Parallel Algorithms for Factoring Multivariate 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`

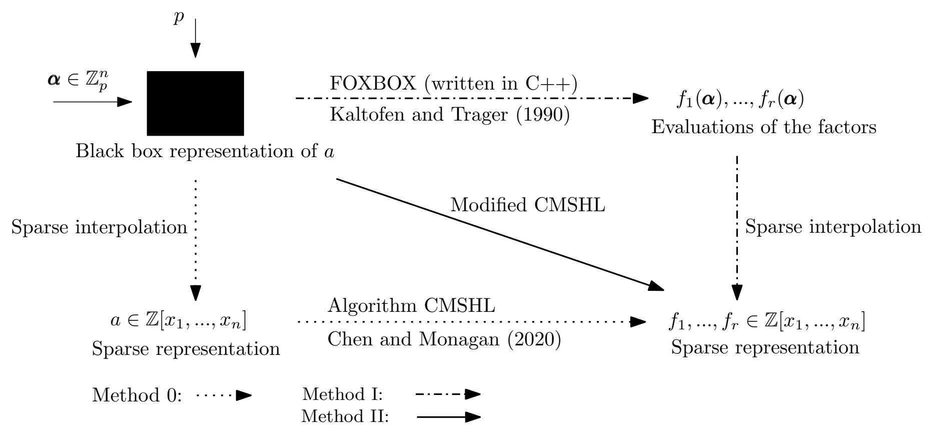
Abstract. Our goal is to develop fast parallel algorithms to factor multivariate polynomials with integer coefficients. The authors recently contributed a parallel sparse Hensel lifting algorithm (CMSHL) to factor multivariate polynomials in their sparse representations (Chen and Monagan (2020)). The dominating cost of CMSHL is polynomial evaluations. To reduce this cost, in this work, we represent the polynomial to be factored by a black box. Instead of first getting the evaluations of the factors and then performing a sparse interpolation (Kaltofen and Trager (1990)), we give a parallel algorithm that computes the factors in their sparse representation directly using our modified CMSHL algorithm. Our new algorithm requires fewer probes to the black box. We show the complexity for both algorithms. We have implemented our algorithm in Maple and are currently implementing parts of it in C.

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

1 Two algorithms for black box factorization

Polynomial factorization has been a central topic in computer algebra. It has applications in diverse fields such as algebraic coding theory, cryptography, number theory, algebraic geometry and biological modelling [3]. Our work focuses on designing and implementing parallel algorithms to factor multivariate polynomials with integer coefficients. In 2020, Chen and Monagan [1] developed a parallel sparse Hensel lifting algorithm (CMSHL) to factor multivariate polynomials input in the *sparse representation.* The dominating cost of CMSHL is polynomial evaluations. To reduce this cost, we represent the polynomial to be factored by a *black box* and aim to compute its factors in their *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 cost of evaluating a as well as the memory space needed to store a in its sparse representation.

The *black box representation* of a polynomial $f \in \mathbb{Z}[x_1, \cdots, x_n]$ is a program which accepts a prime p and an evaluation point $\boldsymbol{\alpha} \in \mathbb{Z}_p^n$ as inputs and outputs $f(\boldsymbol{\alpha}) \bmod p$. It is one of the most space efficient implicit representations [5]. On the other hand, the *sparse representation* of f is explicit. It consists of a list of coefficients a_k and exponents $(e_{k_1}, \cdots, e_{k_n})$ such that $f = \sum_{k=1}^{t} a_k \cdot x_1^{e_{k_1}} \cdots x_n^{e_{k_n}}$, where $a_k \neq 0$ and t is the number of non-zero terms of f (Chap. 16 of [4]).



p
$\boldsymbol{\alpha} \in \mathbb{Z}_p^n$
FOXBOX (written in C++)
Kaltofen and Trager (1990)
$f_1(\boldsymbol{\alpha}), ..., f_r(\boldsymbol{\alpha})$
Evaluations of the factors
Black box representation of a
Modified CMSHL
Sparse interpolation
Sparse interpolation
Algorithm CMSHL
$a \in \mathbb{Z}[x_1, ..., x_n]$
Sparse representation
Chen and Monagan (2020)
$f_1, ..., f_r \in \mathbb{Z}[x_1, ..., x_n]$
Sparse representation
Method 0:
Method I:
Method II:

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

Given $a \in \mathbb{Z}[x_1, \cdots, x_n]$ represented by a black box $\mathbf{B}$, three ways to compute its factors in the sparse representation are shown in Figure 1. Method 0 first interpolates the sparse representation of a then factors it using a sparse Hensel lifting algorithm [1]. Method I adapts Kaltofen and Trager's algorithm [5] to first get black boxes for the factors and then performs sparse interpolation. This algorithm was implemented in FOXBOX [2] in C++. As far as we know, there has not been any black box factorization algorithm developed since Kaltofen and Trager [5] in 1990. We contribute Method II a parallel algorithm which uses a modified CMSHL to output the factors in their sparse representations directly. We have modified algorithm CMSHL in [1] to replace the evaluations of a with probes to the black box and have extended it to multi-factors (still monic). The j^{th} Hensel lifting step is shown in Algorithm 1.

2 Complexity analysis for both algorithms

To analyze black box algorithms we are mostly interested in the number of probes to the black box. The following estimates show that our new algorithm (Method II) requires fewer probes to the black box than Method I since $s \ll \#f_{\max}$ [1]. The quantity s is the number of bivariate images needed in algorithm CMSHL which is $\max \#a_{i,j,k}$ where the $f_i = \sum_{j,k} a_{i,j,k}(x_3, \ldots, x_n) x_1^j x_2^k$ are the factors of a and $\#f_{\max}$ is the maximum number of terms of the factors of a.

– Method I: $O(nd_1 d_{\max} \#f_{\max})$ probes to the black box $\mathbf{B}$, plus $O(nd_{\max} \#f_{\max})$ times the cost of univariate polynomial factorization.
– Method II: $O(nd_1 d_{\max} s)$ probes to the black box $\mathbf{B}$.

In the above, n is the number of variables of a, $d_1 = \deg(a, x_1)$, $d_{\max} = \max_{1\leq j\leq n}(d_j)$. For Method I, we use Zippel's sparse interpolation [7]. Also, we use integer substitutions for each variable $x_2, \cdots, x_n$ and factor univariate polynomials instead of bivariate polynomials in [5]. Since our polynomials have coefficients in the ring of integers, by virtue of the Hilbert irreducibility theorem, the resulting univariate polynomial has the same number of factors as a with high probability. For Method II, Step 10 of Algorithm 1 is the only step to probe the black box $\mathbf{B}$ hence the total number of probes is $O(nd_1 d_{\max} s)$. There are several places where Algorithm 1 can fail (Step 6, 12 and 24) and we still need to quantify the failure probabilities. The failure probability analysis will be similar to the analysis of CMSHL in [1] except Algorithm 1 has been extended to multi-factors.

As an example consider factoring the determinant of T_n the $n\times n$ symmetric Toeplitz matrix shown below. The table shows the number of terms of $\det T_n$, it's two factors, and the parameter s. The data shows that the number of terms of $\det T_n$ is much larger than the largest factor which justifies the black box approach. Also s is significantly smaller than $\#f_{\max}$.

$$\begin{pmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_2 & x_1 & x_2 & & \\ x_3 & x_2 & x_1 & & \\ \vdots & & & \ddots & \vdots \\ x_n & & & \cdots & x_1 \end{pmatrix}$$

Symmetric Toeplitz matrix T_n.

n	$\#\det(T_n)$	$\#f_i$	s
7	427	$30, 56$	8
8	1628	$167, 167$	38
9	6090	$294, 153$	50
10	23797	$931, 931$	229
11	90296	$1730, 849$	337
12	350726	$5579, 5579$	1465
13	1338076	$10611, 4983$	2297
14	5165957	$34937, 34937$	9705

3 Implementation

We aim to make a hybrid Maple + C implementation for our new algorithm (Method II). The major subroutines of Algorithm 1 are Step 10 (probes to the black box), Step 13 (bivariate Hensel lift), and Step 19 (Vandermonde solve). The main program has already been constructed (in Maple) with the black box subroutine computed in C (the determinant is computed in $\mathbb{Z}_p$ via Gaussian elimination). Vandermonde solve has already been successfully linked with the C program. The subroutine Bivariate Hensel lift is currently coded in Maple. We are integrating Garrett Paluck's C implementation of BHL into our software.

For Method I, we need to input the evaluations of the factors into a sparse interpolation procedure. The sparse interpolation procedure has already been implemented in Maple for a single polynomial represented by a black box. It is not difficult to extend it to the multi-factor case. One of the issues that we are still seeking an efficient solution is how to order the factors with identical degree patterns. We plan to implement Method I in Maple, and to compare the number of probes to the black box with Method II.

Algorithm 1 CMSHL for black box: Hensel lifting x_j (multi-factors).

```
1: Input: A prime $p$, $\alpha_j \in \mathbb{Z}_p$, $\mathbf{B}$ (black box representation of $a \in \mathbb{Z}[x_1, \cdots, x_n]$ monic
   in $x_1$), $f_{1,j-1}, \cdots, f_{r,j-1} \in \mathbb{Z}_p[x_1, \cdots, x_{j-1}]$ s.t. $a_j(x_j = \alpha_j) = \prod_{\rho=1}^{r} f_{\rho,j-1}$ with $j > 2$.
2: Output: $f_{1,j}, \cdots, f_{r,j} \in \mathbb{Z}_p[x_1, \cdots, x_j]$ s.t. $a_j = \prod_{\rho=1}^{r} f_{\rho,j}$ where $f_{\rho,j}(x_j = \alpha_j) = f_{\rho,j-1}$ for
   $1 \leq \rho \leq r$; Otherwise, return FAIL.
3: Let $f_{\rho,j-1} = x_1^{df_\rho} + \sum_{i=0}^{df_\rho - 1} \sigma_{\rho,i}(x_2, ..., x_{j-1}) x_1^i$ where $\sigma_{\rho,i} = \sum_{k=1}^{s_{\rho,i}} c_{\rho,ik} M_{\rho,ik}$ and $M_{\rho,ik}$
   are the monomials in $\sigma_{\rho,i}$ for $1 \leq \rho \leq r$.
4: Pick $\boldsymbol{\beta} = (\beta_2, \cdots, \beta_{j-1}) \in \mathbb{Z}_p^{j-2}$ at random.
5: Evaluate: $\{\mathcal{S}_\rho = \{S_{\rho,i} = \{m_{\rho,ik} = M_{\rho,ik}(\boldsymbol{\beta}), 1 \leq k \leq s_{\rho,i}\}, 0 \leq i \leq df_\rho - 1\}, 1 \leq \rho \leq r\}$.
6: if any $|S_{\rho,i}| \neq s_{\rho,i}$ then return FAIL end if
7: Let $s$ be the maximum of $s_{\rho,i}$.
8: for $k$ from 1 to $s$ in parallel do
9:    Let $Y_k = (x_2 = \beta_2^k, \cdots, x_{j-1} = \beta_{j-1}^k)$.
10:   $A_k \leftarrow a_j(x_1, Y_k, x_j)$. // via probes to $\mathbf{B}$ and interpolation . $\mathcal{O}(sd_1d_j \cdot C(\text{probe } \mathbf{B}))$
11:   $F_{1,k}, \cdots, F_{r,k} \leftarrow f_{1,j-1}(x_1, Y_k), \cdots, f_{r,j-1}(x_1, Y_k)$. . . . . . . . . . $\mathcal{O}(s(\#f_1 + \cdots + \#f_r))$
12:   if $\gcd(F_{\rho,k}, F_{\phi,k}) \neq 1$ for any $\rho \neq \phi$ $(1 \leq \rho, \phi \leq r)$ then return FAIL end if
13:   $f_{1,k}, \cdots f_{r,k} \leftarrow$ BivariateHenselLift$(A_k, F_{1,k}, \cdots, F_{r,k}, \alpha_j, p)$. . . . . . $\mathcal{O}(d_1 d_j^2 + d_1^2 d_j)$
14: end for
15: Let $f_{\rho,k} = x_1^{df_\rho} + \sum_{l=1}^{\mu_\rho} \alpha_{\rho,kl} \tilde{M}_{\rho,l}(x_1, x_j)$ for $1 \leq k \leq s$ where $\mu_\rho \leq d_1 d_j$ for $1 \leq \rho \leq r$.
16: for $\rho$ from 1 to $r$ do
17:   for $l$ from 1 to $\mu_\rho$ in parallel do
18:     $i \leftarrow \deg(\tilde{M}_{\rho,l}, x_1)$.
19:     Solve the linear system for $c_{\rho,lk}$: $\left\{\sum_{k=1}^{s_{\rho,i}} m_{\rho,ik}^n c_{\rho,lk} = \alpha_{\rho,nl} \text{ for } 1 \leq n \leq s_{\rho,i}\right\}$.
20:   end for . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . $\mathcal{O}(sd_j(\#f_1 + \cdots + \#f_r))$
21:   Construct $f_{\rho,j} \leftarrow x_1^{df_\rho} + \sum_{l=1}^{\mu_\rho} \left(\sum_{k=1}^{s_{\rho,i}} c_{\rho,lk} M_{\rho,ik}(x_2, ..., x_{j-1})\right) \tilde{M}_{\rho,l}(x_1, x_j)$.
22: end for
23: Pick $\boldsymbol{\beta} \in \mathbb{Z}_p^j$ at random.
24: if $\mathbf{B}(\boldsymbol{\beta}, \alpha_{j+1}, \cdots, \alpha_n) = \prod_{\rho=1}^{r} f_{\rho,j}(\boldsymbol{\beta})$ then return $f_{1,j} \cdots, f_{r,j}$ else return FAIL
    end if
```

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. Diaz, A., Kaltofen E.: FOXBOX: A system for manipulating symbolic objects in black box representation. In Proceedings of ISSAC '98, pp. 30–37. ACM (1998)
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. 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)
6. Wang, P.S.: An improved multivariate polynomial factoring algorithm. *Math. Comp.* **32**, 1215–1231 (1978)
7. Zippel, R.E.: Interpolating polynomials from their values. J. Symb. Comput. **9**(3), 375–403 (1990)