Testing Polynomial Equivalence by Scaling Matrices

Markus Bläser[1(✉)], B.V. Raghavendra Rao[2], and Jayalal Sarma[2]

[1] Saarland Informatics Campus, Saarland University, Saarbrücken, Germany
mblaeser@cs.uni-saarland.de
[2] IIT Madras, Chennai, India
{bvrr,jayalal}@cse.iitm.ac.in

Abstract. In this paper we study the polynomial equivalence problem: test if two given polynomials f and g are equivalent under a non-singular linear transformation of variables.

We begin by showing that the more general problem of testing whether f can be obtained from g by an arbitrary (not necessarily invertible) linear transformation of the variables is equivalent to the existential theory over the reals. This strengthens an NP-hardness result by Kayal [9].

Two n-variate polynomials f and g are said to be equivalent up to scaling if there are scalars $a_1, \ldots, a_n \in \mathbb{F} \setminus \{0\}$ such that $f(a_1x_1, \ldots, a_nx_n) = g(x_1, \ldots, x_n)$. Testing whether two polynomials are equivalent by scaling matrices is a special case of the polynomial equivalence problem and is harder than the polynomial identity testing problem.

As our main result, we obtain a randomized polynomial time algorithm for testing if two polynomials are equivalent up to a scaling of variables with black-box access to polynomials f and g over the real numbers.

An essential ingredient to our algorithm is a randomized polynomial time algorithm that given a polynomial as a black box obtains coefficients and degree vectors of a maximal set of monomials whose degree vectors are linearly independent. This algorithm might be of independent interest. It also works over finite fields, provided their size is large enough to perform polynomial interpolation.

1 Introduction

The polynomial equivalence problem (PolyEq), i.e., testing if two given polynomials are equivalent under a non-singular change of coordinates is one of the fundamental computational tasks related to polynomials. More precisely, two polynomials $p(x_1, x_2, \cdots, x_n)$ and $q(x_1, x_2, \cdots, x_n)$ are said to be *linearly equivalent* if there is an invertible linear transformation, A such that for $y_i = \sum_j A_{ij}x_j$, $p(y_1, y_2, \ldots y_n) = q(x_1, \ldots x_n)$. When A is not restricted to be invertible, the problem is referred to as polynomial projection problem (PolyProj).

Indeed, observing that only a polynomial with all coefficients equal to zero can be equivalent to the zero polynomial, PolyEq is a generalization of the well studied polynomial identity testing problem which has close connections to

R. Klasing and M. Zeitoun (Eds.): FCT 2017, LNCS 10472, pp. 111–122, 2017.
DOI: 10.1007/978-3-662-55751-8_10

arithmetic circuit lower bounds [7]. Further, since a non-singular change of coordinates is one of the fundamental geometric primitives, PolyEq is of primary importance to computational algebraic geometry.

Saxena [14] showed that the graph isomorphism problem is polynomial time many one reducible to the case of PolyEq where the polynomials are of degree three. Thus, the problem simultaneously generalizes the graph isomorphism problem and the polynomial identity testing problem. Further, if the change of coordinates (the matrix A) is not restricted to be invertible, that is, A need not be invertible, then the problem PolyProj is NP-hard under polynomial time many-one reductions [9]. We first strengthen this hardness result over $\mathbb{R}$ (in fact, also over any integral domain) as follows:

Theorem 1. *Given two sparse polynomials $f, g \in \mathbb{R}[x_1, \ldots, x_n]$, deciding whether there is a matrix A such that $f(x) = g(Ax)$ is as hard as the existential theory over the reals.*

Both PolyProj and PolyEq can be solved in polynomial space over $\mathbb{R}$ and $\mathbb{C}$ in the Blum-Shub-Smale model of algebraic computation [4] using existential theories over these fields. Since the best upper bound for existential theory over reals is PSPACE, the above hardness result indicates that the PolyProj is possibly harder than just being NP-hard. However, the hardness result does not apply to when A is restricted to be non-singular, and the complexity of the PolyEq problem remains elusive. Over finite fields the problem is in NP $\cap$ co-AM [17]. However, over the field of rational numbers, it is not known if the problem is decidable [14].

Given the lack of progress in the general problem PolyEq, it is natural to solve special instances of the problem. A natural restriction is to study the problem when the input polynomials are restricted. When both polynomials are restricted to quadratic forms (homogeneous degree 2 polynomials), we know about the structure of equivalent polynomials and this also leads to a polynomial time algorithm for testing equivalence of such polynomials (see Witt's equivalence theorem [12]). As indicated above the problem already becomes harder when the degree is allowed to be even three. Agrawal and Saxena [1] showed that ring isomorphism testing problem, reduces to the PolyEq problem when the degree of the polynomials is at most three. [1] Patarin [13] even designed a cryptosystem which assumes the hardness of the degree bounded (by three) version of the problem to prove security guarantees.

Instead of simultaneously restricting both of the polynomials in the problem, it is even interesting to study the problem when one of the polynomials is fixed to be a well-structured family and the other polynomial is allowed to be arbitrary. In this direction, Kayal [8] obtained randomized polynomial time algorithms to test if a given polynomial (as a black-box) is equivalent to either an elementary symmetric polynomial or to the power symmetric polynomial of a given degree.

[1] For a (partial) converse, they [1] also showed that deciding equivalence of degree k polynomials having n variables over $\mathbb{F}_q$ (such that k and $q-1$ are co-prime), can be reduced to the ring isomorphism problem.

Further, Kayal [9] obtained similar algorithms when one of the polynomials is fixed to be either the permanent polynomial or the determinant polynomial. More recently, Kayal *et al.* [10] obtained randomized polynomial time algorithm for PolyEq when one of the polynomials is the iterated matrix multiplication polynomial.

Another possibility of obtaining restrictions of PolyEq is by restricting the structure of change of coordinates. Grigoriev [6] considered the problem of testing equivalence of polynomials under Taylor shifts[2]: given two polynomials f and g, are there $a_1, \ldots, a_n \in \mathbb{K}$ such that $f(x_1 + a_1, \ldots, x_n + a_n) = g$? Grigoriev obtained a polynomial time algorithm to the problem when the polynomial is given in the sparse representation. The algorithm is deterministic polynomial time if $\mathbb{K}$ is algebraically closed, randomized polynomial time if $\mathbb{K} = \mathbb{Q}$ and quantum polynomial time if $\mathbb{K}$ is finite. More recently, Dvir et al. [5] showed that the shift equivalence problem is polynomial time equivalent to the polynomial identity testing problem in the black-box as well as non-black-box setting.

In this paper, we restrict the structure of the matrices under which the equivalence is tested, to *diagonal matrices*. We obtain a randomized polynomial time algorithm for testing if two polynomials are equivalent up to a scaling of variables with black-box access to polynomials f and g. More precisely, we prove the following theorem:

Theorem 2 (Main). *Given $f, g \in \mathbb{R}[x_1, \ldots, x_n]$ as a blackbox, there exists a randomized algorithm that tests if there is an invertible diagonal matrix A such that $f(X) = g(AX)$. The algorithm runs in time* $\mathsf{poly}(n, \Delta, L)$*, where the degree of f and g is bounded by Δ and all of the coefficients of f and g can be represented by at most L bits.*

As mentioned above, Kayal [9] designed randomized polynomial time algorithms for testing equivalence if one of the polynomials comes from a well-structured family of polynomials like the permanent family or determinant family. These algorithms follow the following general scheme: First, the general problem is reduced to permutation and scaling equivalence testing by studying the Lie algebra of the input polynomial. Then permutation and scaling equivalence testing is reduced to scaling equivalence testing. Our result shows that this last step can always be done in randomized polynomial time, even when one of the polynomials does not come from a nice family but is arbitrary. Thus, the hardness of PolyEq most likely lies in the first step, since we need a large enough Lie algebra to make the approach work. The Lie algebra of a random polynomial is trivial [9].

As an ingredient to our proof of Theorem 2, we obtain a randomized polynomial time algorithm that given a polynomial as a black box obtains coefficients of a maximal set of monomials whose degree vectors are linearly independent, this might be of independent interest.

[2] which is strictly speaking not a (homogeneous) linear change of coordinates.

Theorem 3. *There is a randomized algorithm, that given a polynomial* $f \in \mathbb{R}[x_1, \ldots, x_n]$ *by black box access outputs a maximal collection of*

$$\{(m, \alpha) \mid \alpha \neq 0, \text{ and } \alpha \text{ is the coefficient of the monomial } m \text{ in } f\}$$

such that the set of degree vectors is linearly independent over $\mathbb{R}$*. The running time is polynomial in the degree* Δ *of* f*, the number of variables* n*, and the bit size* L *of the representation of the coefficients.*

We remark that the latter algorithm also works over finite fields provided they are large enough. Here large enough means $p(\Delta, n)$ for some polynomial p for small degree.

2 Preliminaries

In this section, we fix the notations that we use throughout the paper.

For a monomial $m = x_1^{d_1} \cdot \cdots \cdot x_n^{d_n}$ let $\mathrm{Deg}(m) = (d_1, \ldots, d_n)$ denote the degree vector of m. For a polynomial $f \in \mathbb{K}[x_1, \ldots, x_n]$, let $\mathsf{Mon(f)}$ denote the set of monomials that have non-zero coefficients in f. A $\mathsf{degree\text{-}basis}$ for f is a maximal collection $S = \{(m, \alpha) \mid \alpha \neq 0$ is the coefficient of monomial m in $f\}$ such that the set $\{\mathrm{Deg}(m) \mid (m, \alpha) \in S, \text{ for some } \alpha \neq 0\}$ is linearly independent over $\mathbb{Q}$ or equivalently $\mathbb{R}$.

Isolating Monomials: Klivans and Spielman [11] obtained a randomized polynomial time algorithm that tests if a polynomial given as a black-box is identically zero or not. Their algorithm involves a randomized polynomial time algorithm that isolates a monomial in the given polynomial if it is not identically zero. We state the result below:

Theorem 4 (Klivans and Spielman [11]). *There is a probabilistic algorithm that given a non-zero polynomial* $f \in K[x_1, \ldots, x_n]$ *(by blackbox access) outputs a monomial* m *of* f*, its degree vector* $\mathrm{Deg}(m)$ *and its coefficient* α *in* f *with probability* $\geq 1 - \epsilon$ *in time polynomial in* n*,* Δ*, and* $1/\epsilon$*.*

Theorem 4 is going to be a building block for our proof of Theorem 3. We need a bit more insight into the proof of Theorem 4 listed as follows:

- The algorithm in [11] first replaces the variables x_i by y^{a_i} where the a_i are numbers with $O(\log(n\Delta/\epsilon))$ bits. We get a new univariate polynomial $\hat{f}$. Monomials of f get mapped to monomials in $\hat{f}$ and are grouped together. The substitution has the property that with probability $\geq 1 - \epsilon$, there is only one monomial of f getting mapped to the (non-zero) monomial of $\hat{f}$ having minimum degree. Since we have only black-box access to f, this substitution is only conceptual and is simulated when later on plugging values into the blackbox.

- Then we interpolate $\hat{f}$ by evaluating it at $\mathsf{poly}(n, \Delta, 1/\epsilon)$ many values. That is, we plug in values v^{a_i} for each x_i for polynomially many values v. The lowest nonzero coefficient of $\hat{f}$ is also a coefficient of f, however, we do not know the degree pattern of this monomial (yet).
- Then we modify the substituion in the first step by replacing x_1 by $2y^{a_1}$. In this way, the lowest nonzero coefficient of $\hat{f}$ will get multiplied by 2^{d_1} where d_1 is the x_1-degree of the unique monomial in f that is mapped to the lowest degree monomial of $\hat{f}$. Doing this for all x_i, we can also extract the degree vector of the monomial.

Tensors and Tensor Rank: We also fix notations and state the preliminary results about tensors that we need in the paper. We call $t = (t_{i,j,\ell}) \in \mathbb{R}^{n\times n\times n}$ a tensor. A rank one tensor is a tensor that can we written as $u \otimes v \otimes w$ with $u, v, w \in \mathbb{R}^n$. The minimum number of such rank-one-tensors such that t can be written as the sum of them is called the rank of t. For an introduction to this problem, the reader is referred to [2,3].

With a tensor t, we can associate the trilinear form

$$F(x, y, z) = \sum_{i,j,\ell=1}^{n} t_{i,j,\ell} x_i y_j z_\ell.$$

The so called unit tensor $e_r \in \mathbb{R}^{r\times r\times r}$ is given by the trilinear form

$$E_r = \sum_{i=1}^{r} x_i y_i z_i$$

The following fact is well known :

Proposition 1 (see [3]). *Let $t = (t_{i,j,\ell}) \in \mathbb{R}^{n\times n\times n}$ be a tensor. The tensor rank of t is bounded by r if and only if there are matrices $S, T, U \in \mathbb{R}^{r\times n}$ such that*

$$F(x, y, z) = E_r(Sx, Ty, Uz).$$

3 Hardness of the PolyProj Problem

Testing whether there is an arbitrary matrix A such that $f(x) = g(Ax)$ is a hard problem. In this section, we prove Theorem 1. As mentioned in the introduction, this improves the hardness result shown in [9].

Theorem 1. *Given two polynomials $f, g \in \mathbb{R}[x_1, \ldots, x_n]$, as a list of monomials and their coefficients, deciding whether there is a matrix A (not necessarily non-singular) such that $f(x) = g(Ax)$ is as hard as the existential theory over the reals.*

Proof. We proceed by reducing the tensor rank problem to PolyProj problem. Given a tensor $t = (t_{i,j,\ell}) \in \mathbb{R}^{n \times n \times n}$, we observe that Proposition 1 suggests two polynomials of the form $f(x) = g(Ax)$. However, there are two issues. Firstly, we have three sets of variables and secondly, the matrices S, T, and U are not square matrices. The second problem is easy to circumvent. We consider F as a polynomial in the variables $x_1, \ldots, x_r$, $y_1, \ldots, y_r$, and $z_1, \ldots, z_r$ instead and extend the matrices S, T, and U by zero rows.

To address the first problem, we modify the problem we reduce from. A tensor is called *symmetric*, if $t_{i,j,\ell} = t_{i,\ell,j} = \ldots$ for all six permutations of the indices. In the same way as for general tensors, we can associate a trilinear form with symmetric tensors, too:

$$F'(x) = \sum_{i,j,\ell=1}^{n} t_{i,j,\ell} x_i x_j x_\ell.$$

Definition 1 (Symmetric Rank). *The symmetric or Waring rank of a symmetric tensor t is the smallest r such that there is an $r \times n$ matrix A with $F'(x) = E'_r(Ax)$ where*

$$E'_r = \sum_{i=1}^{r} x_i^3.$$

Shitov [16] recently proved that the problem of deciding whether a symmetric tensor t has symmetric rank r is as hard as the existential theory over the underlying ground field. The same is true for the ordinary tensor rank. (Independently, Schaefer and Stefankovic proved a similar result [15], but only for the tensor rank.) □

We remark that, since Shitov's result [16] holds over any integral domain, the above theorem is also true for any integral domain.

4 Extracting a degree-basis of a Polynomial

In this section we obtain a randomized polynomial time algorithm that given a polynomial f as a black-box computes a degree-basis for f. We re-state Theorem 3 for readability:

Theorem 3. *There is a randomized algorithm, that given a polynomial $f \in \mathbb{R}[x_1, \ldots, x_n]$ by black box access outputs a maximal collection of*

$$\{(m, \alpha) \mid \alpha \neq 0, \text{ and } \alpha \text{ is the coefficient of the monomial } m \text{ in } f\}$$

such that the set of degree vectors is linearly independent over $\mathbb{R}$. The running time is polynomial in the degree Δ of f, the number of variables n, and the bit size L of the representation of the coefficients.

Proof. Algorithm 1 is our proposed algorithm. It starts with extracting a first monomial using the algorithm by Klivans and Spielman (Theorem 4). Then it proceeds iteratively and extends the set one by one.

Now assume we have already extracted monomials $m_1, \ldots, m_t$ of f such that the corresponding degree vectors $v_1, \ldots, v_t$ are linearly independent. Let $t < n$. We describe a procedure that finds a new monomial m_{t+1} such that its degree vector v_{t+1} is not contained in the span of $v_1, \ldots, v_t$ or reports that there does not exists such a v_{t+1}.

Let $v_1, \ldots, v_{\hat{t}}$ be a basis of the $\mathbb{R}$-vector space spanned by all degree vectors of f, that is, we extend $v_1, \ldots, v_t$ to a basis. Let p be a prime such that $v_1, \ldots, v_{\hat{t}}$ stay linearly independent over $\mathbb{F}_p$. By the Hadamard bound for the determinant, the matrix formed by $v_1, \ldots, v_t$ has a non vanishing minor whose absolute value is bounded by $(\Delta n)^n$. And by the prime number theorem, a prime number of size $O(n\mathsf{polylog}(\Delta n))$ will have the property stated above with high probability. Note that we only know the vectors $v_1, \ldots, v_t$ so far, but since we simply choose p uniformly at random, we do not need to know the vectors $v_1, \ldots, v_{\hat{t}}$ at all.

Let $u_1, \ldots, u_{n-t}$ be linearly independent vectors such that $v_i u_j = 0$ over $\mathbb{F}_p$ for all $1 \leq i \leq t$, $1 \leq j \leq n-t$, where $v_i u_j$ denotes the standard scalar product. If w is a vector not contained in the span of $v_1, \ldots, v_t$, then there is a j such that $w u_j \neq 0$ over $\mathbb{F}_p$. Consider the substitution $x_i \to y^{u_{j,i}} x_i$, $1 \leq i \leq n$, where $u_{j,i}$ are the entries of u_j. This substitution maps every monomial m of f to some monomial of the form $y^d m$. Let f_j be the resulting polynomial.

By construction, we have:

Lemma 1. *1. The degree of f_j is bounded by $O(\Delta n\mathsf{polylog}\,(\Delta n))$ for all j.*
2. If a monomial m is contained in the span of $v_1, \ldots, v_t$, then for every j, $p|d$ where $y^d m$ is the image of m in f_j.
3. If a monomial m is not contained in the span of $v_1, \ldots, v_t$, then there is a j such that $p \not| d$ where $y^d m$ is the image of m in f_j. □

We continue with the proof of the theorem. The strategy is now clear: We treat each f_j as a univariate polynomial in y with coefficients from $K[x_1, \ldots, x_n]$. Then we use the algorithm from Theorem 4 to extract a monomial from the coefficient polynomial of a power y^d with $p \not| d$. If we find a monomial then we set v_{t+1} to be its degree vector. If we do not find such a monomial, then $v_1, \ldots, v_t$ is a maximal linearly independent set.

Let $f_j = \sum_{d=0}^{\Delta_j} g_d \cdot y^d$. To be able to apply Theorem 4, we have to provide blackbox access to the g_d's but we have only blackbox access to f. We simulate this as follows:

- Given blackbox access to f, it is easy to simulate blackbox access to f_j.
- Now assume we want to evaluate g_d at a point $\xi \in K^n$.
- We evaluate f_j at the points $(\xi, \alpha_i) \in K^{n+1}$, $0 \leq i \leq \Delta_j$, where the α_i are pairwise distinct, that is, we compute values $f_j(\xi, \alpha_i) = \sum_{d=0}^{\Delta_j} g_d(\xi)\alpha_i^d$. From these values, we interpolate the coefficients of f_j, viewed as a univariate polynomial in y. The coefficient of y^d is $g_d(\xi)$.

It is clear from construction that Algorithm 1 returns the correct result if no errors occur in the randomized computations. Thus, if we make every error

Algorithm 1. Gen-Mon(f)

Input: Black box access to polynomial $f \in K[x_1, \ldots, x_n]$
Output: A degree-basis for f

$S \leftarrow \emptyset$.
$t \leftarrow 1$
if f is not identically 0 **then**
  Extract a monomial m_1 of f (using Theorem 4) with coefficient α_1.
  Let v_1 be the degree vector.
  $S \leftarrow S \cup \{(m_1, \alpha_1)\}$.
  while TRUE do
    Randomly choose a prime p of size $O(n\mathsf{polylog}(\Delta n))$.
    Compute linearly independent vectors $u_1, \ldots, u_{n-t}$ such that $v_i u_j = 0$ over $\mathbb{F}_p$ for all $1 \leq i \leq t$, $1 \leq j \leq n-t$.
    Let $f_j(x_1, \ldots, x_n) = f(x_1 y^{u_{j,1}}, \ldots, x_n y^{u_{j,n}})$, $1 \leq j \leq n-t$.
    Write $f_j(x, y) = \sum_{d=0}^{\Delta_j} h_{j,d} y^d$.
    Try to extract a monomial of every $h_{j,d}$ using Theorem 4, $1 \leq j \leq n-t$, $p \not| d$.
    Let m_{t+1} be the first such monomial found, α_{t+1} be its coefficient and let v_{t+1} be its degree vector. Set $S = S \cup \{(m_{t+1}, \alpha_{t+1})\}$.
    If no such monomial m_{t+1} is found, then output S and HALT.
  end while
end if

probability of every randomized subroutine polynomially small in Algorithm 1, then by the union bound, it will compute the correct result with high probability. For the running time observe that the while loop is executed at most $n-1$ times. The degrees Δ_j are bounded by $\mathsf{poly}(n, \Delta)$ by the bound on p. All numbers occurring as coefficients have length bounded by $\mathsf{poly}(n, \Delta, L)$, since the degrees of all polynomials are bounded by $\mathsf{poly}(n, \Delta)$. □

5 Testing for Equivalence by Scaling

Let $f(X)$ and $g(X)$ be polynomials in $\mathbb{R}[x_1, \ldots, x_n]$ given by black box access. We assume that the degree of f and g is bounded by Δ and that all coefficients of f and g can be represented by at most L bits.

Assume there is a non-singular diagonal matrix A such that $f(X) = g(AX)$. Let $(a_1, \ldots, a_n)$ denote the entries of A on the diagonal. Clearly, if $f(X) = g(AX)$ with A diagonal, f and g should have the same set of monomials. We first treat the case that the degree basis has maximum cardinality n.

Lemma 2. *Let $S = \{(m_i, \alpha_i) \mid 1 \leq i \leq n\}$ be a* degree-basis *of f. If $f(X) = g(AX)$ for a non-singular diagonal matrix A, then such an A can be computed deterministically in time polynomial in n, Δ and L, where the a_i are represented by polynomial size expressions with roots.*

Proof. Let $\alpha_i \neq 0$ and $\beta_i \neq 0$ be the coefficient of m_i in f and g, respectively. Suppose $f(X) = g(AX)$ for some non-singular diagonal matrix A with diagonal $(a_1, \ldots, a_n)$. We have n polynomial equations

$$\alpha_i = \beta_i \prod_{j=1}^{n} a_j^{d_{i,j}}$$

where $v_i = \mathrm{Deg}(m_i) =: (d_{i,1}, \ldots, d_{i,n})$. Taking logarithms on both sides, we have

$$\log \alpha_i = \log \beta_i + \sum_{j=1}^{n} d_{i,j} \log a_j.$$

(Formally, you have to choose an appropriate branch of the complex algorithm, since the α_i or β_i can be negative. Since we exponentiate again in the end, the actual choice does not matter.)

Since the vectors $v_1, \ldots, v_n$ are linearly independent over $\mathbb{R}$, there are unique values for $\log a_1, \ldots, \log a_n$ satisfying the above equations. Now $a_1, \ldots, a_n$ can be obtained by inverse logarithms. This proves the uniqueness of $a_1, \ldots, a_n$.

Let $D = (d_{i,j})$. Then the a_i are given by

$$\begin{pmatrix} \log a_1 \\ \vdots \\ \log a_n \end{pmatrix} = D^{-1} \begin{pmatrix} \log(\alpha_1/\beta_1) \\ \vdots \\ \log(\alpha_n/\beta_n) \end{pmatrix}.$$

So each $a_i = \prod_{j=1}^{n} (\alpha_j/\beta_j)^{\bar{d}_{i,j}}$ where $D^{-1} = (\bar{d}_{i,j})$. □

When the set of degree vectors of f has cardinality less than n, we can still use Algorithm 1 to compute a linearly independent set of degree vectors $v_1, \ldots, v_t$ of maximal size. Let $v_i = (d_{i,1}, \ldots, d_{i,n})$, $1 \le i \le t$. Let $m_1, \ldots, m_t$ be the corresponding monomials with coefficients $\alpha_1, \ldots, \alpha_t$. Let $\beta_1, \ldots, \beta_t$ be the corresponding coefficients of the monomials of g. From these values, we can set up a system of equations as in Lemma 2, however, this time there might be more than one solution. The next lemma states that it actually does not matter which of these solutions we choose:

Lemma 3. *Let $a_1, \ldots, a_n$ be any solution to*

$$\log \alpha_i = \log \beta_i + \sum_{j=1}^{n} d_{i,j} \log a_j, \qquad 1 \le i \le t,$$

and let A be the corresponding diagonal matrix. Let $r(x)$ be a monomial with coefficient δ and degree vector $u = (e_1, \ldots, e_n)$ contained in the linear span of $v_1, \ldots, v_t$, i.e., $u = \lambda_1 v_1 + \cdots + \lambda_t v_t$. Then the coefficient of $r(Ax)$ is

$$\delta \cdot \left(\frac{\alpha_1}{\beta_1}\right)^{\lambda_1} \cdots \left(\frac{\alpha_t}{\beta_t}\right)^{\lambda_t},$$

in particular, it is independent of the chosen solution for $a_1, \ldots, a_n$.

Algorithm 2. Scaling equivalence test

Input: Black box access to polynomials $f, g \in K[x_1, \ldots, x_n]$
the degree vectors of f have full rank
Output: Nonsingular diagonal matrix A with $f(x) = g(Ax)$ if such an A exists
Apply Gen-Mon with polynomial f as the black-box to get a set S.
Apply Gen-Mon to g using the *same* random bits as above to get a set S'.
If the monomials in the set S is not the same as S' then Reject.
Solve for the entries of A using Lemma 2 (choosing any solution if there is more than one).
Accept if and only if $f(x) - g(Ax)$ is identically zero.

Proof. Let $u = \sum_{i=1}^{t} \lambda_i v_i$. Now

$$\begin{aligned} r(Ax) &= \delta \cdot (a_1 x_1)^{e_1} \cdots (a_n x_n)^{e_n} \\ &= \delta \cdot a_1^{e_1} \cdots a_n^{e_n} \cdot x_1^{e_1} \cdots x_n^{e_n} \\ &= \delta \cdot a_1^{\sum_{i=1}^{t} \lambda_i d_{i,1}} \cdots a_n^{\sum_{i=1}^{t} \lambda_i d_{i,n}} \cdot x_1^{e_1} \cdots x_n^{e_n} \\ &= \delta \cdot (a_1^{d_{1,1}} \cdots a_n^{d_{1,n}})^{\lambda_1} \cdots (a_1^{d_{t,1}} \cdots a_n^{d_{t,n}})^{\lambda_t} \cdot x_1^{e_1} \cdots x_n^{e_n} \\ &= \delta \cdot \left(\frac{\alpha_1}{\beta_1}\right)^{\lambda_1} \cdots \left(\frac{\alpha_t}{\beta_t}\right)^{\lambda_t} \cdot x_1^{e_1} \cdots x_n^{e_n}. \end{aligned}$$

□

Thus for testing if there is a diagonal matrix A with $f(X) = g(AX)$, it is enough to compute the non-zero coefficients of at most n monomials $m_1, \ldots, m_n$ in f and g the degree vectors of which are linearly independent.

We complete the correctness of Algorithm 2 in the following Theorem, which in turn completes the proof of Theorem 2.

Theorem 5. *Algorithm 2 returns correct the correct answer with high probability. It runs in time polynomial in Δ, n and L.*

Proof. The algorithm calls two times the routine Get-Mon and makes one call to a polynomial identity test. By making the error probabilities of these calls small enough, we can controll the error probability of Algorithm 2 by the union bound.

Now we need to argue that if the polynomials f and g have the same set of monomials, then the calls for Gen-Mon(f) and Gen-Mon(g) with same set of random bits (i.e., by re-using the random bits) will result in sets S and S' with the same set of monomials.

Consider parallel runs of Get-Mon with f and g as inputs respectively such that they use a common random string say R. If f and g have the same set of monomials, then clearly $h_{j,d}^{f}$ and $h_{j,d}^{g}$ both have the same set of monomials at every iteration of the two parallel instances of the algorithm.

Since the randomness is only used in the exponents, if f and g have the same set of monomials, then the algorithm applied to f and to g with the *same*

random bits will result in the same set of degree vectors. Therefore, we get the appropriate coefficients of g.

Once we have the coefficients, we can find the the entries of a scaling matrix using the set of equations in Lemma 2. By Lemma 3, it does not matter which solution we choose. The algorithm is correct by construction. Each single step of the algorithm can be performed in polynomial time. □

Acknowledgments. The work was supported by the Indo-German Max Planck Center for Computer Science.

References

1. Agrawal, M., Saxena, N.: Equivalence of $\mathbb{F}$-algebras and cubic forms. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 115–126. Springer, Heidelberg (2006). doi:10.1007/11672142_8
2. Bläser, M.: Explicit tensors. In: Agrawal, M., Arvind, V. (eds.) Perspectives in Computational Complexity. PCSAL, vol. 26, pp. 117–130. Springer, Cham (2014). doi:10.1007/978-3-319-05446-9_6
3. Bürgisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic Complexity Theory. Grundlehren der mathematischen Wissenschaften, vol. 315. Springer, Heidelberg (1997)
4. Canny, J.F.: Some algebraic and geometric computations in PSPACE. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, Chicago, Illinois, USA, 2–4 May 1988, pp. 460–467 (1988)
5. Dvir, Z., Oliveira, R.M., Shpilka, A.: Testing equivalence of polynomials under shifts. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014. LNCS, vol. 8572, pp. 417–428. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43948-7_35
6. Grigoriev, D.: Testing shift-equivalence of polynomials by deterministic, probabilistic and quantum machines. Theor. Comput. Sci. **180**(1), 217–228 (1997)
7. Kabanets, V., Impagliazzo, R.: Derandomizing polynomial identity tests means proving circuit lower bounds. Comput. Complex. **13**(1/2), 1–46 (2004)
8. Kayal, N.: Efficient algorithms for some special cases of the polynomial equivalence problem. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, 23–25 January 2011, pp. 1409–1421 (2011)
9. Kayal, N.: Affine projections of polynomials: extended abstract. In: Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, 19–22 May 2012, pp. 643–662 (2012)
10. Kayal, N., Nair, V., Saha, C., Tavenas, S.: Reconstruction of full rank algebraic branching programs. In: 32nd IEEE Conference on Computational Complexity (CCC) (2017, to appear)
11. Klivans, A.R., Spielman, D.A.: Randomness efficient identity testing of multivariate polynomials. In: Proceedings on 33rd Annual ACM Symposium on Theory of Computing, 6–8 July 2001, Heraklion, Crete, Greece, pp. 216–223 (2001)
12. Lang, S.: Algebra. Springer, Heidelberg (2002)
13. Patarin, J.: Hidden fields equations (HFE) and Isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996). doi:10.1007/3-540-68339-9_4

14. Saxena, N.: Morphisms of rings and applications to complexity. Ph.D. thesis, Department of Computer Science, Indian Institute of Technology, Kanpur, India (2006)
15. Schaefer, M., Stefankovic, D.: The complexity of tensor rank. CoRR, abs/1612.04338 (2016)
16. Shitov, Y.: How hard is tensor rank? CoRR, abs/1611.01559 (2016)
17. Thierauf, T.: The isomorphism problem for read-once branching programs and arithmetic circuits. Chic. J. Theor. Comput. Sci. **1998**(1) (1998)