Factoring Non-monic Polynomials Represented by Black Boxes

Tian Chen and Michael Monagan *

Department of Mathematics, Simon Fraser University,
Burnaby, British Columbia, V5A 1S6, CANADA
tca71@sfu.ca, mmonagan@sfu.ca

Abstract. We aim to factor a sparse polynomial $a \in \mathbb{Z}[x_1, \cdots, x_n]$ represented by a black box. The authors have previously developed efficient sparse Hensel lifting algorithms for the monic and square-free case that outperforms the algorithm by Kaltofen and Trager in 1990. We complete this black box factorization problem for the non-monic case with a new algorithm that computes the factors of a using many non-monic bivariate Hensel lifts. Our algorithm handles all cases of input $a \in \mathbb{Z}[x_1, \cdots, x_n]$ including the non-square-free and the non-primitive cases. We have implemented the algorithm in Maple with all major subroutines coded in C for efficiency.

Keywords: Multivariate Polynomial Factorization, Black Box Representation, Sparse Hensel Lifting, Bivariate Hensel Lifting

1 Introduction

The black box representation of a polynomial $f \in \mathbb{Z}[x_1, ..., x_n]$ is a program which accepts a prime p and an evaluation point $\boldsymbol{\alpha} \in \mathbb{Z}_p^n$ and outputs $f(\boldsymbol{\alpha}) \bmod p$. It is one of the most space efficient implicit representations [7]. On the contrary, the sparse representation of f is explicit. It consists of a list of coefficients $c_k \neq 0$ 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 [4]).

Given a 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. Figure 1 shows three ways to compute them. Method 0 first interpolates the sparse representation of a and then factors it using a sparse Hensel lifting algorithm, e.g. algorithm CMSHL [1]. Method I is Kaltofen and Trager's method [7] which first constructs black boxes for the factors then applies sparse polynomial interpolation to them. Method II contributed by the authors in [2] computes the factors in the sparse representation directly by a modified algorithm CMSHL and it works for the monic and square-free case. Method II is the most efficient of the three and it outperforms Kaltofen and Trager's algorithm as it requires less number of probes to the black box $\mathbf{B}$ [2].

* We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). [funding reference number RGPIN-2019-04441].

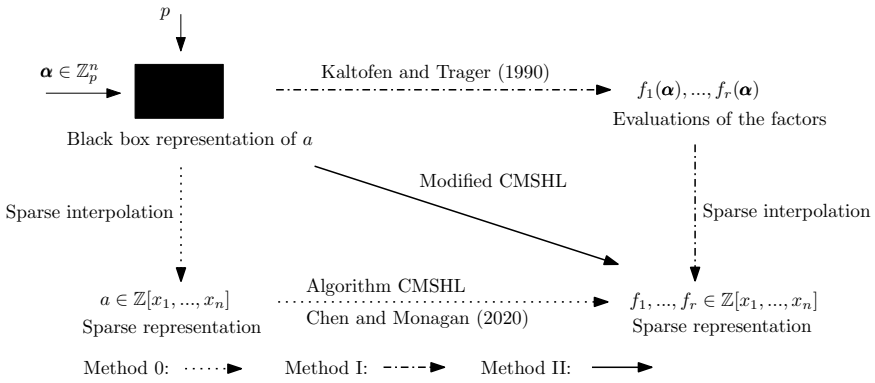

Fig. 1. Factoring $a \in \mathbb{Z}[x_1, ..., x_n]$ represented by a black box.

In this work, we complete the black box factorization problem with a new algorithm that handles the non-monic input a. Our new algorithm accepts all cases of input $a \in \mathbb{Z}[x_1, \cdots, x_n]$ including the non-square-free and the non-primitive cases.

If the input polynomial a is in the sparse representation and is non-monic, there are mainly two methods to pre-compute the leading coefficients of the factors. One is Wang's *leading coefficient correction* [9], and the other is by Kaltofen [6]. However, in our case, we do not need to pre-compute the coefficients of the factors. We only scale the leading coefficients in the bivariate Hensel lifts in algorithm CMSHL (step 16 of Algorithm 1).

For our new algorithm, the following steps are performed (Method II). First, an evaluation point $\boldsymbol{\alpha} = (\alpha_2, \cdots, \alpha_n)$ is chosen and $a(x_1, \boldsymbol{\alpha})$ is factored over $\mathbb{Z}$. By Hilbert's irreducibility theorem [5], the pattern of the irreducible factors remains the same with high probability. To be precise, let $a = h f_1^{e_1} f_2^{e_2} \cdots f_r^{e_r}$ be the irreducible factorization of $a \in \mathbb{Z}[x_1, \cdots, x_n]$ over $\mathbb{Z}$, where $h \in \mathbb{Z}[x_2, \cdots, x_n]$ is the content of a in x_1. Then, with high probability,

$$a(x_1, \boldsymbol{\alpha}) = \hat{h} \hat{f}_1^{\,e_1} \hat{f}_2^{\,e_2} \cdots \hat{f}_r^{\,e_r} \in \mathbb{Z}[x_1], \tag{1}$$

where $f_\rho(x_1, \boldsymbol{\alpha}) = \lambda_\rho \hat{f}_\rho$ for some constant $\lambda_\rho \in \mathbb{Z}$, for $1 \le \rho \le r$, and $\hat{h} = \lambda_h h(\boldsymbol{\alpha})$ for some $\lambda_h \in \mathbb{Z}$.

Let $a_j := a(x_1, \cdots, x_j, \alpha_{j+1}, \cdots, \alpha_n) \bmod p$. Let $\hat{f}_{\rho,1} := \hat{f}_\rho(x_1) \bmod p$ from (1) and let $\hat{f}_{\rho,j}$ denote $\hat{f}_\rho(x_1, \cdots, x_j, \alpha_{j+1}, \cdots, \alpha_n)$ for $2 \le j \le n$ (to be computed). The input to the algorithm CMSHL for the non-monic and non-square-free case is a prime p, the black box $\mathbf{B}$, $\hat{f}_{\rho,1}$ for $1 \le \rho \le r$ and $\boldsymbol{\alpha}$ such that $\gcd(\hat{f}_{k,1}, \hat{f}_{l,1}) = 1$ for $1 \le k, l \le r$ in $\mathbb{Z}_p[x_1]$. Algorithm CMSHL lifts $\hat{f}_{\rho,1}$ to $\hat{f}_{\rho,2}$ then lifts $\hat{f}_{\rho,2}$ to $\hat{f}_{\rho,3}$ etc. At the j^{th} Hensel lifting step, $\text{sqf}(a_j) = \prod \lambda_\rho \prod \hat{f}_{\rho,j} \bmod p$ and $\hat{f}_{\rho,j}(x_j = \alpha_j) = \hat{f}_{\rho,j-1} \bmod p$. At the end, $\text{sqf}(a_n) = \prod \lambda_\rho \prod \hat{f}_{\rho,n} \bmod p$. Here, $\text{sqf}(a)$ denotes the square-free part of the polynomial a.

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 correct coefficients of the

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

1: **Input:** A prime p, $\alpha_\mathsf{j} \in \mathbb{Z}_\mathsf{p}$, the black box $\mathbf{B}$, $\hat{\mathsf{f}}_{\rho,\mathsf{j}-1} \in \mathbb{Z}_\mathsf{p}[\mathsf{x}_1, \cdots, \mathsf{x}_{\mathsf{j}-1}]$ for $1 \le \rho \le \mathsf{r}$ s.t. $\mathrm{sqf}(\mathsf{a}_\mathsf{j}(\mathsf{x}_\mathsf{j} = \alpha_\mathsf{j})) = \prod_{\rho=1}^{\mathsf{r}} \lambda_\rho \prod_{\rho=1}^{\mathsf{r}} \hat{\mathsf{f}}_{\rho,\mathsf{j}-1}$ with $\mathsf{j} > 2$, $\mathsf{d}_\mathsf{i} (= \deg(\mathsf{a}, \mathsf{x}_\mathsf{i}))$ for $1 \le \mathsf{i} \le \mathsf{n}$.
2: **Output:** $\hat{\mathsf{f}}_{\rho,\mathsf{j}} \in \mathbb{Z}_\mathsf{p}[\mathsf{x}_1, \cdots, \mathsf{x}_\mathsf{j}]$ for $1 \le \rho \le \mathsf{r}$ s.t. $\mathrm{sqf}(\mathsf{a}_\mathsf{j}) = \prod_{\rho=1}^{\mathsf{r}} \lambda_\rho \prod_{\rho=1}^{\mathsf{r}} \hat{\mathsf{f}}_{\rho,\mathsf{j}}$ where $\hat{\mathsf{f}}_{\rho,\mathsf{j}}(\mathsf{x}_\mathsf{j} = \alpha_\mathsf{j}) = \hat{\mathsf{f}}_{\rho,\mathsf{j}-1}$ for $1 \le \rho \le \mathsf{r}$; Otherwise, **return** FAIL.
3: Let $\hat{\mathsf{f}}_{\rho,\mathsf{j}-1} = \sum_{\mathsf{i}=0}^{\mathsf{df}_\rho} \sigma_{\rho,\mathsf{i}}(\mathsf{x}_2, ..., \mathsf{x}_{\mathsf{j}-1}) \mathsf{x}_1^\mathsf{i}$ where $\sigma_{\rho,\mathsf{i}} = \sum_{\mathsf{k}=1}^{\mathsf{s}_{\rho,\mathsf{i}}} \mathsf{c}_{\rho,\mathsf{ik}} \mathsf{M}_{\rho,\mathsf{ik}}$, $\mathsf{M}_{\rho,\mathsf{ik}}$ are the monomials in $\sigma_{\rho,\mathsf{i}}$, and $\mathsf{df}_\rho = \deg(\hat{\mathsf{f}}_{\rho,\mathsf{j}-1}, \mathsf{x}_1)$ for $1 \le \rho \le \mathsf{r}$.
4: Pick $\boldsymbol{\beta} = (\beta_2, \cdots, \beta_{\mathsf{j}-1}) \in \mathbb{Z}_\mathsf{p}^{\mathsf{j}-2}$ at random.
5: Evaluate: $\{\mathcal{S}_\rho = \{\mathsf{S}_{\rho,\mathsf{i}} = \{\mathsf{m}_{\rho,\mathsf{ik}} = \mathsf{M}_{\rho,\mathsf{ik}}(\boldsymbol{\beta}), 1 \le \mathsf{k} \le \mathsf{s}_{\rho,\mathsf{i}}\}, 0 \le \mathsf{i} \le \mathsf{df}_\rho\}, 1 \le \rho \le \mathsf{r}\}$.
6: **if** any $|\mathsf{S}_{\rho,\mathsf{i}}| \ne \mathsf{s}_{\rho,\mathsf{i}}$ **then return** FAIL **end if**
7: Let s be the maximum of $\mathsf{s}_{\rho,\mathsf{i}}$.
8: **for** k from 1 to s **do**
9: Let $\mathsf{Y}_\mathsf{k} = (\mathsf{x}_2 = \beta_2^\mathsf{k}, \cdots, \mathsf{x}_{\mathsf{j}-1} = \beta_{\mathsf{j}-1}^\mathsf{k})$.
10: $\mathsf{A}_\mathsf{k} \leftarrow \mathsf{a}_\mathsf{j}(\mathsf{x}_1, \mathsf{Y}_\mathsf{k}, \mathsf{x}_\mathsf{j})$. // via probes to $\mathbf{B}$ and interpolation . $\mathcal{O}(\mathsf{sd}_1\mathsf{d}_\mathsf{j} \cdot \mathrm{C}(\text{probe } \mathbf{B}))$
11: $\mathsf{g}_\mathsf{k} \leftarrow \gcd(\mathsf{A}_\mathsf{k}, \frac{\partial \mathsf{A}_\mathsf{k}}{\partial \mathsf{x}_1}) \bmod \mathsf{p}$. **if** $\deg(\mathsf{g}_\mathsf{k}, \mathsf{x}_1) \ne \mathsf{d}_1 - \sum \mathsf{df}_\rho$ **then return** FAIL **end if**
12: $\mathsf{A}_\mathsf{k} \leftarrow \mathrm{quo}(\mathsf{A}_\mathsf{k}, \mathsf{g}_\mathsf{k}) \bmod \mathsf{p}$. // get $\mathrm{sqf}(\mathsf{A}_\mathsf{k}) \bmod \mathsf{p}$, no content in x_1.
13: $\mathsf{F}_{\rho,\mathsf{k}} \leftarrow \hat{\mathsf{f}}_{\rho,\mathsf{j}-1}(\mathsf{x}_1, \mathsf{Y}_\mathsf{k})$ for $1 \le \rho \le \mathsf{r}$. . . . . $\mathcal{O}(\mathsf{s}(\#\mathsf{f}_1 + \cdots + \#\mathsf{f}_\mathsf{r}))$
14: **if** any $\deg(\mathsf{F}_{\rho,\mathsf{k}}) < \mathsf{df}_\rho$ for $1 \le \rho \le \mathsf{r}$ **then return** FAIL **end if**
15: **if** $\gcd(\mathsf{F}_{\rho,\mathsf{k}}, \mathsf{F}_{\phi,\mathsf{k}}) \ne 1$ for any $\rho \ne \phi$ $(1 \le \rho, \phi \le \mathsf{r})$ **then return** FAIL **end if**
16: $\hat{\mathsf{f}}_{\rho,\mathsf{k}} \leftarrow \mathrm{BivariateHenselLift}(\mathsf{A}_\mathsf{k}(\mathsf{x}_1, \mathsf{x}_\mathsf{j}), \mathsf{F}_{\rho,\mathsf{k}}(\mathsf{x}_1), \alpha_\mathsf{j}, \mathsf{p})$. . . . . $\mathcal{O}(\mathsf{sr}(\mathsf{d}_1\mathsf{d}_\mathsf{j}^2 + \mathsf{d}_1^2\mathsf{d}_\mathsf{j}))$
17: **end for**
18: Let $\hat{\mathsf{f}}_{\rho,\mathsf{k}} = \sum_{\mathsf{l}=1}^{\mathsf{t}_\rho} \alpha_{\rho,\mathsf{kl}} \tilde{\mathsf{M}}_{\rho,\mathsf{l}}(\mathsf{x}_1, \mathsf{x}_\mathsf{j})$ for $1 \le \mathsf{k} \le \mathsf{s}$ where $\mathsf{t}_\rho = \#\hat{\mathsf{f}}_{\rho,\mathsf{k}}$, for $1 \le \rho \le \mathsf{r}$.
19: **for** ρ from 1 to r **do**
20: **for** l from 1 to t_ρ **do**
21: $\mathsf{i} \leftarrow \deg(\tilde{\mathsf{M}}_{\rho,\mathsf{l}}, \mathsf{x}_1)$.
22: Solve the linear system for $\mathsf{c}_{\rho,\mathsf{lk}}$: $\{\sum_{\mathsf{k}=1}^{\mathsf{s}_{\rho,\mathsf{i}}} \mathsf{m}_{\rho,\mathsf{ik}}^\mathsf{n} \mathsf{c}_{\rho,\mathsf{lk}} = \alpha_{\rho,\mathsf{nl}} \text{ for } 1 \le \mathsf{n} \le \mathsf{s}_{\rho,\mathsf{i}}\}$.
23: **end for** . . . . . $\mathcal{O}(\mathsf{sd}_\mathsf{j}(\#\mathsf{f}_1 + \cdots + \#\mathsf{f}_\mathsf{r}))$
24: Construct $\hat{\mathsf{f}}_{\rho,\mathsf{j}} \leftarrow \sum_{\mathsf{l}=1}^{\mathsf{t}_\rho} \left(\sum_{\mathsf{k}=1}^{\mathsf{s}_{\rho,\mathsf{i}}} \mathsf{c}_{\rho,\mathsf{lk}} \mathsf{M}_{\rho,\mathsf{ik}}(\mathsf{x}_2, ..., \mathsf{x}_{\mathsf{j}-1})\right) \tilde{\mathsf{M}}_{\rho,\mathsf{l}}(\mathsf{x}_1, \mathsf{x}_\mathsf{j})$.
25: **end for**
26: Pick $\boldsymbol{\beta} = (\beta_2, \cdots, \beta_\mathsf{j}) \in \mathbb{Z}_\mathsf{p}^{\mathsf{j}-1}$ at random.
27: $\mathsf{A}_{\boldsymbol{\beta}} \leftarrow \mathrm{sqf}(\mathsf{a}_\mathsf{j}(\mathsf{x}_1, \boldsymbol{\beta})) \bmod \mathsf{p}$ // via probes to $\mathbf{B}$, interpolation, and sqrfree compt.
28: **if** $\hat{\mathsf{f}}_{\rho,\mathsf{j}}(\mathsf{x}_1, \boldsymbol{\beta}) \mid \mathsf{A}_{\boldsymbol{\beta}}$ **and** $\deg(\hat{\mathsf{f}}_{\rho,\mathsf{j}}(\mathsf{x}_1, \boldsymbol{\beta})) = \mathsf{df}_\rho$ for $1 \le \rho \le \mathsf{r}$ **then return** $\hat{\mathsf{f}}_{\rho,\mathsf{j}}$ **else return** FAIL **end if**

factors f_ρ in $\mathbb{Z}$. At the end, the content of a in x_1 can be recovered by sparse interpolation (with one less variable) if needed.

2 Algorithm CMSHL: non-monic and non-square free

The j^{th} Hensel lifting step of algorithm CMSHL for the non-monic and non-square-free case is shown in Algorithm 1. There are 4 major steps which are similar to the monic and square-free case in [2], namely (1) the probes to the black box to interpolate the bivariate images $A_k(x_1, x_j)$ of a in step 10, (2) evaluating the factors $\hat{f}_{\rho,j-1}$ for $1 \le \rho \le r$ in step 13, (3) the non-monic bivariate Hensel lifts in step 16, and (4) solving the Vandermonde systems in step 22. The

number of arithmetic operations in $\mathbb{Z}_p$ for these steps is shown in blue. $\#f$ denotes the number of terms in f.

The key is to use the square-free part of the bivariate images of a to do all bivariate Hensel lifts. Step 11–12 computes the square-free part of $A_k(x_1, x_j)$ probabilistically. It also removes the content of $A_k(x_1, x_j)$ in x_1. We yet need to do a complexity analysis for the algorithm with failure probabilities. Another change is that we have modified the algorithm in [8] to make the bivariate Hensel lifts work for the non-monic case (the pseudo-code is omitted in this abstract). After each bivariate Hensel lift, the leading coefficients are also scaled to match the input factors $\hat{f}_{\rho,j-1}(x_1, Y_k)$.

3 Implementation

We have made a hybrid Maple + C implementation for our new algorithm (Method II). In order to get the best performance, we have coded each of the four major steps described in the previous section in C. In particular, for the bivariate Hensel lifts in step 16, we use the new cubic algorithm of Monagan and Paluck [8] that costs $O(d_1 d_j^2 + d_1^2 d_j)$ arithmetic operations in $\mathbb{Z}_p$.

We are currently testing examples to compute the factors of the determinant of matrices with multivariate polynomial entries. The timing benchmarks for the monic case in [2] show that our algorithm is much faster than Maple and Magma's current best determinant and factorization algorithms. We are working on timings for the non-monic, the non-square-free, and the non-primitive cases.

References

1. 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)
2. Chen, T., Monagan, M.: Factoring multivariate polynomials represented by black boxes – A Maple + C implementation. To appear in CASC 2021 Post Conference Proceedings, Mathematics in Computer Science (2022).
3. Geddes, K.O., Czapor, S.R. and Labahn, G.: Algorithms for Computer Algebra. Kluwer Acad. Publ (1992)
4. Von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge University Press (2013)
5. Hilbert, D.: Über die Irreducibilität ganzer rational Functionen mit ganzzahigen Koeffizienten. *J. Renie Agewandte Mathematik*, **110**, 104–129 (1982)
6. Kaltofen, E.: Sparse Hensel lifting. In Proceedings of EUROCAL '85, LNCS **204**, 4–7. Springer (1985)
7. 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)
8. Paluck, G., Monagan, M.: New bivariate Hensel lifting algorithm for n factors. *ACM Communications in Computer Algebra*, **53**(3), 142–145 (2019)
9. Wang, P.S.: An improved multivariate polynomial factoring algorithm. *Math. Comp.* **32**, 1215–1231 (1978)