Revisit Sparse Polynomial Interpolation Based on Randomized Kronecker Substitution

Qiao-Long Huang[1,2] and Xiao-Shan Gao[1(✉)]

[1] KLMM, UCAS, Academy of Mathematics and Systems Science Chinese Academy of Sciences, Beijing, China
xgao@mmrc.iss.ac.cn

[2] Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada
q94huang@uwaterloo.ca

Abstract. In this paper, a new reduction based interpolation algorithm for general black-box multivariate polynomials over finite fields is given. The method is based on two main ingredients. A new Monte Carlo method is given to reduce the black-box multivariate polynomial interpolation problem to the black-box univariate polynomial interpolation problem over any ring. The reduction algorithm leads to multivariate interpolation algorithms with better or the same complexities in most cases when combining with various univariate interpolation algorithms. A modified univariate Ben-Or and Tiwari algorithm over the finite field is proposed, which has better total complexity than the Lagrange interpolation algorithm. Combining our reduction method and the modified univariate Ben-Or and Tiwari algorithm, we give a Monte Carlo multivariate interpolation algorithm, which has better total complexity in most cases for sparse interpolation of black-box polynomial over finite fields.

Keywords: Randomized Kronecker substitution · Sparse polynomial interpolation · Black-box · Finite field · Monte Carlo algorithm

1 Introduction

The interpolation for a sparse multivariate polynomial

$$f = c_1 m_1 + c_2 m_2 + \cdots + c_t m_t \in \mathcal{R}[x_1, \ldots, x_n] \tag{1}$$

given as a black-box is a basic computational problem, where $\mathcal{R}$ is a ring. Here, the challenge is that both the monomials m_i and the coefficients c_i are unknown and the algorithm needs to take advantage of the sparse structure of f.

Partially supported by an NSFC grant No.11688101.

M. England et al. (Eds.): CASC 2019, LNCS 11661, pp. 215–235, 2019.
https://doi.org/10.1007/978-3-030-26831-2_15

In [1], Zippel gave a probabilistic algorithm which needs an upper bound for the number of terms of f and an upper bound for the degree of f in each variable. In [2], Ben-Or and Tiwari gave a deterministic algorithm over the field of complex numbers, which needs an upper bound for the number of terms in f. After these work, many interesting algorithms were given, such as the computational complexity enhancement [3,4], the interpolation with nonstandard bases [5], the interpolation over finite fields [6–10], the early termination algorithm [11,12], the hybrid interpolation algorithm [13–15], the interpolation for modular black-box polynomials [16,17], and the reduction based methods for black-box and SLP polynomials [8,18–23].

The sparse interpolation algorithms can be roughly divided into two types: (1) the direct methods, such as the Ben-Or and Tiwari algorithm, which find the monomials m_i directly and then find the coefficients; (2) the reduction based methods, such as Zippel's algorithm, which reduce the multivariate interpolation problem into the univariate interpolation problem.

The size of an n-variate polynomial f with a degree bound D, a term bound T, and coefficients c_i is $O(nT \log D + T \log c)$, where $c = \max_{i=1}^{t} |c_i|$. The sparse interpolation algorithms can also be roughly divided into two types according to the complexity in D: (1) the supersparse algorithm whose complexity is polynomial in $\log D$; (2) the exponential algorithm whose complexity is polynomial in D. All algorithms have complexity polynomial in n, T.

Since the value of a polynomial of degree D at any point other than $0, \pm 1$ will have D bits or more, any algorithm whose complexity is proportional to $\log D$ cannot perform such an evaluation over $\mathbb{Q}$ or $\mathbb{Z}$. Even for polynomials over the general finite field $\mathbb{F}_q$, there is no supersparse interpolation algorithms for the standard black-box model. On the other hand, supersparse algorithms do exist for the following special models.

The first model is the straight-line program model [8,18,19,21–24], which uses the arithmetic operations in the $\mathcal{R}[x_1, \ldots, x_n]$ to replace the black-box evaluation.

The second model is the modular black-box model [16,17], which works for the polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$. Given a prime p and an element θ in $\mathbb{F}_p$, the model computes $f(\theta)$ over the field $\mathbb{F}_p$. The cost of the evaluation depends on the size of p.

The third model is the precision accuracy black-box model [13,15,25,26], which allows for evaluations on the unit circle in some representation of a subfield of $\mathbb{C}$ or returns only a limited number of bits of precision for an evaluation.

In this paper, we focus on reduction based methods for general black-box models. Our main contribution is to give a new Monte Carlo reduction method, which leads to multivariate interpolation algorithms with better or the same complexities in most cases comparing to existing reduction based methods. We also propose a modified univariate Ben-Or and Tiwari algorithm over the finite field $\mathbb{F}_q$ costing $O^{\sim}(D \log q + TB)$ bit operations, where B is the cost of one query of the black-box and is a function of T, D, q. Note that the Lagrange interpolation algorithm costs $O^{\sim}(D \log q + DB)$ bit operations, which is larger

since $T \leq D$. Let f be an n-variate polynomial with a degree bound D and a term bound T. Combining our reduction method and the modified univariate Ben-Or and Tiwari algorithm, we give a multivariate interpolation algorithm whose bit complexity is $O^{\sim}(nTD \log q + nTB_f)$, where B_f is the cost of evaluating the black-box that gives f and is a function of n, T, D, q.

1.1 Comparing with Other Reduction Based Methods

Our reduction depends on the following Kronecker type substitutions:

$$f(x^{\mathbf{s}}) = f(x^{s_1}, x^{s_2}, \ldots, x^{s_n}) \tag{2}$$

$$f(x^{\mathbf{s}+p\mathbf{I}_k}) = f(x^{s_1}, x^{s_2}, \ldots, x^{s_k+p}, \ldots, x^{s_n}) \tag{3}$$

where p is a prime and $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{N}^n$ is a vector of random integers. The substitution (2) introduced in [20] is called *randomized Kronecker substitution*. (3) was introduced in [27]. Our method builds on the work [20,27]. To compare with [20,27], we first explain how these algorithms work. The algorithm in [20] has three main steps. 1. Randomly choose $O(n+\log T)$ substitutions $\mathbf{s}_i$. 2: Find a diversifying set of terms of f such that a diversifying term has the same coefficient after all substitutions. 3: For each term, solve a linear system to obtain its exponents. The algorithm in [27] also has three main steps. 1: Randomly choose $\log T$ substitutions $\mathbf{s}_i$. 2: Find the $f(x^{\mathbf{s}_u})$ with the maximal number of terms. 3: Find a prime p such that $\#f(x^{\mathbf{s}_u})$ **mod** $(x^p - 1) = \#f(x^{\mathbf{s}_u})$ and half of the terms of f can be recovered from $f(x^{\mathbf{s}_u})$ and $f(x^{\mathbf{s}_u+p\mathbf{I}_k}), k = 1, 2, \ldots, n$.

Our algorithm works as follows. 1: Randomly choose $\log T$ primes p_i of size $O^{\sim}(T \log D)$ and substitutions $\mathbf{s}_i \in \mathbb{Z}_{p_i}^n$. 2: Find a u such that $\#f(x^{\mathbf{s}_u})$ **mod** $(x^{p_u} - 1)$ has the maximal number of terms. 3: Prove that half of the terms of f can be recovered from $f(x^{\mathbf{s}_u})$ and $f(x^{\mathbf{s}_u+p_u\mathbf{I}_k}), k = 1, 2, \ldots, n$.

Our method is different from that in [20,27] in the following aspects. Comparing to [20], we do not need to solve linear systems, so our algorithm is linear in n while theirs are linear in n^ω. Also, our algorithm does not need to find the diversifying set, so it works for more general rings. Comparing to [27], our algorithm chooses a prime p_i first and then chooses the substitutions $\mathbf{s}_i \in \mathbb{Z}_{p_i}^n$, while in [27], the prime is fixed. As a consequence, the univariate polynomials in our algorithm have degrees $O^{\sim}(TD)$, while the degrees of the univariate polynomials in [27] contain either T^2 or D^2.

In Table 1, we list some existing reduction methods, where "#Reductions(N)" is the number of univariate interpolations, "Degree($\widetilde{D}$)" is the degree bound of the univariate polynomials, "Extra bit complexity(η)" is the additional complexities needed besides the univariate interpolations. "Type" means whether the algorithm is deterministic (Det), Monte Carlo (MC), or Las Vegas (LV).

We now compare the complexities of multivariate polynomial interpolation algorithms using the reductions given in Table 1. The complexity of the randomized Kronecker substitution algorithm in [20] already can be improved by removing the $O(n^\omega T)$ term in the complexity, by using structured linear algebra. This is an idea attributed to Pernet in [27], see Lemma 5.7.2. We do not list it

Table 1. Reduction of multivariate polynomial interpolations to univariate ones

	#Reductions(N)	Degree ($\widetilde{D}$)	Extra bit cost(η)	Type
Kronecker	1	D^n	$nT\log D$	Det
Zippel [1]	nT	D	$\geq nT\log D$	MC
Klivans-Spielman [6]	n	nT^2D	$nTD^{O(1)}$	MC
Arnold [27]	$n\log T$ + $\log^2 T$	TD + $TD+\overline{D}T\min(D,T\log(TD))$	$nT\log D$	MC
Arnold-Roche [20]	$n+\log T$	TD	$n^\omega T+nT\log D$	MC
Huang-Gao [22]	$n\log T$	nTD	$nT\log D$	MC
This paper (Cor. 1)	$n\log T+\log^2 T$	TD	$nT\log D$	MC

in Table 1 because the result is spelled out for extended-black-box polynomial. According to our analysis, if using this method to general black-box polynomials, N will be increased to at least $O(n\log T(\log D+\log T))$ and $\widetilde{D}$ will be increased to $O(nTD)$. So its cost is more than the algorithm in [22].

Two cases are considered according to the complexity of the univariate interpolation algorithm to be used.

First, assume a univariate interpolation algorithm is supersparse with complexity $\mathbf{SLin}(T^\alpha, \log^\beta D)$, where $\mathbf{SLin}(a,b,\dots)$ means the complexity is soft-linear in $a,b,\dots$. For simplicity of analysis, we assume $\mathbf{SLin}(a,b,\dots)$ is the product of $a,b,\dots$. Since $\log D$ is the factor of the size of f, $\beta\geq 1$. Then the complexity of the multivariate interpolation algorithm is $\mathbf{SLin}(NT^\alpha, N\log^\beta\widetilde{D})+\eta$, where N, $\widetilde{D}$, and η are from Table 1. We list these complexities in Table 2. From the table, we can see that, for the supersparse algorithms, our reduction method is not worse than the methods in [22,27] and the Kronecker substitution, and is better than others.

Table 2. Complexity for supersparse multivariate interpolation algorithms based on reductions

	Complexity	type
Kronecker	$\mathbf{SLin}(n^\beta, T^\alpha, \log^\beta D)$	Det
Zippel [1]	$\mathbf{SLin}(n, T^{\alpha+1}, \log^\beta D)$	MC
Klivans-Spielman [6]	$\mathbf{SLin}(n, T^\alpha, \log^\beta D)+nTD^{O(1)}$	MC
Arnold [27]	$\mathbf{SLin}(n, T^\alpha, \log^\beta D)$	MC
Arnold-Roche [20]	$\mathbf{SLin}(n, T^\alpha, \log^\beta D, n^\omega T)$	MC
Huang-Gao [22]	$\mathbf{SLin}(n, T^\alpha, \log^\beta D)$	MC
This paper (Cor. 1)	$\mathbf{SLin}(n, T^\alpha, \log^\beta D)$	MC

Second, assume a univariate algorithm is exponential and its complexity is $\mathbf{SLin}(T^\alpha, D^\beta)$. Then the complexities of the multivariate interpolation algorithms are $\mathbf{SLin}(NT^\alpha, N(\widetilde{D})^\beta) + \eta$, which are listed in Table 3. From the table, we can see that, for the exponential algorithms, the complexity of our algorithm is better than all the existed Kronecker-type substitutions [6,20,22,27]. Comparing to Zippel's reduction [1], our method has better, equal, or worse complexities according to $0 < \beta < 1$, $\beta = 1$, or $\beta > 1$, respectively.

Table 3. Complexity for exponential multivariate interpolation algorithms based on reductions

	Complexity	type
Kronecker	$\mathbf{SLin}(T^\alpha, D^{n\beta}) + nT \log D$	Det
Zippel [1]	$\mathbf{SLin}(n, T^{\alpha+1}, D^\beta)$	MC
Klivans-Spielman [6]	$\mathbf{SLin}(n^{\beta+1}, T^{\alpha+2\beta}, D^\beta) + nTD^{O(1)}$	MC
Arnold [27]	$\mathbf{SLin}(n, T^{\alpha+\beta}, D^{2\beta})$ or $\mathbf{SLin}(n, T^{\alpha+2\beta}, D^\beta)$	MC
Arnold-Roche [20]	$\mathbf{SLin}(n, T^{\alpha+\beta}, D^\beta) + n^\omega T$	MC
Huang-Gao [22]	$\mathbf{SLin}(n^{\beta+1}, T^{\alpha+\beta}, D^\beta)$	MC
This paper (Cor. 1)	$\mathbf{SLin}(n, T^{\alpha+\beta}, D^\beta)$	MC

Table 4 is a summary of the comparisons, where "$\surd$", "$=$", "$\times$" means that our reduction method has better, the same, or worse complexity, respectively. We can see that our reduction method achieves better or the same complexities, except one case: exponential algorithms with $\beta > 1$. Since D is the main factor in the complexity, algorithms with high complexities in D are generally not used.

Table 4. Compare to other reduction based interpolation methods

		Kronecker	Zippel [1]	Klivans-Spielman [6]	Arnold [27]	Arnold-Roche [20]	Huang-Gao(MC) [22]
Supersparse	$\beta = 1$	$=$	$\surd$	$\surd$	$=$	$\surd$	$=$
	$\beta > 1$	$\surd$	$\surd$	$\surd$	$=$	$\surd$	$=$
Exponential	$0 \le \beta < 1$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
	$\beta = 1$	$\surd$	$=$	$\surd$	$\surd$	$\surd$	$\surd$
	$\beta > 1$	$\surd$	$\times$	$\surd$	$\surd$	$\surd$	$\surd$

Finally, we remark that for exponential algorithms, the cases $\beta > 1$ and $\beta < 1$ do exist. The original Ben-Or and Tiwari algorithm works for univariate polynomials over the finite field $\mathbb{F}_q$ and costs $O^\sim(T^{1.5}\sqrt{D}\log q + T\log^2 q)$ bit operations (Refer to Remark 2), where $\beta = 0.5$. The bit complexity of the Lagrange interpolation algorithm over $\mathbb{Q}$ is $O^\sim(D^2)$.

1.2 Comparing with Interpolation Algorithms over Finite Fields

In order to obtain a reduction based multivariate interpolation algorithm, we need univariate interpolation algorithms with best complexities.

Let h be a black-box univariate polynomial in $\mathbb{F}_q[x]$ with a degree bound D and a term bound T. Let B_h be the cost of one query of the black-box. In this paper, we give a modified univariate Ben-Or and Tiwari algorithm which costs $O^{\sim}(D\log q)$ bit operations and $O(T)$ evaluations of h, so the total cost is $O^{\sim}(D\log q + TB_h)$. The Lagrange interpolation algorithm costs $O^{\sim}(D\log q)$ bit operations and $O(D)$ evaluations of h and the total complexity is $O(D\log q + DB_h)$. So, the modified univariate Ben-Or and Tiwari algorithm has lower total complexities than the Lagrange algorithm, since $T \leq D$.

A univariate Ben-Or and Tiwari algorithm over the finite filed was given in [24], whose complexity includes the parameter q. Also, the multivariate Ben-Or and Tiwari algorithm was extended to finite fields [9,10], whose complexities are high (see Table 5).

Combining the modified univariate Ben-Or and Tiwari algorithm and our reduction method, we give a new multivariate interpolation algorithm. Table 5 is a comparison with interpolation algorithms over finite fields. "Probes" is the number of evaluations for the polynomials, "Bit complexity" is the complexity besides the probes, and "Size of $\mathbb{F}_q$" means that the algorithm can work for the finite field whose size satisfies this condition, and in the contrary case, the algorithm need to take values in a proper extension field of $\mathbb{F}_q$.

Table 5. "Soft-Oh" comparison of interpolation algorithms over finite field $\mathbb{F}_q$

	Probes (ρ)	Bit complexity (Θ)	Size of $\mathbb{F}_q$	type
Grigoriev-Karpinski-Singer [7]		$n^2T^6\log^2(ntq) + q^{2.5}\log^2 q$		Det
Huang-Rao [9]	T^2D	$(TD)^8((TD)^5 + \log q)\log^2 q + nTD\log q$	$q \geq O(T^2D^2)$	LV
Javadi and Monagan [10]	nT	$T^2(\log q + nD)\log q$	$\phi(q-1) \geq O(nD^2T^2)$	MC
Klivans-Spielman [6]	nT	$n^2T^2D\log q$	$q \geq O(nT^2D)$	MC
Arnold-Roche [20]	nT	$nTD\log q + n^{\omega}T$	$q \geq O(TD)$	MC
Huang-Gao [22]	nT	$n^2TD\log q$	$q \geq O^{\sim}(nDT)$	MC
Zippel [1,10]	nTD	$nTD\log q$	$q \geq O(nD^2T^2)$	MC
This paper (Them. 9)	nT	$nTD\log q$	$q \geq O^{\sim}(TD)$	MC
This paper (Rem. 2)	nT	$nT^{1.5}\sqrt{D}\log q + nT\log^2 q$	$q \geq O^{\sim}(TD)$	MC

The total complexity of an algorithm is $O^{\sim}(\Theta + \rho B)$, where Θ and ρ are from Table 5 and B is the cost of probing the black-box. The bit complexities of the algorithms given in [7,9] are much higher than other algorithms, so we will not compare with them below.

We can see that our algorithm (Theorem 9) has better total complexity than all other methods in [1,6,10,20,22]. Comparing to Zippel's algorithm, our algorithm has the same bit complexity but needs less evaluations and works for a smaller field. Actually, our algorithm is the only one which achieves the best current bounds in all three parameters in Table 5.

The algorithm given in Remark 2 uses the original Ben-Or and Tiwari algorithm for univariate polynomials over the finite field $\mathbb{F}_q$, which costs $O^{\sim}(nT^{1.5}\sqrt{D}\log q + nT\log^2 q)$ bit operations. By Table 4, if using this univariate interpolation algorithm, our reduction method gives a multivariate interpolation algorithm with lowest complexity in D, which can be seen from Table 5.

2 Reduction Based on Randomized Kronecker Substitution

In this section, we give a new Monte Carlo algorithm which reduces multivariate polynomial interpolation to univariate polynomial interpolation based on randomized Kronecker substitutions over any commutative ring with identity.

2.1 Find an "ok" Random Kronecker Substitution

Let $f \in \mathcal{R}[\mathbb{X}]$, where $\mathcal{R}$ is commutative ring with identity and $\mathbb{X} = \{x_1, x_2, \ldots, x_n\}$ is a set of n indeterminates. Denote $\#f$ and $\deg f$ to be the number of terms in f and the total degree of f, respectively. For $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{N}^n$ and a new indeterminate x, let

$$f(x^{\mathbf{s}}) = f(x^{s_1}, x^{s_2}, \ldots, x^{s_n}) \tag{4}$$

$$f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}}) = f(x^{s_1}, x^{s_2}, \ldots, x^{s_n}) \ \mathbf{mod}\ (x^p - 1). \tag{5}$$

For $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{N}^n$, a term cm_1 of f is said to *collide* in $f(x^{\mathbf{s}})$ (or $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}})$) if f has another term em_2 such that $m_1 \neq m_2$ and $m_1(x^{\mathbf{s}}) = m_2(x^{\mathbf{s}})$(or $m_{1(p)}^{\mathbf{mod}}(x^{\mathbf{s}}) = m_{2(p)}^{\mathbf{mod}}(x^{\mathbf{s}})$).

When $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ is chosen randomly, the substitution $x_i = x^{s_i}, i = 1, 2, \ldots, n$ is called a *randomized Kronecker substitution*. For a prime p, a substitution $\mathbf{s}$ is called *"ok" with respect to* p, if a majority, say $\frac{5}{8}$, of the terms of f do not collide in $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}})$.

We need the following Hoeffding's inequility for Bernoulli random variables.

Lemma 1. *[28] Let $X = \sum_{i=1}^{n} X_i$, where X_i,$i = 1, 2, \ldots, n$, are independently distributed in $[0, 1]$. Then for all $\varepsilon > 0$, $\Pr[X > \mathbf{E}[X] + \varepsilon] \leq e^{-2\varepsilon^2/n}$ and $\Pr[X < \mathbf{E}[X] - \varepsilon] \leq e^{-2\varepsilon^2/n}$, where $\mathbf{E}[X]$ is the expected value of X.*

We have the following key lemma.

Lemma 2. *Let $f = \sum_{i=1}^{t} c_i m_i \in \mathcal{R}[\mathbb{X}]$, $T \geq \#f$, $D \geq \deg f$, and $N = \max\{31\lfloor (T-1)\log_2 D \rfloor, 1\}$. Let $p_1, p_2, \ldots, p_N$ be N different primes which satisfy $p_i \geq 32(T-1)$. If we randomly choose a prime p in $\{p_1, p_2, \ldots, p_N\}$ and choose $\mathbf{s} \in \mathbb{Z}_p^n$ uniformly at random, where $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$. Then any fixed term of f collides in $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}})$ with probability $\leq \frac{1}{16}$.*

Proof. If $t = 1$ or $D = 1$, then the proof is obvious. So assume $T \geq t \geq 2$ and $D \geq 2$. In this case, $N = 31\lceil (T-1)\log_2 D\rceil$. Assume $m_i = x_1^{e_{i,1}} \cdots x_n^{e_{i,n}}, i = 1, \ldots, t$. Without loss of generality, we consider the first term $c_1 m_1$. Let $h(s_1, \ldots, s_n) = \prod_{i=2}^{t}[(e_{i,1} - e_{1,1})s_1 + \cdots + (e_{i,n} - e_{1,n})s_n]$ which is a polynomial in $\mathbb{Z}[s_1, \ldots, s_n]$ with degree no more than $T-1$. Assume the variables are ordered as $s_1 \prec \cdots \prec s_n$ and k_i is the largest number such that $e_{i,k_i} - e_{1,k_i} \neq 0$. Then $\prod_{i=2}^{t}[e_{i,k_i} - e_{1,k_i}]s_{k_i}$ is the leading term. Let $C = \prod_{i=2}^{t}[e_{i,k_i} - e_{1,k_i}]$ and let ℓ be the number of different prime factors of C. Since $|e_{i,k_i} - e_{1,k_i}| \leq D$, we have $2^\ell \leq C$ and hence C has at most $\lfloor (T-1)\log_2 D\rfloor$ different prime factors. So if we randomly choose a prime p in $\{p_1, \ldots, p_N\}$, with probability at least $1 - \frac{\lfloor (T-1)\log_2 D\rfloor}{N} = \frac{30}{31}$, $\prod_{i=2}^{t}[e_{i,k_i} - e_{1,k_i}]$ **mod** $p \neq 0$. In this case, $h(s_1, \ldots, s_n)$ **mod** p is a non-zero polynomial in $\mathbb{F}_p[s_1, \ldots, s_n]$. If $h(s_1, \ldots, s_n)$ **mod** $p \neq 0$, then by Zippel's lemma [1], if we choose $\mathbf{s} \in \mathbb{Z}_p^n$ uniformly at random, then $h(\mathbf{s})$ **mod** $p \neq 0$ with probability at least $1 - \frac{T-1}{p} \geq 1 - \frac{T-1}{32(T-1)} = \frac{31}{32}$. So if we randomly choose a prime p in $\{p_1, \ldots, p_N\}$ and then choose $\mathbf{s} \in \mathbb{Z}_p^n$ uniformly at random, with probability at least $\frac{30}{31} \cdot \frac{31}{32} = \frac{15}{16}$, $h(\mathbf{s})$ **mod** $p \neq 0$. Now it suffices to show that when $h(\mathbf{s})$ **mod** $p \neq 0$, $c_1 m_1$ does not collide in $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}})$. Since $h(\mathbf{s})$ **mod** $p \neq 0$, $(e_{i,1} - e_{1,1})s_1 + \cdots + (e_{i,n} - e_{1,n})s_n \neq 0$ **mod** p. So $(e_{i,1}s_1 + \cdots + e_{i,n}s_n)$ **mod** $p \neq (e_{1,1}s_1 + \cdots + e_{1,n}s_n)$ **mod** p, which means that $c_i m_i$ does not collide with $c_1 m_1$ in $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}})$. □

Lemma 3. *Let $B_j, j = 1, 2, \ldots, s$ be nonempty sets of integers and $a_i, i = 1, 2, \ldots, t$ all the different elements in $\cup_{j=1}^{s} B_j$. Let c be the number of a_i satisfying $a_i \in B_j$ and $\#B_j \geq 2$ for some j. Then $t - c \leq s$ and for $s_1 \in [t-c, s] \cap \mathbb{N}$, we have $(t - s_1) \leq c \leq 2(t - s_1)$.*

Proof. B_j is called a single point set if $\#B_j = 1$, and a collision set if $\#B_j \geq 2$. Since $t - c$ is the number of a_i contained in all single point sets, there exist $t - c$ single point sets. So $t - c \leq s$. Since $t - c \leq s_1 \leq s$, we have $(t - s_1) \leq c$. Let k_1 be the number of collision sets. We have $k_1 + t - c = s$. So $c = k_1 + t - s$. Since every collision set contains at least two elements, $k_1 \leq \frac{1}{2}c$. So $c \leq \frac{1}{2}c + t - s$, which is $\frac{1}{2}c \leq t - s \leq t - s_1$. So $c \leq 2(t - s_1)$. □

For $p \in \mathbb{Z}_{>0}$ and $\mathbf{u} \in \mathbb{N}^n$, let $\mathcal{C}_f(p, \mathbf{u})$ be the number of terms of f which collide in $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{u}})$. We have

Lemma 4. *Let $p_{\mathbf{u}}, p_{\mathbf{v}} \in \mathbb{Z}_{>0}$ and $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_{>0}^n$ such that*

$$\#[f_{(p_{\mathbf{u}})}^{\mathbf{mod}}(x^{\mathbf{u}})] \geq \#[f_{(p_{\mathbf{v}})}^{\mathbf{mod}}(x^{\mathbf{v}})].$$

Then $\mathcal{C}_f(p_{\mathbf{u}}, \mathbf{u}) \leq 2\mathcal{C}_f(p_{\mathbf{v}}, \mathbf{v})$.

Proof. Assume $\#[f_{(p_{\mathbf{u}})}^{\mathbf{mod}}(x^{\mathbf{u}})] = k_0, \#[f_{(p_{\mathbf{v}})}^{\mathbf{mod}}(x^{\mathbf{v}})] = k$ and $f_{(p_{\mathbf{u}})}^{\mathbf{mod}}(x^{\mathbf{u}}) = a_1 x^{d_1} + \cdots + a_{k_0} x^{d_{k_0}}, d_i \neq d_j$, when $i \neq j$. Let $f = f_1 + \cdots + f_{k_0} + g$, where $(f_i)_{(p_{\mathbf{u}})}^{\mathbf{mod}}(x^{\mathbf{u}}) = a_i x^{d_i}, i = 1, \ldots, k_0$ and $g_{(p_{\mathbf{u}})}^{\mathbf{mod}}(x^{\mathbf{u}}) = 0$. Let $B_i, i = 1, \ldots, k_0$ be the set of terms in f_i and B_0 be the set of terms in g. By Lemma 3, we have $(t - k_0) < \mathcal{C}_f(p_{\mathbf{u}}, \mathbf{u}) \leq 2(t - k_0)$. By the same reason, we have $(t - k) \leq \mathcal{C}_f(p_{\mathbf{v}}, \mathbf{v}) \leq 2(t - k)$. Now $\mathcal{C}_f(p_{\mathbf{u}}, \mathbf{u}) \leq 2(t - k_0) \leq 2(t - k) \leq 2\mathcal{C}_f(p_{\mathbf{v}}, \mathbf{v})$. □

The following theorem gives a method to compute an "ok" substitution, which is similar to [27, Prop.5.4.2] and has two differences. (1). For each substitution, we choose a random prime, while in [27], the prime is fixed. (2). We choose the substitution $\mathbf{s}$ such that $\#f^{\mathbf{mod}}_{(p)}(x^{\mathbf{s}})$ has the maximal number of terms, while in [27], they choose the one such that $\#f(x^{\mathbf{s}})$ has the maximal number of terms.

Theorem 1. *Let* $f(\mathbb{X}) \in \mathcal{R}[\mathbb{X}]$, $T \geq \#f$, $D \geq \deg f$, $N = \max\{31\lceil(T-1)\log_2 D\rceil, 1\}$ *and* $p_1, \ldots, p_N$ *be* N *different primes which satisfy* $p_i \geq 32(T-1)$. *Let* $\mu \in (0,1)$ *and* $\ell \geq \lceil 32\ln(T\mu^{-1})\rceil$. *For* $i = 1, \ldots, \ell$, *we randomly choose a prime* p_{α_i} *in* $\{p_1, \ldots, p_N\}$ *and then choose* $\mathbf{s}_i \in \mathbb{Z}^n_{p_{\alpha_i}}$ *uniformly at random. Let* $(p, \mathbf{s})$ *be the vector in* $\{(p_{\alpha_1}, \mathbf{s}_1), \ldots, (p_{\alpha_\ell}, \mathbf{s}_\ell)\}$ *such that* $\#[f^{\mathbf{mod}}_{(p)}(x^{\mathbf{s}})] = \max_{i=1}^{\ell} \#[f^{\mathbf{mod}}_{(p_{\alpha_i})}(x^{\mathbf{s}_i})]$. *Then at least* $\frac{5}{8}\#f$ *terms of* f *do not collide in* $f^{\mathbf{mod}}_{(p)}(x^{\mathbf{s}})$ *with probability at least* $1-\mu$.

Proof. First we consider a fixed term $c_i m_i$ and let $f_j(x) = f^{\mathbf{mod}}_{(p_{\alpha_j})}(x^{\mathbf{s}_j})$. By Lemma 2, the probability of $c_i m_i$ colliding in $f_j(x)$ is no more than $\frac{1}{16}$. Define $X_j = 1$ to be the event that $c_i m_i$ collides in $f_j(x)$ and $X_j = 0$ to be the event that $c_i m_i$ does not collide in $f_j(x)$ for some j. Define $X = \sum_{j=1}^{\ell} X_j$, then $\mathbf{E}[X] \leq \frac{1}{16}\ell$. By Hoeffiding's inequality, we have $\Pr(X > \frac{1}{16}\ell + \varepsilon) \leq \Pr(X > \mathbf{E}[X] + \varepsilon) \leq e^{-2\varepsilon^2/\ell}$. Let $\varepsilon = \frac{1}{8}\ell$, then $\Pr(X > \frac{3}{16}\ell) \leq e^{-\ell/32} \leq e^{\ln(\mu/T)} = \frac{\mu}{T}$. So at probability $\leq 1-\mu$, for all term $c_i m_i$ of f, $c_i m_i$ collides in at most $\frac{3}{16}\ell$ of $f_j(x), j = 1, \ldots, \ell$. In other words, with probability $\geq 1-\mu$, at leat $\frac{13}{16}\ell\#f$ terms in $f^{\mathbf{mod}}_{(p_{\alpha_j})}(x^{\mathbf{s}_j}), j = 1, \ldots, \ell$ do not collide. We claim that at least one of $f_j(x)$ has at least $\frac{13}{16}\#f$ non-colliding terms. We prove the claim by contradiction. Assume that each $f_j(x)$ has $< \frac{13}{16}\#f$ non-colliding terms. Then there exist $< \frac{13}{16}\#f\ell$ non-colliding terms in $f_j(x), j = 1, \ldots, \ell$, which contradicts to the fact that these $f_j(x)$ have $\geq \frac{13}{16}\ell\#f$ non-colliding terms. So there must exist one $(p_{\alpha_j}, \mathbf{s}_j)$ for which at most $\frac{3}{16}$ of the terms of f collide. By Lemma 4, the polynomial with maximum $\#f_j(x)$ has at least $\frac{5}{8}\#f$ non-colliding terms. □

2.2 Recover Non-colliding Terms

For $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{N}^n$ and $p \in \mathbb{N}_{>0}$, let

$$f(x^{\mathbf{s}+p\mathbf{I}_k}) = f(x^{s_1}, \ldots, x^{s_k+p}, \ldots, x^{s_n}) \tag{6}$$

to be the univariate polynomial obtained with the substitution: $x_i = x^{s_i}, i = 1, 2, \ldots, n, i \neq k, x_k = x^{s_k+p}$, where $\mathbf{I}_k \in \mathbb{Z}^n_{\geq 0}$ is the k-th unit vector.

In this section, we show how to recover the non-colliding terms of $f \in \mathcal{R}[\mathbb{X}]$ from $f^{\mathbf{mod}}_{(p)}(x^{\mathbf{s}})$, $f(x^{\mathbf{s}})$, and $f(x^{\mathbf{s}+p\mathbf{I}_k})$. Let

$$f^{\mathbf{mod}}_{(p)}(x^{\mathbf{s}}) = a_1 x^{d_1} + \cdots + a_r x^{d_r} \tag{7}$$

Since $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}}) = f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}+p\mathbf{I}_k})$, for $k = 1, 2, \ldots, n$, we can write

$$\begin{aligned} f(x^{\mathbf{s}}) &= f_1 + f_2 + \cdots + f_r + g \\ f(x^{\mathbf{s}+p\mathbf{I}_k}) &= f_{k,1} + f_{k,2} + \cdots + f_{k,r} + g_k \end{aligned} \tag{8}$$

where f_i **mod** $(x^p - 1) = f_{k,i}$ **mod** $(x^p - 1) = a_i x^{d_i}$, g **mod** $(x^p - 1) = g_k$ **mod** $(x^p - 1) = 0$. We define the following key notation

$$\begin{aligned} TS_{(f,p,\mathbf{s},D)} = \{ & a_i x_1^{e_{i,1}} \cdots x_n^{e_{i,n}} | a_i \text{ is from (7), and} \\ & \text{T1} : f_i = a_i x^{u_i}, f_{k,i} = a_i x^{b_{k,i}}, k = 1, 2, \ldots, n. \\ & \text{T2} : e_{i,k} = \frac{b_{k,i} - u_i}{p} \in \mathbb{N}, k = 1, 2, \ldots, n. \\ & \text{T3} : u_i = e_{i,1} s_1 + e_{i,2} s_2 + \cdots + e_{i,n} s_n. \\ & \text{T4} : \sum_{j=1}^{n} e_{i,j} \le D. \} \end{aligned} \tag{9}$$

Lemma 5. *Let $f = \sum_{i=1}^{t} c_i m_i \in \mathcal{R}[\mathbb{X}]$ and $D \ge \deg f$. If cm does not collide in $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}})$, then $cm \in TS_{(f,p,\mathbf{s},D)}$.*

Proof. It suffices to show that cm satisfies the conditions of the definition of $TS_{(f,p,\mathbf{s},D)}$. Assume $m = x_1^{e_1} \cdots x_n^{e_n}$. Since cm is not a collision in $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}})$, without loss of generality, assume $cm(x^{\mathbf{s}})$ **mod** $(x^p - 1) = a_1 x^{d_1}$, where $a_1 x^{d_1}$ is defined in (7). It is easy to show that cm is also not a collision in $f(x^{\mathbf{s}})$ and in $f(x^{\mathbf{s}+p\mathbf{I}_k})$. Hence, $f_1 = a_1 x^{u_1}$ for $u_1 = \sum_{i=1}^{n} e_i s_i$; $b_{k,1} = u_1 + p e_k$. Clearly, T1, T2 and T3 are correct. Since $\deg m = \sum_{j=1}^{n} e_j \le D$, T4 is correct. □

Now we give the algorithm to compute $TS_{(f,p,\mathbf{s},D)}$.

Algorithm 2 (TSTerms).

Input:
- Univariate polynomials $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}}), f(x^{\mathbf{s}}), f(x^{\mathbf{s}+p\mathbf{I}_k})$, where $k = 1, 2, \ldots, n$.
- A prime p.
- A vector $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{Z}_{\ge 0}^n$.
- Degree bound $D \ge \deg f$.

Output: $\mathrm{TS}_{(f,p,\mathbf{s},D)}$.

Step 1: Write $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}})$, $f(x^{\mathbf{s}})$, and $f(x^{\mathbf{s}+p\mathbf{I}_k})$ in the following form

$$\begin{aligned} f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}}) &= a_1 x^{d_1} + a_2 x^{d_2} + \cdots + a_r x^{d_r} \\ f(x^{\mathbf{s}}) &= a_1 x^{u_1} + \cdots + a_\gamma x^{u_\gamma} + f_1 \\ f(x^{\mathbf{s}+p\mathbf{I}_k}) &= a_1 x^{b_{k,1}} + \cdots + a_\gamma x^{b_{k,\gamma}} + f_{k,2} \end{aligned}$$

where $i = 1, 2, \ldots, \gamma$, $k = 1, \ldots, n$, $a_i x^{u_i}$, $a_i x^{b_{k,i}}$ are all the terms satisfying: $x^{b_{k,i}}$ is the unique term in $f(x^{\mathbf{s}+p\mathbf{I}_k})$ such that $\mathbf{mod}(b_{k,i}, p) = d_i$ and x^{u_i} is the unique term in $f(x^{\mathbf{s}})$ such that $\mathbf{mod}(u_i, p) = d_i$.

Step 2: Let $S = \{\}$.
Step 3: For $i = 1, 2, \ldots, \gamma$
a: For $k = 1, 2, \ldots, n$, Let $e_{i,k} = \frac{b_{k,i} - u_i}{p}$. If $e_{i,k} \notin \mathbb{N}$, then break.
b: If $u_i \neq e_{i,1}s_1 + e_{i,2}s_2 + \cdots + e_{i,n}s_n$, then break;
c: If $\sum_{j=1}^{n} e_{i,j} > D$, then break;
d: Let $S = S \bigcup \{a_i x_1^{e_{i,1}} \cdots x_n^{e_{i,n}}\}$.
Step 4: Return S.

Lemma 6. *Algorithm 2 needs $O(nT)$ ring operations in $\mathcal{R}$ and $O^{\sim}(nT \cdot \log(s_{\max}D + pD))$ bit operations, where $s_{\max} = \max\{s_1, s_2, \ldots, s_n\}$.*

Proof. In Step 1, in order to match the terms of $f_{(p)}^{\mathbf{mod}}(x^{\mathbf{s}})$, $f(x^{\mathbf{s}})$, and $f(x^{\mathbf{s}+p\mathbf{I}_k})$, it needs $O^{\sim}(nT\log(s_{\max}D + pD))$ bit operations and $O(nT)$ ring operations in $\mathcal{R}$. In Step 3, **a**, **b** and **c** need $O(nT)$ arithmetic operations in $\mathbb{Z}$. Since the data is $O(s_{\max}D + pD)$, the complexity is $O(nT\log(s_{\max}D + pD))$ bit operations. □

2.3 Algorithms

We will give the reduction algorithm for $f \in \mathcal{R}[\mathbb{X}]$, which works as follows. We first find an "ok" random Kronecker substitution $\mathbf{s}$ based on Theorem 1, then obtain half of the terms of f by applying Algorithm 2, and finally repeat the procedure for at most $\log(\#f)$ times to find f. We assume an interpolation algorithm for univariate polynomials is given in advance.

We first give an algorithm to obtain the polynomials $g(x^{\mathbf{s}+p\mathbf{I}_k}), k = 1, \ldots, n$ from $g(\mathbb{X})$.

Algorithm 3 (PolySubs).

Input: • A polynomial $g \in \mathcal{R}[\mathbb{X}]$.
• A vector $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{Z}_{\geq 0}^n$.
• A prime p.
Output: $g(x^{\mathbf{s}+p\mathbf{I}_k}), k = 1, 2, \ldots, n$.

Step 1: Assume $g = c_1m_1 + \cdots + c_tm_t$, where $m_i = x_1^{e_{i,1}} \cdots x_n^{e_{i,n}}, i = 1, \ldots, t$.
Step 2: For $i = 1, 2, \ldots, n$, let $h_i = 0$;
Step 3: For $i = 1, 2, \ldots, t$ do
a: Let $d = 0$.
b: For $k = 1, 2, \ldots, n$, let $d = d + e_{i,k}s_k$.
c: For $k = 1, 2, \ldots, n$, let $h_k := h_k + c_i x^{d + e_{i,k}p}$.
Step 4: Return $h_i, i = 1, 2, \ldots, n$;

Lemma 7. *The complexity of Algorithm 3 is $O^{\sim}(nt\log(p + s_{\max}) + nt\log\deg f)$ bit operations and $O(nt)$ ring operations in $\mathcal{R}$, where $s_{\max} = \max\{s_1, \ldots, s_n\}$.*

Proof. In **b** of Step 3, d is the degree of $m_i(x^{\mathbf{s}})$. In **c**, since $\deg(m_i(x^{\mathbf{s}+p\mathbf{I}_k})) = \deg(m_i(x^{\mathbf{s}})) + pe_{i,k}$, h_k is $f(x^{\mathbf{s}+p\mathbf{I}_k})$ after finishing Step 3. So the correctness is proved. Now we analyse the complexity. In **b** of Step 3, it needs $O(nt)$ arithmetic operations in $\mathbb{Z}$. Since $\deg(m_i(x^{\mathbf{s}}))$ is $O(s_{\max} \deg f)$, the bit operation is $O(nt \log(s_{\max} \deg f))$. In **c**, it needs $O^{\sim}(nt \log(p \deg f + s_{\max} \deg f))$ bit operations and at most $O(nt)$ arithmetic operations in $\mathcal{R}$. □

Now we give an algorithm which interpolates at least half of the terms.

Algorithm 4 (HalfPoly).

Input: • A black-box procedure $\mathcal{B}_f$ that computes $f \in \mathcal{R}[x_1, \ldots, x_n]$.
- An approximation polynomial $f^* \in \mathcal{R}[x_1, \ldots, x_n]$ to f.
- Term bounds $T \geq \max(\#f, \#f_1), T_1 \geq \#(f - f^*)$ and $T \geq T_1$.
- Degree bound $D \geq \max(\deg f, \deg f^*)$.
- A tolerance ν such that $0 < \nu < 1$.

Output: With probability $\geq 1 - \nu$, return a polynomial h such that $\#(f - f^* - h) \leq \lfloor \frac{T_1}{2} \rfloor$.

Step 1: Let $\ell = \lceil 32 \ln(T_1 \nu^{-1}) \rceil$, $N = \max\{31 \lfloor (T_1 - 1) \log_2 D \rfloor, 1\}$. Find the first N primes $\{p_1, p_2, \ldots, p_N\}$ such that $p_i \geq 32(T_1 - 1)$.

Step 2: For $i = 1, \ldots, \ell$, randomly choose p_{α_i} in $\{p_1, \ldots, p_N\}$, then choose $\mathbf{s}_i \in \mathbb{Z}_{p_{\alpha_i}}^n$ uniformly at random. Deleting the repeated numbers, we still denote these vectors as $(p_{\alpha_1}, \mathbf{s}_1), (p_{\alpha_2}, \mathbf{s}_2), \ldots, (p_{\alpha_\ell}, \mathbf{s}_\ell)$.

Step 3: For $i = 1, 2, \ldots, \ell$, compute $f(x^{\mathbf{s}_i})$ from $\mathcal{B}_f$ by a given univariate interpolation algorithm with degree bound $\|\mathbf{s}_i\|_\infty D$ and term bound T. Let $f_i = f(x^{\mathbf{s}_i}) - f^*(x^{\mathbf{s}_i})$ and $f_i^{\mathbf{mod}} = f_i \ \mathbf{mod}\ (x^{p_{\alpha_i}} - 1)$.

Step 4: Find j_0 such that $\#f_{j_0}^{\mathbf{mod}} = \max\{\#f_i^{\mathbf{mod}} | i = 1, 2, \ldots, \ell\}$.
If $\#f_{j_0}^{\mathbf{mod}} \geq T_1$, return failure.

Step 5: For $k = 1, 2, \ldots, n$, find $f(x^{\mathbf{s}_{j_0} + p_{\alpha_{j_0}} \mathbf{I}_k})$ from $\mathcal{B}_f$ by the given univariate interpolation algorithm with degree bound $\|\mathbf{s}_{j_0} + p_{\alpha_{j_0}} \mathbf{I}_k\|_\infty D$ and term bound T. Let $\{f_1^*, f_2^*, \ldots, f_n^*\} = \mathbf{PolySubs}(f^*, \mathbf{s}_{j_0}, p_{\alpha_{j_0}})$.
Let $g_k = f(x^{\mathbf{s}_{j_0} + p_{\alpha_{j_0}} \mathbf{I}_k}) - f_k^*$.

Step 6: Let $\mathrm{TS} = \mathbf{TSTerms}(f_{j_0}^{\mathbf{mod}}, f_{j_0}, g_1, g_2, \ldots, g_n, p_{\alpha_{j_0}}, \mathbf{s}_{j_0}, D)$.

Step 7: Return $h = \sum_{b \in \mathrm{TS}} b$.

Lemma 8. *Algorithm 4 computes h such that $\#(f - f^* - h) \leq \lfloor \frac{T_1}{2} \rfloor$ with probability $\geq 1 - \nu$. The algorithm needs*

- *$O(n + \log T + \log \frac{1}{\nu})$ interpolations of univariate polynomials of degree $O^{\sim}(TD)$ and sparseness $\leq T$.*
- *$O^{\sim}(nT \log \frac{1}{\nu})$ additional ring operations and $O^{\sim}(nT \log D \log \frac{1}{\nu})$ additional bit operations.*

Proof. We first show that Algorithm 4 returns the polynomial h such that $\#(f - f^* - h) \leq \lfloor \frac{T_1}{2} \rfloor$ with probability $1 - \nu$. In Step 1, Step 2 and Step 4, by Theorem 1, with probability $1 - \nu$, $\mathcal{C}_{f - f^*}(p_{\alpha_{j_0}}, \mathbf{s}_{j_0}) \leq \lfloor \frac{3}{8} T_1 \rfloor$. If j_0 satisfies $\mathcal{C}_{f - f^*}(p_{\alpha_{j_0}}, \mathbf{s}_{j_0}) \leq$

$\lfloor \frac{3}{8}T_1 \rfloor$, then by Lemma 5, there are at most $\lfloor \frac{3}{8}T_1 \rfloor$ terms in $f - f^*$ but not in h. Since the terms of h which are not in $f - f^*$ come from at least three terms in $f - f^*$, then there are at most $\frac{1}{3}\lfloor \frac{3}{8}T_1 \rfloor$ terms of h not in $f - f^*$. So $\#(f - f^* - h) \leq \lfloor \frac{3}{8}T_1 \rfloor + \frac{1}{3}\lfloor \frac{3}{8}T_1 \rfloor \leq \frac{1}{2}T_1$. So we have $\#(f - f^* - h) \leq \lfloor \frac{1}{2}T_1 \rfloor$. The first part is proved.

Now we analyse the complexity. In Step 1, use the sieve of Eratosthenes [29, Them.18.10], the cost of finding the N primes bigger than $32(T_1 - 1)$ is $O^{\sim}(T_1 \log D)$ bit operations.

In Step 2, since probabilistic machines flip coins to decide binary digits, each of these random choices can be simulated with a machine with complexity $O(n \log(T_1 \log D))$. So the complexity of Step 2 is $O(n \log^2 T_1 + n \log T_1 \log \frac{1}{\nu} + n \log T_1 \log\log D + n \log\log D \log \frac{1}{\nu})$ bit operations.

In Step 3, since p_{α_i} is $O^{\sim}(T_1 \log D)$, the degree of $f(x^{\mathbf{s}_i})$ is $O^{\sim}(\|\mathbf{s}_i\|_\infty D) = O^{\sim}(T_1 D)$. So in Step 3, we query $O(\log T_1 + \log \frac{1}{\nu})$ polynomials of degree $O^{\sim}(T_1 D)$. In order to obtain $f^*(x^{\mathbf{s}_i})$, it needs $O(\ell n T)$ ring operations and $O^{\sim}(\ell n T \log D)$ bit operations. In order to obtain f_i, it needs $O(\ell T)$ ring operations in $\mathcal{R}$ and $O^{\sim}(\ell T \log D)$ bit operations. In order to obtain the $f_i^{\mathbf{mod}}$, it needs $O^{\sim}(\ell T_1 \log D)$ bit operations and $O(\ell T_1)$ ring operations.

So it still needs $O^{\sim}(nT \log \frac{1}{\nu} + T_1 \log \frac{1}{\nu})$ ring operations and $O^{\sim}(nT \log D \log \frac{1}{\nu} + T_1 \log D \log \frac{1}{\nu})$ bit operations.

In Step 4, we find the integer j_0. Since $\#f_i^{\mathbf{mod}} \leq T_1$, it needs at most $O^{\sim}(T_1 \log \frac{T_1}{\nu})$ bit operations to compute all $\#f_i^{\mathbf{mod}}, i = 1, 2, \ldots, \ell$. Find j_0 needs $O^{\sim}(\ell \log T_1)$ bit operations. So the bit complexity of Step 4 is $O^{\sim}(T_1 \log \frac{1}{\nu})$.

In Step 5, since the degree of $f(x^{\mathbf{s}_{j_0} + p_{\alpha_{j_0}} \mathbf{I}_k})$ is $O^{\sim}(T_1 D)$, it queries $O(n)$ polynomials of degrees $O^{\sim}(T_1 D)$. By Lemma 7, it needs $O^{\sim}(nT \log D)$ bit operations and $O(nT)$ arithmetic operations in $\mathcal{R}$ to obtain $\{f_1^*, f_2^*, \ldots, f_n^*\}$.

In Step 6, by Lemma 6, the complexity is $O(nT_1)$ ring operations in $\mathcal{R}$ and $O^{\sim}(nT_1 \log D)$ bit operations. Since $T \geq T_1$, the lemma is proved. □

We now give the complete interpolation algorithm.

Algorithm 5 (MulPolySI).

Input: A Black-box procedure $\mathcal{B}_f$ that computes $f \in \mathcal{R}[\mathbb{X}]$, $T \geq \#f$, $D \geq \deg f$, and $\mu \in (0, 1)$.
Output: Return f with probability $\geq 1 - \mu$, or failure.

Step 1: Let $h = 0, T_1 = T, \nu = \frac{\mu}{\lceil \log_2 T \rceil + 1}$.
Step 2: While $T_1 > 0$ do
  a: Let $g = \mathbf{HalfPoly}(\mathcal{B}_f, h, T, T_1, D, \nu)$.
  b: Let $h = h + g$, $T_1 = \lfloor \frac{T_1}{2} \rfloor$.
Step 3: Return h.

Theorem 6. *Algorithm 5 computes f with probability $\geq 1 - \mu$. The algorithm needs*

- $O(n \log T + \log^2 T + \log T \log \frac{1}{\mu})$ *interpolations of univariate polynomials with degree* $O^{\sim}(TD)$ *and sparseness* $\leq T$.
- $O^{\sim}(nT \log \frac{1}{\mu})$ *additional ring operations and* $O^{\sim}(nT \log D \log \frac{1}{\mu})$ *additional bit operations.*

Proof. In **a** of Step 2, since $\#(f-h) \leq T_1$, by Lemma 8, $\#(f-h-g) \leq \lfloor \frac{T_1}{2} \rfloor$ with probability $\geq 1-\nu$. Then, Step 2 will run at most $k = \lceil \log_2 T \rceil + 1$ times and return the correct f with probability $\geq (1-\nu)^k \geq 1-\mu$. The first part is proved.

Now we analyse the complexity. It is easy to see that the complexity is dominated by Step 2. In Step 2, we call at most $O(\log T)$ times Algorithm 4. Since the terms and degrees of $f-h$ are respectively bounded by T and D, by Lemma 8, it needs $O(n \log T + \log^2 T + \log T \log \frac{1}{\nu})$ queries of degree $O^{\sim}(TD)$, $O^{\sim}(nT \log \frac{1}{\nu})$ additional ring operations and $O^{\sim}(nT \log D \log \frac{1}{\nu})$ additional bit operations.

Since $\nu = \frac{\mu}{\lceil \log_2 T \rceil + 1}$, we have proved the theorem. □

Corollary 1. *Set* $\mu = 1/4$. *Then Algorithm 5 computes* f *with probability at lest* $\frac{3}{4}$. *The algorithm needs*

- $O(n \log T + \log^2 T)$ *queries of univariate polynomials with degree* $O^{\sim}(TD)$ *and sparseness* $\leq T$.
- $O^{\sim}(nT)$ *additional ring operations and* $O^{\sim}(nT \log D)$ *additional bit operations.*

3 Sparse Interpolation over Finite Fields

In this section, we give a sparse interpolation algorithm for black-box multivariate polynomials over general finite fields. We first give a univariate Ben-Or and Tiwari algorithm over finite fields and then combine with Algorithm 5 to give a multivariate interpolation algorithm.

3.1 The Ben-Or and Tiwari Sparse Interpolation Algorithm

Following [3], we give a brief introduction to the multivariate Ben-Or and Tiwari sparse interpolation algorithm over $\mathbb{C}$.

Let $f(x_1, \ldots, x_n) = c_1 m_1 + \cdots + c_t m_t \in \mathbb{C}[\mathbb{X}]$ be the polynomial to be interpolated, where $m_i = x_1^{e_{i,1}} \ldots x_n^{e_{i,n}}$ are distinct monomials, c_i are non-zero coefficients, and $t = \#f$ is the number of terms in f. We assume that f is a *black-box*, which means, for $\forall\ (q_1, \ldots, q_n) \in \mathbb{C}^n$, we can obtain the value $f(q_1, \ldots, q_n)$. Note that c_i, m_i, t are not known. In order to determine f uniquely, the algorithm needs as input an upper bound $\tau + 1 \geq t$ on the number of terms in f.

The algorithm proceeds in two stages. The monomials m_i are determined first using an auxiliary polynomial $\zeta(z)$. Once the m_i are known, the coefficients c_i can be obtained easily.

We first determine m_i. Let $v_i = p_1^{e_{i,1}} \dots p_n^{e_{i,n}}$ denote the value of the monomial m_i at $(p_1, \dots, p_n)$, where p_i is the i-th prime number. Clearly, different monomials evaluate to different values under this evaluation. Let $a_0, a_1, \dots, a_{2\tau+1}$ be the values of f at the $2(\tau+1)$ points $\mathbf{p}_i = (p_1^i, \dots, p_n^i), i = 0, 1, \dots, 2\tau+1$, that is, $a_i = \sum_{j=1}^{t} c_j v_j^i$.

The auxiliary polynomial $\zeta(z)$ is defined as follows.

$$\zeta(z) = \prod_{i=1}^{t}(z - v_i) = z^t + \zeta_{t-1}z^{t-1} + \cdots + \zeta_1 z + \zeta_0. \tag{10}$$

Consider the sum $\sum_{i=1}^{t} c_i v_i^j \zeta(v_i) = \sum_{k=0}^{t-1} \zeta_k (c_1 v_1^{k+j} + c_2 v_2^{k+j} + \cdots + c_t v_t^{k+j}) + (c_1 v_1^{t+j} + c_2 v_2^{t+j} + \cdots + c_t v_t^{t+j}) = a_j\zeta_0 + a_{j+1}\zeta_1 + \cdots + a_{j+t-1}\zeta_{t-1} + a_{j+t}$ for $j = 0, \dots, t-1$. Since $\zeta(v_i) = 0$, for $0 \le j \le t-1$, we have

$$a_j\zeta_0 + a_{j+1}\zeta_1 + \cdots + a_{j+t-1}\zeta_{t-1} + a_{j+t} = 0. \tag{11}$$

This is a Toeplitz system $T_{t-1,t-1}\hat{\zeta}_{t-1} = \hat{t}_{2t-1,t-1}$ where

$$T_{u,v} = \begin{pmatrix} a_u & a_{u+1} & \cdots & a_{u+v} \\ a_{u-1} & a_u & \cdots & a_{u+v-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{u-v} & a_{u-v+1} & \cdots & a_u \end{pmatrix}$$

$\hat{\zeta}_v = (\zeta_0, \zeta_1, \dots, \zeta_v)^\tau$, $\hat{t}_{u,v} = -(a_u, a_{u-1}, \dots, a_{u-v})^\tau$. This system is non-singular as can be seen from the factorization.

$$T_{t-1,t-1} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ v_1 & v_2 & \cdots & v_t \\ \vdots & \vdots & \ddots & \vdots \\ v_1^{t-1} & v_2^{t-1} & \cdots & v_t^{t-1} \end{pmatrix} \begin{pmatrix} c_1 & 0 & \cdots & 0 \\ 0 & c_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_t \end{pmatrix} \begin{pmatrix} 1 & v_1 & \cdots & v_1^{t-1} \\ 1 & v_2 & \cdots & v_2^{t-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & v_t & \cdots & v_t^{t-1} \end{pmatrix} \tag{12}$$

Since the v_i are distinct, the two Vandermonde matrices are nonsingular and as no c_i is zero, the diagonal matrix is nonsingular, too. If the input value of the upper bound $\tau+1$ is greater than t, then the coefficients c_k, for $k > t$, can be regarded as zero and the resulting system $T_{\tau,\tau}$ would be singular.

Lemma 9. *[3] If t is the exact number of terms in f, then*

a) $T_{i,t-1}$ is non-singular for all $i \ge t-1$.
b) $T_{i,t+j}$ is singular for all $i \ge t-1, j \ge 0$.

By Lemma 9, when considering $2\tau+2$ values $a_0, \dots, a_{2\tau+1}$ of f, the coefficients of $\zeta(z)$ can be uniquely recovered from the system $T_{\tau,\tau}\hat{\zeta}_\tau = \hat{t}_{2\tau+1,\tau}$. By finding the roots $v_i = p_1^{e_{i,1}} \dots p_n^{e_{i,n}}$ of $\zeta(z)$, the monomials m_i can be recovered.

By choosing the first t evaluations $a_0, \dots, a_{t-1}$ of f, we obtain the following transposed Vandermonde system $A\hat{c} = \hat{a}$ for the coefficients of f, where

$$A = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ v_1 & v_2 & \cdots & v_t \\ \vdots & \vdots & \ddots & \vdots \\ v_1^{t-1} & v_2^{t-1} & \cdots & v_t^{t-1} \end{pmatrix}, \hat{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_t \end{pmatrix}, \hat{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} \tag{13}$$

The deterministic Ben-Or and Tiwari's algorithm over $\mathbb{Z}$ needs $O(T)$ evaluations of f plus $O(nT^2d)$ $\mathbb{Z}$-operations and the height of the data is $O(Td)$ [3], where $d = \deg f$.

If the coefficients of the polynomials are from a finite field, then it is difficult to find the exponents from $v_i = p_1^{e_{i,1}} \dots p_n^{e_{i,n}}$, which is a multi-variate discrete logarithm problem.

3.2 Univariate Ben-Or and Tiwari Algorithm over Finite Field

In this section, we give a modified univariate Ben-Or and Tiwari algorithm over the finite field $\mathbb{F}_q$. Assume $f(x) = \sum_{i=1}^{t} c_i m_i \in \mathbb{F}_q[x], D \geq \deg f$. Since $f(x)$ is univariate, $\#f \leq D$. We consider two cases: $q > D$ or $q \leq D$.

First, consider the case $q > D$. Let ω be a primitive element of $\mathbb{F}_q$. Assume $m_i = x^{d_i}$ and denote $v_i = \omega^{d_i}$. Let $a_i = \sum_{j=1}^{t} c_j v_j^i, i = 0, 1, \dots, 2\tau + 1$. $T_{t-1,t-1}$ still can be factored as (12). Since ω is a primitive element of $\mathbb{F}_q$ and $q > D$, $v_i \neq v_j$ when $i \neq j$. So the two Vandermonde matrices in (12) are nonsingular and Lemma 9 is still correct. Now we can give the algorithm.

Algorithm 7 (UniBoTFq).

Input: A black-box procedure $\mathcal{B}_f$ to compute $f(x) \in \mathbb{F}_q[x]$, $\tau + 1 \geq \#f$, and $D \geq \deg f$.
Output: The polynomial $f = \sum_{i=1}^{t} c_i m_i$.

Step 1: Let ω be a primitive element of $\mathbb{F}_q$. Evaluate f at the $2(\tau + 1)$ points ω^i, $i = 0, \dots, 2\tau + 1$. Let $a_i, i = 0, \dots, 2\tau + 1$ be the corresponding values.
Step 2: Solve the Toeplitz system $T_{\tau,\tau}\hat{\zeta}_\tau = \hat{t}_{2\tau+1,\tau}$ (or the largest non-singular subsystem $T_{j,2\tau-j}\,\widehat{\zeta}_{2\tau-j} = \hat{t}_{2\tau+1,2\tau-j}$ of $T_{\tau,\tau}$, where j is the smallest positive integer that makes $T_{j,2\tau-j}$ non-singular) to obtain the polynomial $\zeta(z) = \sum_{i=0}^{t} \zeta_i z^i$.
Step 3: Find the monomial set M of f. $M = \emptyset$. For $i = 0, 1, \dots, D$, compute ω^i and if $\zeta(\omega^i) = 0$ then let $M = \{x^i\} \cup M$.
Step 4: Find the coefficients c_i by solving the transposed Vandermonde system $A\hat{c} = \hat{a}$ in (13).

Lemma 10. *If $q > D$, Algorithm 7 is correct and it needs $2(\tau + 1)$ evaluations of f plus $O^{\sim}(D \log q)$ bit operations.*

Proof. The correctness comes from Lemma 9. Now we analyse the complexity. Due to the fast integer and polynomial multiplication algorithms [29], one can perform an arithmetic operation in $\mathbb{F}_q$ in $O^{\sim}(\log q)$ bit operations. In Step 1, it

needs $O(\tau \log q)$ bit operations to obtain ω^i, $i = 1, 2, \ldots, 2\tau + 1$. In Step 2, it needs $O(M(\tau) \log \tau \log q)$ bit operations, where $M(\tau) = \tau \log(\tau) \log\log(\tau)$ [3].

In Step 3, computing $\omega^i, i = 0, 1, \ldots, D$ needs $O(D \log q)$ bit operations. Then we evaluate $\zeta(\omega^i), i = 0, 1, \ldots, D$, by fast multi-point evaluation method [29, Them.10.6], which needs $O(\frac{D}{T} M(T) \log T \log q) = O^{\sim}(D \log T \log q)$ bit operations, where $T = \tau + 1$.

In Step 4, it needs $O(M(t) \log t \log q)$ bit operations [3]. So the complexity of the total algorithm is $O^{\sim}(D \log T \log q + T \log q) = O^{\sim}(D \log q)$ bit operations, since $\#f \leq D$. □

Second, consider the case $q < D$. We need evaluate the polynomial in an extended field of $\mathbb{F}_q$. We extends $\mathbb{F}_q$ into $\mathbb{F}_{q^m}$ such that $q^m \geq D + 1$, where $m = \lceil \frac{\log(D+1)}{\log q} \rceil$. Due to the fast integer and polynomial multiplication algorithms [29], one can perform an arithmetic operation in $\mathbb{F}_{q^m}$ in $O^{\sim}(m \log q) = O^{\sim}(\log D)$ bit operations, since $m = \lceil \frac{\log(D+1)}{\log q} \rceil$.

Now we can extend Algorithm 7 into the case $q \leq D$. The only change is to replace the primitive element of $\mathbb{F}_q$ by a primitive element of $\mathbb{F}_{q^m}$ in Step 1. Similar to the proof of Lemma 10, the complexity of the algorithm is $O^{\sim}(D \log Tm \log q + Tm \log q)$, which is $O^{\sim}(D \log T + T \log D) = O^{\sim}(D \log D) = O^{\sim}(D)$ bit operations. We thus have

Lemma 11. *If $q \leq D$, Algorithm 7 needs $2(\tau + 1)$ evaluations of f plus $O^{\sim}(D)$ bit operations.*

Following Lemmas 10 and 11, we have

Theorem 8. *Let f be a black-box univariate polynomial in $\mathbb{F}_q[x]$ with $T \geq \#f$ and $D \geq \deg f$. We can compute f with $O(T)$ evaluations of f plus $O^{\sim}(D \log q)$ bit operations.*

Remark 1. In Step 3 of Algorithm 7, we may follow the original Ben-Or and Tiwari algorithm to find the exponents. First, find the roots v_i of $\zeta(z) = 0$, which costs $O^{\sim}(t \log^2 q)$ bit operations [29] for $t = \#f$. Second, solve the discrete logarithm problem $v_i = \omega^{e_i}$ to find the exponents e_i, which costs $O^{\sim}(\sqrt{D} \log q)$ bit operations [30]. Therefore, the total complexity of the algorithm is $O^{\sim}(T \log^2 q + T\sqrt{D} \log q)$ bit operations plus $O(T)$ evaluations.

3.3 Multivariate Polynomial Interpolation over Finite Fields

Combing the reduction algorithm given in Sect. 2 and the univariate interpolation given in Sect. 3.2, we give a multivariate interpolation algorithm over finite fields.

Theorem 9. *Let $f \in \mathbb{F}_q[\mathbb{X}]$ be a black-box polynomial. Given $T \geq \#f$ and $D \geq \deg f$, with probability greater than $\frac{3}{4}$, one can find f using $O^{\sim}(nTD \log q)$ bit operations plus $O^{\sim}(nT)$ evaluations of f.*

Proof. We use the Algorithm 5 to compute f and use Algorithm 7 for univariate polynomial interpolation in Step 3 and Step 5 of Algorithm 4.

The complexity consists of two parts. By Corollary 1, we needs $O(n \log T + \log^2 T)$ queries of univariate polynomials with degree $O^{\sim}(TD)$ and sparseness $\leq T$. Then by Theorem 8, we need $O^{\sim}((n \log T + \log^2 T)T) = O^{\sim}(nT)$ evaluations of f and $O^{\sim}((n \log T + \log^2 T)(TD \log q)) = O^{\sim}(nTD \log q)$ bit operations to query these univariate polynomials.

By Corollary 1, we needs additional $O^{\sim}(nT)$ operations in $\mathbb{F}_q$ if $q > D$ (or in $\mathbb{F}_{q^m}$ if $q < D$ for $m = \lceil \frac{\log(D+1)}{\log q} \rceil$) and $O^{\sim}(nT \log D)$ bit operations. $O^{\sim}(nT)$ operations in $\mathbb{F}_q$ costs $O^{\sim}(nT \log q)$ bit operations. $O^{\sim}(nT)$ operations in $\mathbb{F}_{q^m}$ costs $O^{\sim}(nT \log D)$ bit operations. Therefore, the query of f is the dominating step and the bit complexity of the algorithm is $O^{\sim}(nTD \log q)$. □

Remark 2. If using the original Ben-Or and Tiwari algorithm mentioned in Remark 1 to interpolation the univariate polynomials, the total complexity of our algorithm is $O^{\sim}(nT^{1.5}\sqrt{D} \log q + nT \log^2 q)$ bit operations.

Remark 3. Let $f \in \mathbb{F}_q[\mathbb{X}]$ be a black-box polynomial and q a prime. If quantum algorithms can be used, the quantum complexity of finding f is $O^{\sim}(nT \log^2 q)$ plus $O^{\sim}(nT)$ evaluations of f and $O^{\sim}(nT)$ black-box evaluations for solving the discrete logarithm problem.

We need to change Step 3 of Algorithm 7 as follows:

(1) Find the roots v_i of $\zeta(z)$, which costs an expected $O^{\sim}(T \log^2 q)$ bit operations [29, p.368].
(2) Solve the discrete logarithm problem $v_i = \omega^{e_i}$ **mod** q to find e_i using Shor's quantum algorithm, which costs $O^{\sim}(T \max\{\log^2 D, \log^2 q\})$ plus T black-box evaluations [31, p.238].

By Corollary 1, the total complexity is $O^{\sim}(nT \max\{\log^2 D, \log^2 q\})$.

4 Experimental Results

In this section, practical performances of the interpolation algorithm over finite fields given in Remark 2 will be reported. The algorithm uses Algorithm 5 to reduce multivariate interpolation to univariate interpolation and uses Algorithm 7 for univariate polynomial interpolation. In Algorithm 7, we use the Berlekamp-Massey algorithm to solve the Toeplitz systems, use the command *Roots* in Maple to find the roots, and use the command *mlog* in Maple to solve the discrete logarithm problem.

The data are collected on a desktop with Windows system, 3.60 GHz Core *i*7-4790 CPU, and 8GB RAM memory. The implementations in Maple can be found in

`http://www.mmrc.iss.ac.cn/~xgao/software/rkron.zip`

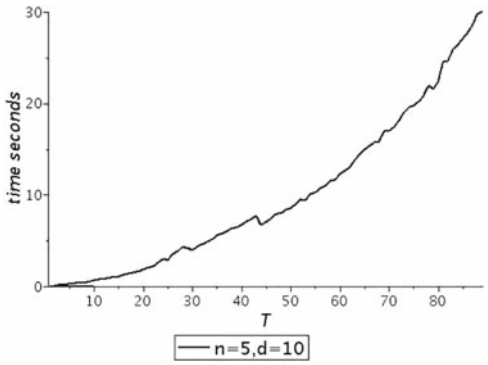

Fig. 1. Average times with varying T

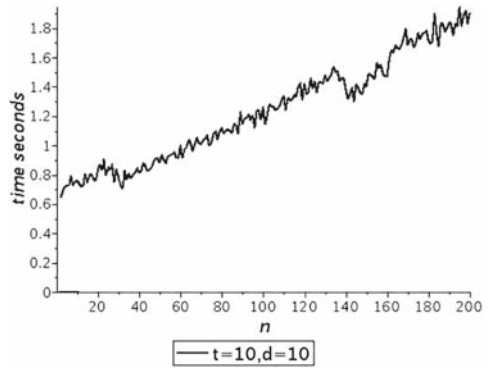

Fig. 2. Average times with varying n

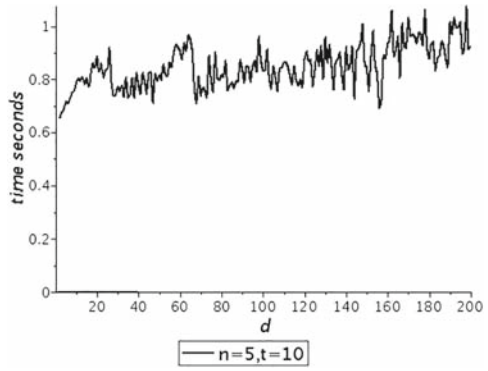

Fig. 3. Average times with varying d

We randomly construct five polynomials over the finite field $\mathbb{F}_q$, then regard them as black-box polynomials and reconstruct them with the algorithm. The actual size and degree of the polynomials are used as the term bound and degree bound, respectively. The average times are collected. In our testing, we fix $q = 30000000001$ and use the primitive element 29 of $\mathbb{F}_q$.

The results are shown in Figs. 1, 2, 3. In each figure, two of the parameters n, T, D are fixed and one of them is variant. These data are basically in accordance with the complexity $O^{\sim}(nT^{1.5}\sqrt{D}\log q + nT\log^2 q)$ of the algorithm.

References

1. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Ng, E.W. (ed.) Symbolic and Algebraic Computation. LNCS, vol. 72, pp. 216–226. Springer, Heidelberg (1979). https://doi.org/10.1007/3-540-09519-5_73
2. Ben-Or, M., Tiwari, P.: A deterministic algorithm for sparse multivariate polynomial interpolation. In: Proceedings STOC 1988, pp. 301–309. ACM Press (1988)
3. Kaltofen, E.L., Lakshman, Y.N.: Improved sparse multivariate polynomial interpolation algorithms. In: Proceedings ISSAC 1988, pp. 467–474. Springer-Verlag (1988)
4. Zippel, R.: Interpolating polynomials from their values. J. Symb. Comp. **9**, 375–403 (1990)
5. Lakshman, Y.N., Saunders, B.D.: Sparse polynomial interpolation in nonstandard bases. SIAM J. Comput. **24**(2), 387–397 (1995)
6. Klivans, A.R., Spielman, D.: Randomness efficient identity testing of multivariate polynomials. In: Proceedings STOC 2001, pp. 216–223. ACM Press (2001)
7. Grigoriev, D.Y., Karpinski, M., Singer, M.F.: Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. SIAM J. Comput. **19**, 1059–1063 (1990)
8. Giesbrecht, M., Roche, D.S.: Diversification improves interpolation. In: Proceedings ISSAC 2011, pp. 123–130, ACM Press (2011)
9. Huang, M.D.A., Rao, A.J.: Interpolation of sparse multivariate polynomials over large finite fields with applications. J. Algorithms **33**, 204–228 (1999)
10. Javadi, S.M.M., Monagan, M.: Parallel sparse polynomial interpolation over finite fields. In: Proceedings PASCO 2010, pp. 160–168. ACM Press (2010)
11. Kaltofen, E.L., Lee, W.S.: Early termination in sparse interpolation algorithms. J. Symbolic Comput. **36**, 365–400 (2003)
12. Hao, Z., Kaltofen, E.L., Zhi, L.: Numerical sparsity determination and early termination. In: Proceedings ISSAC 2017, pp. 247–254. ACM Press (2016)
13. Giesbrecht, M., Labahn, G., Lee, W.: Symbolic-numeric sparse interpolation of multivariate polynomials. In: Proceedings ISSAC 2006, pp. 116–123. ACM Press (2006)
14. Kaltofen, E.L., Lee, W.S., Yang, Z.: Fast estimates of hankel matrix condition numbers and numeric sparse interpolation. In: SNC 2011, pp. 130–136. ACM Press (2011)
15. Cuyt, A., Lee, W.S.: A new algorithm for sparse interpolation of multivariate polynomials. Theor. Comput. Sci. **409**(2), 180–185 (2008)
16. Giesbrecht, M., Roche, D.S.: Interpolation of shifted-lacunary polynomials. Comput. Complex. **19**(3), 333–354 (2010)

17. Bläser, M., Jindal, G.: A new deterministic algorithm for sparse multivariate polynomial interpolation. In: Proceedings ISSAC 2014, pp. 51–58. ACM Press (2014)
18. Arnold, A., Giesbrecht, M., Roche, D.S.: Faster sparse multivariate polynomial interpolation of straight-line programs. J. Symbolic Comput. **75**, 4–24 (2016)
19. Arnold, A., Giesbrecht, M., Roche, D.S.: Faster sparse interpolation of straight-line programs. In: Gerdt, V.P., Koepf, W., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2013. LNCS, vol. 8136, pp. 61–74. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-02297-0_5
20. Arnold, A., Roche, D.S.: Multivariate sparse interpolation using randomized Kronecker substitutions. In: ISSAC 2014, pp. 35–42. ACM Press (2014)
21. Garg, S., Schost, E.: Interpolation of polynomials given by straight-line programs. Theor. Comput. Sci. **410**, 2659–2662 (2009)
22. Huang, Q.L., Gao, X.S.: Faster interpolation algorithms for sparse multivariate polynomials given by straight-Line programs. arXiv 1709.08979v4 (2017)
23. Huang, Q.L.: Sparse polynomial interpolation over fields with large or zero characteristic. In: Proceedings ISSAC 2019, ACM Press, to appear (2019)
24. Avendaño, M., Krick, T., Pacetti, A.: Newton-Hensel interpolation lifting. Found. Comput. Math. **6**(1), 82–120 (2006)
25. Alon, N., Mansour, Y.: Epsilon-discrepancy sets and their application for interpolation of sparse polynomials. Inform. Process. Lett. **54**(6), 337–342 (1995)
26. Mansour, Y.: Randomized interpolation and approximation of sparse polynomials. SIAM J. Comput. **24**(2), 357–368 (1995)
27. Arnold, A.: Sparse polynomial interpolation and testing. PhD Thesis, Waterloo Unversity (2016)
28. Dubhashi, D.P., Panconesi, A.: Concentration of Measure for the Analysis of Randomized Algorithms. Cambridge University Press, Cambridge (2009)
29. Gathen, J., von zur Gerhard, J.: Modern Computer Algebra. Cambridge University Press, Cambridge (1999)
30. Pollard, J.M.: Monte Carlo Methods for Index Computation (mod p). Math. Comput. **32**(143), 918–924 (1978)
31. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2015)