A New Sparse Polynomial GCD by Separating Terms

Qiao-Long Huang
School of Mathematics, Shandong University
Jinan, China
huangqiaolong@sdu.edu.cn

Michael Monagan*
Department of Mathematics, Simon Fraser University
Burnaby, British Columbia, Canada
mmonagan@sfu.ca

ABSTRACT

We propose a new sparse GCD algorithm for multivariate polynomials over finite fields. Our algorithm uses a new type of substitution to recover the terms of the GCD in batches. We present a detailed complexity analysis and experimental results which show that our algorithm is faster than Zippel's GCD algorithm and competitive with the Monagan-Hu GCD algorithm.

CCS CONCEPTS

• **Computing methodologies** → Symbolic and algebraic manipulation; • **Theory of Computation** → *Analysis of Algorithms and Problem Complexity*.

KEYWORDS

Polynomial GCD, Sparse Polynomial, Kronecker Substitution

ACM Reference Format:
Qiao-Long Huang and Michael Monagan. 2024. A New Sparse Polynomial GCD by Separating Terms. In *International Symposium on Symbolic and Algebraic Computation (ISSAC '24), July 16–19, 2024, Raleigh, NC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3666000.3669684

1 INTRODUCTION

Let A and B be polynomials in $\mathbb{Z}[x_1, \ldots, x_n]$ and let $G = \gcd(A, B)$ be their greatest common divisor (GCD). Computing G is a key operation in a Computer Algebra System. One application is to simplify the fraction A/B. GCD computation is interesting because all modifications of the Euclidean algorithm [3, 5] to compute G result in an expression swell where the size of the intermediate polynomials grows exponentially in n the number of variables.

In 1971, Brown [2] solved the intermediate expression swell problem by interpolating $x_2, x_3, \ldots, x_n$ in G from many univariate images of G in x_1. For polynomials of degree d Brown's algorithm uses $O(d^{n-1})$ univariate images which is effective for dense polynomials but not sparse polynomials.

Early sparse GCD algorithms include Zippel's algorithm [24] from 1979 and Wang's EEZ-GCD algorithm [23] from 1980. Zippel's algorithm is currently the main GCD algorithm in Fermat, Magma, Maple and Mathematica. The literature for the polynomial GCD problem is large. Many ideas have been tried. We cite the works [4, 6, 8, 11–17, 19–21]. See also Ch. 7 of [7].

Let $\mathbb{F}_q$ be a finite field with q elements and let $\#G$ denote the number of terms of the polynomial G. In this work we present a new sparse GCD algorithm for $\mathbb{F}_q[x_1, \ldots, x_n]$. The main result of the paper is given below.

Theorem 1.1. *Let A, B be polynomials in $\mathbb{F}_q[x_1, \ldots, x_n]$ with $d = \max_{i=1}^{n} \max(\deg(A, x_i), \deg(B, x_i))$. Then there exists a randomized algorithm that takes as inputs A, B and returns $G = \gcd(A, B)$ with probability $\geq 11/12$, using expected $O^{\sim}(nT_{\text{in}}d \log q \log^3 T_{\text{o}} + n^2 d^2 T_{\text{o}} \log q)$ bit operations where $T_{\text{in}} = \#A + \#B$ and $T_{\text{o}} = \#G$.*

We've implemented our algorithm in Maple with parts of it implemented in C for increased efficiency. We use our algorithm to compute a GCD in $\mathbb{Z}[x_1, \ldots, x_n]$ by computing it modulo primes and using the Chinese remainder theorem to recover the integer coefficients of G. Our benchmarks in Section 7 show that it is faster than the implementations of Zippel's algorithm in Maple and Magma and compares well with the Monagan-Hu algorithm [8, 15].

1.1 Overview of the Algorithm

Let A, B be in $\mathbb{F}_q[x_1, \ldots, x_n]$ and $G = \gcd(A, B)$. Our new algorithm uses substitutions of the form

$$x_i = (\gamma_i z - \alpha_i) y^{s_i} \text{ for } 1 \leq i \leq n$$

where γ_i, α_i are chosen from $\mathbb{F}_q$ and s_i are non-negative integers. The substitution converts a GCD problem in n variables into a bivariate GCD problem in $\mathbb{F}_q[y, z]$. The purpose of the γ_i is to prevent a degree loss in z. For now assume $\gamma_i = 1$ works.

Our algorithm chooses α_i from $\mathbb{F}_q$ at random and distinct. It then chooses s_i from $[0, T)$ at random where T is a parameter of the algorithm that takes on the values $2, 4, 8, 16, \ldots$ until the algorithm succeeds. Suppose q is a large prime and

$$G = x_1^2 + 3x_1x_2 + 2x_2^3 - x_2.$$

Suppose we choose $\alpha = [2, 5]$ and $\mathbf{s} = [2, 3]$. Let $\phi(f) = f((z-2)y^2, (z-5)y^3)$ be our substitution. We have

$$\phi(G) = (z-2)^2 y^4 + 3(z-2)(z-5)y^5 + 2(z-5)^3 y^9 - (z-5)y^3.$$

Notice that each monomial in G mapped to a unique monomial in y. We say $\mathbf{s}$ separated the terms of G. Since $z-2$ and $z-5$ are relatively prime, we can recover all terms in G from $\phi(G)$ by dividing the coefficients of $\phi(G)$ by $z-2$ and $z-5$. If a coefficient factors as $c(z-2)^{d_1}(z-5)^{d_2}$ for a constant c, we recover the term $cx_1^{d_1}x_2^{d_2}$.

If G has t terms, we need $T \sim t^2$ for $\phi(G)$ to be separated with reasonable probability. If $D = \max(\deg A, \deg B)$, this would create a bivariate gcd problem of degree $O(Dt^2)$ in y and D in z which will be expensive to compute for large t. Instead, we use a much

*Corresponding author

ISSAC '24, July 16–19, 2024, Raleigh, NC, USA

ACM ISBN 979-8-4007-0696-7/24/07
https://doi.org/10.1145/3666000.3669684

smaller T and several choices for $\mathbf{s}$. Suppose we choose $\mathbf{s} = [1, 1]$. Let $\phi_1(f) = f((z-2)y^1, (z-5)y^1)$. Then

$$\phi_1(G) = 2(z-5)^3y^3 + (z-2)(4z-17)y^2 - (z-5)y.$$

Since GCDs are unique up to a scalar, the GCD algorithm will normalize $\phi_1(G)$. Suppose it makes $\phi_1(G)$ monic in lexicographical order with $y > z$. Here $\mathrm{LC}(\phi_1(G)) = 2$ so we obtain

$$G_1 := \tfrac{1}{2}\phi_1(G) = (z-5)^3y^3 + \tfrac{1}{2}(z-2)(4z-17)y^2 - \tfrac{1}{2}(z-5)y.$$

The factor $4z - 17$ is divisible by neither $z - 2$ nor $z - 5$ thus at least two monomials collided in the term $(z-2)(4z-17)y^2$. If a coefficient of G_1 factors as $c(z-\alpha_1)^{d_1}(z-\alpha_2)^{d_2}$, it is unlikely it comes from a collision because we chose the α_i randomly from $[0, q)$. Thus x_2^3 and x_2 are monomials in G with high probability. We extract the good terms of G found in G_1 as $G^* := x_2^3 - \frac{1}{2}x_2$.

Our algorithm now tries a new $\mathbf{s}$ and α, say $\mathbf{s} = [2, 1]$ and $\alpha = [2, 5]$. Let $\phi_2(f) = f((z-2)y^2, (z-5)y^1)$. We have

$$G_2 := \phi_2(G) = (z-2)^2y^4 + (z-5)(2z^2 - 17z + 44)y^3 - (z-5)y.$$

This time x_1x_2 and x_2^3 collided under ϕ_2. We could recover one new monomial x_1^2 in G but we can do much better. We use G^* to recover more new monomials from G_2.

Notice that the monomial x_2 is recovered from both G_1 and G_2 but the coefficient is $-\frac{1}{2}$ and -1 respectively. As long as G_1 and G_2 share at least one monomial M, we can scale G^* (or G_2) so that G^* and G_2 have the same coefficient for M. In our example we multiply G^* by 2 so that $2G^* = 2x_2^3 - x_2$. Now we compute

$$H_2 := G_2 - \phi_2(2G^*) = (z-2)^2y^4 + 3(z-2)(z-5)y^3.$$

The terms of H_2 yield **two** new monomials x_1^2 and x_1x_2. We obtain the new terms $G^{**} := x_1^2 + 3x_1x_2$ of G. We set

$$G^* := 2G^* + G^{**} = x_1^2 + 3x_1x_2 + 2x_2^3 - x_2.$$

Since there were no collisions detected in H_2 our algorithm stops and outputs G^*. Otherwise it would try another $\mathbf{s}$.

Our algorithm tries $T = 2, 4, 8, 16, \ldots$ until $\log_2 T$ choices for $\mathbf{s}$ are enough to recover all the terms of G. For example, on one of our benchmarks where G has $n = 9$ variables, $\#G = 996$ terms, and $\deg(G) = 30$, using $T = 64$, it recovered 279 good terms from G_1 then 309, 226, 118, 56, 8 new terms from G_2, G_3, G_4, G_5, G_6.

Consider $G = x_1^2 - x_2^2 + 1$. Suppose $\gamma_i = 1$ so that $x_i = (z-\alpha_i)y^{s_i}$. If $s_1 = s_2$, which happens with probability $1/T$, then $\deg(\phi(G), z) < \deg(G)$ which means we cannot recover x_1^2 and x_2^2. To avoid this we choose γ_i from $\mathbb{F}_q$ at random. Randomized Kronecker substitutions $x_i = y^{s_i}$ were previously studied in [1, 9, 10].

2 DEFINITIONS

Fix the lexicographical monomial order with $x_1 > \cdots > x_n$. For a polynomial $A \in \mathbb{F}_q[x_1, \ldots, x_n]$, denote $\mathrm{LC}(A)$ as the leading coefficient of A. For a polynomial $F \in \mathbb{F}_q[x_1, \ldots, x_n, y]$, denote $\mathrm{LC}(F, y)$ as the leading coefficient of F w.r.t. the main variable y.

Definition 2.1. Let $A, B \in \mathbb{F}_q[x_1, \ldots, x_n]$. Then G is called the greatest common divisor (GCD) of A, B if

(1) G divides A and B,
(2) every common divisor of A and B divides G, and
(3) $\mathrm{LC}(G) = 1$.

We say that A is similar to B, $A \sim_y B$, if $A = a \cdot y^k \cdot B$, where $a \in \mathbb{F}_q^*$ and $k \in \mathbb{Z}$. In particular, if $k = 0$, we use the notation $A \sim B$.

Definition 2.2. Let $f \in \mathbb{F}_q[x_1, \ldots, x_n]$. For $\gamma_i, \alpha_i \in \mathbb{F}_q$, $s_i \in \mathbb{N}$ and new variables y, z we define

$$\phi(f) = f(x_i = (\gamma_i z - \alpha_i)y^{s_i} \text{ for } 1 \le i \le n).$$

Definition 2.3. For $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{N}^n$ define

$$f_{\mathbf{s}} = f(x_i = x_i y^{s_i} \text{ for } 1 \le i \le n) = g_1 y^{d_1} + \cdots + g_k y^{d_k},$$

where $g_i \in \mathbb{F}_q[x_1, \ldots, x_n]$ and $d_1 > \cdots > d_k$. If g_i has only one term cm, where $c \in \mathbb{F}_q^*$ and m is a monomial, then we call it a non-colliding term in $f_{\mathbf{s}}$. Collect all non-colliding terms in $f_{\mathbf{s}}$ and generate the following set

$$\mathrm{NC}(f, \mathbf{s}) = \{\ cm \mid cm \text{ is a non-colliding term in } f_{\mathbf{s}}\}.$$

Define the other terms in $f_{\mathbf{s}}$ as the colliding terms by

$$\mathrm{C}(f, \mathbf{s}) = \{\ cm \mid cm \text{ is a colliding term in } f_{\mathbf{s}}\}.$$

Definition 2.4. Let $f = \sum_{i=1}^{t} a_i M_i(x_1, \ldots, x_n)$ be a polynomial with coefficients a_i and monomials M_i. Define the monomial content $\mathrm{MoCont}(f) = \gcd(M_1, M_2, \ldots, M_t) = \prod_{i=1}^{n} x_i^{d_i}$ for some $d_i \in \mathbb{N}$. We say f is monomial primitive if $\mathrm{MoCont}(f) = 1$.

3 PRELIMINARY RESULTS

Lemma 3.1. *[22, Lemma 6.44] (the Schwartz-Zippel Lemma). Let $\mathcal{R}$ be an integral domain and $A \in \mathcal{R}[x_1, \ldots, x_n]$ be non-zero and with total degree D and let $S \subset \mathcal{R}$ be a finite set. If $\mathbf{a} = (a_1, \ldots, a_n)$ is chosen at random from S^n then* $\mathrm{Prob}[A(\mathbf{a}) = 0] \le \frac{D}{|S|}$.

Lemma 3.2. *[8, Lemma 4] Let $F_1, F_2 \in \mathbb{F}_q[x_1, \ldots, x_n, y]$ with $d = \deg(F_1, y) > 0$ and $\ell = \deg(F_2, y) > 0$. Let $a_d = \mathrm{LC}(F_1, y), b_\ell = \mathrm{LC}(F_2, y)$ and the resultant $R = \mathrm{res}_y(F_1, F_2) \in \mathbb{F}_q[x_1, \ldots, x_n]$. For $\alpha \in \mathbb{F}_q^n$, if $a_d(\alpha) \neq 0$ and $b_\ell(\alpha) \neq 0$ then*

(i) $\deg_y \gcd(F_1(\alpha, y), F_2(\alpha, y)) > 0 \iff R(\alpha) = 0$ *and*
(ii) $\mathrm{res}_y(F_1(\alpha, y), F_2(\alpha, y)) = R(\alpha)$.

Lemma 3.3. *Let $A, B \in \mathbb{F}_q[x_1, \ldots, x_n]$, $G = \gcd(A, B)$. If γ_i, α_i for $1 \le i \le n$ are regarded as variables and $\phi(A), \phi(B), \phi(G) \in \mathbb{F}_q[z, y, \gamma_1, \ldots, \gamma_n, \alpha_1, \ldots, \alpha_n]$. Then $\phi(G) \sim_y \gcd(\phi(A), \phi(B))$.*

Proof. For $f \in \mathbb{F}_q[z, y, \gamma_1, \ldots, \gamma_n, \alpha_1, \ldots, \alpha_n]$, define $\psi(f) = f(\alpha_i = \gamma_i z - x_i y^{-s_i} \text{ for } 1 \le i \le n)$. Directly checking the composite maps $\psi \circ \phi$ and $\phi \circ \psi$, it can be seen that $\psi = \phi^{-1}$.

Now, as $G|A$ and $G|B$, we have $\phi(G)|\phi(A)$ and $\phi(G)|\phi(B)$. So $\phi(G)|\gcd(\phi(A), \phi(B))$. For the opposite direction suppose $P = \gcd(\phi(A), \phi(B))$ and $H = \phi(A)/P$. Then $\psi(P) \cdot \psi(H) = \psi(\phi(A)) = A$. Here $\psi(P) = P(z, y, \gamma_1, \ldots, \gamma_n, \gamma_1 z - x_1 y^{-s_1}, \ldots, \gamma_n z - x_n y^{-s_n})$, which may be a rational function as it may has negative degree in y. Let $k = \deg(\psi(P), y)$. Then $\psi(P)/y^k$ and $\psi(H)y^k$ are polynomials of degree zero in y as A doesn't have y. Thus $\psi(P)/y^k$ divides A. For the same reason, $\psi(P)/y^k$ divides B. Thus $\psi(P)/y^k|\gcd(A, B) = G$. By mapping ϕ, we have $\phi(\psi(P)) \cdot y^{-k}|\phi(G)$, which is $P \cdot y^{-k}|\phi(G)$. □

4 OUR NEW GCD ALGORITHM

Let $A, B \in \mathbb{F}_q[x_1, \ldots, x_n]$ and $G = \gcd(A, B)$. This section presents and analyses three algorithms. (1) Algorithm 1: for any vector $\mathbf{s} \in \mathbb{N}^n$, this algorithm computes the non colliding set $\mathrm{NC}(G, \mathbf{s})$. (2)

Algorithm 2: for an approximation G^* of G, this algorithm computes half of the terms in $G - G^*$ using Algorithm 1. (3) Algorithm 3: this is the main algorithm for computing the $G = \gcd(A, B)$. It calls Algorithm 2 in a loop.

4.1 Computing the non-colliding set

We hope to recover the polynomial G from the bivariate images $\phi(G) = G((\gamma_1 z - \alpha_1)y^{s_1}, \ldots, (\gamma_n z - \alpha_n)y^{s_n})$. The variable y is used to separate the terms, and the variable z is used to compute the exponents of terms.

Let $G = c_1 m_1 + \cdots + c_t m_t \in \mathbb{F}_q[x_1, \ldots, x_n]$. For $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{N}^n$, assuming $G(x_i = x_i y^{s_i} \text{ for } 1 \le i \le n) = g_1 y^{d_1} + \cdots + g_t y^{d_t}$, where $g_i \in \mathbb{F}_q[x_1, \ldots, x_n]$ and d_i's are different degrees. Then

$$\phi(G) = \textstyle\sum_{i=1}^{t} g_i(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n) y^{d_i}.$$

If $\#g_i = 1$ and $g_i = c x_1^{e_1} \cdots x_n^{e_n}$, then $g_i(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n) = c(\gamma_1 z - \alpha_1)^{e_1} \cdots (\gamma_n z - \alpha_n)^{e_n}$. If the α_i/γ_i's are different, then $c(\gamma_1 z - \alpha_1)^{e_1} \cdots (\gamma_n z - \alpha_n)^{e_n}$ is one-to-one corresponding to $c x_1^{e_1} \cdots x_n^{e_n}$ by unique factorization. However, for some g_i even with $\#g_i > 1$, $g_i(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n)$ may still have the form $c(\gamma_1 z - \alpha_1)^{e_1} \cdots (\gamma_n z - \alpha_n)^{e_n}$. The intuitive way to avoid this is to randomly select α_i, γ_i from a larger set. The following theorem indicates the success probability. We first assume $\mathrm{NC}(G, \mathbf{s}) \neq \emptyset$.

Theorem 4.1. *Let $G \in \mathbb{F}_q[x_1, \ldots, x_n]$, $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{N}^n$ and $\mathrm{MoCont}(G) = 1$. Assuming $\mathrm{NC}(G, \mathbf{s}) \neq \emptyset$ and $G(x_1 y^{s_1}, \ldots, x_n y^{s_n}) = g_1 y^{d_1} + \cdots + g_k y^{d_k}$ where d_i's are different and $g_i \neq 0 \in \mathbb{F}_q[x_1, \ldots, x_n]$. If $(\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n)$ is randomly chosen from $\mathbb{F}_q^{2n}$, then with a probability of $\ge 1 - \frac{(3n+1)\|\mathbf{s}\|_\infty D^2 + 3nD + n^2}{q}$, the following hold.*

1. $\mathrm{Cont}(\phi(G), y) = 1$.
2. *If $\#g_i = 1$, then $g_i(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n)$ has the form $c(\gamma_1 z - \alpha_1)^{e_1} \cdots (\gamma_n z - \alpha_n)^{e_n}$ for some $e_i \in \mathbb{N}$.*
3. *If $\#g_i > 1$, then $g_i(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n)$ does not have a form $c(\gamma_1 z - \alpha_1)^{e_1} \cdots (\gamma_n z - \alpha_n)^{e_n}$ for some $e_i \in \mathbb{N}$.*

Proof. Suppose that $\phi(G) = h_1 y^{d_1} + \cdots + h_k y^{d_k}$, where $h_i(z) = g_i(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n)$. Thus $\mathrm{Cont}(\phi(G), y) = \gcd(h_1, \ldots, h_k)$. Since $\mathrm{NC}(G, \mathbf{s}) \neq \emptyset$, there exists a h_i corresponding to a term in $\mathrm{NC}(G, \mathbf{s})$. Therefore $\mathrm{Cont}(\phi(G), y) = (z - \alpha_1/\gamma_1)^{w_1} \cdots (z - \alpha_n/\gamma_n)^{w_n}$ for some $w_i \in \mathbb{N}$. To prove $\mathrm{Cont}(\phi(G), y) = 1$, it suffices to show that $(z - \alpha_i/\gamma_i) \nmid \mathrm{Cont}(\phi(G), y)$ for $i = 1, \ldots, n$. Due to $\mathrm{MoCont}(G) = \gcd(g_1, \ldots, g_k) = 1$, for each x_i, there is a g_{j_i} in $\{g_1, \ldots, g_k\}$ such that $x_i \nmid g_{j_i}$. Substitute $x_i = \gamma_i z - \alpha_i$ into g_{j_i}, then $h_{j_i}(z) = g_{j_i}(x_i = \gamma_i z - \alpha_i \text{ for } 1 \le i \le n)$. Let $\kappa_0 := \prod_{i=1}^{n} \gamma_i^D \cdot h_{j_i}(\frac{\alpha_i}{\gamma_i})$. Claim: if $\kappa_0 \neq 0$, then $\mathrm{Cont}(\phi(G), y) = 1$. This is because that if $\kappa_0 \neq 0$, then $h_{j_i}(\alpha_i/\gamma_i) \neq 0$ for $i = 1, \ldots, n$, which implies that $(z - \alpha_i/\gamma_i) \nmid h_{j_i}(z)$, thus $(z - \alpha_i/\gamma_i) \nmid \mathrm{Cont}(\phi(G), y)$. Therefore $\ell_i = 0$ and $\mathrm{Cont}(\phi(G), y) = 1$. If α_i, γ_i are regarded as variables then $\kappa_0 \in \mathbb{F}_q[\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n]$. Claim: κ_0 is a non-zero polynomial with degree $\le 3nD$. Proof of the claim: we prove the case $i = 1$ that $\gamma_1^D \cdot h_{j_1}(\frac{\alpha_1}{\gamma_1})$ is a non-zero polynomial. Let $g_{j_1} = x_1\theta + \eta$, where $\theta \in \mathbb{F}_q[x_1, \ldots, x_n], \eta \in \mathbb{F}_q[x_2, \ldots, x_n]$ and $\eta \neq 0$. Then $h_{j_1}(\alpha_1/\gamma_1) = \eta(x_\ell = \gamma_\ell(\alpha_1/\gamma_1) - \alpha_\ell \text{ for } 2 \le \ell \le n) = \eta(x_\ell = (\gamma_\ell\alpha_1 - \alpha_\ell\gamma_1)/\gamma_1 \text{ for } 2 \le \ell \le n)$. As $\deg \eta \le D$, then $\gamma_1^D \cdot h_{j_1}(\frac{\alpha_1}{\gamma_1})$ is a non-zero polynomial and the degree of $\gamma_1^D \cdot h_{j_1}(\frac{\alpha_1}{\gamma_1})$ is $\le D + 2\deg g_{j_1} \le 3D$. Other cases for $i = 2, \ldots, n$ can be similarly proven. Thus κ_0 is a non-zero polynomial with degree $\le 3nD$.

Case (2) is always correct. Consider case (3). Without loss of generality assume $\#g_1 > 1$. Let $g_1 = cm \cdot Q$, where $x_i \nmid Q$ for $1 \le i \le n$. Assuming $\beta(z) := Q(x_i = \gamma_i z - \alpha_i \text{ for } 1 \le i \le n) = r_1 z^{u_1} + \cdots + r_t z^{u_t}$ where $u_1 > \cdots > u_t$ and $r_i \in \mathbb{F}_q$. Here r_i depends on the choice of $\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n$. Let $\kappa_1 := r_1 \cdot \prod_{i=1}^{n} \gamma_i^D \cdot \beta(\alpha_i/\gamma_i)$. Claim: if $\kappa_1 \neq 0$, then $g_1(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n)$ does not have the form like $c(\gamma_1 z - \alpha_1)^{e_1} \cdots (\gamma_n z - \alpha_n)^{e_n}$. Proof of claim: it suffices to show that (i) $(z - \alpha_i/\gamma_i) \nmid Q(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n)$, $i = 1, \ldots, n$ and (ii) $Q(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n) \notin \mathbb{F}_q$. $\kappa_1 \neq 0$ implies that $r_1 \neq 0$ and $\beta(\alpha_i/\gamma_i) \neq 0$ for $1 \le i \le n$. First, $r_1 \neq 0$ is enough to prove $Q(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n) \notin \mathbb{F}_q$. Second, $\beta(\alpha_i/\gamma_i) \neq 0$ implies that $(z - \alpha_i/\gamma_i) \nmid \beta(z) = Q(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n)$. The claim is proved.

If α_i, γ_i are regarded as variables then $\kappa_1 \in \mathbb{F}_q[\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n]$. Claim: κ_1 is a non-zero polynomial with degree $\le 3nD + D$. As $\#g_1 > 1$, then $u_1 > 0$ and r_1 is a non-zero polynomial with degree $\le D$. As $x_i \nmid Q$, for the same reason for κ_0, $\gamma_i^D \cdot \beta(\alpha_i/\gamma_i)$ is a non-zero polynomial with degree $\le 3D$. The claim is proved.

Assuming $g_{i_1}, \ldots, g_{i_\ell}$ are all in $G_\mathbf{s}$ with more than one term, for the same reason, there are corresponding nonzero polynomials $\kappa_i \in \mathbb{F}_q[\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n], i = 1, \ldots, \ell$. Then $\kappa_1 \cdots \kappa_\ell \neq 0$ implies that case (3) is correct. Let

$$\Gamma = \prod_{i=1}^{n} \gamma_i \cdot \prod_{1 \le i < j \le n} (\alpha_i\gamma_j - \alpha_j\gamma_i) \cdot \prod_{i=0}^{\ell} \kappa_i \in \mathbb{F}_q[\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n].$$

Therefore, if $\Gamma \neq 0$, then $\gamma_i z - \alpha_i$ are different irreducible polynomials and Cases (1), (2) and (3) are all met. As $\deg \kappa_0 \le 3nD$ and $\deg \kappa_i \le 3nD + D, i = 1, \ldots, \ell$ and $\ell \le \deg(G_\mathbf{s}, y) \le \|\mathbf{s}\|_\infty D$, we have $\deg \Gamma \le (3n+1)\|\mathbf{s}\|_\infty D^2 + 3nD + n^2$. By Lemma 3.1, if $(\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n)$ are randomly chosen from $\mathbb{F}_q^{2n}$, the probability that $\Gamma(\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n) \neq 0$ is $\ge 1 - \frac{(3n+1)\|\mathbf{s}\|_\infty D^2 + 3nD + n^2}{q}$. □

The condition $\mathrm{NC}(G, \mathbf{s}) \neq \emptyset$ is necessary for Theorem 4.1. The following shows a counter-example.

Example 4.2. Suppose $G = x_1^2 x_2 + x_2 x_3^2 + x_1^2 + x_3^2$ and $\mathbf{s} = (1, 1, 1)$. Then $G(x_1 y, x_2 y, x_3 y) = (x_1^2 x_2 + x_2 x_3^2) y^3 + (x_1^2 + x_3^2) y^2 = g_1 y^3 + g_2 y^2$ and $\gcd(g_1, g_2) = x_1^2 + x_3^2$. Now consider $\phi(x_1^2 + x_3^2) = ((\gamma_1^2 + \gamma_3^2) z^2 - 2(\alpha_1\gamma_1 + \alpha_3\gamma_3) z + \alpha_1^2 + \alpha_3^2) y^2$. Since $\mathrm{Prob}[\gamma_1^2 + \gamma_3^2 = 0] \le 2/q$ by Lemma 3.1, $\mathrm{Cont}(\phi(G), y) \neq 1$ with high probability.

4.2 Solving the case $\mathrm{NC}(G, \mathbf{s}) = \emptyset$

Let $G = \gcd(A, B)$, then $G_\mathbf{s} \sim_y \gcd(A_\mathbf{s}, B_\mathbf{s})$ (see Def. 2.3). We can quickly detect situations similar to Example 4.2 based on $A_\mathbf{s}$ and $B_\mathbf{s}$. Suppose that A and B are monomial primitive and

$$A(x_i = x_i y^{s_i} \text{ for } 1 \le i \le n) = A_1 y^{v_1} + \cdots + A_k y^{v_k}, \tag{1}$$

$$B(x_i = x_i y^{s_i} \text{ for } 1 \le i \le n) = B_1 y^{u_1} + \cdots + B_\ell y^{u_\ell}. \tag{2}$$

The main idea comes from the following two facts:

1. If $\gcd(A_1, \ldots, A_k, B_1, \ldots, B_\ell) \neq 1$, then $\mathrm{NC}(G, \mathbf{s}) = \emptyset$;
2. If $\gcd(A_1, \ldots, A_k, B_1, \ldots, B_\ell) = 1$, then $\mathrm{Cont}(\phi(G), y) = 1$ with high probability if α_i, γ_i are randomly selected in $\mathbb{F}_q$.

Lemma 4.3. *Let $A, B \in \mathbb{F}_q[x_1, \ldots, x_n]$ and A and B be monomial primitive. If $\gcd(A_1, \ldots, A_k, B_1, \ldots, B_\ell) \neq 1$, then $\mathrm{NC}(G, \mathbf{s}) = \emptyset$.*

Proof. Denote $H := \gcd(A_1, \ldots, A_k, B_1, \ldots, B_\ell)$. Then $H|A$ and $H|B$. Thus $H|\gcd(A, B) = G$. Therefore $H_{\mathbf{s}}|G_{\mathbf{s}}$. We prove $H_{\mathbf{s}} = H \cdot y^d$ for some $d \in \mathbb{N}$. Assume to a contradiction $H_{\mathbf{s}} = H_1 y^{d_1} + \cdots + H_t y^{d_t}$ where $d_1 > \cdots > d_t$. Since $H|A_1$, let $A_1 = H \cdot \overline{H}$. Then $(A_1)_{\mathbf{s}} = A_1 y^{v_1} = H_{\mathbf{s}} \cdot \overline{H}_{\mathbf{s}} = (H_1 y^{d_1} + \cdots + H_t y^{d_t}) \cdot \overline{H}_{\mathbf{s}}$. Then the number of terms w.r.t y in $A_1 y^{v_1}$ is at least two, a contradiction.

Thus $H \cdot y^d | G_{\mathbf{s}}$. So H divides all the coefficients of $G_{\mathbf{s}}$ w.r.t y. Since A and B are monomial primitive and $H \neq 1$, H has at least two terms, which implies that each coefficient of $G_{\mathbf{s}}$ w.r.t y has at least two terms. Thus there is no non-colliding term in $G_{\mathbf{s}}$. □

Lemma 4.4. *Let $G = \gcd(A, B)$ and let $D = \max(\deg A, \deg B)$. Then there exists a non-zero polynomial Γ with degree $\leq 2D^2 + 2D$, such that if $\Gamma(\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n) \neq 0$ for $\alpha_i, \gamma_i \in \mathbb{F}_q$, then $G(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n) \sim$*
$\gcd(A(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n), B(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n))$.

Proof. If α_i, γ_i's are regarded as variables, then by Lemma 3.3, $\phi(G) \sim_y \gcd(\phi(A), \phi(B))$ for $\mathbf{s} = (0, \ldots, 0)$. As both sides have degree 0 in y, $\phi(G) \sim \gcd(\phi(A), \phi(B))$. Let $\Gamma := \mathrm{LC}(\phi(A), z) \cdot \mathrm{LC}(\phi(B), z) \cdot \mathrm{res}_z(\phi(A)/\phi(G), \phi(B)/\phi(G))$. As the degrees of the coefficients of $\phi(A)$ and $\phi(B)$ in z are at most D, $\deg(\Gamma) \leq D + D + 2D^2$. For $\alpha_i, \gamma_i \in \mathbb{F}_q$, by Lemma 3.2, if $\Gamma(\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n) \neq 0$, then $\phi(G), \phi(A), \phi(B) \in \mathbb{F}_q[z]$ and $\phi(G) \sim \gcd(\phi(A), \phi(B))$. □

Corollary 4.5. *Suppose $\gcd(A_1, \ldots, A_k, B_1, \ldots, B_\ell) = 1$. Let $D = \max(\deg A, \deg B)$. If α_i, γ_i's are randomly chosen from $\mathbb{F}_q$, then with probability $\geq 1 - \frac{(2D^2+2D)(2\|\mathbf{s}\|_\infty D+1)}{q}$, $\gcd(A_i(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n), B_j(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n), i = 1, \ldots, k, j = 1, \ldots, \ell) = 1$.*

Proof. As $\gcd(A_1, \ldots, A_k) = \gcd(A_1, \gcd(\cdots \gcd(A_{k-1}, A_k)))$ and $\gcd(B_1, \ldots, B_\ell) = \gcd(B_1, \gcd(\cdots \gcd(B_{\ell-1}, B_\ell)))$, we need to compute $k + \ell - 1$ GCDs. For each GCD, there exists a non-zero polynomial $\Gamma_i(\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n)$, so that if α_i, γ_i's are chosen from $\mathbb{F}_q$ and $\Gamma_i(\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n) \neq 0$, then the GCD is still correct when replacing $x_i = \gamma_i z - \alpha_i$. Multiply all polynomials Γ_i together, the degree of the product is $\leq (2D^2 + 2D)(k + \ell - 1) \leq (2D^2 + 2D)(2\|\mathbf{s}\|_\infty D + 1)$. By Lemma 3.1, we proved it. □

4.3 Reduce Multivariate GCD to Univariate GCD

To compute $\phi(G) = \gcd(\phi(A), \phi(B))$ in $\mathbb{F}_q[y, z]$ we interpolate z in $\phi(G)$ from $\gcd(\phi(A)(z = b_k, y), \phi(B)(z = b_k, y))$ for some $b_k \in \mathbb{F}_q$. Condition (2) in Lemma 4.6 identifies which b_k can be used.

Lemma 4.6. *Let $A, B \in \mathbb{F}_q[x_1, \ldots, x_n]$ and $G = \gcd(A, B)$. Let $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{N}^n$. If the α_i, γ_i's are randomly chosen from $\mathbb{F}_q$ and b_k for $0 \leq k \leq D$ are randomly chosen from $\mathbb{F}_q$, then with probability $\geq 1 - \frac{(4\|\mathbf{s}\|_\infty D^2+5D)(D+1)+n^2}{q}$, we have*

(1) *the α_i/γ_i are distinct and the b_k are distinct;*
(2) *$\phi(G)(z = b_k, y) \sim_y \gcd(\phi(A)(z = b_k, y), \phi(B)(z = b_k, y))$.*

Proof. Regard α_i, γ_i's as variables. By Lemma 3.3, we have $\phi(G) \sim_y \gcd(\phi(A), \phi(B))$. Suppose $y^\ell \cdot \phi(G) \sim \gcd(\phi(A), \phi(B))$ for some $\ell \in \mathbb{N}$. Let $R := \mathrm{res}_y(\phi(A)/(\phi(G) \cdot y^\ell), \phi(B)/(\phi(G) \cdot y^\ell))$ and $\Gamma := R \cdot \mathrm{LC}(\phi(A), y) \cdot \mathrm{LC}(\phi(B), y) \in \mathbb{F}_q[\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n, z]$. By [7, p288, Sylvester's Criterion], $R \neq 0$, so $\Gamma \neq 0$. By the definition of resultant, we have $\deg R \leq 4\|\mathbf{s}\|_\infty D^2$. So $\deg \Gamma \leq 4\|\mathbf{s}\|_\infty D^2 + 4D$.

For each k, let $\Gamma_k := \Gamma(\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n, z = b_k)$. Let

$$\Omega = \prod_{k=0}^{D} \Gamma_k \cdot \prod_{0 \leq i < j \leq D} (b_i - b_j) \cdot \prod_{1 \leq i < j \leq n} (\alpha_i \gamma_j - \alpha_j \gamma_i) \cdot \prod_{i=1}^{n} \gamma_i.$$

Claim: if α_i, γ_i, b_k are chosen from $\mathbb{F}_q$ and $\Omega \neq 0$, then conditions (1) and (2) are satisfied. Proof of claim: $\Omega \neq 0$ implies $\Gamma_k \neq 0$ and $\prod_{0 \leq i < j \leq D}(b_i - b_j) \neq 0$ and $\prod_{1 \leq i < j \leq n}(\alpha_i \gamma_j - \alpha_j \gamma_i) \neq 0$ and $\gamma_i \neq 0$. The last three inequalities imply that α_i/γ_i's are distinct and b_k's are distinct. By (i) of Lemma 3.2, $\Gamma_k \neq 0$ implies that $\gcd(\phi(A)(z = b_k, y), \phi(B)(z = b_k, y)) \sim \phi(G)(z = b_k, y) \cdot y^\ell$. We proved it. As $\deg \Omega \leq (4\|\mathbf{s}\|_\infty D^2 + 4D)(D + 1) + (D + 1)D/2 + n^2$, by Lemma 3.1, if α_i, γ_i's and b_k are randomly chosen from $\mathbb{F}_q$, the probability that $\Omega \neq 0$ is $\geq 1 - \frac{(4\|\mathbf{s}\|_\infty D^2+5D)(D+1)+n^2}{q}$. □

4.4 An Algorithm for Computing $N(G, \mathbf{s})$

Let G^* be an approximation polynomial containing some terms of G and let $\mathrm{Terms}(G^*)$ denote those terms. For an $\mathbf{s} \in \mathbb{N}^n$, the following algorithm computes the set $(\mathrm{NC}(G, \mathbf{s}) \setminus \mathrm{Terms}(G^*)) \bigcup \mathrm{NC}(G - G^*, \mathbf{s})$ which is a set that contains all the terms in $\mathrm{NC}(G, \mathbf{s})$ and $\mathrm{NC}(G - G^*, \mathbf{s})$ but not in $\mathrm{Terms}(G^*)$.

Algorithm 1 Computing the Non-colliding Set

Require: Monomial primitive polynomials $A, B \in \mathbb{F}_q[x_1, \ldots, x_n]$; an approximation polynomial G^* containing some terms of G; a vector $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{N}^n$; a tolerance $\varepsilon \in (0, 1)$.

Ensure: If $\mathrm{NC}(G, \mathbf{s}) \bigcap \mathrm{Terms}(G^*) \neq \emptyset$ or $G^* = 0$, return the set $(\mathrm{NC}(G, \mathbf{s}) \setminus \mathrm{Terms}(G^*)) \bigcup \mathrm{NC}(G - G^*, \mathbf{s})$ with a probability of $\geq 1 - \varepsilon$; or "Failure".

1: Let $D := \max(\deg A, \deg B)$ and let $\varepsilon := \min(\varepsilon, 1/D)$.
2: Find ℓ such that $q^\ell > \frac{8\|\mathbf{s}\|_\infty (D+1)^3 + 6n\|\mathbf{s}\|_\infty (D+1)^2 + 3n^2}{\varepsilon}$.
3: Extend $\mathbb{F}_q$ to $\mathbb{F}_{q^\ell}$ if $\ell > 1$. We still denote $\mathbb{F}_{q^\ell}$ as $\mathbb{F}_q$.
4: Let $D_{min} := \min(\deg A, \deg B)$.
5: Compute $A_{\mathbf{s}} = A(x_1 y^{s_1}, \ldots, x_n y^{s_n})$ and $B_{\mathbf{s}} = B(x_1 y^{s_1}, \ldots, x_n y^{s_n})$. Assume $A_{\mathbf{s}} = A_1 y^{v_1} + \cdots + A_r y^{v_r}$ where $v_1 > \cdots > v_r$ and $B_{\mathbf{s}} = B_1 y^{u_1} + \cdots + B_\ell y^{u_\ell}$ with $u_1 > \cdots > u_\ell$.
6: Randomly pick $\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n, b_0, b_1, \ldots, b_{D_{min}}$ from $\mathbb{F}_q$ until α_i/γ_i's and b_i's are distinct and $\deg_z A_1(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n) = \deg A_1$ and $A_1(\gamma_1 b_i - \alpha_1, \ldots, \gamma_n b_i - \alpha_n) \neq 0$ and $B_1(\gamma_1 b_i - \alpha_1, \ldots, \gamma_n b_i - \alpha_n) \neq 0$ for all $i = 0, \ldots, D_{min}$.

Stage 1: Test if $\mathrm{NC}(G, \mathbf{s}) = \emptyset$. See section 4.2.

7: **if** $\gcd(A_i(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n), B_j(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n), i = 1, \ldots, r, j = 1, \ldots, \ell) \neq 1$ **then**
8: return "Failure". {* This means $\mathrm{NC}(G, \mathbf{s}) = \emptyset$ (Lemma 4.3).*}
9: **end if**

Stage 2: Interpolate the bivariate GCD $\phi(G)$ (upto y^m) from $D_{min} + 1$ monic univariate GCDs $\bar{g}_k \in \mathbb{F}_q[y]$.

10: **if** $\deg A < \deg B$ **then** $\Gamma := A_1$ **else** $\Gamma := B_1$ **end if**
11: **for** $k = 0, 1, 2, \ldots, D_{min}$ **do**
12: Compute $\overline{g}_k := \gcd(A((\gamma_1 b_k - \alpha_1) y^{s_1}, \ldots, (\gamma_n b_k - \alpha_n) y^{s_n}), B((\gamma_1 b_k - \alpha_1) y^{s_1}, \ldots, (\gamma_n b_k - \alpha_n) y^{s_n}))$.
13: Set $g_k := \Gamma(\gamma_1 b_k - \alpha_1, \ldots, \gamma_n b_k - \alpha_n) \cdot \overline{g_k}$ and assume $g_k = c_{k,1} y^{d_1} + \cdots + c_{k,t} y^{d_t}$.
14: **end for**

15: **for** $i = 1, 2, \ldots, t$ **do**
16: Interpolate $\overline{C}_i(z)$ from $(b_k, c_{k,i})$ so that $\overline{C}_i(b_k) = c_{k,i}$ for $0 \le k \le D_{min}$.
17: **end for**
18: Compute $\overline{C}(z) = \gcd(\overline{C}_1(z), \ldots, \overline{C}_t(z))$ then for $i = 1, \ldots, t$ $C_i(z) := \overline{C}_i(z)/\overline{C}(z)$. {* $\phi(G) \sim_y C_1(z)y^{d_1} + \cdots + C_t(z)y^{d_t}$. *}

Stage 3: compute the non-colliding set $\mathrm{NC}(G, \mathbf{s})$.

19: $\mathrm{NC} := \emptyset$.
20: **for** $i = 1, \ldots, t$ **do**
21: **if** $C_i(z)$ factors as $c_i \prod_{j=1}^{n} (\gamma_j z - \alpha_j)^{e_{i,j}}$ **then**
22: $\mathrm{NC} := \mathrm{NC} \bigcup \{c_i \cdot x_1^{e_{i,1}} \cdots x_n^{e_{i,n}}\}$.
23: **end if**
24: **end for**
25: **if** $G^* = 0$ **then** return NC **end if**

Stage 4: Adjust the coefficients of $\phi(G)$ and $\phi(G^*)$ using a common term to make them consistent to get $\phi(G - G^*)$.

26: Let $h_1 := C_1(z)y^{d_1} + \cdots + C_t(z)y^{d_t}$. {* $h_1 \sim_y \phi(G)$. *}
27: Compute $h_2 := G^*((\gamma_1 z - \alpha_1)y^{s_1}, \ldots, (\gamma_n z - \alpha_n)y^{s_n})$ and assume $h_2 = E_1(z)y^{w_1} + \cdots + E_\gamma(z)y^{w_\gamma}$. {* $h_2 = \phi(G^*)$. *}
28: **if** the monomials of NC and G^* do not have a common one **then** return "Failure" **end if**
29: Assume one of the same monomials corresponds to terms $C_\rho(z)y^{d_\rho}$ and $E_\delta(z)y^{w_\delta}$ in h_1 and h_2.
30: Let $h_3 := h_1 \cdot y^{\max(d_\rho, w_\delta) - d_\rho} \cdot \frac{\mathrm{LC}(E_\delta(z))}{\mathrm{LC}(C_\rho(z))} - h_2 \cdot y^{\max(d_\rho, w_\delta) - w_\delta}$.
Assume $h_3 = F_1(z)y^{\eta_1} + \cdots + F_\tau(z)y^{\eta_\tau}$. {* $h_3 = \phi(G - G^*)$. *}
31: **if** $h_3 = 0$ return $\emptyset$ **end if**
32: Multiply each term in NC by the scalar $\frac{\mathrm{LC}(E_\delta(z))}{\mathrm{LC}(C_\rho(z))}$ in $\mathbb{F}_q$.

Stage 5: Compute the non-colliding terms of $\mathrm{NC}(G - G^*, \mathbf{s})$.

33: Set $\mathbf{NCG}^* := \emptyset$. {* Store the elements of $\mathrm{NC}(G - G^*, \mathbf{s})$. *}
34: **for** $k = 1, \ldots, \tau$ **do**
35: **if** $F_k(z)$ factors as $c_k \prod_{i=1}^{n} (\gamma_i z - \alpha_i)^{e_{k,i}}$ **then**
36: $\mathbf{NCG}^* := \mathbf{NCG}^* \bigcup \{c_k \cdot x_1^{e_{k,1}} \cdots x_n^{e_{k,n}}\}$.
37: **end if**
38: **end for**
39: **if** $\mathbf{NCG}^* = \emptyset$ **then** return "Failure" **end if**
40: **return** $(\mathrm{NC} \setminus \mathrm{Terms}(G^*)) \bigcup \mathbf{NCG}^*$.

In Stage 2 of Algorithm 1 we interpolate z in $\phi(G)$ a bivariate polynomial in $\mathbb{F}_q[y, z]$. The following Lemma shows that $D_{min} + 1$ values for z are sufficient where $D_{min} = \min(\deg A, \deg B)$.

Lemma 4.7. *Let $b_0, b_1, \ldots, b_N$ be distinct points in $\mathbb{F}_q$. Let $a = \phi(A)$, $b = \phi(B)$, and $g = \gcd(a, b)$ in $\mathbb{F}_q[y, z]$. Let $h_k = \gcd(a(y, b_k), b(y, b_k))$ for $0 \le k \le N$. If (1) $\mathrm{Cont}(g, y) = 1$ and (2) $h_k \sim g(y, b_k)$ in $\mathbb{F}_q[y]$ for $0 \le k \le N$ then g can be interpolated from the h_k for $N = D_{min}$.*

Proof. WLOG assume $\deg A \le \deg B$. Let $\Gamma(z) = \mathrm{LC}(a, y)$ and $c = a/g$. Then $\Gamma(z) = \mathrm{LC}(g, y)\,\mathrm{LC}(c, y)$. Let

$$\begin{aligned} g_k(y) &= \Gamma(b_k)\,\mathrm{monic}(h_k(y)) \\ &= \mathrm{LC}(c, y)(b_k)\,\mathrm{LC}(g, y)(b_k)\,\mathrm{monic}(h_k(y)) \\ &= \mathrm{LC}(c, y)(b_k)\,g(y, b_k). \end{aligned}$$

Interpolating the $(b_k, g_k(y))$ gives us $h = \mathrm{LC}(c, y)\,g$ not g. To compute g we compute $\mathrm{Cont}(h, y) = \mathrm{LC}(c, y)$ and remove it from h. So we need sufficient values for z to interpolate z in h. We have $\deg(h, z) = \deg(\mathrm{LC}(c, y), z) + \deg(g, z) \le \deg(c, z) + \deg(g, z) = \deg(a, z)$ thus $\deg(a, z) + 1$ values are sufficient. The Lemma follows since $\deg(a, z) \le \deg A = Dmin$. □

Theorem 4.8. *Algorithm 1 works correctly as specified.*

Proof. Consider two cases.
Case 1: If $Q := \gcd(A_1, \ldots, A_r, B_1, \ldots, B_\ell) \ne 1$, then by Lemma 4.3, $\mathrm{NC}(G, \mathbf{s}) = \emptyset$. In Step 6, as $\deg_z A_1(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n) = \deg A_1$, then $\deg_z Q(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n) = \deg Q \ge 1$. Thus this case can be detected in Step 7. We analyse the success rate. In Step 6, the leading coefficient of $A_1(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n)$ is a polynomial in γ_i's with degree $\le D$. To make α_i/γ_i distinct, we should make $\prod_{1 \le i < j \le n} (\alpha_i \gamma_j - \alpha_j \gamma_i) \cdot \prod_{i=1}^{n} \gamma_i \ne 0$. This polynomial has degree $\le n^2$. As $\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n$ are randomly chosen from $\mathbb{F}_q$, the output of Step 8 is correct with probability $\ge 1 - \frac{D+n^2}{q} \ge 1 - \varepsilon$.
Case 2: If $\gcd(A_1, \ldots, A_r, B_1, \ldots, B_\ell) = 1$, Step 22 computes the correct $\mathrm{NC}(G, \mathbf{s})$ if the following three conditions are met.

(1) The gcd in Step 7 is 1.
(2) In Step 12, $\overline{g}_k \sim_y \phi(G)(z = b_k, y))$ for all k.
(3) In Step 21, C_i corresponds to a non-colliding term in $G_{\mathbf{s}}$ iff C_i has the form $c_i(\gamma_1 z - \alpha_1)^{e_{i,1}} \cdots (\gamma_n z - \alpha_n)^{e_{i,n}}$.

By Corollary 4.5, $\Pr[(1) \text{ occurs}] \ge 1 - \frac{(2D^2+2D)(2\|\mathbf{s}\|_\infty D+1)}{q}$.
By Lemma 4.6, $\Pr[(2) \text{ occurs}] \ge 1 - \frac{(4\|\mathbf{s}\|_\infty D^2+5D)(D+1)+n^2}{q}$.
By Theorem 4.1, $\Pr[(3) \text{ occurs}] \ge 1 - \frac{(3n+1)\|\mathbf{s}\|_\infty D^2+3nD+n^2}{q}$.

Step 36 computes the correct $\mathrm{NC}(G - G^*, \mathbf{s})$ if the following condition in Step 35 is met.

(4) F_k corresponds to a non-colliding term in $(G - G^*)_{\mathbf{s}}$ iff F_k has the form $c_k(\gamma_1 z - \alpha_1)^{e_{k,1}} \cdots (\gamma_n z - \alpha_n)^{e_{k,n}}$.

By Theorem 4.1, $\Pr[(4) \text{ occurs}] \ge 1 - \frac{(3n+1)\|\mathbf{s}\|_\infty D^2+3nD+n^2}{q}$.
Thus our algorithm returns the correct set with probability (1) times (2) times (3) times(4) which simplifies to $\ge 1 - Q/q$ where $Q = 8\|\mathbf{s}\|_\infty (D+1)^3 + 6n\|\mathbf{s}\|_\infty (D+1)^2 + 3n^2$. Since Step 2 imposes $q > Q/\varepsilon$, the probability is $\ge 1 - \varepsilon$. □

Theorem 4.9. *The expected bit complexity of Algorithm 1 is $O^\sim(nT_{\mathrm{in}}(\log d + \log \|\mathbf{s}\|_\infty) + \log^2 \frac{1}{\varepsilon} + \log \frac{1}{\varepsilon} \log q + (T_{\mathrm{in}} D + \|\mathbf{s}\|_\infty D^2 + T_{\mathrm{o}} D^2 + n\|\mathbf{s}\|_\infty D) \cdot (\log q + \log(n\|\mathbf{s}\|_\infty D/\varepsilon)))$ where $T_{\mathrm{in}} = \#A + \#B$ and $T_{\mathrm{o}} = \#G$, $d = \max_{i=1}^{n} \max(\deg(A, x_i), \deg(B, x_i))$ and $D = \max(\deg A, \deg B)$.*

Proof. In Step 2, as $\ell = O(\log \|\mathbf{s}\|_\infty + \log D + \log n + \log \frac{1}{\varepsilon})$, the complexity of extension is $O(\ell^2 + \ell \log q)$ bit operations [18], which is $O(\log^2 \|\mathbf{s}\|_\infty + \log^2 D + \log^2 n + \log^2 \frac{1}{\varepsilon} + \log \|\mathbf{s}\|_\infty \log q + \log D \log q + \log n \log q + \log \frac{1}{\varepsilon} \log q)$. Step 5 costs $O^\sim(nT_{\mathrm{in}}(\log d + \log \|\mathbf{s}\|_\infty) + T_{\mathrm{in}} \log q)$ bit operations. In Step 6, the cost of randomly choosing $\alpha_1, \ldots, \alpha_n, \gamma_1, \ldots, \gamma_n, b_0, \ldots, b_{D_{\min}}$ is $O((n + D) \log q)$ bit operations. As analyzed in Theorem 4.8, the probability of all conditions being satisfied is $\ge 1 - \frac{5D(D+1)+n^2}{q}$. Due to $q > 8\|\mathbf{s}\|_\infty (D+1)^3 + 6n\|\mathbf{s}\|_\infty (D+1)^2 + 3n^2$, the probability $\ge \frac{3}{8}$. Thus the expected cost of Step 6 can be absorbed by the complexities of Step 7 and Step 12. In Step 7, first, compute $A_i(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n)$ and

$B_i(\gamma_1 z - \alpha_1, \ldots, \gamma_n z - \alpha_n)$, which costs up to $O^\sim(T_{\text{in}} D \log q)$ bit operations. Then compute $r + \ell - 1$ polynomial GCDs with degree D, resulting in a complexity of $O^\sim((r+\ell)D \log q)$ bit operations. As $r + \ell$ is $O(\|\mathbf{s}\|_\infty D)$, the complexity is $O^\sim(\|\mathbf{s}\|_\infty D^2 \log q)$ bit operations. In Step 12, compute all $\phi(A)(z = b_k, y)$ and $\phi(B)(z = b_k, y)$, which costs $O^\sim(T_{\text{in}} D \log q)$ bit operations. Then, we totally compute $O(D)$ univariate GCDs of degree $O(\|\mathbf{s}\|_\infty D)$, the complexity is $O^\sim(\|\mathbf{s}\|_\infty D^2 \log q)$ bit operations. In Step 16, we interpolate t polynomials with degrees $O(D)$. As $t \in O(\|\mathbf{s}\|_\infty D)$, the complexity is $O^\sim(\|\mathbf{s}\|_\infty D^2 \log q)$ bit operations. In Step 18, we compute $t-1$ polynomial GCDs of degree $O(D)$. As $t \in O(\|\mathbf{s}\|_\infty D)$, the complexity is $O^\sim(\|\mathbf{s}\|_\infty D^2 \log q)$ bit operations. Step 21 costs $O(tD^2 \log q)$ bit operations to factor all $C_i(z)$ by continuously dividing them by $\gamma_j z - \alpha_j$ in $\mathbb{F}_q[z]$. If $t > T_{\text{o}}$, then we computed the wrong result, as stated in Theorem 4.8, the probability of this case happening is $\le \varepsilon$. In this case, $t \le \|\mathbf{s}\|_\infty D$, therefore the complexity is $O(\|\mathbf{s}\|_\infty D^3 \log q)$ bit operations. If $t \le T_{\text{o}}$, $O(tD^2 \log q)$ is $O(T_{\text{o}} D^2 \log q)$. In Step 1, we let $\varepsilon \le \frac{1}{D}$. The expected complexity is $O(\varepsilon \cdot \|\mathbf{s}\|_\infty D^3 \log q + T_{\text{o}} D^2 \log q)$, which is $O^\sim(\|\mathbf{s}\|_\infty D^2 \log q + T_{\text{o}} D^2 \log q)$ bit operations. Step 27 does $O^\sim(T_{\text{o}} D \log q)$ bit operations as $\#G^* \le \#G$. In Step 30, the complexity is $O^\sim((t + T_{\text{o}}) \log q + (t + T_{\text{o}}) \log \|\mathbf{s}\|_\infty + (t + T_{\text{o}}) \log D)$ bit operations, as $t \le \|\mathbf{s}\|_\infty D$, the cost is $O^\sim(\|\mathbf{s}\|_\infty D \log q + T_{\text{o}} \log q + T_{\text{o}} \log \|\mathbf{s}\|_\infty + T_{\text{o}} \log D)$ bit operations. Steps 34-36 have the same the complexity as Steps 20-22, which is $O^\sim(\|\mathbf{s}\|_\infty D^2 \log q + T_{\text{o}} D^2 \log q + n\|\mathbf{s}\|_\infty D \log q)$ bit operations. After Step 4, $\mathbb{F}_q$ is actually $\mathbb{F}_{q^\ell}$, so in the complexity analysis, $\log q^\ell \in \max(O(\log q, \log(\frac{n\|\mathbf{s}\|_\infty D}{\varepsilon})))$. To simplify it, we replace it with $O(\log q + \log(\frac{n\|\mathbf{s}\|_\infty D}{\varepsilon}))$. □

4.5 Good Kronecker Substitutions

When $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ is chosen randomly, the substitution $x_i = x_i y^{s_i}, i = 1, 2, \ldots, n$ is called a *randomized Kronecker substitution*. We call a vector $\mathbf{s}$ that causes $\#\text{NC}(G, \mathbf{s}) \ge \frac{1}{2}\#G$ a "good" Kronecker substitution for G. The following key lemma shows that there is an upper bound on the number of "bad" vectors $\mathbf{s}$.

Lemma 4.10. *Let $G \in \mathbb{F}_q[x_1, \ldots, x_n]$ and $t = \#G$. If there exist K different integer vectors $\mathbf{s} \in [0, N)^n$, such that $\#\text{C}(G, \mathbf{s}) \ge \ell$ then $K \le t(t-1)N^{n-1}/\ell$.*

Proof. Assume $G = c_1 m_1 + \cdots + c_t m_t$ and $m_i = x_1^{e_{i,1}} \cdots x_n^{e_{i,n}}$. Let $h_{i,j}(s_1, \ldots, s_n) = \sum_{k=1}^n (e_{i,k} - e_{j,k}) s_k$ for $1 \le i < j \le t$. Denote $R_{i,j}$ as the number of integer roots in $[0, N)^n$ and let $R = \sum_{1 \le i < j \le t} R_{i,j}$. For each $h_{i,j}$, there are up to N^{n-1} different integer roots in $[0, N)^n$. Therefore $R_{i,j} \le N^{n-1}$ and $R \le \frac{t(t-1)}{2} N^{n-1}$. Assuming $\mathbf{s}$ is a vector such that $\#\text{C}(G, \mathbf{s}) \ge \ell$. Without loss of generality, assume $c_1 m_1, \ldots, c_\ell m_\ell$ are colliding terms, then at least $\lceil \frac{\ell}{2} \rceil$ pairs of terms in $\text{C}(G, \mathbf{s})$ collide together. Therefore, $\mathbf{s}$ is the root of at least $\lceil \frac{\ell}{2} \rceil$ different $h_{i,j}$. There are K such roots, so for all $h_{i,j}$'s, there are at least $\lceil \frac{\ell}{2} \rceil \cdot K$ roots. So we have $\frac{\ell}{2} \cdot K \le \lceil \frac{\ell}{2} \rceil \cdot K \le R \le \frac{t(t-1)}{2} N^{n-1}$, therefore $\ell \cdot K \le t(t-1)N^{n-1}$. □

Theorem 4.11 provides a method to find a vector $\mathbf{s}$ so that, with high probability, $G_{\mathbf{s}}$ has at least $\beta \cdot \#G$ non-colliding terms.

Theorem 4.11. *Let $G(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$, $T \ge \#G = t$. Let $\beta \in (0, 1)$ and $\mu \in (0, 1)$. Let $N = \lceil \frac{T-1}{\mu(1-\beta)} \rceil$. If we choose $\mathbf{s} \in [0, N)^n$ at random, then $\Pr[\#\text{NC}(G, \mathbf{s}) > \beta \cdot \#G] \ge 1 - \mu$.*

Proof. Observe that $\Pr[\#\text{NC}(G, \mathbf{s}) > \beta \cdot \#G] = 1 - \Pr[\#\text{NC}(G, \mathbf{s}) \le \beta \cdot \#G] = 1 - \Pr[\#\text{C}(G, \mathbf{s}) \ge (1 - \beta) \cdot \#G]$. The second equality follows from $\#\text{NC}(G, \mathbf{s}) + \#\text{C}(G, \mathbf{s}) = \#G$. Thus it suffices to show that $\Pr[\#\text{C}(G, \mathbf{s}) \ge (1 - \beta) \cdot \#G] \le \mu$. According to Lemma 4.10, the number of integer vectors $\mathbf{s}$ in $[0, N)^n$ such that $\#\text{C}(G, \mathbf{s}) \ge (1 - \beta)\#G$ is $\le \frac{t(t-1)N^{n-1}}{(1-\beta)\#G}$. Since there are N^n integer vectors in $[0, N)^n$, we have $\Pr[\#\text{C}(G, \mathbf{s}) \ge (1 - \beta) \cdot \#G] \le ((t-1)N^{n-1}/(1-\beta))/N^n = ((t-1)/(1-\beta))/N \le \mu(t-1)/(T-1) \le \mu$. So $\Pr[\#\text{NC}(G, \mathbf{s}) > \beta \cdot \#G] \ge 1 - \mu$. □

If we choose $\beta = \frac{1}{2}, \mu = \frac{1}{4}\lceil \log_2 T \rceil^{-1}$, we have :

Corollary 4.12. *Let $G \in \mathbb{F}_q[x_1, \ldots, x_n]$, $T \ge \#G$ and $N = 8(T-1)\lceil \log_2 T \rceil$. If we choose $\mathbf{s} \in [0, N)^n$ at random then*

$$\Pr[\#\text{NC}(G, \mathbf{s}) > \tfrac{1}{2} \cdot \#G] \ge 1 - \tfrac{1}{4}\lceil \log_2 T \rceil^{-1}.$$

Actually, if we choose $\beta = \mu = \frac{1}{2}$, then $\Pr[\#\text{NC}(G, \mathbf{s}) > \frac{1}{2} \cdot \#G] \ge \frac{1}{2}$, the probability that at least half of the terms in $G_{\mathbf{s}}$ do not collide, is $\ge \frac{1}{2}$. This is a very satisfactory result and $N = 4(T-1)$. However, because we only recover $T/2$ terms each time, in order to find all the terms, we need to loop $\lceil \log_2 T \rceil$ times, which reduces the probability of success to $2^{-\lceil \log_2 T \rceil} \le \frac{1}{T}$, which is too low. Therefore, in the Corollary 4.12, we increase N a little to increase the probability of success from $\frac{1}{2}$ to $1 - \frac{1}{4}\lceil \log_2 T \rceil^{-1}$, so the probability of computing all terms is $(1 - \frac{1}{4}\lceil \log_2 T \rceil^{-1})^{\lceil \log_2 T \rceil} \ge \frac{3}{4}$.

4.5.1 Structure of Our GCD Algorithm. **Stage 1**: First, randomly choose a vector $\mathbf{s} \in [0, N)^n$ where $N = 8(T-1)\lceil \log_2 T \rceil$. Then by Corollary 4.12, with high probability, $\#\text{NC}(G, \mathbf{s}) > \frac{1}{2} \cdot \#G$. By Algorithm 1, we can compute $\text{NC}(G, \mathbf{s})$. Denote G^* as the sum of all terms in $\text{NC}(G, \mathbf{s})$. Then G^* is an approximation polynomial of G and satisfies $\#G^* \ge \frac{1}{2}\#G$ (or $\#(G - G^*) \le \frac{1}{2}\#G$). Generally $G \ne G^*$, so we choose other vectors $\mathbf{s}'$ to find the remaining terms in $G - G^*$.

Stage 2: Let $\text{Terms}(G^*)$ denote the set of all terms in G^*. We want to choose a new vector $\mathbf{s}' \in \mathbb{N}^n$ that satisfies

(1) $\#\text{NC}(G - G^*, \mathbf{s}') > \frac{1}{2}\#(G - G^*)$;
(2) $\text{NC}(G, \mathbf{s}') \bigcap \text{Terms}(G^*) \ne \emptyset$.

Condition (1) ensures $\text{NC}(G - G^*, \mathbf{s}')$ contains at least half the terms in $G - G^*$. Let G^{**} be the sum of the terms in $\text{NC}(G - G^*)$. Then we have $\#(G - G^* - G^{**}) < \frac{1}{2}(G - G^*) < \frac{1}{4}\#G$. By performing the same steps for $G - G^* - G^{**}$, we can find a polynomial G^{***} such that $\#(G - G^* - G^{**} - G^{***}) < \frac{1}{2^3}\#G$. Repeating this $\lceil \log_2 T \rceil$ times we obtain all the terms of G.

Condition (2) is used to match the terms in $\text{NC}(G, \mathbf{s}')$ and the previous approximation polynomial G^*. This is because $\text{NC}(G, \mathbf{s}')$ is computed based on the factorization of the coefficients of y in $G_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i$ for $1 \le i \le n)$. To compute $G_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i)$, we compute the GCD of $A_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i)$ and $B_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i)$. We will get a polynomial $\kappa y^m \cdot G_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i)$, where $\kappa \ne 1$ and $m \ne 0$ are likely. We need to identify κ and m for our algorithm to work. Let $H := G - G^*$. $\text{NC}(G - G^*, \mathbf{s}')$ is computed based on the factorization of the coefficients of y in $H_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i)$. Here $H_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i) = G_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i) - G^*_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i)$. G^* is known, so $G^*_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i)$ is known. But the κ and y^m are unknown to us, which means $G_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i)$ is also unknown. Condition (2) means that G^* and $\text{NC}(G, \mathbf{s}')$ have a common monomial, so we can

remove the factor κy^m in $\kappa y^m \cdot G_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i)$ by comparing its coefficients with the coefficients of G^*, as they have at least one common term. Therefore we can compute $H_{\mathbf{s}'}(x_i = \gamma_i z - \alpha_i)$ and ultimately find $\mathrm{NC}(G - G^*, \mathbf{s}')$.

The following theorem gives the probability of successfully choosing a vector $\mathbf{s}$ that satisfies the two conditions.

Theorem 4.13. *Let $G \in \mathbb{F}_q[x_1, \ldots, x_n]$, $T \geq \#G$. Let G^* be a polynomial containing some terms of G, and $\#G^* \geq \frac{1}{2}\#G$. Let $N = 8(T-1)\lceil \log_2 T \rceil$. If we choose $\mathbf{s} \in [0, N)^n$ at random then* $\Pr[\#\mathrm{NC}(G - G^*, \mathbf{s}) \geq \frac{1}{2}\#(G - G^*)$ and $\mathrm{NC}(G, \mathbf{s}) \bigcap \mathrm{Terms}(G^*) \neq \emptyset] \geq 1 - \frac{1}{4\lceil \log_2 T \rceil}$.

Proof. Denote $H := G - G^*$. Assuming there are K_1 integer vectors in $[0, N)^n$, such that $\#\mathrm{NC}(H, \mathbf{s}) < \frac{1}{2}\#(H)$ and there are K_2 integer vectors in $[0, N)^n$ such that $\mathrm{NC}(G, \mathbf{s}) \bigcap \mathrm{Terms}(G^*) = \emptyset$. Below we give an upper bound for $K_1 + K_2$. Let $t_H := \#H$. Then $t_H \leq \frac{T}{2}$. $\#\mathrm{NC}(H, \mathbf{s}) < \frac{1}{2}\#(H)$ is equivalent to $\#\mathrm{C}(H, \mathbf{s}) > \frac{1}{2}t_H$, by Lemma 4.10, $K_1 \leq (t_H(t_H - 1)N^{n-1})/(\frac{1}{2}t_H) = 2(t_H - 1)N^{n-1} \leq (T-2)N^{n-1}$. Now consider K_2. Assume $G = \sum_{i=1}^{t} c_i m_i$ and $m_i = x_1^{e_{i,1}} \cdots x_n^{e_{i,n}}$. W.l.o.g., assume $c_1 m_1 \in \mathrm{Terms}(G^*)$. As $\mathrm{NC}(G, \mathbf{s}) \bigcap \mathrm{Terms}(G^*) = \emptyset$, $c_1 m_1 \notin \mathrm{NC}(G, \mathbf{s})$, which means $c_1 m_1 \in \mathrm{C}(G, \mathbf{s})$. Set $h(s_1, \ldots, s_n) = \prod_{j=2}^{t}[(e_{j,1} - e_{1,1})s_1 + (e_{j,2} - e_{1,2})s_2 + \cdots + (e_{j,n} - e_{1,n})s_n]$. Then $c_1 m_1 \in \mathrm{C}(G, \mathbf{s})$ means $h(s_1, \ldots, s_n) = 0$. Since $\deg h(s_1, \ldots, s_n) \leq T - 1$, by Lemma 3.1, there exist at most $(T-1)N^{n-1}$ points $\mathbf{s} \in [0, N)^n$ such that $h(s_1, \ldots, s_n) = 0$. So $K_2 \leq (T-1)N^{n-1}$. Therefore we have $K_1 + K_2 \leq 2(T-1)N^{n-1}$. So the probability $\Pr[\#\mathrm{NC}(G - G^*, \mathbf{s}) \geq \frac{1}{2}\#(G - G^*)$ and $\mathrm{NC}(G, \mathbf{s}) \bigcap \mathrm{Terms}(G^*) \neq \emptyset] \geq 1 - \frac{K_1 + K_2}{N^n} \geq 1 - \frac{2(T-1)}{N} = 1 - \frac{1}{4}\lceil \log_2 T \rceil^{-1}$. □

Because we may only recover $T/2$ terms each time, in order to find all the terms, we need to loop $\lceil \log_2 T \rceil$ times, so the probability of computing all terms becomes $(1 - \frac{1}{4}\lceil \log_2 T \rceil^{-1})^{\lceil \log_2 T \rceil} \geq \frac{3}{4}$.

4.5.2 Algorithms.

Algorithm 2 Generating a Newly Added Polynomial G^{**}

Require: Monomial primitive polynomials $A, B \in \mathbb{F}_q[x_1, \ldots, x_n]$; an approximation polynomial G^* of G, satisfying $\#(G - G^*) \leq \frac{1}{2}\#G$ or $G^* = 0$; an upper bound $T \geq \#G$.
Ensure: A polynomial G^{**} such that $\#(G - G^* - G^{**}) \leq \frac{1}{2}\#(G - G^*)$ with probability $\geq 1 - \frac{1}{3\lceil \log_2 T \rceil}$ or "Failure".
1: Let $N = 8(T-1)\lceil \log_2 T \rceil$. Randomly choose $\mathbf{s} \in [0, N)^n$.
2: Compute $C := (\mathrm{NC}(G, \mathbf{s}) \setminus \mathrm{Terms}(G^*)) \bigcup \mathrm{NC}(G - G^*, \mathbf{s})$ by Algorithm 1 with input A, B, G^*, $\mathbf{s}$ and $\varepsilon = \frac{1}{12\lceil \log_2 T \rceil}$.
3: **if** C is "Failure" **then** return "Failure" **end if**
4: Let G^{**} be the sum of the terms in C.
5: **return** G^{**}.

Theorem 4.14. *Algorithm 2 works correctly as specified.*

Proof. We consider two cases. (1) $G^* = 0$. In Step 1, as we choose $\mathbf{s} \in [0, N)^n$, by Corollary 4.12, we have $\Pr[\#\mathrm{NC}(G, \mathbf{s}) > \frac{1}{2} \cdot \#G] \geq 1 - \frac{1}{4}\lceil \log_2 T \rceil^{-1}$. In Step 2, we compute $(\mathrm{NC}(G, \mathbf{s}) \setminus \mathrm{Terms}(G^*)) \bigcup \mathrm{NC}(G - G^*, \mathbf{s}) = \mathrm{NC}(G, \mathbf{s})$ by Algorithm 1 with probability $\geq 1 - \frac{1}{12\lceil \log_2 T \rceil}$. So in Step 5, we have $\#(G - G^{**}) \leq \frac{1}{2}\#G$ with probability $\geq (1 - \frac{1}{4\lceil \log_2 T \rceil})(1 - \frac{1}{12\lceil \log_2 T \rceil}) \geq 1 - \frac{1}{3\lceil \log_2 T \rceil}$.

(2) $\#(G - G^*) \leq \frac{1}{2}\#G$. In Step 1, as we choose $\mathbf{s} \in [0, N)^n$, by Theorem 4.13, we have $\Pr[\#\mathrm{NC}(G - G^*, \mathbf{s}) \geq \frac{1}{2}\#(G - G^*)$ and $\mathrm{NC}(G, \mathbf{s}) \bigcap \mathrm{Terms}(G^*) \neq \emptyset] \geq 1 - \frac{1}{4\lceil \log_2 T \rceil}$. In Step 2, we compute $(\mathrm{NC}(G, \mathbf{s}) \setminus \mathrm{Terms}(G^*)) \bigcup \mathrm{NC}(G - G^*, \mathbf{s})$ by Algorithm 1 with probability $\geq 1 - \frac{1}{12\lceil \log_2 T \rceil}$. So in Step 5, we have $\#(G - G^* - G^{**}) \leq \frac{1}{2}\#(G - G^*)$ with probability $\geq (1 - \frac{1}{4\lceil \log_2 T \rceil})(1 - \frac{1}{12\lceil \log_2 T \rceil}) \geq 1 - \frac{1}{3\lceil \log_2 T \rceil}$. □

Now we analyse the complexity of Algorithm 2. We assume $T \in O(T_\mathrm{o})$, that is, T is not a bad bound.

Theorem 4.15. *The expected complexity of Algorithm 2 is $O^\sim(nT_\mathrm{in}(\log d + \log T_\mathrm{o}) + D(T_\mathrm{in} + nT_\mathrm{o} + T_\mathrm{o}D)(\log q + \log(nT_\mathrm{o})))$ bit operations, where $T_\mathrm{in} := \#A + \#B$, $T_\mathrm{o} := \#G$, $d := \max_{i=1}^{n} \max(\deg(A, x_i), \deg(B, x_i))$ and $D := \max(\deg A, \deg B)$.*

Proof. In Step 2, we call Algorithm 1. As $\|\mathbf{s}\|_\infty$ is $O^\sim(T_\mathrm{o})$ and $\varepsilon = \frac{1}{12\lceil \log_2 T \rceil}$, the expected complexity is $O^\sim(nT_\mathrm{in}(\log d + \log T_\mathrm{o}) + D(T_\mathrm{in} + nT_\mathrm{o} + T_\mathrm{o}D)(\log q + \log(nT_\mathrm{o})))$ bit operations, by Theorem 4.9. □

Algorithm 3 GCD algorithm

Require: $A, B \in \mathbb{F}_q[x_1, \ldots, x_n]$; an upper bound $T \geq \#G$.
Ensure: $G^* = \gcd(A, B)$ up to a constant with probability $\geq \frac{2}{3}$; or "Failure".
1: Compute monomial contents $\mathrm{MoCont}(A)$ and $\mathrm{MoCont}(B)$ and set $\mathrm{MoCont}(G) := \gcd(\mathrm{MoCont}(A), \mathrm{MoCont}(B))$. (see 2.4)
2: $A := A/\mathrm{MoCont}(A)$. $B := B/\mathrm{MoCont}(B)$.
3: $G^* := 0$;
4: **for** $i = 1, 2, \ldots, \lceil \log_2 T \rceil$ **do**
5: Let G^{**} be the output of Algorithm 2 with inputs A, B, G^* and T.
6: **if** G^{**} = "Failure" **then** return "Failure" **end if**
7: **if** $G^{**} = 0$ **then** return $G^* \cdot \mathrm{MoCont}(G)$ **end if**
8: $G^* := G^* + G^{**}$.
9: **end for**
10: **return** "Failure".

Theorem 4.16. *Algorithm 3 works correctly as specified.*

Proof. Steps 5-8 compute the new approximation $G^* + G^{**}$ from G^* using Algorithm 2. As $\#(G - G^* - G^{**}) \leq \frac{1}{2}\#(G - G^*)$, $\lceil \log_2 T \rceil$ loops are enough. If $G^{**} = 0$, then all terms have been discovered, so G^* is the monomial primitive part of the GCD G. Therefore we output $G^* \cdot \mathrm{MoCont}(G)$ in Step 7.

If when Algorithm 2 called in Step 5 it always returns the correct newly added polynomial G^{**}, Algorithm 3 returns the correct GCD in Step 7. Since Algorithm 2 is correct with probability $\geq 1 - \frac{1}{3\lceil \log_2 T \rceil}$, the probability is $\geq (1 - \frac{1}{3\lceil \log_2 T \rceil})^{\lceil \log_2 T \rceil} \geq 1 - \frac{1}{3} = \frac{2}{3}$. □

We analyze the complexity of Algorithm 3. Again we assume $T \in O(T_\mathrm{o})$, that is, T is not a bad bound.

Theorem 4.17. *The expected complexity of Algorithm 3 is* $O^{\sim}(nT_{\text{in}}(\log d \log T_{\text{o}} + \log^2 T_{\text{o}}) + D \log T_{\text{o}}(T_{\text{in}} + nT_{\text{o}} + T_{\text{o}}D)(\log q + \log(nT_{\text{o}})))$ *bit operations where* $T_{\text{in}} := \#A + \#B$, $T_{\text{o}} := \#G$, $D = \max(\deg A, \deg B)$ *and* $d = \max_{i=1}^{n} \max(\deg(A, x_i), \deg(B, x_i))$.

Proof. The cost of computing the monomial contents in Step 1 and the monomial primitive parts in Step 2 is $O^{\sim}(nT_{\text{in}} \log d)$ which negligible. As we call Algorithm 2 at most $\lceil \log_2 T_{\text{o}} \rceil$ times, by Theorem 4.15, the complexity is $O^{\sim}(nT_{\text{in}}(\log d \log T_{\text{o}} + \log^2 T_{\text{o}}) + D(T_{\text{in}} + nT_{\text{o}} + T_{\text{o}}D)(\log q \log T_{\text{o}} + \log(nT_{\text{o}}) \log T_{\text{o}}))$ bit operations. □

Remark 4.1. *If Algorithm 3 outputs a polynomial, with probability* $\geq \frac{11}{12}$*, this polynomial is the GCD of A and B, up to a constant. This probability is greater than* 2/3 *because we have excluded the case where the output is "Failure". We analyze the rate. In Step 2 of Algorithm 2, Algorithm 1 is called. If Algorithm 1 returns a correct set, then Algorithm 2 also returns a correct newly added polynomial* G^{**}*. Therefore, in Algorithm 3, we get a truly better approximation* $G^{*} + G^{**}$*. Due to a maximum of* $\lceil \log_2 T \rceil$ *calls to Algorithm 1, and the selection of* $\varepsilon = \frac{1}{12\lceil \log_2 T \rceil}$*, the probability is* $\geq (1 - \frac{1}{12\lceil \log_2 T \rceil})^{\lceil \log_2 T \rceil} \geq \frac{11}{12}$.

Remark 4.2. *In addition, from the above analysis, it can be seen that even if T is not the upper bound of G, if Algorithm 3 returns a polynomial, then with probability* $\geq \frac{11}{12}$*, it is still the correct GCD of A and B up to a constant. This is because the correctness of Algorithm 1 ensures the output of Algorithm 3 is always correct.*

5 DROPPING THE TERM BOUND

Algorithm 3 requires a term bound T for $\#G$ as input. In this section we remove this requirement. Remark 4.2 means we can simply call Algorithm 3 with $T = 2, 2^2, 2^3, 2^4, \ldots$. Once a polynomial is output instead of "Failure", it is the GCD with probability $\geq \frac{11}{12}$.

Algorithm 4 GCD algorithm for $\mathbb{F}_q[x_1, \ldots, x_n]$.

Require: Two polynomials $A, B \in \mathbb{F}_q[x_1, \ldots, x_n]$.
Ensure: $H = \gcd(A, B)$ up to a constant with probability $\geq \frac{11}{12}$.
1: $Tm := \prod_{i=1}^{n}(1 + \min(\deg(A, x_i), \deg(B, x_i)))$. {* $Tm \geq \#G$ *}
2: $T := 2$.
3: **repeat**
4: Call Algorithm 3 to compute the GCD of A, B using T as a guess for $\#G$. Let H be the output of Algorithm 3.
5: **if** $H \neq$ "Failure" **then** return H **end if**
6: **if** $T < Tm$ **then** $T := 2T$ **end if**
7: **end repeat**

Theorem 5.1. *Algorithm 4 works correctly as specified.*

Proof. We let $T = 2, 4, 8, \ldots$, once $T \geq T_{\text{o}}$, then with probability $\geq \frac{2}{3}$, Algorithm 3 returns $G = \gcd(A, B)$ up to a constant. As mentioned in Remark 4.2, once Algorithm 3 returns a polynomial, with probability $\geq \frac{11}{12}$, it is the GCD of A and B up to a constant. □

The following theorem gives the complexity of Algorithm 4.

Theorem 5.2. *Let* $A, B \in \mathbb{F}_q[x_1, \ldots, x_n]$*. Algorithm 4 computes the correct GCD* $G = \gcd(A, B)$ *using expected* $O^{\sim}(nT_{\text{in}} d \log q \log^3 T_{\text{o}} + n^2 d^2 T_{\text{o}} \log q)$ *bit operations where* $T_{\text{in}} := \#A + \#B$ *and* $T_{\text{o}} := \#G$ *and* $d := \max_{i=1}^{n} \max(\deg(A, x_i), \deg(B, x_i))$.

Proof. When T is fixed. In Step 4, the bit complexity is $C_T := O^{\sim}(nT_{\text{in}} d \log^2 T \log q + n^2 d^2 T \log q)$ as Algorithm 3 calling at most $\lceil \log_2 T \rceil$ times Algorithm 1 and $D \in O(nd)$. We obtain the complexity due to Theorem 4.17. Now $T = 2, 2^2, \ldots, 2^{\kappa-1}, 2^{\kappa}, 2^{\kappa}, 2^{\kappa}, \ldots$, where $\kappa = \lceil \log_2(Tm) \rceil$. We keep calling Algorithm 3 until it outputs a polynomial. When $T \geq \#G$, Algorithm 3 returns G with probability $\geq \frac{2}{3}$. Let $L := \lceil \log_2 T_{\text{o}} \rceil$. Then $2^L \geq T_{\text{o}}$. Set the event $\text{E}_\ell :=$ { when Algorithm 4 calls Algorithm 3 ℓ-th times, it returns a polynomial}. Denote $\mathbf{Com}_\ell := \sum_{i=1}^{\ell} C_{2^i} \in O^{\sim}(nT_{\text{in}} d \log q \ell^3 + n^2 d^2 2^\ell \log q)$ as the bit complexity of Algorithm 4 when event E_ℓ occurs. For $\ell \leq T$, we have $\mathbf{Com}_\ell \leq \sum_{i=1}^{L} C_{2^i} \in O^{\sim}(nT_{\text{in}} d \log q \log^3 T_{\text{o}} + n^2 d^2 T_{\text{o}} \log q)$. For $\ell > L$, when event E_ℓ occurs, it means that in the $L, L+1, \ldots, (\ell-1)$-th calling of Algorithm 3, it does not return a polynomial. So the probability $P(\text{E}_\ell) \leq (\frac{1}{3})^{\ell - L}$. So the expected complexity is $\leq \mathbf{Com}_L + \sum_{\ell=L+1}^{\infty} P(\text{E}_\ell)\mathbf{Com}_\ell$. As $\sum_{\ell=L+1}^{\infty} \ell^3 (\frac{1}{3})^{\ell-L} \in O(L^3)$ and $\sum_{\ell=L+1}^{\infty} 2^\ell (\frac{1}{3})^{\ell-L} \in O(2^L)$, we have $\sum_{\ell=L+1}^{\infty} P(\text{E}_\ell)\mathbf{Com}_\ell + \mathbf{Com}_L \in O^{\sim}(nT_{\text{in}} d \log q \log^3 T_{\text{o}} + n^2 d^2 T_{\text{o}} \log q)$ bit operations. □

6 IMPLEMENTATION NOTES

We have implemented our new GCD algorithm in Maple and Stage 2 of Algorithm 1 coded in C for prime fields $\mathbb{F}_p$ for $p < 2^{63}$ using signed 64 bit integer arithmetic. We use Algorithm 4 to compute $G = \gcd(A, B)$ of polynomials in $\mathbb{Z}[x_1, \ldots, x_n]$ by computing G modulo a sequence of primes and using Chinese remaindering. We wait until the result of Chinese remaindering does not change then use trial division over $\mathbb{Z}$ to prove correctness.

Algorithm 4 tries $T = 2, 4, 8, \ldots$ until it succeeds. The total number of calls to Algorithm 1, equivalently, the total number of bivariate gcds in $\mathbb{F}_q[y, z]$ done, is then $\sum_{i=1}^{\log_2 T} i \in O(\log_2^2 T)$. We can reduce this for the second and subsequent primes to $O(\log_2 T)$ by, if Algorithm 4 used $T = t$, then for the next prime we initialize $T := \max(2, \frac{t}{2})$ in Step 2 of Algorithm 4.

In Algorithm 1 t is the number of terms in y of our the bivariate gcd. If $|G^*| \ll t$, this value for T is unlikely to succeed. So we require $2|G^*| \geq t$ to try to recover more terms in G. This reduces the total number of calls to Algorithm 1 to $O(\log_2 T)$.

Let $A, B \in \mathbb{F}_p[x_1, \ldots, x_n]$, $G = \gcd(A, B)$, $C = A/G$ and $D = B/G$. Another key improvement is to reconstruct the smaller of G, C and D. If $\#C \ll \#G$ then reconstructing C instead of G will require a smaller value of T. Maple's gcd algorithm does not do this and it is very evident in our Benchmarks.

Let $T_{\text{in}} = \#A + \#B$ and $T_{\text{o}} = \min(\#G, \#C, \#D)$. For sparse inputs $T_{\text{in}} \gg T_{\text{o}}$. This means the dominating cost of our algorithm will usually be evaluating the inputs at $x_i = (\gamma_i z - \alpha_i) y^{s_i}$ at $z = b_k$ in Step 12 of Algorithm 1. In Step 12, Algorithm 1 computes

$$\bar{g}_k = \gcd(A(x_i = (\gamma_i b_k - \alpha_i) y^{s_i}), B(x_i = (\gamma_i b_k - \alpha_i) y^{s_i}))$$

for each b_k in a loop. Because we evaluate z and not y, if we store the evaluations of the monomials of A and B at $x_i = (\gamma_i b_k - \alpha_i)$, for each b_k, we can reuse them for all choices of $\mathbf{s}$. Similarly, if we also store the evaluations of the monomials of A and B at $x_i = y^{s_i}$, we can reuse them for each b_k.

7 EXPERIMENTAL RESULTS

Our benchmarks were run on an Intel Gold 6342 server using one core. Maple 2022 and Magma V2.26-12 were used.

Our first benchmark is for GCD problems with $n = 9$ variables. We create t terms for G and s terms for C and D where each monomial is chosen randomly from the set of monomials of total degree at most 30 and each integer coefficient is chosen randomly from $[-99, 99]$. The values of s and t mean the input polynomials $A = CG$ and $B = GD$ have about 10^6 terms. The different choices of s and t include cases where $\#G \ll \#C$, $\#G = \#C$ and $\#G \gg \#C$.

The timings in Table 1 in column MGCD are for our new algorithm. It used two primes $p_1 = 2^{62} - 57$ and $p_2 = 2^{62} - 87$ to compute $\gcd(A, B)$. Column eval is the time spent evaluating the input polynomials A and B. Column T is the value of the T parameter in our algorithm. Notice T is much smaller than $\min(s, t)$.

The timings in columns Maple and Magma are for their main gcd algorithm. Maple and Magma both use Zippel's algorithm [24]. The Maple implementation is described in [6]. The timings in column MonHu are for the Monagan-Hu gcd algorithm from [8, 15].

s	t	MGCD	T	eval	Maple	Magma	MonHu
10^5	10^1	11.69	4	10.22	78.92	39.90	0.661
10^4	10^2	13.55	8	10.24	197.6	9.98	1.488
10^3	10^3	29.65	64	10.27	1054.9	37.49	6.868
10^2	10^4	13.43	16	10.24	14568.	27.68	1.087
10^1	10^5	10.67	4	9.47	NA	144.8	0.696

Table 1: Benchmark 1: Timings in CPU seconds for n=9

One reason why the Monagan-Hu algorithm is faster than ours on Benchmark 1 is that its evaluation points form a geometric sequence which reduces the number of multiplications needed by a factor of n for each evaluation. Monagan-Hu does a Kronecker substitution to map the coefficients of A and B from $\mathbb{F}_p[x_2, \ldots, x_n]$ into $\mathbb{F}_p[y]$. If the inputs have more variables or have higher degree, the prime p needed may overflow machine precision and then Monagan-Hu will use multi-precision integer arithmetic. This happens for Benchmark 1 when $n \geq 12$. Benchmark 2 below is the same as Benchmark 1 but with $n = 18$ variables instead of $n = 9$. For Benchmark 2 our new algorithm is the fastest. %begingroup

s	t	MGCD	T	eval	Maple	Magma	MonHu
10^5	10^1	38.17	4	22.67	494.9	166.5	310.2
10^4	10^2	48.76	16	24.62	1473.2	79.50	450.8
10^3	10^3	92.60	128	24.68	14287.	447.8	4358.
10^2	10^4	50.54	16	24.64	NA	76.73	605.7
10^1	10^5	39.61	4	22.59	NA	188.1	150.6

Table 2: Benchmark 2 timings in CPU seconds for n=18

8 CONCLUSION

In this paper, we proposed a new method for computing the GCD of sparse multivariate polynomials over finite fields. GCD computation is an important operation of a Computer Algebra System. We gave the explicit bit complexity for the algorithm, which is polynomial in the sparse representation of the input and output and their degrees. Our initial experimental results are very good. The core of our algorithm, Steps 11 to 14 of Algorithm 1, is easily parallelized.

ACKNOWLEDGMENTS

This first author was supported by the National Key R&D Program of China (Grant No.2021YFB3100200). The second author was supported by Maplesoft and NSERC of Canada.

REFERENCES

[1] A. Arnold and D. S. Roche. Multivariate sparse interpolation using randomized kronecker substitutions. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pages 35–42, 2014.
[2] W. S. Brown. On Euclid's algorithm and the computation of polynomial greatest common divisors. *Journal of the ACM (JACM)*, 18(4):478–504, 1971.
[3] W. S. Brown and J. F. Traub. On Euclid's algorithm and the theory of subresultants. *J. ACM*, 18(4):505–514, 1971.
[4] B. W. Char, K. O. Geddes, and G. H. Gonnet. GCDHEU: heuristic polynomial GCD algorithm based on integer GCD computation. *J. Symb. Comput.*, 7(1):31–48, 1989.
[5] G. E. Collins. Subresultants and reduced polynomial remainder sequences. *J. ACM*, 14(1):128–142, 1967.
[6] J. de Kleine, M. B. Monagan, and A. D. Wittkopf. Algorithms for the non-monic case of the sparse modular GCD algorithm. In *Proceedings of ISSAC 2005*, pages 124–131. ACM, 2005.
[7] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic, 1992.
[8] J. Hu and M. Monagan. A fast parallel sparse polynomial GCD algorithm. *Journal of Symbolic Computation*, 105:28–63, 2021.
[9] Q.-L. Huang and X.-S. Gao. Revisit sparse polynomial interpolation based on randomized kronecker substitution. In *Computer Algebra in Scientific Computing: 21st International Workshop, CASC 2019, Moscow, Russia, August 26–30, 2019, Proceedings 21*, pages 215–235. Springer, 2019.
[10] Q.-L. Huang and X.-S. Gao. New sparse multivariate polynomial factorization algorithms over integers. In *Proceedings of the 2023 International Symposium on Symbolic and Algebraic Computation*, pages 315–324, 2023.
[11] E. Kaltofen. Sparse hensel lifting. In *Proceedings of EUROCAL '85*, volume 204 of *LNCS*, pages 4–17, 1985.
[12] E. Kaltofen. Greatest common divisors of polynomials given by straight-line programs. *Journal of the ACM (JACM)*, 35(1):231–264, 1988.
[13] E. Kaltofen and B. M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symb. Comput.*, 9(3):301–320, 1990.
[14] H.-C. Liao and R. J. Fateman. Evaluation of the heuristic polynomial gcd. In *Proceedings of the 1995 international symposium on Symbolic and algebraic computation*, pages 240–247, 1995.
[15] M. Monagan. Speeding up polynomial gcd, a crucial operation in maple. *Maple Transactions*, 2(1), 2022.
[16] J. Moses and D. Y. Y. Yun. The EZ GCD algorithm. In I. E. Perlin and T. J. M. Jr., editors, *Proceedings of the ACM annual conference, Atlanta, Georgia, USA, August 27-29, 1973*, pages 159–166. ACM, 1973.
[17] T. Sasaki and M. Suzuki. Three new algorithms for multivariate polynomial GCD. *J. Symb. Comput.*, 13(4):395–412, 1992.
[18] V. Shoup. Fast construction of irreducible polynomials over finite fields. *J. Symb. Comput.*, 17(5):371–391, 1994.
[19] M. Tang, B. Li, and Z. Zeng. Computing sparse GCD of multivariate polynomials via polynomial interpolation. *Journal of Systems Science and Complexity*, 31(2):552–568, 2018.
[20] K. Tsuji. An improved EZ-GCD algorithm for multivariate polynomials. *J. Symb. Comput.*, 44(1):99–110, 2009.
[21] J. van der Hoeven and G. Lecerf. On sparse interpolation of rational functions and gcds. *Communications in Computer Algebra*, 55(1):1–12, 2021.
[22] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra (3rd ed.)*. Cambridge University Press, 2013.
[23] P. S. Wang. The EEZ-GCD algorithm. *SIGSAM Bull.*, 14(2):50–60, 1980.
[24] R. Zippel. Probabilistic algorithms for sparse polynomials. In E. W. Ng, editor, *Proceedings of EUROSAM '79*, volume 72 of *LNCS*, pages 216–226. Springer, 1979.