On Blackbox Polynomial Identity Testing of sparse polynomials

Devansh Shringi

Advisor: Prof. Markus Bläser

November 1, 2021

Contents

1 Introduction

In this report we will be studying the problem of Polynomial Identity Testing (PIT). It is the problem to efficiently test whether a given polynomial P is identically zero. The problem is

interesting when the polynomial is given in compact form as a blackbox or arithmetic circuit. There are 2 variants of PIT, Whitebox where we have access to polynomials computed in gates of the arithmetic circuit and the Other Blackbox, where we can only do evaluations of the polynomial given as blackbox. We will be interested only in the Blackbox setting in this report.

It is notable that the problem of PIT has a very simple and elegant randomized solution, thanks to the PIT lemma.

Lemma 1.1. ***(PIT Lemma)****(Schwartz-Zippel[Sch80]) Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-zero polynomial of total degree $d \geq 0$. Let S be any finite subset of $\mathbb{F}$, and let $\alpha_1, \ldots, \alpha_n$ be elements selected independently, uniformly and randomly from S. Then,*

$$Pr_{\alpha_1,\ldots,\alpha_n \in S}[f(\alpha_1, \ldots, \alpha_n) = 0] \leq \frac{d}{|S|}$$

The above lemma can be easily proved inductively, with the base case being the univariate case. This puts PIT in $coRP$. The problem of derandomizing PIT in polynomial time, to put it in P is still open.

We will look at the restricted version of Blackbox PIT where the number of monomials in the input polynomial is bounded by m. We will briefly discuss the previous works done on this problem, and give an algorithm that requires only $\log(m)$ random bits to solve it for the field $\mathbb{R}$. This is also the information theoretic bound on the problem. We will be using the Descartes Rule similar to [BE11] and the construction of dense Sidon Sets from [BP20].

We also approach to give a hitting set and a randomized algorithm, if we are given variable order such that number of monomials of $x_1 \leq x_2 \leq \ldots \leq x_n$.

1.1 The Motivation

The problem of PIT has many applications, like the problem of deciding the existence of perfect matching in a graph efficiently can be seen as a question of finding an efficient PIT algorithm for the determinant polynomial of the graph's Tutte matrix. The idea of PIT was also very useful in the proof of the complexity result $IP = PSPACE$. Even the problem of Primality testing was solved by working with a PIT formulation. It was observed that a positive integer n is prime iff $(x+1)^n = (x^n+1)(mod n)$, which can be considered as checking $P(x) = (x+1)^n - (x^n+1)$ to be identity over $\mathbb{Z}/n\mathbb{Z}$. The problem of derandomizing PIT has relations to complexity results like $PIT \in P \implies NEXP \not\subseteq P/poly$ or $VP \neq VNP$. For more details into PIT and it's application, look at the surveys [Sax09], [Sax14] and [SY10].

Bläser and Engels [BE11], used the Descartes rule of signs, to give a deterministic algorithm with $\tilde{O}(m^3n^3)$ running time. They also created a randomized algorithm that used $O(\log^2(mn/\epsilon))$ random bits. The theoretical lower bound for the problem is $O(\log m)$ random bits, and hence we work on reaching this bound.

Recently, Sidon Sets were used in [BP20] to obtain an algorithm with optimal randomness for low degree polynomials. They were used to bring the problem to univariate polynomials,

as they mapped each monomial to a different univariate monomial. This has been a key concept in the solutions for sparse PIT, and hence, it was natural to see if we can use them to reduce the number of random bits required for solving sparse PIT.

1.2 Preliminaries and Notation

We use $\mathbb{F}$ to represent fields. We will mainly be working with $\mathbb{R}$, and the variables will be denoted mainly by $x_1, \ldots, x_n$ with n denoting the number of indeterminants. $\mathbb{R}_m[x_1, \ldots, x_n]$ will denote the set of polynomials over $\mathbb{R}$ with at most m monomials. $PIT(n, m)$ is the problem of deciding if a polynomial in $\mathbb{R}_m[x_1, \ldots, x_n]$ given by a black box is identically zero.

1.2.1 Hitting Sets Generators and Transformations

We will give definitions of terms related to hitting sets that we will require

Definition 1. (Hitting Set)[BE11] A Hitting set $H \subseteq \mathbb{R}^n$ for the PIT(n,m) is a set of points such that for a non-zero polynomial $f \in \mathbb{R}_m[x_1, \ldots, x_n]$, for a random point $(\alpha_1, \ldots, \alpha_n) \in H$, probability that $f(\alpha_1, \ldots, \alpha_n) \neq 0$ is $\geq 1 - \epsilon$, i.e.

$$Pr_{\alpha \in H}[f(\alpha) \neq 0] \geq 1 - \epsilon$$

We will focus on hitting sets with integer elements (we can work with any set of $\mathbb{R}$ which can be efficiently represented). For our purpose $\mathbb{Z}$ suffices.

Definition 2. (Hitting Set Generator)[BE11] of probability $1 - \epsilon$ is a function $\mathcal{H} : \{0,1\}^\rho \to \mathbb{Z}^n$ whose image is a hitting set with hitting probability $1 - \epsilon$. ρ is the seed length .

We can use a hitting set generator of $PIT(n, m)$ running in time $t(n, m)$ and seed length $\rho(n, m)$ with $1 - \epsilon > 0$, to get a deterministic algorithm with running time $O(2^{\rho(n,m)} t(n, m))$, by simply running on all seeds.

Definition 3. (Hitting Set Transformation)[BE11] from $PIT(n', m')$ to $PIT(n, m)$ of probability $1 - \beta$ is a function $\mathcal{T} : \{0,1\}^\rho \times \mathbb{Z}^{n'} \to \mathbb{Z}^n$ which for every hitting set H of $PIT(n', m')$ with hitting probability $1 - \epsilon$

$$\{T(r, x) | r \in \{0,1\}^\rho, x \in H\}$$

is a hitting set with hitting probability $\geq (1 - \beta)(1 - \epsilon)$

Using a hitting set transformation, we will be able to transform the hitting set for $PIT(n', m')$ to hitting set for $PIT(n, m)$, and thus allows us to decrease the number of variables.

Theorem 1.2. *[BE11] Let $\mathcal{H}_{n',m'} : \{0,1\}^{\rho'} \to \mathbb{Z}^{n'}$ be a hitting set generator of $PIT(n',m')$ with hitting probability $1-\epsilon$ and evaluation time $t(n',m')$. Let $T_{n,m} : \{0,1\}^{\rho} \times \mathbb{Z}^{n'} \to \mathbb{Z}^{n}$ be a hitting set transformation from $PIT(n',m')$ to $PIT(n,m)$ with success probability $1-\beta$ which can be computed in time $g(n,m,s)$ where s is the maximal size of an element in the input hitting set. Then there are*

1. *a deterministic algorithm for $PIT(n,m)$ the running time of which is $O(2^{\rho+\rho'} \cdot g(n,m,t(n',m')) + t(n',m'))$*
2. *a randomized algorithm for $PIT(n,m)$ using $\rho(n,m)+\rho'(n',m')$ random bits with running time $O(g(n,m,t(n',m')) + t(n',m'))$ and success probability $(1-\beta)(1-\epsilon)$*

1.2.2 Sidon Sets

We made an attempt to use A set $\mathcal{S} := \{s_1, \ldots, s_n\} \subset \mathbb{Z}_{\geq 0}$ is said to be a Sidon set B_d if every element in the set $d\mathcal{S} := \{s_{i_1} + s_{i_2} + \ldots + s_{i_d} | \forall k \in [d], s_{i_k} \in \mathcal{S}\}$ are distinct upto rearrangements of summands.

We will use the Sidon sets used in BP20, given by the following theorem

Theorem 1.3. *[BP20, Theorem 2] For every n, d , there exists a poly(n,d) time constructible Sidon $B_{\leq d}$ set $S_{n,d} := \{b_1, \ldots, b_n\}$, with $b_n \leq \left\lceil (d+1)(2n\log n)^d \log(2n\log n) \right\rceil$*

They show that the setting $b_k := \left\lceil (d+1)(2n\log n)^d \log(p_k) \right\rceil$ gives us the required Sidon set. In this $p_1, \ldots, p_n$ are the first n primes. We can compute the first n primes using, the Sieve of Eratosthenes, which takes $O(n \log\log n)$ time.We can calculate the log also efficiently. We can calculate the powering in $poly(n,d)$ time as well using repeated squaring. Thus, the time required for construction of Sidon set is $poly(n,d)$).

2 Previous work

Assume the polynomial has degree $\delta_1, \ldots, \delta_n$ in the variables $x_1, \ldots, x_n$. The problem of PIT can be solved using the Schwartz-Zippel hitting set, but it requires $\sum_{i=1}^{n} \lceil \log(\delta_i + 1) \rceil + n \log n$ random bits, where the δ_i is degree of x_i in the polynomial. This was improved to $\sum_{i=1}^{n} \lceil \log(\delta_i + 1) \rceil$ for integer coefficient by Kao and Chen and for field of positive characteristic by Lewin and Vadhan.

The problem of Sparse PIT was given a solution using $\log(mn\delta/\epsilon)$ random bits by Klivans and Spielman, with error probability bounded by ϵ and algorithm running in time polynomial in $\log m, n, \log \delta, \log \frac{1}{\epsilon}$. Bläser and Engels, used the Descartes rule of signs, to give a deterministic algorithm with $\tilde{O}(m^3 n^3)$ running time. They also created a randomized algorithm that used $O(\log^2(mn/\epsilon))$ random bits. We will briefly describe the approach used by them for the randomized algorithm.

2.1 Using $\log^2(mn/\epsilon)$ random bits

The idea is to start with multivariate polynomials with a significantly smaller number of variables, $\lceil \log(q) \rceil + 1$, where $\log q$ is $\log(m) + \log(n)$. Then, use a transformation from $PIT(\lceil \log(q) \rceil + 1, m)$ to $PIT(n, m)$ using only number of random bits that is linear in the number of variables. We start with describing the Hitting set generator, we will use based on Descartes rule for multiple variables.

2.1.1 Hitting set for multiple variables

The hitting set generator will be very similar to the Schwartz-Zippel generator. The following lemma similar to PIT lemma, extending Descartes rule to multiple variables.

Lemma 2.1. *[BE11] Let $f \in \mathbb{R}_m[x_1, \ldots, x_n]$ be a non-zero polynomial. Let $z = (z_1, \ldots, z_n)$, where each z_i is drawn uniformly and independently from $Z \subseteq \mathbb{Z}$. Then*

$$Pr_{z \in Z^n}[f(z) = 0] \leq \frac{mn}{|Z|}$$

Proof. The proof is inductive similar to the PIT lemma.The base case, i.e. for $n = 1$, it follows from the Descartes' rule of signs.

Consider f as polynomial in $\mathbb{R}[x_2, \ldots, x_n][x_1]$, i.e. as $\sum_{i=1}^{m'} x_1^{i_1} f_i(x_2, \ldots, x_n)$ with $m' \leq m$.

$$\begin{aligned}
Pr_{z \in Z^n}[f(z) = 0] &= Pr_z[f(z_1, \ldots, z_n) = 0 | f_1(z_2, \ldots, z_n) = 0] \cdot Pr_z[f_1(z_2, \ldots, z_n) = 0] \\
&\quad + Pr_z[f(z_1, \ldots, z_n) = 0 | f_1(z_2, \ldots, z_n) \neq 0] \cdot Pr_z[f_1(z_2, \ldots, z_n) \neq 0] \\
&\leq Pr_z[f_1(z_2, \ldots, z_n) = 0] + Pr_z[f(z_1, \ldots, z_n) = 0 | f_1(z_2, \ldots, z_n) \neq 0] \\
&\leq \frac{m(n-1)}{|Z|} + Pr_z[f(z_1, \ldots, z_n) = 0 | f_1(z_2, \ldots, z_n) \neq 0] \qquad \text{induction hypothesis} \\
&\leq \frac{m(n-1)}{|Z|} + \frac{m}{|Z|} \qquad \text{Descartes rule} \\
&= \frac{mn}{|Z|}
\end{aligned}$$

□

This gives us the following $PIT(n, m)$ hitting set generator, with hitting probability $1 - \epsilon$, seed length $n \log(\frac{mn}{\epsilon})$ and the output size is $n \log(\frac{mn}{\epsilon})$.

Algorithm 1 Schwatrz-Zippel Generator

1: **Input:** Seed $r \in \{0,1\}^{n \log(mn/\epsilon)}$
2: Use r to choose $z \in \{1, \ldots, \frac{mn}{\epsilon}\}^n$
3: **Output:** z

2.1.2 Hitting set Transformation

We want a hitting set transformation from $PIT(s+1,m)$ to $PIT(n,m)$ with success probability $1-\beta$. Let $N = \frac{mn}{\beta}$ and q be a prime in $[N, 2N]$. Let vector a_i be $(1, i \mod q, i^2 \mod q, \ldots, i^{n-1} \mod q$. It can easily be seen that for any non-zero vector $b \in \mathbb{Z}^n$, $a_i \cdot b$ is zero for at most $n-1$ indices. Further more we set $a_{i,v} = \sum_{k=0}^{\lceil \log q \rceil} a_{i,v,k} 2^k$, i.e. $a_{i,v,k}$ is the binary expansion of $a_{i,v}$. Denote $\lceil \log q \rceil$ with s. Now we use the variable map, which takes $x_v \to y_0^{a_{i,v,0}} \ldots y_s^{a_{i,v,s}}$. Substituting $y_j = y^{2^j}$, gets us $x_v \to y^{a_{i,v}}$. The k^th monomial is represented as $c_k x_1^{b_{k,1}} x_2^{b_{k,2}} \ldots x_n^{b_{k,n}}$, and $b_k = (b_{k,1}, \ldots, b_{k,n})$. Consider the difference power vector of first and k^{th} monomial, $b_1 - b_k$. As we saw $a_i \cdot b$ is zero for atmost $n-1$ i's, for a random choice of i gives us a probability that $(n-1)/N$ that it will be zero. The $m-1$ difference vectors, at least one will be zero with probability $\leq (m-1)(n-1)/N \leq \beta$. Thus, with probability $\geq 1-\beta$ the first monomial isn't cancelled. Thus, we have a $1-\beta$ success probability Hitting set transformation from $PIT(s+1,m)$ to $PIT(n,m)$ using $\log(nm/\beta)$ random bits.

Algorithm 2 Transformation from $PIT(s+1,m)$ to $PIT(n,m)$

1: **Input:** Seed $r \in \{0,1\}^{\log(mn/\beta)}, y_0, \ldots, y_s$
2: $N \leftarrow \frac{mn}{\beta}$, $q \in [N, 2N]$ be a prime
3: Use r to choose $i \in [N]$ uniformly at random
4: Set $x_v \to y_0^{a_{i,v,0}} \ldots y_s^{a_{i,v,s}}$ for every $v \in [n]$
5: **Output:** $(x_1, \ldots, x_n)$

To obtain the prime, we use $\log^2 N$ pairwise independent bit stings, using $2 \log N$ random bits, and check each using the AKS algorithm in $O(\log^6 N)$ time. Using prime number theorem, we get each of this string will be prime with probability $\geq \frac{1}{\log(N)}$. Using the Chebyshev bound, atleast one of these will be prime using with prob $1 - o(1)$. This can be brought to $1-\epsilon$ using a random walk on an expander. Thus, it takes $O(\log^8 \frac{mn}{\beta})$ steps to find the prime. Computing a_i takes time $O(n \log^2 \frac{2mn}{\beta})$ using repeated squaring. Let $y = max(|y_0|, \ldots, |y_s|)$. Then computing x_v takes $O(ns \log y)$ time. Overall, the time is $poly(\log m, n, \log \frac{1}{\beta}, \log y)$. **Note:** This is the reason of not going to univaraite, as the calculation will take $O(m)$ time.

2.1.3 Final algorithm

We just use Algorithm 1 and 2 with Theorem 1.2. Set both error parameters to $\epsilon/2$. Algorithm works on $n = s+1 = \lceil \log q \rceil + 1$ variables. $\log y$ in our case will be $\leq \log N$. Therefore, the running time is $poly(\log m, n \log \frac{1}{\epsilon})$. The random bits used are $O((s+1) \log \frac{2m(s+1)}{\epsilon} + \log \frac{mn}{\epsilon})$ which is just $O(\log^2 \frac{mn}{\epsilon})$.

2.2 An optimal Hitting Set

In [BP20], they give a hitting set of size $\binom{n+d}{d}$ for d-degree polynomials in $\mathbb{F}[x_1, \ldots, x_n]$(denoted by $(\mathbb{F}, n, d)$ as follows:

Theorem 2.2. *[BP20, Theorem 10] If $\{0, 1, \ldots, d\} \subseteq \mathbb{F}$, then $\mathcal{H}_1 := \{(x_1, \ldots, x_n) \in \{0, 1, \ldots, d\}^n | x_1 + \ldots + x_n \leq d\}$, is a hitting set for $(\mathbb{F}, n, d)$.*

Proof. We will be to use induction on n. For $n = 1$, $\{0, \ldots, d\}$ is a hitting set as a univariate can have atmost d zeros. We have our induction hypothesis as "For any d, $\{(x_1, \ldots, x_{n-1}) \in \{0, 1, \ldots, d\}^{n-1} | x_1 + \ldots + x_{n-1} \leq d\}$, is a hitting set for $(\mathbb{F}, n-1, d)$". Now input $f \in (\mathbb{F}, n, d)$, we write it as a univariate in x_n as $\sum_{i=1}^{d'} P_i(x_1, \ldots, x_{n-1}) x_n^i$, where d' is maximum degree of x_n in f. If $f \neq 0$, then $P_{d'}(x_1, \ldots, x_{n-1}) \neq 0$, where $P_{d'}$ has degree $\leq d - d'$. By inductive hypothesis, there is a point in $\{(s_1, \ldots, s_{n-1}) \in \{0, 1, \ldots, d\}^{n-1}$, such that $P_{d'}(s_1, \ldots, s_{n-1}) \neq 0$ and $s_1 + \ldots + s_{n-1} \leq d - d'$. Fixing $x_i = s_i$, we get f is a univariate in x_n with degree d'. Choosing $x_n = s_n$ from $\{0, 1, \ldots, d'\}$ will give us a point at which f evaluates to non-zero. Also, $\sum_{i=1}^{n} s_i \leq d - d' + d' = d$. □

Our second approach to the problem is based on this approach.

3 Derandomizing Algorithm 2 using Sidon sets

In our new approach, we will present a deterministic hitting set transformation, using Sidon sets, which will reduce the problem to univariate case, where we will use algorithm 1, with $n = 1$.

3.1 Hitting Set Transformation using Sidon Sets

Given a polynomial $f \in \mathbb{R}_m[x_1, \ldots, x_n]$ with total degree $\leq \delta$, we will first construct a Sidon set $S_{n,\delta} = \{b_1, \ldots, b_n\}$ according to theorem 1.3. Then we simply transform each $x_i \to y^{b_i}$. As we know all summations of size $\leq \delta$ are distinct in the Sidon set, we have all monomials mapping to different powers of y and hence distinct monomials after applying the map. Thus we have the following Hitting set transformation

Algorithm 3 Transformation from $PIT(1, m)$ to $PIT(n, m)$

1: **Input:** y
2: Compute the Sidon set for n, δ as $S = \{b_1, \ldots, b_n\}$
3: Set $x_v \to y^{b_v}$ for every $v \in [n]$
4: **Output:** $(x_1, \ldots, x_n)$

Clearly, our algorithm doesn't require any random bits. The construction of S takes $poly(n, \delta)$ time as per theorem 1.3. The running time is $poly(\log y, \delta, n)$.

3.2 Final Algorithm

We combine Algorithm 1 and Algorithm 3 using theorem 1.2. Algorithm 1 runs for $n = 1$ and the error parameter ϵ. The random bits requirement of Algorithm 1 is $n \log \frac{mn}{\epsilon}$ bits, which is $\log \frac{m}{\epsilon}$ for $n = 1$. Therefore, Random bits used is $\log \frac{m}{\epsilon}$. Also, the output is $\log \frac{m}{\epsilon}$ bits, i.e. $\log y = \log \frac{m}{\epsilon}$. Thus, the run time of Algorithm 3 is $poly(\log m, \log \frac{1}{\epsilon}, n, \delta)$.

3.3 Drawbacks

We did manage to decrease the number of random bits required to $\log(\frac{m}{\epsilon})$, but the running time of our algorithm is polynomial in the total degree bound δ as computation of Sidon set requires this. We don't want dependence on degree, so this method is not successful.

An approach where a smaller Sidon set could work was if the separation between the degrees of monomials is huge, i.e. total degree might be large, but there is atleast one small degree monomial. It can be shown if the separation is larger than $d \log d$, where d is degree of small degree monomial, the sidon set with this degree bound can work. But the case where there is no small degree monomial cannot be solved using this.

We drew inspiration from the hitting set in Theorem 2.2 for following hitting set.

4 Optimal Hitting set

We will in this section give a hitting set for sparse polynomials, assuming we have the variables in increasing order of the number of different degrees in the polynomial. For easier analysis, we will need to introduce another parameter r, such that the number of different degrees of any x_i in the polynomial is $\leq r$. So We define the hitting set $\mathcal{S}(n, m, r)$ as follows:

Let $\mathcal{S}(n, m, r)$ be the set of all vectors $v = (v_1, \ldots, v_n)$ such that $v_n \in \{0, \ldots, min(m, r)\}$ and $(v_1, \ldots, v_{n-1}) \in \cup_{s=1}^{min(m,r)} \mathcal{S}(n, m/s, s)$. $\mathcal{S}(1, m, r)$ such that $m \neq r$ is empty set, as for univariate the number of different degrees is equal to the number of monomials. Also, $\mathcal{S}(n, 1, r)$ will be just 1^n.

Claim 4.1. *$\mathcal{S}(n, m, m)$ is a hitting set for m monomial polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ assuming we have the variables in increasing order of the number of different degrees in the polynomial.*

Proof. The proof is similar to the one in Theorem 2.2. For the base case of $n = 1$, we should have $m = r$ and $v_1 \in \{0, \ldots, m\}$ and we know from Descartes' principle that $f(x)$ with m monomials can have at most m zeros. Also, $\mathcal{S}(n, 1, r)$ will be just one element with all 1's. We have our induction hypothesis as for all m' and r, $\mathcal{S}(n-1, m', r)$ as hitting set for m' monomial polynomials in $\mathbb{F}[x_1, \ldots, x_{n-1}]$. Consider $f \in \mathbb{F}[x_1, \ldots, x_n]$ with m monomials. We have from assumption x_n as the variable with maximum distinct degrees in f with degrees from set D_j, $|D_j| = s$. We can write it as $f = \sum_{i \in D_j} P_i(x_1, \ldots, x_{n-1}) x_n^i$. Let i' be such that $P_{i'}$ has the maximum number of monomials among all P_i. Number of monomials in $P_{i'} \geq \left\lceil \frac{m}{s} \right\rceil$ as if all P_i had number of monomials $< \left\lceil \frac{m}{s} \right\rceil$, then total number of variables

$< m$. By induction hypothesis, we have $S(n-1, \lceil \frac{m}{s} \rceil)$ a hitting set for $P_{i'}$, which gives us an evaluation for $x_1, \ldots, x_{n-1}$ for which $P_{i'} \neq 0$, leaving f as a univariate in x_n with s monomials. As $s \leq m$, choosing $x_n \in \{0, \ldots, m\}$, will sure give us an evaluation from where f is not zero. Thus, $\mathcal{S}(n, m, m)$ is a hitting set for f. □

Lemma 4.2. *Size of the hitting set ($|\mathcal{S}(n, m, m)|$) is $O(m^4)$.*

Proof. From definition of $\mathcal{S}(n, m, r)$, we have

$$|\mathcal{S}(n, m, r)| = \sum_{s=1}^{min(m,r)} (s+1) * |\mathcal{S}(n-1, m/s, s)|$$

It is clear that $|\mathcal{S}(1, m, r)| = m$ and $|\mathcal{S}(n, 1, r)| = 1$.

By above relation, we can easily see that for $\mathcal{S}(n, m_1, r_1)$ and $\mathcal{S}(n, m_2, r_2)$ if $min(m_1, r_1) < min(m_2, r_2)$, then $|\mathcal{S}(n, m_1, r_1)| < |\mathcal{S}(n, m_2, r_2)|$.

Not only that, we can also see that if $min(m_1, r_1) < min(m_2, r_2)$, then the choice for net variable is smaller as well as possibilities for s is also smaller. Therefore, $\mathcal{S}(n, m_1, r_1) \subset \mathcal{S}(n, m_2, r_2)$.

Considering $\mathcal{S}(n, m, m)$, we have

$$|\mathcal{S}(n, m, m)| = \sum_{s=1}^{m} (s+1) * |\mathcal{S}(n-1, m/s, s)|$$

As $min(m/s, s) <= \sqrt{(}m)$ and the argument above, we have for all s, $|\mathcal{S}(n, m/s, s)| <= |\mathcal{S}(n-1, sqrt(m), sqrt(m)|$. Therefore, we have

$$|\mathcal{S}(n, m, m)| <= \sum_{s=1}^{m} (s+1) * |\mathcal{S}(n-1, sqrt(m), sqrt(m)| <= m^2 * |\mathcal{S}(n-1, sqrt(m), sqrt(m)|$$

Using the above recurrence, we can get $|\mathcal{S}(n, m, m)| <= m^{2+1+\ldots+2^{-(n-2)}} = O(m^4)$. Also, as $min(m/s, s) <= \sqrt{(}m)$, we have $\mathcal{S}(n, m/s, s) \subseteq \mathcal{S}(n, \sqrt{m}, \sqrt{m})$. Therefore, $S(n, m, m) = \{0, 1, \ldots, m\} \times S(n-1, \sqrt{m}, \sqrt{m})$. □

In the above analysis, we saw that the actual worst case for $S(n, m, m)$ happens when $s = \sqrt{m}$. All other cases can be said to be subsumed in this case, as $min(m, r) = min(m/s, s) \leq \sqrt{m}$. Thus, we give the following very simple hitting set.

Lemma 4.3. *Let $\mathcal{H}(n, m) := \{(v_1, \ldots, v_n) | v_i \in \{0, \ldots, m^{2^{i-n}}\}\}$. $\mathcal{H}(n, m)$ is a hitting set for polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ with m monomials and variables in increasing order of distinct degree.*

Proof. The proof is clear from the analysis above, where we showed $S(n, m, m) = \{0, 1, \ldots, m\} \times S(n-1, \sqrt{m}, \sqrt{m})$. The main idea as to why this happens is if the number of monomials of x_n is greater than $\sqrt{m}$, then there is a coefficient polynomial of some degree with less than $\sqrt{m}$ monomials, and if it is less than $\sqrt{m}$, by our assumption the rest of variables. It is easy to see that, $|\mathcal{H}(n, m)| = O(m^2)$ and it can be computed in $poly(m)$ time. □

4.1 Randomized Algorithm

Using the above, we can give the following randomized algorithm for polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ with m monomials and variables in increasing order of distinct degree.

We choose x_n randomly uniformly from $\{0, 1, \ldots, \left\lceil \frac{m}{\epsilon^{1/n}} \right\rceil$, and x_{n-1} from $\{0, 1, \ldots, \left\lceil \frac{\sqrt{m}}{\epsilon^{1/n}} \right\rceil$, and so on. That is, we choose x_n from $\{0, 1, \ldots, \left\lceil \frac{m^{2^{i-n}}}{\epsilon^{1/n}} \right\rceil$.

The probability that we get a non-zero evaluation for a non-zero polynomial will be ϵ. The number of random bits used will be

$$\sum_{i=1}^{n} \log \left(\frac{m^{2^{i-n}}}{\epsilon^{1/n}} \right) \leq 2 \log(m) + \log \left(\frac{1}{\epsilon} \right)$$

Thus, we only need $O(\log(m))$ bits to check if it's an identity or not.

4.2 Drawbacks

The assumption we made about the variable order is too strong and of course cannot be used for general polynomials in $\mathbb{F}[x_1, \ldots, x_n]$. We are not sure yet on how to remove this dependence on variable order.

We believe that the variable transformation of 2 should mix all the variables, with high probability that the number of distinct degree of any variable is almost the same. But showing this for any general polynomial is still open.

Also, this approach cannot be used for finite fields, as Descartes' principle doesn't work in them.

References

[BE11] Markus Blaser and Christian Engels. Randomness efficient testing of sparse black box identities of unbounded degree over the reals. *28th Symposium on Theoretical Aspects of Computer Science (STACS'11).*, 2011.

[BP20] Markus Blaser and Anurag Pandey. Polynomial identity testing for low degree polynomials with optimal randomness. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, 2020.

[Sax09] Nitin Saxena. Progress on polynomial identity testing. *Bulletin of the EATCS, 99:49–79*, 2009.

[Sax14] Nitin Saxena. Progress on polynomial identity testing- ii. *In Perspectives in Computational Complexity, volume 26 of Progress in Computer Science and Applied Logic, pages 131–146. Springer International Publishing*, 2014.

[Sch80] Jacob T Schwart. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 1980.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science: Vol. 5*, 2010.