplicial* tensors – These are basically tensorial analogs of triangular matrices and we can prove that their rank exceeds k. This implies that a simplicial tensor can never be obtained as the image of a Basic Set Multilinear polynomial. We then devise a set of evaluations such that all non-trivial polynomials evaluating to zero on this set correspond to simplicial tensors, which gives us an $n^{O(\log n \log\log n)}$ time blackbox PIT algorithm. At this point, it is worth stressing that [11] have already devised an $n^{O(\log\log n)}$ time algorithm for this model. Why then, should someone care about our results? In a nutshell, it seems that the techniques employed by [11] have reached a saturation point, and it is unlikely that any straightforward extension would allow us to jump from quasi-polynomial to polynomial time. For the tensor rank approach, we seem to run into some interesting problems that we hope can be resolved. In Chapter 4, we outline the next steps that one could take in order to take the program forward.

Chapter 2

Diagonal Depth 3

2.1 Introduction

The Diagonal depth-3 model is one of the simplest models for which we do not yet have polynomial time blackbox PIT. In this model, the polynomial is expressed as the sum of powers of linear functions. The beauty of this model is that it sits at a remarkably fruitful inflection point – It's analytical tractability and aesthetic appeal makes it a good workhorse for new ideas and the fact that it is not easily solvable means that those ideas generally have a wide range of applicability. We elucidate this point of view in the next section, by providing a brief account of the previous work on the Diagonal model. First though, we introduce the notation we use in the rest of the thesis.

2.2 Notation and Preliminaries

Boldface lower case letters denote tuples, such as $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ which is an an n-tuple of variables. For a degree tuple $\boldsymbol{d} = (d_1, d_2, \ldots, d_n) \in$

$[0, 1, 2, \cdots d]^n$, we denote by $\boldsymbol{x}^{\boldsymbol{d}}$ the monomial $\prod_{i=1}^{n} x_i^{d_i}$, where d is the degree of the polynomial. A degree tuple $\boldsymbol{d}$ can be, and is, viewed from a host of different perspectives. It can be thought of as a vector, and in the special case where the polynomial is multilinear (Each degree is either 0/1), it can also be thought of as a subset of n elements, or a bit string. We abuse notation and associate all of these instances with the same notation $\boldsymbol{d}$. Naturally, the associated notations from these different contexts carry over as well, for example we use the 1-norm notation, which in this context translates to $|\boldsymbol{d}|_1 = d_1 + d_2 \cdots + d_n$.

In these thesis, we deal with fields $\mathbb{F}$ of characteristic zero. Let $A(\boldsymbol{x})$ be a polynomial in $\mathbb{F}[\boldsymbol{x}]$. By $\text{coeff}_A(\boldsymbol{x}^d) \in \mathbb{F}$ we denote the coefficient of the monomial $\boldsymbol{x}^{\boldsymbol{d}}$ in $A(\boldsymbol{x})$.

2.3 Previous Work

The Diagonal depth-3 model was first introduced by Saxena in [30]. This paper showed how to perform polynomial time whitebox PIT, by introducing the now famous Duality Trick. This is a simple lemma, stated as follows:

Lemma 2.1. *[30] Let $t = nd + d + 1$. Then there can be found, in poly(nd) time, univariate polynomials q_{ij} of degree at most d, such that:*

$$(x_1 + x_2 + \cdots + x_n)^d = \sum_{i=1}^{t} q_{i1}(x_1) q_{i2}(x_2) \cdots q_{in}(x_n)$$

As we can see, this lemma helps us to decouple the polynomial into a product over disjoint univariates. Applying this transform to a diagonal circuit gives us a commutative ROABP, whitebox PIT for which was given by [28]. The

Duality Trick was also one of the key ingredients in the recent beautiful depth reduction work of [16].

In terms of blackbox PIT as well, there has been significant progress on this model. The earliest results for this model were by [4] and [12], which both gave $n^{O(\log n)}$ time algorithms, albeit via completely different techniques. The PIT algorithm given by [4] is one of the prototypical examples of the notion called *rank concentration*. In order to comfortably discuss their approach we must introduce the concept of Hadamard algebras. A friendly warning is in order – At various points in this thesis we will utilise standard terminology that is sometimes more complicated than it needs to be. At such points, we advise the reader to read on without getting intimidated by the names, as the underlying concepts are pretty simple.

Definition 2.2. The Hadamard algebra $H_k(\mathbb{F})$ is a commutative ring given by $(\mathbb{F}^k, +, \star)$, where $(\mathbb{F}^k, +)$ is simply the regular k-dimensional vector space, and $\star$ is the Hadamard product, which is a binary operation that multiplies the vectors co-ordinatewise.

Given this definition, it it easy to see that diagonal circuits can be rephrased as the dot product of a depth-2 circuit over the Hadmard algebra with a vector $\boldsymbol{c}$, i.e. $P(\boldsymbol{x}) = \boldsymbol{c}^{\intercal}(1+\alpha_1 x_1+\cdots+\alpha_n x_n)^d$, where $\alpha_i \in H_k(\mathbb{F})$. Rank concentration is the idea that the coefficient vectors of the low-support monomials span $\mathbb{F}^k$, and that hence, the dot product of at least one of them with $\boldsymbol{c}$ must be non-zero. While the result did not eventually make it to the paper, [4] showed that Diagonal depth-3 circuits are l-concentrated, where $l = O(\log n)$. The proof is easy to follow and quite beautiful.

Theorem 2.3. *Consider a non zero diagonal circuit $P(\boldsymbol{x})$ over the Hadamard algebra $H_k(\mathbb{F})$. Then $P(\boldsymbol{x})$ is rank concentrated among the log-support monomials, i.e. the co-efficients of the monomials with support $\leq \log k$ span the vector space $\mathbb{F}^k$.*

Proof. Expand out the depth-2 circuit and arrange the terms of the polynomial in deg-lex order, i.e. order them first by lower to higher total degree, and within each degree, arrange them according to a lexicographic order, i.e.

$$P(\boldsymbol{x}) = \underbrace{1}_{\text{degree 0}} + \underbrace{\alpha_1 x_1 + \cdots + \alpha_n x_n}_{\text{degree 1}} + \underbrace{\alpha_1^2 x_1^2 + \alpha_1\alpha_2 x_1 x_2 + \cdots + \alpha_n^2 x_n^2}_{\text{degree 2}} + \cdots \tag{2.1}$$

Relabel the coefficients under the deg-lex ordering as $c_1, c_2, \ldots c_N$, where $N = \sum_{i=0}^{d} \binom{i+n}{n}$. For any i, define S_i to be the vector space spanned by the first i coefficients. Then the least basis is a set of coefficients B_L defined as follows: $B_L = \{\ c_i :\ \ c_i \notin S_{i-1}, i \in [1, 2, \cdots, N]\}$. Observe that if the coefficient of a monomial is in B_L, the coefficients of all of it's factors must be in B_L as well. In order to prove this, assume that $\text{coeff}_P(\boldsymbol{x}^{\boldsymbol{a}}) \in B_L$ and that $\text{coeff}_P(\boldsymbol{x}^{\boldsymbol{b}}) \notin B_L$ for some factor $\boldsymbol{x}^{\boldsymbol{b}}$ of $\boldsymbol{x}^{\boldsymbol{a}}$. In this case, if $\text{coeff}_P(\boldsymbol{x}^{\boldsymbol{b}})$ can be expressed as a linear combination of lesser coefficients (Under the deg-lex ordering), we can multiply both sides of the equation by $\text{coeff}_P(\boldsymbol{x}^{\boldsymbol{a}-\boldsymbol{b}})$, and express $\text{coeff}_P(\boldsymbol{x}^{\boldsymbol{a}})$ as a linear combination of lesser coefficients as well, which is a contradiction. In order to show that this leads to log-concentration, note that an l-support monomial has at least 2^l factors. The number of elements in B_L is at most k, which is the dimension of the vector space. Together, these facts imply that the largest support monomial whose coefficient can be in B_L has support $\leq \log k$, which shows that $P(\boldsymbol{x})$ is $\log k$ rank concentrated. □

This gives us a simple $n^{O(\log n)}$ time algorithm – We can simply interpolate the polynomial and find the coefficients of all monomials that are a product $\leq l$ distinct variables – At least one of them is non-zero.

[12] also show the existence of an l-support monomial with non-zero coefficient, but to do this, they leverage lower bounds against the Diagonal depth-3 model. It was first observed by Kayal, and reported in [31], that the seminal partial derivative techniqe of [26] can be used to obtain strong lower bounds for diagonal circuits – In particular any diagonal circuit that computes the monomial $x_1 x_2 \cdots x_n$ must have exponential top fan-in. [12] scale this lower bound down to show that a diagonal ciruit with polynomial top fan-in must have low support monomials, and then proceed in the same way as before. We do not give a formal statement of their proof, as it requires an introduction to the theory of monomial orderings, which we want to avoid for now (Though they make a guest appearance in Chapter 4).

These results were dramatically improved by [11], who gave an $n^{O(\log \log n)}$ time algorithm. In their algorithm, they exploit the fact that diagonal circuits are log-concentrated, and then use hashing techniques [36] to further reduce the question of identity to a commutative ROABP over $O(\log n)$ variables. After this they simply invoke one of the many existing identity tests for this model, and end up with the desired result.

This Work: In this thesis, we provide an alternate $n^{O(\log \log n)}$ time algorithm. At a high level, our algorithm works in much the same way as [11]. Our main technical contribution is in designing a simple map that reduces the diagonal circuit to a commutative ROABP over $O(\log n)$ variables.

2.4 l-Support Preserving Variable Reduction

In this section we introduce a simple map, which reduces the number of variables in a polynomial to l, while preserving the coefficients of all the l-support monomials. Note that this is a general result, and is not specific to diagonal circuits. For ease of exposition, we will first introduce a related map, and then convert it to our desired form by an application of the standard Kronecker trick [23].

Definition 2.4. Define the map $\Psi : \mathbb{F}[\boldsymbol{x}] \to \mathbb{F}[\boldsymbol{a}, \boldsymbol{b}, t]$, where $|\boldsymbol{a}| = |\boldsymbol{b}| = l$, as follows:

$$x_i \to \sum_{j=1}^{l} a_j^i b_j^{i^2} t$$

Theorem 2.5. *Under the map $\Psi : \mathbb{F}[\boldsymbol{x}] \to \mathbb{F}[\boldsymbol{a}, \boldsymbol{b}, t]$, each $\leq l$-support monomial in $P(\boldsymbol{x})$ is mapped to a unique monomial in $P(\Psi(\boldsymbol{x}))$.*

Proof. Let us fix an l-support monomial $\boldsymbol{x^r} = x_{i_1}^{r_1} \cdots x_{i_l}^{r_l}$. Let $\boldsymbol{i} = (i_1, i_2, \cdots, i_l)$. On applying the substitution Ψ, we have:

$$\Psi(\boldsymbol{x}^r) = \boldsymbol{a^p b^q} t^{|\boldsymbol{r}|_1} + \text{other monomials} \ldots$$

where $\boldsymbol{p} = r \star \boldsymbol{i}$, $\boldsymbol{q} = \boldsymbol{r} \star \boldsymbol{i} \star \boldsymbol{i}$ and $|\boldsymbol{r}|_1 = (r_1 + \cdots + r_l)$. For the sake of convenience, we have abused notation a bit and used the Hadamard product ($\star$) for the exponent vectors as well. It is easy to see how the first monomial occurs in the image $\Psi(\boldsymbol{x}^r)$ – On expanding the substitution, simply pick the first term $= a_1^{i_1} b_1^{i_1^2}$ from the first r_1 brackets, the second term from the next r_2 brackets and so on. We will now show that this monomial can occur only in the image $\Psi(\boldsymbol{x}^r)$. Let us say that in forming this monomial, we picked the first term from k_1 brackets. Further, assume that the exponents of a_1 in each of these brackets were $(e_1, e_2, \cdots, e_{k_1})$. Then, we have the relations:

$$\sum_{j=1}^{k_1} e_j = r_1 i_1$$
$$\sum_{j=1}^{k_1} e_j^2 = r_1 i_1^2$$
$$\sum_{j=1}^{k_1} e_j^2 \geq \frac{1}{k_1} \left(\sum_{j=1}^{k_1} e_j \right)^2$$

The third equation is the Cauchy-Schwartz inequality, which implies that $k_1 \geq r_1$, with equality occurring if and only if $e_1 = e_2 = \cdots = e_{k_1} = i_1$. In fact, we can prove relations of the form $k_j \geq r_j$, for all values of j. We now use the information in the degree counter t, which enforces the relation $\sum k_j = \sum r_j$. This means that each of the inequalities were equalities ($k_j = r_j$, for all j), which in turn fixes each of the individual exponents for $(a_1, a_2, \cdots, a_l)$. This uniquely gives us the degrees and the variables of the pre-image monomial. $\square$

Now that we have proved that this map preserves the l-support monomials of a polynomial, we can convert it to a more convenient form via the application of Kronecker trick. In the image polynomial $P(\Psi(\boldsymbol{x}))$, the individual degree of each variable is bounded above by $D = n^2 d + 1$. This means that for each j, we can merge a_j and b_j into a single variable z_j by performing the substitution $a_j \to z_j$ and $b_j \to z_j^D$. Such tricks are standard in the PIT literature, and we can use them to arrive at a map $\Phi : \mathbb{F}[\boldsymbol{x}] \to \mathbb{F}[\boldsymbol{z}], |\boldsymbol{z}| = l$, which preserves l-support monomials:

$$\Phi(x_i) = \sum_{j=1}^{l} (z_j)^{h(i)}$$

where $h(i) = i^2 D^2 + iD + 1$.

2.5 Diagonal Circuit Identity Test

We can now combine the above ideas to arrive at a simple $n^{O(\log \log n)}$ time PIT algorithm for Diagonal depth-3. Set $l = \log k$ and make the substitution $\boldsymbol{x} \to \Phi(\boldsymbol{x})$. On rearranging the terms a little, the polynomial becomes $P(\boldsymbol{z}) = c^{\intercal}(1+ g(z_1) + \cdots + g(z_l))^d$, where g is a univariate polynomial over $H_k(\mathbb{F})$. Saxena's Duality can now be applied to convert this to a commutative ROABP over l variables, whose degree and top fan-in are both bounded above by $\mathsf{poly}(\mathrm{n})$, following which we may directly apply the results of [12] to get an $n^{O(\log l)} = n^{O(\log \log n)}$ time algorithm, as promised.

2.6 Addendum

It is worth noting that our map Φ also gives us, as a byproduct, a way to reduce blackbox PIT for a general polynomial to blackbox PIT for a symmetric polynomial, i.e. a polynomial that is unchanged under permutation of it's variables. In order to see this, note that every non-zero polynomial in n variable is n-supported. Following our theorem, we can define a map that preserves the n-support monomials and transforms the polynomial to a symmetric one.

Chapter 3

Basic Set Multilinear

3.1 Introduction

The Basic Set Multilinear model is arguably the simplest model for which we currently do not have explicit hitting sets. In terms of the Hadamard algebra introduced in Chapter 3, we can write a Basic Set Multilinear circuit $A(\boldsymbol{x})$ as: $A(\boldsymbol{x}) = \boldsymbol{c}^{\intercal}\Pi(1 + \alpha_i x_i)$, where $\alpha_i \in H_k(\mathbb{F})$. The first explicit mention of the problem that we can find is in [32], where it was shown that such polynomials can be transformed to a log-concentrated polynomial by appropriately shifting each one of the variables i.e. $x_i \to x_i + t_i$. As in Chapter 2, this immediately gives an $n^{O(\log n)}$ time algorithm. The current best algorithm for this problem has time complexity $n^{O(\log \log n)}$ [11]. This algorithm follows the same strategy as the blackbox test for Diagonal depth-3 – It uses a hash function family to reduce the circuit to an ROABP over $O(\log n)$ variables, and then uses the algorithm from [12]. In this work, we approach PIT for this model via tensor rank lower bounds and provide an algorithm with time complexity $n^{\tilde{O}(\log n)}$, where $\tilde{O}(\log n)$ is $O(\log n \log \log n)$. Readers will notice that the running time of our algorithm is (much) worse than existing bounds. Why then, should

one care about this result? The potential value of our method stems from it's freshness – The hope is that we can push the results of this thesis further in order to achieve polynomial size hitting sets; Something that we do not think is possible with existing methods. Whether this turns out to be possible or not remains to be seen, but in Chapter 4 we outline the problems that would need to be solved in order to achieve this. Along the way, we also prove a tensorial generalization of a simple result in linear algebra, which may be of independent interest.

3.2 Notation and Preliminaries

We continue using the notation from Chapter 2, wherein lowercase boldfont $\boldsymbol{x}$ is used to denote an n-tuple. This n-tuple can, once again be used in different contexts, as the index of a tensor entry, as the degree of a monomial, or as an element of a vector space. As before, we abuse notation and address these different versions of the tuple by the same name, and carry over all associated notions, such as the dot product: $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = x \cdot y$, and the 1-norm.

Given a subset $S \subset [n]$, the restriction of the vector $\boldsymbol{x}$ to S is the vector composed of the entries $\{x_i : i \in S\}$ and is denoted by $\boldsymbol{x}_S$. In this chapter, we also make use of uppercase bold font $\mathbf{R}$ to denote a tuple of vectors, or a matrix. Such uses will be made clear in the context.

A *tensor* is an n-dimensional array of field elements, and for our purposes, we can think of a tensor as a map $T : \{0, 1, \cdots, d\}^n \to \mathbb{F}$. Given n maps $v_i : \{0, 1, \cdots, d\} \to \mathbb{F}$, their tensor product $V = \otimes_{i=1}^{n} v_i$ is an n-dimensional tensor, with entries: $V(\boldsymbol{b}) = \prod_{i=1}^{n} v_i(b_i)$. Any tensor that can be expressed as such a product is called a *rank one* tensor. The rank of tensor T is then defined as $\text{rank}(T) = min\ \{k : T = \sum_{j=1}^{k} V_j;\ \text{rank}(V_j) = 1\}$.

As noted in Chapter 1, there is a simple correspondence between a n-variate, d degree polynomial, and an n-dimensional tensor, with side length d – The coefficient of a monomial $x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$ is the $(i_1, i_2, \cdots, i_n)^{\text{th}}$ entry of the tensor. We can sharpen this correspondence by observing that the coefficients of a Basic Set Multinear polynomial $A(\boldsymbol{x})$ with top fan-in bounded by k, can be thought of as coming from the entries of a tensor T_A, with $\text{rank}(T_A) \leq k$.

Our approach to identity testing (also outlined in [12]) relies on this correspondence as follows: The polynomial evaluating to zero at a particular point corresponds to linear relationship holding among the entries of the tensor. Hence, in this setting, a hitting set can be thought of as a low dimensional subspace $\mathcal{L}$, spanned by rank-one tensors[1], such that all non-zero tensors in the orthogonal complement $\text{Null}(\mathcal{L})$ have rank $> k$.

Hopping to a higher level of abstraction, it is clear that we need a structural property that distinguishes high rank tensors. At this point, we should probably clarify our usage of the term 'high rank tensor' in this thesis. Finding explicit tensors with high tensor rank is a famous open problem – We still do not have *explicit* three-dimensional tensor whose rank is $\omega(d)$, while a simple counting argument shows a rank lower bound of $\Omega(d^2)$ for almost all such tensors[2]. If we do not even have an explicit example, how can we hope to specify a general pattern that forces high rank? The catch of course, is that in this work we consider any tensor with rank $\geq k$ as 'high rank'. This is (comparatively) a meagre demand, since for generic tensors, the rank grows exponentially with the dimension, i.e. as $d^{\Omega(n)}$. In order to bound the rank, we introduce the concept of reshaping, a classic tensor rank lower bound that has (unfortunately)

[1]The tensors are rank-one because evaluating it the polynomial can be viewed as taking it's dot product with a tensor formed with the evaluated values of the corresponding monomials. The tensor formed by taking the values of each monomial evaluation is of rank one.

[2]For those that smell a connection with circuit lower bounds and want to know more, wait for a couple of paragraphs.

stood the test of time. Reshaping gives us rank lower bounds for the tensor in terms of the rank of a derived matrix, which is easier to reason about.

The simplest example of a high rank matrix pattern (and the one that we exploit) is a triangular matrix – With non-zero entries on the diagonal and all entries above (or below) set to zero. The tensors we show have high rank are a simple higher-dimensional twist on triangular matrices that we call simplicial tensors. We define this property formally later in the chapter; The following intuition should suffice for now – Consider a matrix with a particular zero-nonzero pattern: It contains a right triangular region of zero entries, with a large number of nonzero entries on the hypotenuse. It is clear, from the fact that triangular matrices have full rank, that such a matrix has high rank. We show that an obvious generalization of this result also holds for tensors (Though the parameteric dependence of tensor rank on the size of the triangle, or rather the simplex, is much weaker, which is probably a manifestation of the intractability of tensor rank.)

Happily enough, it turns out that simplicial tensors are 'PIT compatible'. As it turns out, the property of being a polynomial corresponding to a simplicial tensor can easily be enforced by composing two simple PIT tools, namely interpolation followed by sparse PIT [22].

3.3 Tensor Rank Lower Bounds

In this section we explore the notion of tensor rank in greater detail. From a computational perspective, while matrix rank is in P, for even 3-dimensional tensors, tensor rank is known to be NP-hard [17], which is a strong hint about the expressiveness of this notion. Indeed tensor rank is a fundamental concept in the study of arithmetic circuits, and is strongly linked with circuit lower

bounds.The first work to explore this connection was [38], and proved the following result:

Theorem 3.1 ([38]). *Given a tensor* $T : [d]^3 \to \mathbb{F}$*, the smallest size of the arithmetic circuit computing the polynomial* $\sum_{(i,j,k)\in[d]} T(i,j,k)x_iy_jz_k$ *is* $\Omega(\text{rank}(T))$.

Most recently, [27] showed that for a tensor T with $\text{rank}(T) \geq d^{n(1-o(1))}$, and $n = \frac{\log d}{\log\log d}$, the associated polynomial has super polynomial formula complexity. Given these connections, and how little we know about circuit lower bounds, it should come as no surprise that we do not know much about tensor lower bounds either.

Having ensured that the reader is intimidated enough by tensors, we now describe the most basic technique for lower bounding tensor rank. In mathematics, whenever one encounters a hard nonlinear problem, one possible strategy is to replace it by an easy linear problem, and in this case we achieve that by reshaping the tensor into a matrix [24, 26]:

Definition 3.2. Let T be an n-dimensional tensor. Given a subset of indices, $S \subset [n]$, define the reshaped matrix $M_S(T)$ to be the matrix with rows indexed by subsets of S and columns indexed by subsets of the complement $\bar{S}$. Hence, the (i,j)-th entry of the matrix M_S is given by the tensor entry $T(S = i, \bar{S} = j)$ i.e. the entry is found by taking the indices in S to be set to i and the ones in $\bar{S}$ to be set to j.

Lemma 3.3 ([24]). *For a tensor* T*, and a reshaping* S*, we have* $\text{rank}\,(M_S(T)) \leq \text{rank}\,(T)$.

Proof. Observe that if T were a rank one tensor, M_S would be a rank one matrix. This implies that any k-rank decomposition of the tensor T yields a k-rank decomposition of M_S. Hence $\text{rank}\,(M_S(T)) \leq \text{rank}\,(T)$. □

Once we reshape our tensor, we are still left with the task of enforcing rank lower bounds for M_S, with the important difference that we are now in the world of matrices, with a rich menagerie of high-rank patterns to choose from. In this work, we only make use of triangular matrices – Arguably the simplest pattern that causes high rank. The reasons for this choice become clear later, but for now note that the rank of this pattern depends only on the support of the matrix. Having already referred to it many times so far, we finally provide a formal definition of the triangular pattern:

Definition 3.4. A matrix M is said to contain an upper k-triangle, if there exists a sequence of matrix indices $\{(r_1, c_1), (r_2, c_2), \ldots, (r_k, c_k)\}$, such that

- $\forall\ i \in [k]$: $M(r_i, c_i) \neq 0$, and
- $\forall\ i, j \in [k]$, such that $i < j$: $M(r_j, c_i) = 0$

The index (r_j, c_i) is referred to as the index between (r_i, c_i) and (r_j, c_j).

The fact the matrix is upper triangular is not essential, and is mentioned to provide geometric clarity to the definition. Hence, the strategy we pursue in order to prove tensor rank lower bounds is to show that there exists a reshaping of the tensor that contains a k-triangle (In fact, we prove a stronger result – That a random reshaping contains a triangle with high probability). What kind of tensor would be likely to contain a triangle? We answer this question in the next section.

3.4 Simplicial Tensors

In this section we give a description of simplicial tensors, which are a natural, higher dimensional generalization of triangular matrices. We will show that

under a random reshaping (corresponding to a uniformly chosen subset of variables S), such simplicial tensors contain a k-triangle, with high probability. By 3.3, this proves that the rank of simplicial tensors is lower bounded by k.

Definition 3.5. An n-dimensional tensor T is said to be $(\boldsymbol{r}, \rho)$-simplicial, where $\boldsymbol{r} \in \mathbb{N}^n$, and $\rho \in \mathbb{N}$, if it has the following structure: Define the weight function as $\mathrm{Wt}(\boldsymbol{b}) = \langle \boldsymbol{r}, \boldsymbol{b} \rangle$ and partition the tensor indices into the level sets $\{L_1, L_2, \ldots, L_m\}$ of this function, i.e. $L_j = \{\boldsymbol{b} \in \{0,1\}^n : \mathrm{Wt}(\boldsymbol{b}) = j\}$. In order for the tensor to be $(\boldsymbol{r}, \rho)$ simplicial, we require that for each set L_j, it's support (the number of non-zero tensor entries in it) is either large or non-existent, i.e. $\forall j : |\operatorname{supp}(L_j)| \geq \rho$ or $|\operatorname{supp}(L_j)| = 0$.

3.4.1 Connection with PIT

This definition may seem a little artificial, but those familiar with PIT will realise that the property of being $(\boldsymbol{r}, \rho)$-simplicial can be enforced with some basic techniques. First we map our variables $x_i \to x_i \cdot t^{r_i}$. This gives us the polynomial $A(\boldsymbol{x}, t) = A_0(\boldsymbol{x}) + A_1(\boldsymbol{x}) \cdot t + \cdots + A_m(\boldsymbol{x}) \cdot t^m$, where $m = |\boldsymbol{r}|_1$. By interpolating this polynomial in t, we can isolate the monomials with equal weighted degree, and conduct sparse PIT with sparsity ρ for each of these groups. Then every non-zero polynomial mapped to zero by this test will be $(\boldsymbol{r}, \rho)$-simplicial. Assuming an appropriate choice of parameters, no Basic Set Multilinear polynomial with top fan-in $\leq k$ can be $(\boldsymbol{r}, \rho)$-simplicial. Hence this test preserves the non-zeroness of Basic Set Multilinear circuits. The time complexity of this PIT algorithm is $O(m\rho)$. All that is left is to show rank lower bounds on simplicial tensors.

3.5 Triangles in a Simplex?

Having defined simplicial tensors, and understood their correspondence with PIT, we will list a few obvious facts about them. Note that the number of level sets is $m = r_1 + \cdots + r_n + 1$. Also note that the tail sets, corresponding to the smallest and largest values will be quite small, and as such, the sparsity constraint will ensure that all those entries are zero. Looking at it geometrically, the resulting tensor will have a large simplicial region of zero entries, with the first bounding plane containing many non-zero entries. Vague geometric intuition suggests that such a tensor would contain triangles – Under a reshaping S, the index between two non-zero indices in the bounding plane most likely belongs to a lower set, and is thus a zero entry.

Let p be the smallest integer such that $\mathrm{supp}(L_p) \neq 0$. Let $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k\}$ be the indices of k non-zero entries in L_p. Given a reshaping S, define the weight under a reshaping $\mathrm{Wt}_S(\boldsymbol{b}) = \langle \boldsymbol{r}_S, \boldsymbol{b}_S \rangle$. Note that we have the general relation $\mathrm{Wt}(\boldsymbol{b}) = \mathrm{Wt}_S(\boldsymbol{b}) + \mathrm{Wt}_{\bar{S}}(\boldsymbol{b})$.

Lemma 3.6. *Under a reshaping S, if the value $\mathrm{Wt}_S(\boldsymbol{b}_i)$ is distinct for each index $\boldsymbol{b}_i$, the indices form a k-triangle in the reshaped matrix M_S.*

Proof. Since the weights under reshaping are all distinct, assume w.l.o.g. that $\mathrm{Wt}_S(\boldsymbol{b}_1) > \mathrm{Wt}_S(\boldsymbol{b}_2) > \cdots > \mathrm{Wt}_S(\boldsymbol{b}_k)$. Under the reshaping S, a pair of indices $\boldsymbol{y}$ and $\boldsymbol{z}$ are mapped to $(\boldsymbol{y}_S, \boldsymbol{y}_{\bar{S}})$ and $(\boldsymbol{z}_S, \boldsymbol{z}_{\bar{S}})$ and the index between them is $(\boldsymbol{z}_S, \boldsymbol{y}_{\bar{S}})$. Hence, for any $i < j$, the weight of the index between $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ is given by $\mathrm{Wt}_S(\boldsymbol{b}_j) + \mathrm{Wt}_{\bar{S}}(\boldsymbol{b}_i) \;=\; p + (\mathrm{Wt}_S(\boldsymbol{b}_j) - \mathrm{Wt}_S(\boldsymbol{b}_i))$, which, by the above inequalities, is clearly less than p. As p was taken to be the weight of the first non-zero plane, for any ordered pair of indices in this set, the index between them contains a zero entry. □

In the next section, we look at the conditions on a set of indices, and a weight vector $\boldsymbol{r}$, that give us a set of k distinct weights under a reshaping.

3.6 Random Reshapings and Flat Distributions

Let $W(\boldsymbol{b})$ denote the random variable corresponding to the weight of $\boldsymbol{b}$ over a random reshaping, i.e. $W(\boldsymbol{b}) = \mathrm{Wt}_S(\boldsymbol{b}) : S \subseteq_R [n]$. Hence, $W(\boldsymbol{y})$ and $W(\boldsymbol{z})$ are random variables over $\mathbb{N}$, and we are required to show that with high probability, they are distinct. In this section, we show the following result:

Theorem 3.7. *Let $l = \max(10 \log k, \log n)$. There exists a set on n integers $\boldsymbol{r}$, with $|\boldsymbol{r}|_1 = k^{O(l \log l)}$, such that for all pairs of indices $\boldsymbol{y}, \boldsymbol{z} : |\boldsymbol{y} \oplus \boldsymbol{z}| > \Omega(\log k)$, the probability $\Pr(W(\boldsymbol{y}) = W(\boldsymbol{z}))$ is upper bounded by $\frac{1}{k^2}$.*

Given this theorem, we can appeal to the union bound and get a reshaping across which the weights of k indices are distinct. We can ensure the existence of a set of k non-zero, pairwise $\Omega(l)$-distant binary indices, by setting the sparsity ρ of the simplicial tensor to be $kn^{\Omega(l)}$.

We start by analysing the expression $\Pr(W(\boldsymbol{y}) = W(\boldsymbol{z}))$. It is handy to re-express the probability in terms of the set difference: It is clear that $\Pr(W(\boldsymbol{y}) = W(\boldsymbol{z})) = \Pr(W(\boldsymbol{y} \backslash \boldsymbol{z}) = W(\boldsymbol{z} \backslash \boldsymbol{y}))$ (Here set difference is defined in the regular way: $\boldsymbol{y} \backslash \boldsymbol{z} = \boldsymbol{y} - (\boldsymbol{y} \oplus \boldsymbol{z})$). Since $\boldsymbol{y} \backslash \boldsymbol{z}$ and $\boldsymbol{z} \backslash \boldsymbol{y}$ are disjoint, the corresponding random variables are independent, and we can write $\Pr(W(\boldsymbol{y} \backslash \boldsymbol{z}) = W(\boldsymbol{z} \backslash \boldsymbol{y})) = \sum_{i \in [m]} \Pr(W(\boldsymbol{y} \backslash \boldsymbol{z}) = i) \cdot \Pr(W(\boldsymbol{z} \backslash \boldsymbol{y}) = i)$. Hence, one way to reduce the probability of equality, would be to ensure that these random variables are 'well spread out', that the probability that either $\Pr(W(\boldsymbol{y} \backslash \boldsymbol{z}) = i)$ or $\Pr(W(\boldsymbol{z} \backslash \boldsymbol{y}) = i)$ is low for every i. To flesh out this idea, we define ϵ-flat distributions and prove a couple of simple lemmas about them.

Definition 3.8. The distribution of a discrete random variable X over $\mathbb{N}$ is said to be ϵ-flat if $\forall\ i \in \mathbb{N} : \Pr(X = i) \leq \epsilon$.

Lemma 3.9. *If Y and Z are independent random variables, such that the distribution of Y is ϵ-flat, then:*

1. $\Pr(\ Y = Z) \leq \epsilon$, *and*
2. *$Y + Z$ is ϵ-flat distributed.*

Proof. The proofs of these statements are almost identical:

1. $\Pr(\ Y = Z) \ = \ \sum_{i\in[m]} \Pr(Y = i)\cdot\Pr(Z = i) \ \leq \ \epsilon\cdot\sum_{i\in[m]} \Pr(Z = i) \ \leq \ \epsilon.$

2. $\Pr(\ Y + Z = j) \ = \ \sum_{i\in[m]} \Pr(Y = j - i)\cdot\Pr(Z = i) \ \leq \ \epsilon\cdot\sum_{i\in[m]} \Pr(Z = i) \ \leq \ \epsilon.$ □

To recap, we expressed the probability $\Pr(W(\boldsymbol{y}) = W(\boldsymbol{z}))$ in terms of the probability that $W(\boldsymbol{y}\backslash\boldsymbol{z})$ (w.l.o.g) takes on any one particular value. Setting $\boldsymbol{b} = \boldsymbol{y}\backslash\boldsymbol{z}$, we note that the lowest possible flatness of a random variable $W(\boldsymbol{b})$ is $\frac{1}{2^{|\boldsymbol{b}|}}$, which corresponds to a value of $m = 2^{|\boldsymbol{b}|}$, i.e. each non-empty subset of the set bits of $\boldsymbol{b}$ gives rise to a distinct sum. Setting $r_i = 2^i$ helps us attain this bound, but since the cost of the algorithm is proportional to $\sum r_i$, this idea is no good (Note that this 'solution' corresponds exactly to brute force by the Kronecker trick discussed in Chapter 2).

It is clear that we wish to pick the integers $\boldsymbol{r} = (r_1, r_2, \cdots r_n)$ such that the weights across random subsets of any $\Omega(\log k)$-sized subset of $\boldsymbol{r}$, corresponding to the set bits in $\boldsymbol{b} = \boldsymbol{y}\backslash\boldsymbol{z}$, are distributed across many *distinct* values. If the r_i's were vectors instead of integers, proving the distinctness of these subset weights might be easier – Given a set of linearly independent vectors, the sum of each subset of these vectors would be distinct. From this point of view, the

problem reduces to looking for a set of vectors that are (approximately) l-wise linearly independent, i.e. a set of n vectors $\mathbf{R}$, such that for any l-sized subset of $\mathbf{R}$, the dimension of the space $\text{span}(R_{i_1}, R_{i_2}, \cdots, R_{i_l})$ is at least $l/2$ (Say). Note that this results in a flatness of $\leq 2^{-l/2}$, since every subset of these $l/2$ vectors gives a distinct sum.

To proceed, we solve the above problem for vectors, and then map the vectors to integers in a way that the addition of the integers mimics the addition of the vectors. The technical term for such a map is a Freiman Isomorphism:

Definition 3.10 ([15])**.** Let A be a subset of an abelian group G. Then a Freiman Isomorphism of order l is a map $\Phi : G \to \mathbb{Z}$ such that $a_1+a_2+\cdots+a_l = a'_1+a'_2+\cdots a'_l$ if and only if $\Phi(a_1)+\Phi(a_2)+\cdots+\Phi(a_l) = \Phi(a'_1)+\Phi(a'_2)+\cdots \Phi(a'_l)$.

Note that if G is the vector space $\mathbb{R}^L$, and A is the set of binary strings, $A = \{0,1\}^L$, then for $\boldsymbol{a} \in A$, the map $\Phi(\boldsymbol{a}) = a_1+a_2\cdot(l+1)+\cdots a_L\cdot(l+1)^{L-1}$ is a Freiman Isomorphism of order l. Basically, the map 'embeds' vector addition in integer addition by expressing the vectors as integers in a sufficently high base, and treating each digit of the representation as a different dimension. Since the base is large enough, there are no carries and addition happens digit-wise, just as in a vector.

3.7 l-wise Linearly Independent Vectors

Approximate l-wise independence of a set of vectors means that any subset of l vectors should span a space of dimension at least $l/2$. In this section, we design a set of n approximately l-wise independent bit vectors. This is done using combinatorial designs, an extremely handy tool when it comes to notions of approximate independence, and thus heavily used when it comes to pseudorandom generation and randomness extraction.

Definition 3.11. A collection of subsets $D = (D_1, D_2, \cdots, D_m)$ of a universe U, is called an (q, r) design if, for every i, $|D_i| = q$ and for every $i \neq j$, $|D_i \cap D_j| < r$.

Such designs were first constructed by [25]. If $L = l^2$, [25] show how we can build an $(l^{3/2}, l)$ design over a universe of L elements. Such a design is guaranteed to have at least n subsets, because of the way in which we chose l. The set $\mathbf{R}$ is a set of vectors in $\{0,1\}^L$, and is simply obtained by taking the indicator vectors of each of the D_i's – This means that the j^{th} entry of R_i is 1 if $j \in D_i$ and 0 otherwise. Using some simple linear algebra, we can re-express the dimension of a subset of l vectors as the rank of a diagonally dominant matrix and then appeal to the following result from [6]:

Theorem 3.12. *Let* $\mathbf{A} = (a_{ij})$ *be an* $n \times n$ *real, symmetric matrix, with* $a_{ii} = 1$ *and* $a_{ij} < \frac{1}{\sqrt{n}}$, *for all* $i \neq j$. *Then* $\text{rank}(\mathbf{A}) \geq n/2$.

This shows us that the collection of vectors $\mathbf{R}$ is l-wise linearly independent. A more general analysis of these design matrices was carried out by [8], but given the importance of this result to our thesis, we include the proof for completeness:

Theorem 3.13. *Let* $\mathbf{R}$ *be an* $N \times L$ *matrix whose rows are the indicator vectors of an* $(l^{3/2}, l)$ *design, as constructed above. Let* $\mathbf{P}$ *be the submatrix formed by taking any* l *rows of* $\mathbf{R}$. *Then* $\text{rank}(\mathbf{P}) > l/2$.

Proof. We wish to show a lower bound on $(\mathbf{P})$. It is well known that for any real matrix $\mathbf{P}$, we have $\text{rank}(\mathbf{P}) = \text{rank}(\mathbf{P}^\intercal\mathbf{P})$. In this case, we can argue that the matrix $\mathbf{P}^\intercal\mathbf{P}$ is an $l \times l$ diagonally dominant matrix – The diagonal entries are much larger than the off diagonal entries. This is simply because the rows of $\mathbf{P}$ formed a design, hence the value of each diagonal entry in $\mathbf{P}^\intercal\mathbf{P}$ is $l^{3/2}$,

while the value of each off-diagonal entry is $\leq l$. Dividing each column by $l^{3/2}$ preserves the rank of the matrix, and appealing to the previous theorem, we get a lower bound of $l/2$ for rank($\mathbf{P}$). □

This theorem shows that any $\mathbf{P}$ contains a set of $l/2$ linearly independent rows. All linear combinations of these rows are distinct: In particular, we have a set of $l/2$ vectors $(R_{i_1}, R_{i_2}, \cdots, R_{i_{l/2}})$ such that the sum $(s_1 R_{i_1} + s_2 R_{i_2} + \cdots + s_{l/2} R_{i_{l/2}})$ is unique for every possible binary assignment of the variables s_i. If we use the order l Freiman isomorphism to obtain $r_i = \Phi(R_i)$, we can conclude that for any $\boldsymbol{b} \in \{0,1\}^n$ such that $|\boldsymbol{b}| = l$, the distribution of the random variable $W(\boldsymbol{b})$ is $\frac{1}{2^{l/2}} \leq \frac{1}{k^2}$-flat. The vectors R_i are L dimensional and hence applying Φ give numbers r_i whose size is upper bounded by $l^L = n^{O(l \log l)}$.

3.8 Putting it all together

Having seen all the different components of our algorithm, we provide a broad summary of what we did. We defined simplicial tensors and proved the existence of $\boldsymbol{r} : |\boldsymbol{r}|_1 = n^{O(l \log l)}$ and $\rho = kn^{O(l)}$, such that the rank of every non-zero $(\boldsymbol{r}, \rho)$-simplicial tensor is greater than k. This is achieved for two reasons:

The sparsity lower bound of ρ on the first non-zero slice guarantees the existence of k non-zero indices that are a Hamming distance of $\geq l$ from each other. For a suitable choice of $\boldsymbol{r}$, the weights of any two l-separated indices are different under a random reshaping with probability $1 - O(\frac{1}{k^2})$. Using the Union bound, we can obtain the existence of reshaping under which the weights of all k points are different. As we proved, this is a sufficient condition to obtain a rank lower bound of k for the tensor.

The PIT algorithm corresponding to this, as we mentioned, is to isolate the indices $\boldsymbol{x}$ with the same value of $\mathrm{Wt}(\boldsymbol{x})$ by interpolation, and then apply the sparsity bound across this slice by sparse PIT. This ensures that the only non-zero polynomials that evaluate to zero on our test correspond to high rank tensors. The running time of the PIT algorithm is dominated in this case by the interpolation step, or the size of the r_i's, which is $n^{\tilde{O}(\log n)}$.

Chapter 4

Conclusion

4.1 Further Work

In terms of future work, one candidate for improvement is the weight vector $\boldsymbol{r}$ we use for the simplicial tensor. Recall that we designed a set of numbers $\boldsymbol{r} = (r_1, r_2, \cdots, r_n)$ such that for any $x : |x| = l$ the distribution of $W(x)$ is $\frac{1}{2^{\Omega(l)}}$ flat. In order to achieve this, the numbers in $\boldsymbol{r}$ could get as large as $n^{\tilde{O}(\log n)}$. One obvious improvement we desire is to achieve the same flatness, but with polynomially large r_i's instead. At this point, we are unsure of whether such a design even exists. If, instead of a flatness of $\frac{1}{2^{\Omega(l)}}$, we require the lowest possible flatness of $\frac{1}{2^l}$, we can easily show that such designs do not exist for $|\boldsymbol{r}| = n^{O(1)}$. Note that a flatness of $\frac{1}{2^l}$ implies that any subset of an $\leq l$-size subset of $\boldsymbol{r}$ would sum to a unique value. Consider the Hamming ball of radius $l/2$ centred at the origin (say). Given two points $\boldsymbol{x}$ and $\boldsymbol{y}$ in this ball, if $\mathrm{Wt}(\boldsymbol{x}) = \mathrm{Wt}(\boldsymbol{y})$, we have $\mathrm{Wt}(\boldsymbol{x} \backslash \boldsymbol{y}) = \mathrm{Wt}(\boldsymbol{y} \backslash \boldsymbol{x})$. Since $\boldsymbol{x}$ and $\boldsymbol{y}$ are in the same Hamming ball, we have $|\boldsymbol{x} \backslash \boldsymbol{y}| + |\boldsymbol{y} \backslash \boldsymbol{x}| \leq l$. This gives an $\leq l$-size subset, two distinct subsets of which have the same weight. This means that all points in the Hamming ball must have different weights. Since there are $n^{\Omega(l)}$ points inside the Hamming

ball, this immediately gives a quasi-polynomial lower bound for $|\boldsymbol{r}|$, in the case where we require the flattest posible distribution. Fortunately, this simple 'top kill' does not seem to extend to the relaxed version given above, which gives us hope that it may be possible to come up with polynomially large designs after all.

4.2 Results and Conclusion

In this thesis we have obtained blackbox PIT algorithms for the Diagonal depth-3 model and the Basic Set Multilinear model. The recent work [11] gives algorithms with running time $n^{O(\log \log n)}$ for both these models. For Diagonal depth-3, we match their result with a runtime of $n^{O(\log \log n)}$. For the Basic Set MultiLinear Model, we achieve a runtime of $n^{O(\log n \log \log n)}$. In this case, our main technical innovation is to define a class of tensors called simplicial tensors and to prove rank lower bounds for them. Our hope is that our results on simplicial tensors can be tightened.

Bibliography

[1] Agrawal, M. (2005). Proving lower bounds via pseudo-random generators. In *FSTTCS*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105.

[2] Agrawal, M., Gurjar, R., Korwar, A., and Saxena, N. (2014). Hitting-sets for ROABP and sum of set-multilinear circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:85. (to appear in SICOMP, 2015).

[3] Agrawal, M., Kayal, N., and Saxena, N. (2004). Primes is in p. *Annals of Mathematics*, pages 781–793.

[4] Agrawal, M., Saha, C., and Saxena, N. (2013). Quasi-polynomial hitting-set for set-depth-D formulas. In *STOC*, pages 321–330.

[5] Agrawal, M. and Vinay, V. (2008). Arithmetic circuits: A chasm at depth four. In *FOCS*, pages 67–75.

[6] Alon, N. (2003). Problems and results in extremal combinatorics, i. *Discrete Mathematics*, pages 31–53.

[7] Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M. (1998). Proof verification and the hardness of approximation problems. *Journal of the ACM*, pages 501–555.

[8] Barak, B., Dvir, Z., Yehudayoff, A., and Wigderson, A. (2011). Rank bounds for design matrices with applications to combinatorial geometry and locally correctable codes. In *STOC*, pages 519–528.

[9] Dvir, Z. and Shpilka, A. (2007). Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM J. Comput.*, 36(5):1404–1434.

[10] Forbes, M. A. (2014). *Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs*. PhD thesis, MIT.

[11] Forbes, M. A., Saptharishi, R., and Shpilka, A. (2014). Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 867–875.

[12] Forbes, M. A. and Shpilka, A. (2012a). On identity testing of tensors, low-rank recovery and compressed sensing. In *STOC*, pages 163–172.

[13] Forbes, M. A. and Shpilka, A. (2012b). Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. *CoRR*, abs/1209.2408.

[14] Forbes, M. A. and Shpilka, A. (2013). Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *FOCS*, pages 243–252.

[15] Freiman, G. (1973). *Foundations of a structural theory of set addition*. Number 37. AMS.

[16] Gupta, A., Kamath, P., Kayal, N., and Saptharishi, R. (2013). Arithmetic circuits: A chasm at depth three. *FOCS*, pages 578–587.

[17] Hastad, J. (1990). Tensor rank is np complete. *Journal of Algorithms*, pages 644–654.

[18] Kabanets, V. and Impagliazzo, R. (2003). Derandomizing polynomial identity tests means proving circuit lower bounds. *STOC*, pages 355–364.

[19] Karnin, Z. S. and Shpilka, A. (2011). Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. *Combinatorica*, 31(3):333–364.

[20] Kayal, N. and Saraf, S. (2009). Blackbox polynomial identity testing for depth 3 circuits. In *FOCS*, pages 198–207.

[21] Kayal, N. and Saxena, N. (2007). Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138.

[22] Klivans, A. and Spielman, D. A. (2001). Randomness efficient identity testing of multivariate polynomials. In *STOC*, pages 216–223.

[23] Kronecker, L. (1882). Grundzuge einer arithmetischen theorie der algebraischen grossen. *G. Reimer*.

[24] Nisan, N. (1991). Lower bounds for non-commutative computation (extended abstract). In *Proceedings of the 23rd ACM Symposium on Theory of Computing, ACM Press*, pages 410–418.

[25] Nisan, N. and Wigderson, A. (1994). Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167.

[26] Nisan, N. and Wigderson, A. (1997). Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3):217–234.

[27] Raz, R. (2010). Tensor-rank and lower bounds for arithmetic formulas. In *STOC*, pages 659–666.

[28] Raz, R. and Shpilka, A. (2005). Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19.

[29] Raz, R. and Yehudayoff, A. (2009). Lower bounds and separations for constant depth multilinear circuits. *Computational Complexity*, 18(2):171–207.

[30] Saxena, N. (2008). Diagonal circuit identity testing and lower bounds. *ICALP*, pages 60–71.

[31] Saxena, N. (2009). Progress on polynomial identity testing. *Bulletin of the EATCS*, 99:49–79.

[32] Saxena, N. (2014). Progress on polynomial identity testing-II. In *Perspectives in Computational Complexity*, pages 131–146. Birkhäuser Basel.

[33] Saxena, N. and Seshadhri, C. (2011). An almost optimal rank bound for depth-3 identities. *SIAM J. Comput.*, 40(1):200–224.

[34] Saxena, N. and Seshadhri, C. (2012). Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn't matter. *SIAM J. Comput.*, 41(5):1285–1298.

[35] Schwartz, J. T. (1980). Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717.

[36] Shpilka, A. and Volkovich, I. (2009). Improved polyomial identity testing for read-once formulas. pages 709–713.

[37] Shpilka, A. and Yehudayoff, A. (2010). Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388.

[38] Strassen, V. (1973). Vermeidung von divisionen. *The Journal fur die Reine und Angewandte Mathematik*, pages 182–202.

[39] Tutte, W. T. (1947). The factorization of linear graphs. *Journal of the London Mathematical Soceity*, pages 107–111.