A New Black Box Factorization Algorithm - the Non-monic Case

Tian Chen
tca71@sfu.ca
Simon Fraser University
Burnaby, BC, Canada

Michael Monagan
mmonagan@sfu.ca
Simon Fraser University
Burnaby, BC, Canada

ABSTRACT

Given a sparse polynomial $a \in \mathbb{Z}[x_1, \cdots, x_n]$ represented by a black box, we aim to find its factors in the sparse representation. The authors have previously developed an efficient algorithm for the monic and square-free case. In this work, we contribute a new algorithm that also handles the non-monic, non-square-free and non-primitive cases. We give a worst case complexity analysis with failure probabilities. The required number of probes to the black box in our algorithm is much less than the previously best known algorithm by Rubinfeld and Zippel in 1994. We have also implemented our new algorithm in Maple with all major subroutines in C. Our benchmarks show that our algorithm is much faster than the current best determinant and factorization algorithms in Maple and Magma.

KEYWORDS

black box representation, multivariate polynomial factorization, sparse Hensel lifting, bivariate Hensel lifting

ACM Reference Format:
Tian Chen and Michael Monagan. 2023. A New Black Box Factorization Algorithm - the Non-monic Case. In *International Symposium on Symbolic and Algebraic Computation 2023 (ISSAC 2023), July 24–27, 2023, Tromsø, Norway*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3597066.3597119

1 INTRODUCTION

The black box representation of a polynomial $f \in \mathbb{Z}[x_1, ..., x_n]$ is one of the most space efficient implicit representations [13]. It is a program which accepts a prime p and an evaluation point $\boldsymbol{\alpha} \in \mathbb{Z}_p^n$ and outputs $f(\boldsymbol{\alpha}) \bmod p$ (Figure 1). On the other hand, the sparse representation of f is explicit. It consists of a list of coefficients $c_k \neq 0$, $c_k \in \mathbb{Z}$ and exponents $(e_{k_1}, \cdots, e_{k_n})$ such that $f = \sum_{k=1}^{t} c_k \cdot x_1^{e_{k_1}} \cdots x_n^{e_{k_n}}$, where t is the number of non-zero terms of f (Chap. 16 of [7]).

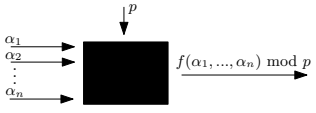

Figure 1: $f \in \mathbb{Z}[x_1, ..., x_n]$ represented by a black box.

ISSAC 2023, July 24–27, 2023, Tromsø, Norway

ACM ISBN 979-8-4007-0039-2/23/07…$15.00
https://doi.org/10.1145/3597066.3597119

Given a sparse polynomial $a \in \mathbb{Z}[x_1, \cdots, x_n]$ represented by a black box, we aim to compute its factors in the sparse representation. An example is to factor the determinant of a matrix A with multivariate polynomial entries. Usually the factors of $a = \det A \in \mathbb{Z}[x_1, \ldots, x_n]$ have a lot fewer terms than a. We save the memory space needed to store a in its sparse representation, as well as the cost of evaluating a at each Hensel lifting step [5].

In 1990, Kaltofen and Trager [13] contributed the first black box factorization algorithm for multivariate polynomials with coefficients in a field. Their algorithm first computes the black boxes of the factors, then the sparse representation of the factors can be recovered using *sparse polynomial interpolation*. Early references for sparse polynomial interpolation include [1, 14, 33]. For a recent bibliography we refer the reader to Roche [24]. Then in 1994, a simpler algorithm for factoring polynomials in $\mathbb{Z}[x_1, \cdots, x_n]$ is presented by Rubinfeld and Zippel [25]. Instead of using bivariate transformations to compute black boxes of the factors in [13], Rubinfeld and Zippel's algorithm uses simple evaluations for each variable $x_2, \cdots, x_n$. We refer the algorithms described above as Approach I, shown in Figure 2.

Figure 2 shows another two ways to compute the factors of a in the sparse representation. Approach 0 is the least efficient as it first interpolates the sparse representation of a and then factors it using a sparse Hensel lifting algorithm, e.g. Algorithm CMSHL [4]. In 2022, the authors contributed Approach II [5] which computes the factors in the sparse representation directly by a modified CMSHL algorithm. It works only for the monic and square-free case. Approach II is the most efficient of the three. It outperforms Approach I as it requires less number of probes to the black box than Approach I [5]. We shall give a more rigorous complexity analysis in Section 4.

In this work, we present the following new contributions. First, we give a new black box factorization algorithm that also handles the non-monic, non-square-free, and non-primitive polynomial input $a \in \mathbb{Z}[x_1, \cdots, x_n]$. Second, we implemented our new algorithm in Maple with all major subroutines in C and tested on a variety of problems. We compared our timings with Maple and Magma's current best determinant and factorization algorithms and our algorithm is much faster. We also give a detailed complexity analysis with failure probabilities. We show that the number of probes to the black box required in our algorithm is much less than the algorithm in [25] (the best known algorithm for Approach I).

If the input polynomial a is in the sparse representation and is non-monic, two methods are known to pre-compute the leading coefficients of the factors. One is Wang's *leading coefficient correction* [28], and the other is by Kaltofen [11]. However, in our algorithm, we do not need to pre-compute the coefficients of the factors. In order to obtain the correct leading coefficients of the factors, we

scale the bivariate images at each Hensel lifting step to match their leading coefficients with the input factors.

This paper is organized as follows. Section 2 provides a detailed description of our algorithm CMSHL and a proof of correctness. Section 3 presents our implementation results with two timing benchmarks. Section 4 gives a worst case complexity analysis with failure probabilities of our new algorithm, and a comparison of the number of probes to the black box in our algorithm with the algorithm in [25]. Section 5 is conclusion and future work.

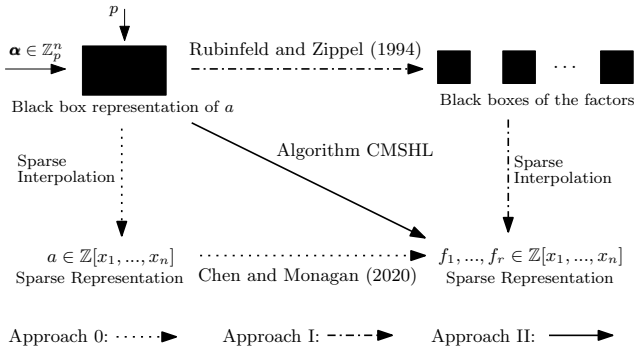

Figure 2: Factorize $a \in \mathbb{Z}[x_1, ..., x_n]$ represented by a black box.

2 ALGORITHM CMSHL: NON-MONIC CASE

The following steps are performed by our new algorithm. First, we choose a large prime p (e.g. $p = 2^{62} - 57$) and a positive integer $\tilde{N}$ with $\tilde{N} < p$. Then, an evaluation point $\boldsymbol{\alpha} = (\alpha_2, \cdots, \alpha_n) \in \mathbb{Z}^{n-1}$ is chosen randomly from $[1, \tilde{N} - 1]^{n-1}$ and $a(x_1, \boldsymbol{\alpha})$ is factored over $\mathbb{Z}$. Since we have a modular black box **B**, in order to compute $a(x_1, \boldsymbol{\alpha}) \in \mathbb{Z}[x_1]$, we used Chinese remaindering with different primes to get the coefficients of $a(x_1, \boldsymbol{\alpha})$ in $\mathbb{Z}$.

By Hilbert's irreducibility theorem [9], the pattern of the irreducible factors remains the same with high probability.

Let $P \in \mathbb{Z}[x_1, \cdots, x_n]$ be an irreducible polynomial in $\mathbb{Z}$. We call a point $(\alpha_2, \cdots, \alpha_n) \in \mathbb{Z}^{n-1}$ *Hilbertian* if $P(x_1, \alpha_2, \cdots, \alpha_n)$ remains irreducible [16]. The sharpest result on a bound for the number of non-Hilbertian points of $P(x_1, \cdots, x_n)$ was obtained by Cohen [6], stated in [25]:

Proposition 2.1. *Let $R(d, n, \tilde{N})$ be the number of non-Hilbertian points $(\alpha_2, \cdots, \alpha_n) \in \mathbb{Z}^{n-1}$ with $0 \le \alpha_i < \tilde{N}$ ($\tilde{N} \in \mathbb{Z}^+$) for an irreducible polynomial $P(x_1, \cdots, x_n)$ of degree d. Then,*

$$R(d, n, \tilde{N}) < \bar{c}(d)\tilde{N}^{n-3/2}\log(\tilde{N}), \tag{1}$$

where $\bar{c}$ depends only on the degree of the irreducible polynomial.

It is conjectured in [25] that $\bar{c}(d) < c_1 d^{c_2}$ for some absolute constants c_1, c_2. We can see that $\lim_{\tilde{N}\to\infty} R(d, n, \tilde{N})/\tilde{N}^{n-1} = 0$. Thus, a sufficiently large $\tilde{N}$ ensures a very low failure probability.

Now, let the factorization of a over $\mathbb{Z}$ be of the form

$$a = h f_1^{e_1} f_2^{e_2} \cdots f_r^{e_r} \in \mathbb{Z}[x_1, \cdots, x_n], \tag{2}$$

where $\deg(f_\rho, x_1) > 0$, f_ρ is irreducible over $\mathbb{Z}$ $(1 \le \rho \le r)$ with $\operatorname{sgn}(f_\rho) = 1$, i.e. $\operatorname{lcoeff}(f_\rho) > 0$ and $h = \operatorname{cont}(a, x_1) \in \mathbb{Z}[x_2, \cdots, x_n]$ is the content of a in x_1 (not necessarily factored at this stage).

Then, with high probability (w.h.p.),

$$a(x_1, \boldsymbol{\alpha}) = \hat{h}\hat{f}_1^{e_1}\hat{f}_2^{e_2}\cdots\hat{f}_r^{e_r} \in \mathbb{Z}[x_1], \tag{3}$$

where $\hat{f}_\rho(x_1, \boldsymbol{\alpha}) := (1/\lambda_\rho) f_\rho(x_1, \boldsymbol{\alpha})$ for some constant $\lambda_\rho \in \mathbb{Z}$ $(1 \le \rho \le r)$ and $\hat{f}_\rho$ is irreducible in $\mathbb{Z}[x_1]$ with $\operatorname{sgn}(\hat{f}_\rho) = 1$. Thus, $\hat{h}(\boldsymbol{\alpha}) = \lambda_h h(\boldsymbol{\alpha}) \in \mathbb{Z}$ with $\lambda_h = \prod_{\rho=1}^r \lambda_\rho^{e_\rho} \in \mathbb{Z}$.

More explicitly,

$$\begin{aligned} a(x_1, \boldsymbol{\alpha}) &= h(\boldsymbol{\alpha}) f_1(x_1, \boldsymbol{\alpha})^{e_1} \cdots f_r(x_1, \boldsymbol{\alpha})^{e_r} \\ &= h(\boldsymbol{\alpha}) \left(\lambda_1 \hat{f}_1(x_1, \boldsymbol{\alpha})\right)^{e_1} \cdots \left(\lambda_r \hat{f}_r(x_1, \boldsymbol{\alpha})\right)^{e_r} \text{ w.h.p.} \\ &= \underbrace{h(\boldsymbol{\alpha}) \left(\prod_{\rho=1}^r \lambda_\rho^{e_\rho}\right)}_{\hat{h}(\boldsymbol{\alpha})} \hat{f}_1(x_1, \boldsymbol{\alpha})^{e_1} \cdots \hat{f}_r(x_1, \boldsymbol{\alpha})^{e_r}. \end{aligned} \tag{4}$$

The evaluation point $\boldsymbol{\alpha}$ also must satisfy the *weak SHL assumption* [4, 21] for each factor at every Hensel lifting step. The following Lemma from [4] is essential for our algorithm to succeed w.h.p.

Lemma 2.2. *Let $f \in \mathbb{Z}_p[x_1, \cdots, x_j]$ and α_j be a randomly chosen element in $\mathbb{Z}_p$. Let $f = \sum_{i=0}^{df_j} \sigma_i(x_1, \cdots, x_{j-1})(x_j - \alpha_j)^i$ where $df_j = \deg(f, x_j)$. Then*

$$Pr[Supp(\sigma_i) \not\subseteq Supp(\sigma_0)] \le |Supp(\sigma_i)| \frac{df_j}{p - df_j + i} \text{ for } 1 \le i \le df_j,$$

where $|Supp(\sigma_i)|$ denotes the number of monomials in σ_i.

The assumption that $\operatorname{Supp}(\sigma_i) \not\subseteq \operatorname{Supp}(\sigma_0)$ for $1 \le i \le df_j$ is called *the weak SHL assumption* [4, 21].

Now we define the *square-free part* of the polynomial a as

Definition 2.3.

$$\operatorname{sqf}(a) := \prod_{\rho=1}^r f_\rho = \frac{a}{\gcd(a, \partial a/\partial x_1)} \in \mathbb{Z}[x_1, \cdots, x_n]. \tag{5}$$

Thus, w.h.p.,

$$\operatorname{sqf}(a) = \prod_{\rho=1}^r \lambda_\rho \prod_{\rho=1}^r \hat{f}_\rho(x_1, \cdots, x_n) \in \mathbb{Z}[x_1, \cdots, x_n], \tag{6}$$

where r is the number of factors in $\operatorname{sqf}(a)$, and $\hat{f}_\rho(x_1, \cdots, x_n) \in \mathbb{Q}[x_1, \cdots, x_n]$ if $|\lambda_\rho| > 1$.

Let $a_j := a(x_1, \cdots, x_j, \alpha_{j+1}, \cdots, \alpha_n) \bmod p$. Let $\hat{f}_{\rho,1} := \hat{f}_\rho(x_1, \boldsymbol{\alpha}) \bmod p$. Define $\hat{f}_{\rho,j} := \hat{f}_\rho(x_1, \cdots, x_j, \alpha_{j+1}, \cdots, \alpha_n) \bmod p$ for $2 \le j \le n$ (to be computed). Let **B** denote the black box representation of the polynomial a. The input and output to our new algorithm CMSHL for the non-monic and non-square-free case is:

- Input: A prime p, the black box **B**, $\boldsymbol{\alpha} \in \mathbb{Z}^{n-1}$, $\deg(a, x_j)$ $(1 \le j \le n)$ (pre-computed), $\hat{f}_{\rho,1} \in \mathbb{Z}_p[x_1]$ $(1 \le \rho \le r)$ s.t.
  (i) $\gcd(\hat{f}_{k,1}, \hat{f}_{l,1}) = 1$ for $k \ne l$ in $\mathbb{Z}_p[x_1]$,
  (ii) $\operatorname{sqf}(a(x_1, \boldsymbol{\alpha})) = \prod_{\rho=1}^r \lambda_\rho \prod_{\rho=1}^r \hat{f}_{\rho,1} \bmod p$.
- Output: $\hat{f}_{\rho,n} \in \mathbb{Z}_p[x_1, \cdots, x_n]$ $(1 \le \rho \le r)$ s.t.
  (i) $\operatorname{sqf}(a(x_1, \cdots, x_n)) = \prod_{\rho=1}^r \lambda_\rho \prod_{\rho=1}^r \hat{f}_{\rho,n} \bmod p$,
  Or FAIL.

Algorithm CMSHL lifts $\hat{f}_{\rho,1}(x_1)$ to $\hat{f}_{\rho,2}(x_1, x_2)$ then lifts $\hat{f}_{\rho,2}(x_1, x_2)$ to $\hat{f}_{\rho,3}(x_1, x_2, x_3)$ etc. After the j^{th} Hensel lifting step (see Algorithm 1), $\operatorname{sqf}(a_j) = \prod_{\rho=1}^r \lambda_\rho \prod_{\rho=1}^r \hat{f}_{\rho,j} \bmod p$ and $\hat{f}_{\rho,j}(x_j = \alpha_j) = \hat{f}_{\rho,j-1}$

mod p. At the end, $\text{sqf}(a_n) = \prod_{\rho=1}^{r} \lambda_\rho \prod_{\rho=1}^{r} \hat{f}_{\rho,n} \bmod p$. (Proof of correctness is given in Sect. 2.3.)

After the last Hensel lifting step, rational number reconstruction is performed on the coefficients of $\hat{f}_{\rho,n}$ for $1 \le \rho \le r$ to get the integer coefficients of the factors f_ρ in $\mathbb{Z}$.

More detailed, suppose

$$f_\rho = \sum_{k=1}^{\#f_\rho} c_k M_k \in \mathbb{Z}[x_1, \cdots, x_n],$$

where $c_k \in \mathbb{Z}$ (to be determined), M_k's are the monomials of f_ρ and $\#f_\rho$ is the number of terms in f_ρ. After the last Hensel lifting step, we have computed $\hat{f}_{\rho,n} = \sum_{k=1}^{\#f_\rho} \hat{c}_k M_k \in \mathbb{Z}_p[x_1, \cdots, x_n]$. Thus,

$$\hat{f}_{\rho,n} = \sum_{k=1}^{\#f_\rho} \hat{c}_k M_k \equiv \frac{1}{\lambda_\rho} f_\rho \bmod p \equiv \sum_{k=1}^{\#f_\rho} \frac{c_k}{\lambda_\rho} M_k \bmod p.$$

Thus, we use rational number reconstruction to obtain λ_ρ and c_k from $\hat{c}_k \equiv \frac{c_k}{\lambda_\rho} \bmod p$ $(1 \le k \le \#f_\rho)$.

To recover the factors of the content we construct a black box $\mathbf{C}$ for the content. First, we construct a black box $\mathbf{F}$ as the product of the factors found, i.e. $\mathbf{F}(x_1, \cdots, x_n) = \prod_{\rho=1}^{r} f_\rho$. Then $\mathbf{C}(x_2, \cdots, x_n) = \mathbf{B}(\alpha, x_2, \cdots, x_n)/\mathbf{F}(\alpha, x_2, \cdots, x_n)$ for α chosen at random from $\mathbb{Z}_p$. Then we use our black box algorithm to get the factors of $\mathbf{C}$ recursively. The black box $\mathbf{C}$ returns FAIL if $\mathbf{F}$ evaluates to 0 in which case we need to restart with a different α.

Example 2.4. Consider $a = f_1 f_2 \in \mathbb{Z}[x_1, \cdots, x_4]$ where

$$f_1 = (2x_2^2x_3^3 + 4)x_1^8 + (4x_2^2x_3^3 + 22x_2^2x_4^3 + 1452x_2^2x_4)x_1 + x_2^2x_3x_4 - 4x_3,$$
$$f_2 = (3x_2 + 39x_4 + 3x_3)x_1^8 + (5x_2x_3^2x_4 + 33x_2x_3x_4^2)x_1^2 - 363x_4^2 + 44.$$

In this case, $h = 1$ (a has no content in x_1, neither integer content) and $\text{sqf}(a) = a$. Let $\boldsymbol{\alpha} = (2, 3, 9)$,

$$\begin{aligned} a(x_1, \boldsymbol{\alpha}) &= 80520x_1^{16} + 3706560x_1^{10} + \cdots - 3430775304x_1 - 2818464 \\ &= \underbrace{4}_{\lambda_1} \underbrace{(55x_1^8 + 29214x_1 + 24)}_{\hat{f}_1} \underbrace{(366x_1^8 + 16848x_1^2 - 29359)}_{\hat{f}_2} \\ &= f_1(x_1, \boldsymbol{\alpha}) f_2(x_1, \boldsymbol{\alpha}). \end{aligned}$$

We have $\lambda_1 = 4$ and $\lambda_2 = 1$, thus $f_1(x_1, \boldsymbol{\alpha}) = \lambda_1 \hat{f}_1 = 4\hat{f}_1$ and $f_2(x_1, \boldsymbol{\alpha}) = \hat{f}_2$. The input to algorithm CMSHL is $p = 2^{31} - 1$, $\boldsymbol{\alpha}$, the black box $\mathbf{B}$, $\hat{f}_{\rho,1} = \hat{f}_\rho \bmod p$ $(\rho = 1, 2)$.

After the 1st (denoted as the 2nd) Hensel lifting step (a bivariate Hensel lift only), the algorithm outputs $\hat{f}_{\rho,2} \in \mathbb{Z}_p[x_1, x_2]$ $(\rho = 1, 2)$ s.t. $a_2 = \text{sqf}(a_2) = (\lambda_1\lambda_2)\hat{f}_{1,2}\hat{f}_{2,2}$ with

$$\hat{f}_{1,2} = (1073741837x_2^2 + 1)x_1^8 + 1073749127x_2^2x_1 + 1610612742x_2^2 + 2147483644,$$
$$\hat{f}_{2,2} = (3x_2 + 360)x_1^8 + 8424x_2x_1^2 + 2147454288.$$

After the 3rd Hensel lifting step,

$$\hat{f}_{1,3} = (1073741824x_2^2x_3^3 + 1)x_1^8 + (x_2^2x_3^3 + 1073749100x_2^2)x_1 + 536870914x_2^2x_3 + 2147483646x_3,$$
$$\hat{f}_{2,3} = (3x_2 + 3x_3 + 351)x_1^8 + (45x_2x_3^2 + 2673x_2x_3)x_1^2 + 2147454288.$$

The last Hensel lifting step outputs $\hat{f}_{\rho,4}$ $(\rho = 1, 2)$ s.t. $a_4 = \text{sqf}(a_4) = (\lambda_1\lambda_2)\hat{f}_{1,4}\hat{f}_{2,4}$ with

$$\hat{f}_{1,4} = (1073741824x_2^2x_3^3 + 1)x_1^8 + (x_2^2x_3^3 + 1073741829x_2^2x_4^3 + 363x_2^2x_4)x_1 + 536870912x_2^2x_3x_4 + 2147483646x_3$$
$$\hat{f}_{2,4} = (3x_2 + 39x_4 + 3x_3)x_1^8 + (5x_2x_3^2x_4 + 33x_2x_3x_4^2)x_1^2 + 2147483284x_4^2 + 44.$$

Now, we notice that $4\hat{f}_{1,4} \bmod p = f_1$ and $\hat{f}_{2,4} \bmod p = f_2$ (mod is taken in the symmetric range). The values for λ_1, λ_2 are still unknown, so we perform rational number reconstruction on coefficients of $\hat{f}_{\rho,4}$ to find λ_ρ and hence get the true factors f_ρ $(\rho = 1, 2)$. In Maple, we do the following for the first factor (similarly for the second factor):

```
> iratrecon(f_hat[1,4],p);
```

$$\text{ff}_1 := \frac{1}{2}x_2^2x_1^8x_3^3 + x_1^8 + x_1x_2^2x_3^3 + \frac{11}{2}x_2^2x_1x_4^3 + 363x_2^2x_1x_4 + \frac{1}{4}x_2^2x_3x_4 - x_3$$

λ_1 is the least common multiple of the denominators of coefficients of ff_1. Multiply ff_1 by λ_1, we get the true factor $f_1 \in \mathbb{Z}[x_1, \cdots, x_n]$:

```
> f[1] := numer(ff[1]);
```

$$\text{f}_1 := 2x_2^2x_1^8x_3^3 + 4x_1^8 + 4x_1x_2^2x_3^3 + 22x_2^2x_1x_4^3 + 1452x_2^2x_1x_4 + x_2^2x_3x_4 - x_3$$

Note: `iratrecon` could return FAIL. If we use Wang's Euclidean algorithm for rational reconstruction [29], to guarantee the correct answer, we need

$$2\lambda_\rho \max_{\rho=1}^{r} \|f_\rho\|_\infty < p \tag{7}$$

for all $1 \le \rho \le r$, where $\|f_\rho\|_\infty$ is the max-norm of the factor $f_\rho \in \mathbb{Z}[x_1, \cdots, x_n]$. Form various examples we tested, $\lambda_\rho \ll p$.

For our benchmarks, we used $p = 2^{62} - 57$. If p is not big enough to recover $a \in \mathbb{Z}[x_1, \cdots, x_n]$, we can simply use larger and larger primes, e.g. $p_2 > p_1^2$, $p_3 > p_2^2$, etc. until the algorithm succeeds.

The j^{th} Hensel lifting step of algorithm CMSHL for the non-monic and non-square-free case is shown in Algorithm 1. The key idea to our non-monic algorithm is to interpolate the square-free part of the bivariate images of a and then use them to perform non-monic bivariate Hensel lifts. In step 12–14, a square-free image of a, $\text{sqf}((a(x_1, Y_k, x_j))$, is computed probabilistically via a bivariate gcd computation and a division (correct up to a constant). The content of $a(x_1, Y_k, x_j)$ in x_1 has also been removed. Step 15 then makes A_{sf} monic, i.e. $\text{lc}(\text{sqf}(a(x_1, Y_k, x_j)), x_1)$ becomes monic in x_j. Step 16 evaluates the input factors to get a univariate image $\hat{f}_{\rho,j-1}(x_1, Y_k)$. Then at step 19, a non-monic bivariate Hensel lift (BHL) is performed, so we get a bivariate image of the factors, $\hat{f}_{\rho,j}(x_1, Y_k, x_j)$. After obtaining s bivariate images of the factors (s is defined in step 7 in Algorithm 1), we use them to recover the factor $\hat{f}_{\rho,j} \in \mathbb{Z}_p[x_1, \cdots, x_j]$ via Vandermonde solves at step 25.

Our algorithm also features that every major subroutine is parallelizable. In our benchmarks, the bottleneck is probes to the black box which involves evaluations of polynomials at multiple points. This could be done in parallel to speed up the computation.

Algorithm 1 CMSHL: Hensel lifting x_j (non-monic).

Input: A prime p, $\alpha_j \in \mathbb{Z}_p$, the black box **B**, $d_i = \deg(a, x_i)$ $(1 \le i \le n)$, $\hat{f}_{\rho,j-1} \in \mathbb{Z}_p[x_1, \cdots, x_{j-1}]$ $(1 \le \rho \le r)$ s.t. $\text{sqf}(a_j(x_j = \alpha_j)) = \prod_{\rho=1}^{r} \lambda_\rho \prod_{\rho=1}^{r} \hat{f}_{\rho,j-1}$ with $j > 2$.
Output: $\hat{f}_{\rho,j} \in \mathbb{Z}_p[x_1, \cdots, x_j]$ $(1 \le \rho \le r)$ s.t. $\text{sqf}(a_j) = \prod_{\rho=1}^{r} \lambda_\rho \prod_{\rho=1}^{r} \hat{f}_{\rho,j}$ and $\hat{f}_{\rho,j}(x_j = \alpha_j) = \hat{f}_{\rho,j-1}$; Otherwise, **return** FAIL.
Let $\hat{f}_{\rho,j-1} = \sum_{i=0}^{df_\rho} \sigma_{\rho,i}(x_2, ..., x_{j-1}) x_1^i$ $(1 \le \rho \le r)$ where $\sigma_{\rho,i} = \sum_{k=1}^{s_{\rho,i}} c_{\rho,ik} M_{\rho,ik}$ with $M_{\rho,ik}$ the monomials in $\sigma_{\rho,i}$ and $df_\rho = \deg(\hat{f}_{\rho,j-1}, x_1)$.
Pick $\boldsymbol{\beta} = (\beta_2, \cdots, \beta_{j-1}) \in \mathbb{Z}_p^{j-2}$ at random.
Evaluate (for $1 \le \rho \le r$): $\mathcal{S}_\rho = \{S_{\rho,i} = \{m_{\rho,ik} = M_{\rho,ik}(\boldsymbol{\beta}), 1 \le k \le s_{\rho,i}\}, 0 \le i \le df_\rho\}$.
if any $|S_{\rho,i}| \ne s_{\rho,i}$ **then return** FAIL **end if**
Let s be the maximum of $s_{\rho,i}$.
for k from 1 to s **do**
  Let $Y_k = (x_2 = \beta_2^k, \cdots, x_{j-1} = \beta_{j-1}^k)$.
  $A_k \leftarrow a_j(x_1, Y_k, x_j) \in \mathbb{Z}_p[x_1, x_j]$. // via probes to **B** and dense interpolation ..... $O(s(d_1^2 d_j + d_1 d_j^2 + d_1 d_j C(\text{probe } \mathbf{B})))$
  if $\deg(A_k, x_1) \ne d_1$ **then return** FAIL **end if**
  $g_k \leftarrow \gcd(A_k, \frac{\partial A_k}{\partial x_1}) \bmod p$. ..... $O(s(d_1^2 d_j + d_1 d_j^2))$
  if $\deg(g_k, x_1) \ne d_1 - \sum_{\rho=1}^{r} df_\rho$ **then return** FAIL **end if**
  $A_{sf} \leftarrow \text{quo}(A_k, g_k) \bmod p$.
  $A_{sfm} \leftarrow A_{sf}/(\text{lc}(\text{lc}(A_{sf}, x_1), x_j)) \bmod p$.
  $F_{\rho,k} \leftarrow \hat{f}_{\rho,j-1}(x_1, Y_k) \in \mathbb{Z}_p[x_1]$ for $1 \le \rho \le r$.
  if any $\deg(F_{\rho,k}) < df_\rho$ (for $1 \le \rho \le r$) **then return** FAIL **end if**
  if $\gcd(F_{\rho,k}, F_{\phi,k}) \ne 1$ for any $\rho \ne \phi$ $(1 \le \rho, \phi \le r)$ **then return** FAIL **end if**
  $\hat{f}_{\rho,k} \leftarrow \text{BivariateHenselLift}(A_{sfm}(x_1, x_j), F_{\rho,k}(x_1), \alpha_j, p)$. ..... $O(s(\tilde{d}_1 \tilde{d}_j^2 + \tilde{d}_1^2 \tilde{d}_j)) \subseteq O(s(d_1 d_j^2 + d_1^2 d_j))$
end for
Let $\hat{f}_{\rho,k} = \sum_{l=1}^{t_\rho} \alpha_{\rho,kl} \tilde{M}_{\rho,l}(x_1, x_j) \in \mathbb{Z}_p[x_1, x_j]$ for $1 \le k \le s$ where $t_\rho = \#\hat{f}_{\rho,k}$, for $1 \le \rho \le r$.
for ρ from 1 to r **do**
  for l from 1 to t_ρ **do**
    $i \leftarrow \deg(\tilde{M}_{\rho,l}, x_1)$.
    Solve the linear system $\left\{\sum_{k=1}^{s_{\rho,i}} m_{\rho,ik}^t c_{\rho,lk} = \alpha_{\rho,tl} \text{ for } 1 \le t \le s_{\rho,i}\right\}$ for $c_{\rho,lk}$.
  end for ..... $O(s\tilde{d}_j(\sum_{\rho=1}^{r} \#\hat{f}_{\rho,j-1}))$
  $\hat{f}_{\rho,j} \leftarrow \sum_{l=1}^{t_\rho} \left(\sum_{k=1}^{s_{\rho,i}} c_{\rho,lk} M_{\rho,ik}(x_2, ..., x_{j-1})\right) \tilde{M}_{\rho,l}(x_1, x_j)$.
end for
Pick $\boldsymbol{\beta} = (\beta_2, \cdots, \beta_j) \in \mathbb{Z}_p^{j-1}$ at random.
$A_{\boldsymbol{\beta}} \leftarrow \text{sqf}(a_j(x_1, \boldsymbol{\beta})) \bmod p$ // via probes to **B**, interpolation, and sqrfree compt.
if $\hat{f}_{\rho,j}(x_1, \boldsymbol{\beta}) \mid A_{\boldsymbol{\beta}}$ **and** $\deg(\hat{f}_{\rho,j}(x_1, \boldsymbol{\beta})) = df_\rho$ for $1 \le \rho \le r$ **then return** $\hat{f}_{\rho,j}$ for $1 \le \rho \le r$
else return FAIL **end if**

2.1 The non-monic bivariate Hensel lift (BHL) in the j^{th} Hensel lifting step of CMSHL

The non-monic bivariate Hensel lift (step 19 of Algorithm 1) has the following input and output:

- Input: A prime p, $\alpha_j \in \mathbb{Z}_p$, $\text{monic}(\text{sqf}(a(x_1, Y_k, x_j)) = A_{\text{sfm}}$ (step 15), $\hat{f}_{\rho,j-1}(x_1, Y_k)$ $(1 \le \rho \le r)$ (step 16) s.t.
  (i) $\gcd(\hat{f}_{\rho,j-1}(x_1, Y_k), \hat{f}_{\phi,j-1}(x_1, Y_k)) = 1$ for $\rho \ne \phi$,
  (ii) $A_{\text{sfm}}(x_j = \alpha_j) = \xi \prod_{\rho=1}^{r} \hat{f}_{\rho,j-1}(x_1, Y_k) \bmod p$, $\xi \in \mathbb{Z}_p$.
- Output: $\hat{f}_{\rho,j}(x_1, Y_k, x_j)$ $(1 \le \rho \le r)$ s.t.
  (i) $A_{\text{sfm}} = \xi \prod_{\rho=1}^{r} \hat{f}_{\rho,j}(x_1, Y_k, x_j) \bmod p$, $\xi \in \mathbb{Z}_p$,
  (ii) $\hat{f}_{\rho,j}(x_1, Y_k, x_j)(x_j = \alpha_j) = \hat{f}_{\rho,j-1}(x_1, Y_k)$ $(1 \le \rho \le r)$,
  Or FAIL.

In the above, monic(.) means to make the $\text{lc}(\text{sqf}(a(x_1, Y_k, x_j)), x_1)$ monic in x_j. Also, we have $\xi = (\prod_{\rho=1}^{r} \lambda_\rho)/lc_{Asf} \in \mathbb{Z}_p$, where $lc_{Asf} := \text{lc}(\text{lc}(\text{sqf}(a(x_1, Y_k, x_j), x_1), x_j)$.

Notice that the output of BHL $\hat{f}_{\rho,j}(x_1, Y_k, x_j)$ satisfies

$$\hat{f}_{\rho,j}(x_1, Y_k, x_j)(x_j = \alpha_j) = \hat{f}_{\rho,j-1}(x_1, Y_k) \ (1 \le \rho \le r).$$

Thus, when evaluating the output bivariate factors $\hat{f}_{\rho,j}(x_1, Y_k, x_j)$ at $x_j = \alpha_j$, their leading coefficients equal the leading coefficients of the input factors $\hat{f}_{\rho,j-1}(x_1, Y_k)$. The correct return of the leading coefficients from BHL ensures that we have the correct leading coefficients after each Hensel lifting step of CMHSL, i.e. $\hat{f}_{\rho,j} \in \mathbb{Z}_p[x_1, \cdots, x_j]$ satisfies

$$\hat{f}_{\rho,j}(x_j = \alpha_j) = \hat{f}_{\rho,j-1} \ (1 \le \rho \le r).$$

We have modified the monic BHL algorithm developed by Monagan and Paluck in 2022 [23] to handle the non-monic case. It has a cubic cost of $O(d_1 d_j^2 + d_1^2 d_j)$. Pseudocode for non-monic bivariate Hensel lifting is shown in Algorithm 2.

There is a potential issue when $\gamma(y) = \text{lc}(a, x)$ has a high degree. Thus after step 3 of Algorithm 2, $a(x, y) \leftarrow \gamma(y)^{r-1} a(x, y)$ has a high degree in y. This happens especially when the number of factors is large. In such case, we could implement a recursive algorithm to break down the factors into a binary tree for bivariate Hensel lifts.

2.2 Pre-computing the degrees of a

Our algorithm CMSHL requires $\deg(a, x_j)$ $(1 \le j \le n)$ to be pre-computed as input. We do the following steps to compute $\deg(a, x_j)$ with high probability (w.h.p.):

- Pick $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_{j-1}, \alpha_{j+1}, \cdots, \alpha_n) \in {\mathbb{Z}_p}^{n-1}$.
- Define $g(v) := a(\alpha_1, \cdots, \alpha_{j-1}, v, \alpha_{j+1}, \cdots, \alpha_n) \bmod p$.
- Interpolate $g(v)$ w.h.p. as follows:

  for k from 1 **do**
    Pick $\beta_k \in \mathbb{Z}_p$ at random.
    $b_k \leftarrow \mathbf{B}([\alpha_1, \cdots, \alpha_{j-1}, \beta_k, \alpha_{j-1}, \cdots, \alpha_n], p)$.
    $h_k(v) \leftarrow \text{interp}([\beta_1, \cdots, \beta_k], [b_1, \cdots, b_k]) \bmod p$.
    if $h_k(v) = h_{k-1}(v)$ **then break end if**
  end for
  $h \leftarrow h_k$
  if $h = 0$ **return** -1 **else return** $\deg(h, v)$ **end if**

Algorithm 2 Non-monic bivariate Hensel lifting - cubic cost.

1: **Input:** A prime p, $\alpha \in \mathbb{Z}_p$, $a \in \mathbb{Z}_p[x, y]$ where a is primitive in x and lcoeff(lcoeff(a, x), y) = 1, $f_{\rho,0} \in \mathbb{Z}_p[x]$ for $1 \le \rho \le r$ s.t.
(i) $\gcd(f_{k,0}, f_{l,0}) = 1$ for $k \ne l$,
(ii) $a(y = \alpha) = \xi \prod_{\rho=1}^{r} f_{\rho,0}$, $\xi \in \mathbb{Z}_p$.
2: **Output:** $f_\rho \in \mathbb{Z}_p[x, y]$ s.t.
(i) $a = \xi \prod_{\rho=1}^{r} f_\rho$,
(ii) $f_\rho(y = \alpha) = f_{\rho,0}$.
Or FAIL.
3: $\gamma(y) \leftarrow lc(a, x) \in \mathbb{Z}_p[y]$;
$a(x, y) \leftarrow \gamma(y)^{r-1} a(x, y) \in \mathbb{Z}_p[x, y]$.
4: $f_{\rho,0} \leftarrow \gamma(y) \cdot \text{monic}(f_{\rho,0}(x)) \bmod (y - \alpha) \in \mathbb{Z}_p[x]$ $(1 \le \rho \le r)$.
5: $dx \leftarrow \deg(a, x)$; $dy \leftarrow \deg(a, y)$; $df_{\rho,0} \leftarrow \deg(f_{\rho,0}, x)$.
6: $M \leftarrow \prod_{\rho=1}^{r} f_{\rho,0} \in \mathbb{Z}_p[x]$.
7: **for** ρ from 1 to r **do** $f_\rho \leftarrow f_{\rho,0}$; $M_\rho \leftarrow M/f_{\rho,0}$ **end for**
8: **for** k from 0 to dy **do**
9: $\gamma_k \leftarrow \text{coeff}(\gamma, (y - \alpha)^k)$.
10: **for** ρ from 1 to r **do** $Tf_{\rho,k} \leftarrow \gamma_k x^{df_{\rho,0}}$ **end for**
11: **end for**
12: **for** k from 1 to dy **do**
13: $ac_k \leftarrow \text{coeff}(a, (y - \alpha)^k)$.
14: $\Delta_k \leftarrow \text{coeff}(\prod_{\rho=1}^{r} f_\rho, (y - \alpha)^k)$. // via eval and interpolation.
15: $\delta_k \leftarrow \sum_{\rho=1}^{r} Tf_{\rho,k} \cdot M_\rho$.
16: $c_k \leftarrow ac_k - \Delta_k - \delta_k$.
17: **if** $\sum_{\rho=1}^{r} \deg(f_\rho, y) = dy$ and $c_k \ne 0$ **return FAIL end if**
18: **if** $c_k \ne 0$ **then**
19: Solve $\sum_{\rho=1}^{r} \bar{f}_{\rho,k} M_\rho = c_k$ for $\bar{f}_{\rho,k} \in \mathbb{Z}_p[x]$
with $\deg(\bar{f}_{\rho,k}, x) < \deg(f_{\rho,0}, x)$ for $1 \le \rho \le r$.
20: **for** ρ from 1 to r **do**
21: $f_{\rho,k} \leftarrow Tf_{\rho,k} + \bar{f}_{\rho,k}$; $f_\rho \leftarrow f_\rho + f_{\rho,k}(y - \alpha)^k$.
22: **end for**
23: **end if**
24: **end for**
25: **if** $\sum_{\rho=1}^{r} \deg(f_\rho, y) \ne dy$ **then return FAIL end if**
26: **for** ρ from 1 to r **do**
27: $\tilde{f}_\rho \leftarrow \text{primpart}(f_\rho(x, y), x))$. // $lc(lc(\tilde{f}_\rho, x), y) = 1$.
28: $lc_{eval} \leftarrow lc(\tilde{f}_\rho(x, y), x)(y = \alpha)$; $\eta \leftarrow lc(f_{\rho,0}, x)/lc_{eval}$.
29: $f_\rho \leftarrow \eta \tilde{f}_\rho$.
30: **end for**
31: **return** f_ρ for $1 \le \rho \le r$.

Note: $h(v)$ does not necessarily equal to $g(v)$. We have the following failure probability.

Proposition 2.5. *Suppose* $\deg(a, x_j) = \deg(a \bmod p, x_j)$. *Then,*

$$Pr[\deg(h, v) \ne \deg(a, x_j)] \le \frac{\deg(a, x_j)^2 + \deg(a) - \deg(a, x_j)}{p},$$

where $\deg(a)$ *is the total degree of* a.

Proof: By Schwartz-Zippel lemma [26, 31], details omitted. □

To obtain a total degree bound of a, in the case of computing the determinant of an $N \times N$ matrix A, where $\det(A) \in \mathbb{Z}[x_1, \cdots, x_n]$, we can get the maximum total degree of each row (or column) of A, i.e. $\max_{j=1}^{N}(\deg(A_{ij}))$ where $A_{ij} \in \mathbb{Z}[x_1, \cdots, x_n]$. A total degree bound is given by

$$\sum_{i=1}^{N} \max_{j=1}^{N}(\deg(A_{ij})).$$

2.3 Correctness of CMSHL

We need the following Proposition to prove Theorem 2.7.

Proposition 2.6. *The output factors* $f_\rho \in \mathbb{Z}_p[x, y]$ $(1 \le \rho \le r)$ *from Algorithm 2 (non-monic BHL) are uniquely determined.*

Proof. The multi-term Diophantine equation (step 19) gives unique solutions $\bar{f}_{\rho,k}$ with $\deg(\bar{f}_{\rho,k}, x) < \deg(f_{\rho,0}, x)$ (Theorem 2.6 in [8]). By construction in step 21, $f_{\rho,k} = \gamma_k x^{df_{\rho,0}} + \bar{f}_{\rho,k}$, where $\gamma_k x^{df_{\rho,0}}$ is the leading term of $f_{\rho,k}$. Thus, the leading coefficients of $f_{\rho,k}$ do not change, and the solution $f_\rho^{(k+1)} = f_{\rho,0} + f_{\rho,1}(y - \alpha) + \cdots + f_{\rho,k}(y - \alpha)^k$ obtained at the k^{th} iteration (step 21) is uniquely determined.

Since $lc(lc(a, x), y) = 1$, for each $\tilde{f}_\rho = \text{primpart}(f_\rho(x, y), x))$, $lc(lc(\tilde{f}_\rho, x), y) = 1$ and $\tilde{f}_\rho$'s $(1 \le \rho \le r)$ are also uniquely determined. This means

$$a = \prod_{\rho=1}^{r} \tilde{f}_\rho \in \mathbb{Z}_p[x, y]. \tag{8}$$

By evaluating (8) at $y = \alpha$, we get

$$a|_{y=\alpha} = \tilde{f}_1 \tilde{f}_2 \cdots \tilde{f}_r|_{y=\alpha} = \xi f_{1,0} f_{2,0} \cdots f_{r,0} \in \mathbb{Z}_p[x].$$

Since $\mathbb{Z}_p[x]$ is a UFD, there exists $\eta_\rho \in \mathbb{Z}_p$ $(1 \le \rho \le r)$ s.t. $\tilde{f}_\rho(y = \alpha) = (1/\eta_\rho) f_{\rho,0}$ $(1 \le \rho \le r)$.
Define $f_\rho := \eta_\rho \tilde{f}_\rho(x, y)$ $(1 \le \rho \le r)$. Since $\mathbb{Z}_p[x, y]$ is also a UFD,

$$a = \prod_{\rho=1}^{r} \tilde{f}_\rho = \frac{\prod_{\rho=1}^{r} f_\rho}{\prod_{\rho=1}^{r} \eta_\rho} = \xi \prod_{\rho=1}^{r} f_\rho \in \mathbb{Z}_p[x, y],$$

where $\xi = 1/(\prod_{\rho=1}^{r} \eta_\rho)$, and f_ρ's are uniquely determined. □

Theorem 2.7. *Let* $sqf(a_j) = sqf(a(x_1, \cdots, x_j, \alpha_{j+1}, \cdots, \alpha_n))$ *mod* $p \in \mathbb{Z}_p[x_1, \cdots, x_j]$. *Let* r *be the number of square-free factors of* $a \in \mathbb{Z}[x_1, \cdots, x_n]$ *and* $\Lambda = \prod_{\rho=1}^{r} \lambda_\rho \in \mathbb{Z}$. *Let* $\hat{f}_{\rho,j} \in \mathbb{Z}_p[x_1, \cdots, x_j]$ *be the output factors after the* j^{th} *Hensel lifting step of CMHSL (Algoirhtm 1). Then,*

$$sqf(a_j) = \Lambda \prod_{\rho=1}^{r} \hat{f}_{\rho,j} \bmod p, \tag{9}$$

for all $1 \le j \le n$.

Proof. We want to show that (9) is satisfied for each j $(1 \le j \le n)$. The first Hensel lifting step $(j = 1)$ is the initial input, and (9) is satisfied. For $j = 2$, it is a bivariate Hensel lift, and from Proposition 2.6, (9) is satisfied.

Suppose (9) is satisfied at the beginning of Hensel lifting step j, i.e. (9) is satisfied for $j - 1$. Before each bivariate Hensel lift,

$$\text{sqf}(a_j(x_1, Y_k, \alpha_j)) = \Lambda \prod \hat{f}_{\rho,j-1}(x_1, Y_k) \bmod p$$
$$\Rightarrow \text{monic}(\text{sqf}(a_j(x_1, Y_k, \alpha_j))) = \Lambda / lc_{Asf} \prod \hat{f}_{\rho,j-1}(x_1, Y_k) \bmod p.$$

And after each BHL, Proposition 2.6 ensures unique $\hat{f}_{\rho_j}$'s s.t.

$$\text{monic}(\text{sqf}(a_j(x_1, Y_k, x_j))) = \Lambda/lc_{Asf} \prod \hat{f}_{\rho,j}(x_1, Y_k, x_j) \bmod p$$
$$\Rightarrow \text{sqf}(a_j(x_1, Y_k, x_j)) = \Lambda \prod \hat{f}_{\rho,j}(x_1, Y_k, x_j) \bmod p.$$

After all bivariate Hensel lifts, Vandermonde solves give unique solutions for coefficients $c_{\rho,lk}$ to recover $\hat{f}_{\rho,j}$. Therefore,

$$\text{sqf}(a_j) = \Lambda \prod_{\rho=1}^{r} \hat{f}_{\rho,j} \bmod p.$$

□

3 IMPLEMENTATION RESULTS

Before presenting the complexity analysis, we would like to show our timing benchmarks. We have made a hybrid Maple + C implementation to our new algorithm. The main program is coded in Maple with major subroutines coded in C. The following subroutines in each Hensel lifting step are coded in C to speed up the computation:

- Step 10: Probes to the black box **B** and dense interpolation,
- Step 16: Evaluations of the factors $\hat{f}_{\rho,j-1}$,
- Step 19: Non-monic bivaraite Hensel lifts,
- Stpe 25: Vandermonde solves.

For step 10, the matrix A is converted to a list of polynomials to be passed into a C program for evaluations (BB eval). After evaluating each polynomial entry, another C program is called to calculate its determinant in $\mathbb{Z}_p$ (BB det). To compute $A_k = a_j(x_1, Y_k, x_j)$, we acquire $O(d_1 d_j)$ such evaluations and then perform dense interpolations to get this bivariate image.

For step 16, we evaluate the polynomials $\hat{f}_{\rho,j-1}$ using two arrays for each factor. One array stores the coefficients of $\hat{f}_{\rho,j-1}$ in x_1, the other array stores its monomial evaluations. This enables us to make use of the previous evaluation points, since our evaluation points $(\beta_2^k, \cdots, \beta_{j-1}^k)$ are simply powers of the β_i's. For details of this step, see also [5].

Step 19 uses the new cubic bivariate Hensel lifting (BHL) algorithm developed by Monagan and Paluck [23] in 2022. One multifactor BHL costs $O(d_1 d_j^2 + d_1^2 d_j)$ arithmetic operations in $\mathbb{Z}_p$.

The Vandermonde solves in step 25 use the classical algorithm of Zippel [33]. It does $O(s_{\rho,i}^2)$ arithmetic operations in $\mathbb{Z}_p$.

3.1 Benchmarks

We present two timing benchmarks. All timings were obtained on 2 Intel Xeon E5-2660 8 core CPUs with 64 GB RAM. We used $p = 2^{62} - 57$ and $\tilde{N} = 4001$.

The first benchmark presents timings to compute the determinants of matrices B_n, where each B_n consists of four factors. For example, B_4 is of size 8×8 and it has 4 variables:

$$B_4 = \begin{bmatrix} uvw & v & uvw+v+w & \cdots & uvw+v \\ v & uvw & uvw+2v & \cdots & uvw+v \\ w & v & uvw+v+w & \cdots & v+w \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w & v & uvw+v+w & \cdots & 2vwx+2ux+3v+4w \end{bmatrix}.$$

$$\begin{aligned} \det(B) = -\,&(-v^2w^2x^2 + uvwx^2 + vw^2x - uwx + v^2 - 2vw + w^2) \\ &(v^2w^2x^2 + uvwx^2 + vw^2x + uwx - v^2 - 2vw - w^2) \\ &(u^2v^2w^2 + u^2vwx + uv^2w + uvx - v^2 - 2vw - w^2) \\ &(u^2v^2w^2 - u^2vwx - uv^2w + uvx - v^2 + 2vw - w^2). \end{aligned}$$

The number of terms in $\det(B_4)$ is 120, and each factor has 7 terms. And the leading coefficients in each variable is non-monic, e.g.

$$\begin{aligned} \text{lcoeff}(B, u) &= -v^6w^6x^4 + v^4w^4x^6 + v^4w^6x^2 - v^2w^4x^4, \\ \text{lcoeff}(B, v) &= u^4w^8x^4 - 2u^4w^6x^2 - 3u^2w^6x^4 + u^4w^4 + 6u^2w^4x^2 \\ &\quad + w^4x^4 - 3u^2w^2 - 2w^2x^2 + 1. \end{aligned}$$

All the matrices we used for our benchmarks are available online: `http://www.cecm.sfu.ca/~mmonagan/code/BBfactor/`

Table 1 shows the CPU timings (in seconds) for our new algorithm, compared with Maple and Magma's current best determinant and factorization algorithms. We used Maple 2022 and Magma V2.25-5 to compute the determinants of B_n and factored them. Maple 2022 uses Monagan and Tuncer's algorithm MTSHL [22] for factoring multivariate polynomials. The timings for Maple det were obtained by using Gentleman and Johnson's algorithm [10].

In Table 1, n is the number of variables of $a = \det(B_n) \in \mathbb{Z}[x_1, \cdots, x_n]$. The size of matrix B_n is of $N \times N$ with $N = 2n$. $\#f_i$'s are the number of terms in each factor of a. $\#\det(B_n)$ is the number of terms of $\det(B_n)$ in expanded form. CMSHL tot is the total time for our algorithm, and probes tot is the total number of probes to the black box **B** for CMHSL. Maple det is the time for determinant computation in Maple and Maple fac is the time for Maple's factorization. Similarly for the last section of Magma's timings.

We see that our algorithm outperformed both Maple and Magma at $n = 5$. At $n = 7$, our algorithm is more than 100 times faster than Maple and Magma. At $n = 8$, Maple ran out of memory at computing $\det(B_8)$ and CMSHL only took 19.68 seconds in total.

Table 2 shows a breakdown of timings for each subroutine at Hensel lifting the last variable x_n. The number s is the number of bivariate images needed at the last Hensel lifting step (s is defined in 7 of Algorithm 1). BB tot is the total time for step 10 (probes to the black box **B**). BB eval is the time for evaluating the polynomial entries of the matrix B_n. BB det is the time for computing the determinant in $\mathbb{Z}_p$. Eval $\hat{f}_{\rho,j}$ is the time for evaluating the factors $f_{\rho,j-1}$ at step 16. BHL is the time for bivariate Hensel lifts at step 19. VSolve is the time for Vandermonde solves at step 25.

The first benchmark is non-monic but square-free and the matrices are relatively small. We tested more examples for non-square-free and non-primitive cases, and the matrices are much larger. Table 3 shows timings for four different matrices with various n and N. For example, heron3d is of 13×13 and it has the following determinant with 7 variables:

$$\det(A) = 64as^7(as - bs + cs)(as - bs - cs)(as + bs + cs)(as+bs-cs) \underbrace{(as^4es^2 + as^2bs^2cs^2 - \cdots - cs^2es^2fs^2 + 144vo^2)^2}_{23 \text{ terms}}$$

In Table 3, n is the number of variables, the size of matrices are of $N \times N$. r is the number of square-free factors of a. $\#f_i$ are the

Table 1: Timings (in seconds) for computing $\det(B_n)$

n	5	6	7	8	9
$N = 2n$	10	12	14	16	18
$\#f_i$	12,7 12,7	32,32 32,32	56,30 56,30	167,167 167,167	153,294 253,294
$\#\det(B_n)$	701	5162	79740	1716810	7490224
CMSHL tot	0.257	0.972	3.618	19.677	40.219
probes tot	2112	6453	19584	85189	145065
Maple det	0.455	7.880	382.80	> 64 gigs	N/A
Maple fac	0.109	0.326	1.270	N/A	N/A
Maple tot	0.564	8.206	384.07	N/A	N/A
Magma det	1.680	6.290	594.60	> 3h	N/A
Magma fac	0.120	0.480	33.140	N/A	N/A
Magma tot	1.800	6.770	627.74	N/A	N/A

Table 2: Breakdown of timings for H.L. x_n.

n	5	6	7	8	9
$N = 2n$	10	12	14	16	18
H.L. x_n total	0.088	0.385	0.868	5.931	12.163
s (H.L. x_n)	9	25	31	131	201
BB tot	0.028	0.138	0.429	3.389	7.678
BB eval	0.017	0.083	0.322	2.639	5.936
BB det	0.004	0.038	0.078	0.450	1.015
Eval $\hat{f}_{\rho,j-1}$	0.001	0.014	0.015	0.074	0.128
BHL	0.003	0.012	0.018	0.050	0.082
VSolve	0.001	0.007	0.007	0.016	0.020

number of terms in each square-free factor and e_i are the corresponding powers for those factors in $\det(A)$ (as factored form). The number $\max \lambda_\rho$ is the maximum of λ_ρ $(1 \le \rho \le r)$, computed from rational number reconstruction, after the last Hensel lifting step. We compared our timings with Maple's determinant computation and factorization. Maple ran out of memory for computing the determinant of heron4d.

Table 4 shows a breakdown of timings for our algorithm at the last Hensel lifting step. The matrix robotarms has a relatively smaller size (20×20), but it has larger number of terms (about 100) in each polynomial entry. We can see that BB tot is much larger for robotarms than heron4d. heron5d is a much larger matrix, and BB tot is the bottleneck.

4 COMPLEXITY ANALYSIS WITH FAILURE PROBABILITIES

First, we state the Schwartz-Zippel Lemma [26, 31]:

Lemma 4.1. *Let F be a field and $f \in F[x_1, x_2, \cdots, x_n]$ $(f \neq 0)$ with total degree d and let $S \subseteq F$. Then the number of roots of f in S^n is at most $d|S|^{n-1}$. Hence if $\boldsymbol{\beta}$ is chosen at random from S^n then* $\Pr[f(\boldsymbol{\beta}) = 0] \le \frac{d}{|S|}$.

We have the following bound of failure probability for the j$^{\text{th}}$ Hensel lifting step of Algorithm CMSHL:

Table 3: Timings (in seconds) for computing determinants of large matrices.

	heron3d	heron4d	robotarms	heron5d
n	7	11	8	16
$N \times N$	13×13	63×63	20×20	399×399
r	6	4	3	8
$\#f_i$	3,23,3, 3,1,3	22,1, 6,131	2124,4,7	823,130,22,3 3,3,3,1
e_i	1,2,1, 1,7,1	2,37, 7,4	1,4,4	8,8,20,46 46,46,1831
$\#\det(A)$	525	37666243	178053	-
$\max \lambda_\rho$	1	1	169	1
CMSHL tot	1.096	81.376	1083.335	155054.324
probes tot	8560	339840	540834	36008392
Maple det	0.614	O/M	N/A	N/A
Maple fac	0.084	O/M	N/A	N/A
Maple tot	0.698	-	-	-

Table 4: Breakdown of timings (in seconds) at H.L. x_n.

	heron3d	heron4d	robotarms	heron5d
n	7	11	8	16
$N \times N$	13×13	63×63	20×20	399×399
H.L. x_n tot	0.229	16.612	441.593	10361.995
s	13	85	806	571
BB tot	0.046	12.801	421.366	9940.302
BB eval	0.028	5.428	415.676	4809.717
BB det	0.011	6.507	7.193	5087.231
Eval $\hat{f}_{\rho,j-1}$	0.011	0.132	0.374	0.467
BHL	0.005	0.023	0.298	0.196
VSolve	0.003	0.001	0.333	0.021

Proposition 4.2. *Let p be a large prime. Let r be the number of factors of sqf(a). Let $d = \deg(a)$, $\tilde{d} = \deg(sqf(a))$, $\tilde{d}_j = \deg(sqf(a), x_j)$ and s be the number defined at step 7 in Algorithm 1. Let $\#\hat{f}_{\rho,j-1}$ denote the number of terms in the input factors $\hat{f}_{\rho,j-1}$ at the j^{th} Hensel lifting step of Algorithm CMSHL. Then, Algorithm 1 fails to compute $\hat{f}_{\rho,j} \in \mathbb{Z}_p[x_1, \cdots, x_j]$ with a probability less than*

$$\underbrace{\frac{((r^2 - r + 2)\tilde{d}^2 + 2d)s^2 + (2\tilde{d}^2 + \tilde{d}\sum_{\rho=1}^{r} \#\hat{f}_{\rho,j-1} + 2d)s}{2(p-1)}}_{\text{step } 6,11,13,17,18} + \underbrace{\frac{\tilde{d}_j^2 \sum_{\rho=1}^{r} \#\hat{f}_{\rho,j-1}}{p - \tilde{d}_j + 1}}_{\text{Lemma } 2.2}. \quad (10)$$

Proof. For step 13, let $\bar{A}_k = A_k/g_k$ and $\frac{\partial \bar{A}_k}{\partial x_1} = \frac{\partial A_k}{\partial x_1}/g_k$. Then,

$$\deg\left(\gcd\left(\bar{A}_k, \frac{\partial \bar{A}_k}{\partial x_1}\right), x_1\right) > 0 \Leftrightarrow \operatorname{res}\left(\bar{A}_k, \frac{\partial \bar{A}_k}{\partial x_1}, x_1\right) = 0.$$

Let $\bar{a}_j = a_j/g_j$ and $\frac{\partial \bar{a}_j}{\partial x_1} = \frac{\partial a_j}{\partial x_1}/g_j$ where $g_j := \gcd(a_j, \frac{\partial a_j}{\partial x_1}) \in \mathbb{Z}_p[x_1, \cdots, x_j]$. Define $R := \text{res}\left(\bar{a}_j, \frac{\partial \bar{a}_j}{\partial x_1}, x_1\right) \in \mathbb{Z}_p[x_2, \cdots, x_j]$ (for simplicity, we assume $R \neq 0$). Let $R_k := R(x_2^k, \cdots, x_{j-1}^k, x_j)$ and $S = \prod_{k=1}^{s} R_k$.

Algorithm CMSHL step j fails at step 13 if $R(Y_k, x_j) = 0$ for some k. Let $(\beta_2, \cdots, \beta_j)$ be chosen at random from $\mathbb{Z}_p^{j-1}$,

$$\begin{aligned}\Pr[R(Y_k, x_j) = 0 \text{ for some } k] &= \Pr[S(\beta_2, \cdots, \beta_{j-1}, x_j) = 0] \\ &\leq \Pr[S(\beta_2, \cdots, \beta_{j-1}, \beta_j) = 0] \\ &\leq \frac{\deg(S)}{p-1} \text{ by Lemma 4.1.}\end{aligned}$$

Now,

$$\deg(S) = \sum_{k=1}^{s} \deg(R_k) < \sum_{k=1}^{s} 2k\tilde{d}^2 = \tilde{d}^2 s(s+1).$$

Thus, CMSHL step j fails at step 13 with a probability less than

$$\frac{\tilde{d}^2 s(s+1)}{p-1}.$$

The proofs for failure probabilities at step 6, 11, 17, 18 follow from [21] and [4]. And we have the following:

$$\begin{aligned}\Pr[\text{step 6 fails at step j}] &< \frac{\tilde{d}s \sum_{\rho=1}^{r} \#\hat{f}_{\rho,j-1}}{2(p-1)}, \\ \Pr[\text{step 11 fails at step j}] &< \frac{ds(s+1)}{2(p-1)}, \\ \Pr[\text{step 17 fails at step j}] &< \frac{\tilde{d}s(s+1)}{2(p-1)}, \\ \Pr[\text{step 18 fails at step j}] &< \frac{\tilde{d}^2 s^2 r(r-1)}{2(p-1)}.\end{aligned}$$

Adding up the above, we get the first term in (10). The second term in (10) comes directly from Lemma 2.2.

□

We have the following theorem for the complexity of CMSHL:

Theorem 4.3. *Let p be a large prime and $\tilde{N} < p$, $\tilde{N} \in \mathbb{Z}^+$. Let $a \in \mathbb{Z}[x_1, \cdots, x_n]$ and $\boldsymbol{\alpha} = (\alpha_2, \cdots, \alpha_n) \in \mathbb{Z}_p^{n-1}$ be randomly chosen such that $0 < \alpha_i < \tilde{N}$. Suppose $\boldsymbol{\alpha}$ is Hilbertian and condition (i) of the input of CMSHL is satisfied. Then, with a high probability of success, the total number of arithmetic operations in $\mathbb{Z}_p$ in the worst case for lifting $\hat{f}_{\rho,1}$ to $\hat{f}_{\rho,n}$ using Algorithm CMSHL in $n-1$ steps is*

$$O\left((n-2)s_{\max}d_{\max}\left(\sum_{\rho=1}^{r} \#\hat{f}_{\rho,j-1} + d_1^2 + d_1 d_{\max} + d_1 C(\text{probe } \boldsymbol{B})\right)\right). \tag{11}$$

where $d_1 = \deg(a, x_1)$, $d_{\max} = \max_{j=2}^{n}(\deg(a, x_j))$, $s_{\max} = \max(s_j)$ where s_j is the number s defined at step 7 of Algorithm 1 and $C(\text{probe } \boldsymbol{B})$ is the cost of one probe to the black box $\boldsymbol{B}$. The total number of probes to the black box is $O(nd_1 d_{\max} s_{\max})$.

Proof. Let $d_j = \deg(a, x_j)$, $\tilde{d}_1 = \deg(\text{sqf}(a), x_1)$ and $\tilde{d}_j = \deg(\text{sqf}(a), x_j)$. For step 10, we use dense interpolations to get a bivariate image $a_j(x_1, Y_k, x_j)$. Thus, it requires $O(d_1 d_j)$ probes to $\mathbf{B}$ and $O(d_1 d_j^2 + d_1^2 d_j)$ arithmetic operations in $\mathbb{Z}_p$ for one image. The total cost for step 10 for CMHSL step j is $O(s(d_1 d_j C(\text{probe } \mathbf{B})))$ plus $O(s(d_1^2 d_j + d_1 d_j^2))$ operations in $\mathbb{Z}_p$ for dense interpolations.

For step 12, we can use Brown's GCD algorithm [3] for GCDs in $\mathbb{Z}_p[x_1, x_j]$ which costs $O(d_1^2 d_j + d_1 d_j^2)$ arithmetic operations in $\mathbb{Z}_p$. An alternative with the same asymptotic cost would be to use the bivariate Hensel lifting of Monagan and Paluck [23]. The total cost for step 12 for step j of Algorithm CMSHL is $O(s(d_1^2 d_j + d_1 d_j^2))$ operations in $\mathbb{Z}_p$.

The proofs of the following steps follow from [4]. We give the total count of arithmetic operations in $\mathbb{Z}_p$ for step j:

Step 16 costs $O(s \sum_{\rho=1}^{r} \#\hat{f}_{\rho,j-1})$.

Step 19 costs $O(s(\tilde{d}_1^2 \tilde{d}_j + \tilde{d}_1 \tilde{d}_j^2)) \subseteq O(s(d_1^2 d_j + d_1 d_j^2))$.

Step 25 costs $O(s\tilde{d}_j \sum_{\rho=1}^{r} \#\hat{f}_{\rho,j-1})$.

Adding up, we get the total number of arithmetic operations in $\mathbb{Z}_p$ for step j of Algorithm CMSHL:

$$O\left(s\left(d_1^2 d_j + d_1 d_j^2\right) + s\tilde{d}_j \sum_{\rho=1}^{r} \#\hat{f}_{\rho,j-1} + sd_1 d_j C(\text{probe } \mathbf{B})\right). \tag{12}$$

And (11) follows from (12).

□

The following theorem gives the complexity and the number of probes to the black box by Rubinfeld and Zippel's algorithm [25]:

Theorem 4.4. *Let a be a square-free polynomial over $\mathbb{Q}$ with n variables and let r be the number of factors of a. To determine the factors of a with high likelihood of success, the total number of arithmetic operations by Rubinfeld and Zippel's algorithm [25] is $O(n^2 d_{\max}^3 \#f_{\max}^3 + n^4 d_{\max}^{10} \#f_{\max})$ and the number of probes to the black box is $O(d_1 (rnd_{\max})^2 \#f_{\max})$, where $\#f_{\max} = \max_{\rho=1}^{r} \#f_\rho$.*

Rubinfeld and Zippel's algorithm [25] is the best known algorithm for Approach I. However, our algorithm CMSHL requires much less number of probes to the black box since $s_{\max} < \#f_{\max}$.

5 CONCLUSION AND FUTURE WORK

We have contributed a new black box factorization algorithm that handles the non-monic, non-square-free and non-primitive cases. Our benchmarks show that our algorithm is much faster than Maple and Magma's current best determinant and factorization algorithms. We give a complexity analysis with failure probabilities. The number of probes the black box in our new algorithm is much less than Approach I by Rubinfeld and Zippel's algorithm [25].

From our benchmarks, the bottleneck of our algorithm is probes to the black box. One way to speed this up is to use the previous evaluation points for polynomial evaluations. Also, our algorithm is highly parallelizable. We hope to design a parallel algorithm to further speed up the computation.

Acknowlegement

This work was supported by Maplesoft and by NSERC of Canada. We gratefully acknowledge the contributions of the anonymous referees.

REFERENCES

[1] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In Proceedings of STOC '88, pp. 301–309. ACM (1988)

[2] Bernardin, L.: On Bivariate Hensel Lifting and its Parallelization. In Proceedings of ISSAC '98, pp. 96–100. ACM (1998)

[3] W. S. Brown. On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors. *J. ACM* **18**:478–504, 1971.

[4] Chen, T., Monagan, M.: The complexity and parallel implementation of two sparse multivariate Hensel lifting algorithms for polynomial factorization. In Proceedings of CASC 2020, LNCS **12291**, 150–169. Springer (2020)

[5] Chen, T., Monagan, M.: Factoring multivariate polynomials represented by black boxes: A Maple + C Implementation. *Math. Comput. Sci.* **16**,18 (2022)

[6] Cohen, S.D.: The distribution of Galois groups and Hilbert's irreducibility theorem. In Proceedings of the London Mathematical Society (3), 43:227–250 (1981)

[7] Von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge University Press (2013)

[8] Geddes, K.O., Czapor, S.R. and Labahn, G.: Algorithms for Computer Algebra. Kluwer Acad. Publ (1992)

[9] David Hilbert. Über die Irreducibilität ganzer rational Functionen mit ganzzahigen Koeffizienten. *J. Renie Agewandte Mathematik*, **110**, 104–129 (1982)

[10] Gentleman, W.M. and Johnson, S.C.: Analysis of algorithms, a case study: determinants of matrices with polynomial entries. *ACM Trans. on Math. Soft.* **2**(3), 232–241. ACM (1976)

[11] Kaltofen, E.: Sparse Hensel lifting. In Proceedings of EUROCAL '85, LNCS **204**, 4–17. Springer (1985)

[12] Khovanova, T., Scully, Z.: Efficient Calculation of Determinants of Symbolic Matrices with Many Variables, ArXiv: 1304.4691 (2013)

[13] Kaltofen E., Trager, B.M.: Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symb. Cmpt.* **9**(3), 301–320. Elsevier (1990)

[14] Kaltofen, E., Yagati, L.: Improved sparse multivariate polynomial interpolation algorithms. In Proceedings of ISSAC '88, LNCS **358**, 467–474. Springer (1988)

[15] Lang, S.: Fundamentals of Diophantine Geometry. Springer-Verlag New York (1983)

[16] Lee, W.S.: Early termination strategies in sparse interpolation algorithms. Ph.D. Thesis (2001)

[17] Monagan, M.: Linear Hensel lifting for $\mathbb{F}_p[x, y]$ and $\mathbb{Z}[x]$ with cubic cost. In Proceedings of ISSAC 2019, pp. 299–306. ACM (2019)

[18] Monagan, M., Tuncer, B.: Using Sparse Interpolation in Hensel Lifting. In Proceedings of CASC 2016, LNCS **9890**, 381–400. Springer (2016)

[19] Monagan, M., Tuncer, B.: Factoring multivariate polynomials with many factors and huge coefficients. In Proceedings of CASC 2018, LNCS **11077**, 319–334. Springer (2018)

[20] Monagan, M., Tuncer, B.: Sparse multivariate Hensel lifting: a high-performance design and implementation. In Proceedings of ICMS 2018, LNCS **10931**, 359–368. Springer (2018)

[21] Monagan, M., Tuncer, B.: The complexity of sparse Hensel lifting and sparse polynomial factorization. *J. Symb. Cmpt.* **99**, 189–230. Elsevier (2020)

[22] Monagan, M., Tuncer, B.: Polynomial factorization in Maple 2019. In Maple in Mathematics Education and Research. *Communications in Computer and Information Science*, **1125**, 341–345. Springer (2020)

[23] Monagan, M., Paluck, G.,: Linear Hensel lifting for $Zp[x, y]$ for n factors with cubic cost. In Proceedings of ISSAC 2022, 169-166. ACM (2022)

[24] Roche, D.: What can (and can't) we do with sparse polynomials? In Proceedings of ISSAC 2018, pp. 25–30. ACM (2018)

[25] Rubinfeld, R., Zippel, R.E.: A new modular interpolation algorithm for factoring multivariate polynomials. In Proceedings of Algorithmic Number Theory, First International Symposium, ANTS-I (1994)

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

[27] Wang, P.S., Rothschild, L.P.: Factoring multivariate polynomials over the integers. *Math. Comp.* **29**, 935–950 (1975)

[28] Wang, P.S.: An improved multivariate polynomial factoring algorithm. *Math. Comp.* **32**, 1215–1231 (1978)

[29] Wang, P.S., Guy, M.J.T., Davenport, J.H.: p-adic reconstruction of rational numbers. SIGSAM Bulletin, **16**(2) (1982)

[30] Yun, D.Y.Y.: The Hensel Lemma in algebraic manipulation. Ph.D. Thesis (1974)

[31] Zippel, R.E.: Probabilistic algorithms for sparse polynomials. LNCS **72**, 216–226 (1979)

[32] Zippel, R.E.: Newton's iteration and the sparse Hensel algorithm. In Proceedings of the ACM Symposium on Symbolic Algebraic Computation, pp. 68–72 (1981)

[33] Zippel, R.E.: Interpolating polynomials from their values. *J. Symb. Cmpt.* **9**(3), 375–403 (1990)