SCIENCE CHINA
Mathematics

• ARTICLES •

October 2014 Vol. 57 No. 10: 2123–2142
doi: 10.1007/s11425-014-4850-y

Sparse bivariate polynomial factorization

WU WenYuan, CHEN JingWei* & FENG Yong

Chongqing Key Laboratory of Automated Reasoning and Cognition, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714*, China*
Email: wuwenyuan@cigit.ac.cn, chenjingwei@cigit.ac.cn, yongfeng@cigit.ac.cn

Received March 28, 2013; accepted October 15, 2013; published online June 25, 2014

Abstract Motivated by Sasaki's work on the extended Hensel construction for solving multivariate algebraic equations, we present a generalized Hensel lifting, which takes advantage of sparsity, for factoring bivariate polynomial over the rational number field. Another feature of the factorization algorithm presented in this article is a new recombination method, which can solve the extraneous factor problem before lifting based on numerical linear algebra. Both theoretical analysis and experimental data show that the algorithm is efficient, especially for sparse bivariate polynomials.

Keywords polynomial factorization, sparse polynomial, generalized Hensel lifting

MSC(2010) 12Y05, 68W30, 11Y16, 12D05, 13P05

Citation: Wu W Y, Chen J W, Feng Y. Sparse bivariate polynomial factorization. Sci China Math, 2014, 57: 2123–2142, doi: 10.1007/s11425-014-4850-y

1 Introduction

Polynomial factorization has been one of the most attractive areas in symbolic computation for a long time. Roughly speaking, it can be dated back to the Babylonians' algorithm to solve quadratic equations, around 1900–1600 BC [18]. The first polynomial-time algorithm for univariate polynomial factorization is due to Lenstra et al. [44], and the first kind of polynomial-time algorithms for multivariate polynomial factorization are due to Kaltofen [29–31]. Nowadays, polynomial factorization plays a significant role for both challenges and open problems and it addresses as well its usefulness in applications in various fields including the simplification, primary decomposition, solving polynomial equations, algebraic coding theory, cryptography, etc.

In this article, we present an efficient method for computing the irreducible factorization of any bivariate polynomial over the rationals satisfying Hypothesis (H) (see Subsection 1.1.1 for the hypothesis).

We start this introduction with some preliminaries and hypotheses. Then we present our main results and give an overview of the main steps of our algorithm. At the end of this section, we conclude with discussing related works.

1.1 Preliminaries

Let $\mathbb{Z}$ and $\mathbb{Q}$ represent the integer ring and the rational number field, respectively. Throughout this article, let f be the bivariate polynomial in $\mathbb{Z}[x, y]$ that we want to factorize, and we assume that it is squarefree; it has no univariate factors; its degrees are d_x in x and d_y in y.

*Corresponding author

1.1.1 *Hypothesis* (H)

Among so many different methods for bivariate polynomial factorization, the most popular strategy is the so-called *lifting and recombination.* One firstly factorizes $f(x,0)$ over $\mathbb{Q}$, then lifts the resulting univariate factors over a power series algebra, and lastly retrieves the bivariate factorization from recombination of the lifted factors. The lifting is the *classical Hensel lifting* (see, e.g., [21, Chapter 6]), and we call the univariate factors of $f(x,0)$ the *initial factors* for the classical Hensel lifting. For the *generalized Hensel lifting* presented in this article, the initial factors play a similar role, but they are different from that for the classical Hensel lifting. We will redefine the initial factors in this section. Before we do that, we need some basic concepts.

The *convex hull* of a set V of vectors in $\mathbb{R}^n$ (n-dimensional real vector space) is defined to be

$$\operatorname{conv}(V)=\left\{\sum_{i=1}^{k}\lambda_i\boldsymbol{v}_i:\boldsymbol{v}_i\in V,\ \lambda_i\in\mathbb{R}_{\geqslant 0},\text{ and }\sum_{i=1}^{k}\lambda_i=1\right\},$$

where $\mathbb{R}_{\geqslant 0}$ denotes the non-negative real numbers. If $|V|$ is finite, then the convex hull of V is called a *convex polytope*, where $|V|$ denotes the cardinality of V. A point of a polytope is called a *vertex* if it is not on the line segment of any two other points of the polytope. An edge of a polytope is the line segment of two vertices. It is well known that a polytope is always the convex hull of its vertices. The *Newton polytope* of a bivariate polynomial $f=\sum f_{i,j}x^iy^j$ is defined to be the convex hull in $\mathbb{R}^2$ of all the points (i,j) with $f_{i,j}\neq 0$, and denoted by N_f.

Definition 1.1. The low degree of a univariate polynomial is the lowest degree of non-zero terms in the polynomial expressed in canonical form, i.e., as a sum of terms.

Definition 1.2. The *Newton line* of f is defined to be an edge of N_f decided by the right-bottom-most point of N_f and another point such that no vertex of N_f is below this edge. The sum of all non-zero terms of f lying on the Newton line is called *Newton polynomial* of f, denoted by $f^{(0)}(x,y)$.

From this definition, the Newton line and the corresponding Newton polynomial of a given bivariate polynomial are unique, respectively. Let $\hat{\delta}$ and $\hat{d}>0$ be two integers such that $\gcd(\hat{\delta},\hat{d})=1$ and $\hat{\delta}/\hat{d}$ is the slope of the Newton line of f. If the slope is 0 then $\hat{\delta}=0$ and $\hat{d}=1$.

Definition 1.3. Let $\hat{\delta}$ and $\hat{d}>0$ be the two integers as above. For an integer $k\geqslant 0$ and an integer $n>0$, define

$$I_k(n):=\{x^jy^{-(n-j)\hat{\delta}/\hat{d}}\cdot y^{k/\hat{d}}:j=0,\ldots,n\}.$$

Definition 1.4. Let f and g be two polynomials in $\mathbb{Z}[x,y]$. For an integer $k\geqslant 0$ and an integer $n>0$, define $f-g\equiv 0\mod I_k(n)$ (or written as $f\equiv g\mod I_k(n)$) such that the low degree with respect to y of the coefficient of x^j in $f-g$ is greater than or equal to $(k-\hat{\delta}(n-j))/\hat{d}$, where j ranges from 0 to n.

We note that the meaning of *mod* in this article is completely different from the convention, where $f\equiv g\mod I$ usually indicates $f-g$ belongs to an ideal I. We shall give more details about these definitions in Subsection 2.2.

Definition 1.5. Let f be a bivariate polynomial in $\mathbb{Z}[x,y]$, $G_1,\ldots,G_r$ its irreducible factors over $\mathbb{Q}$, and d_i the degree of G_i with respect to x. For $i=1,\ldots,r$, define the *initial factors* of f, denoted by $G_i^{(0)}$, as the polynomials such that

$$\begin{cases}G_i^{(0)}\equiv 0\mod I_0(d_i),\\ G_i^{(0)}\equiv G_i\mod I_1(d_i).\end{cases}$$

Example 1.6. Let $f=x^8-3\,x^4y^2+5\,x^4y^5-4\,y^4+5\,y^7+2\,y^3x^4-8\,y^5+10\,y^8$. Over $\mathbb{Q}$, f has 2 irreducible factors $G_1=x^4+y^2+2\,y^3$ and $G_2=x^4-4\,y^2+5\,y^5$.

Figure 1 illustrates the Newton polytope and the Newton line of f, where e_x and e_y represent the exponents of the monomial $x^{e_x}y^{e_y}$ in x and y, repectively. Hence, the Newton polynomial of f is

$$f^{(0)}(x,y)=x^8-3\,x^4y^2-4\,y^4,$$

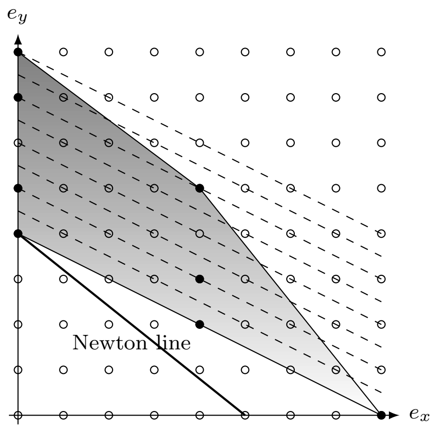

Figure 1 The Newton polytope of f

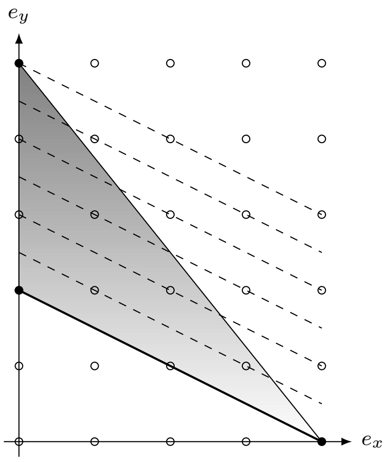

Figure 2 The Newton polytope of G_2

each of whose monomials corresponds to an integral point on the Newton line (see Figure 1). Here, $\hat{\delta} = -1$ and $\hat{d} = 2$. Thus

$$I_0(4) = \{y^2, xy^{\frac{3}{2}}, x^2y, x^3y^{\frac{1}{2}}, x^4\},$$

which corresponds to points with integral e_x on the Newton line of G_2, as in Figure 2, and

$$I_1(4) = \{y^{\frac{5}{2}}, xy^2, x^2y^{\frac{3}{2}}, x^3y, x^4y^{\frac{1}{2}}\}.$$

It follows from Definition 1.5 that $G_2^{(0)} = x^4 - 4\,y^2$ is an initial factor of f. Note that $G_2^{(0)}$ is reducible over $\mathbb{Q}$.

It is worth mentioning that the Newton polynomial and some similar definitions were introduced by Sasaki et al. [25–28, 49–51, 53] for solving multivariate algebraic equation and Puiseux series factorization of multivariate polynomials. In Subsection 1.2, we will discuss the differences between the work of Sasaki et al. [49–54] and that in this article.

Throughout this article, we assume on $f(x, y) \in \mathbb{Z}[x, y]$ and its initial factors such that the following hypothesis hold:

(H$_\text{a}$) f is squarefree and has no univariate factors;

(H$_\text{b}$) f is non-constant and monic in x;

(H$_\text{c}$) the initial factors of f are mutually coprime.

Note that (H$_\text{a}$) and (H$_\text{b}$) are not really restrictive, but for simple illustration: For any f having univariate factors, we can reduce it to (H$_\text{a}$) by computing $f/\gcd(f, \frac{\partial f}{\partial x})$ or $f/\gcd(f, \frac{\partial f}{\partial y})$; the leading coefficient problem can also be solved by the methods in [32, 55, 56]. However, (H$_\text{c}$) is crucial since it indicates the applicability of our method.

1.1.2 *Complexity model*

Following a popular way, such as in [8, 40, 43], we use computational tree model (see [9, Chapter 4]) for our complexity analysis. Roughly speaking, this means that the cost of each arithmetic operation $(+, -, \times, \div)$ or the equality test is a constant in the ground field. Denote by $\boldsymbol{M}(n)$ the complexity of multiplying two polynomials of degree at most n. As in [19, Chapter 8], we assume that $\boldsymbol{M}$ is supper-additive: For two positive integers m and n, $\boldsymbol{M}(m+n) \geqslant \boldsymbol{M}(m) + \boldsymbol{M}(n)$. The constant ω represents the linear algebra exponent, i.e., the multiplication of two $n \times n$ matrices can be computed in $\mathcal{O}(n^\omega)$ operations $(2 < \omega \leqslant 3)$, and so can the matrix inversion of an $n \times n$ matrix (see, e.g., [48, Chapter 2]). The $\tilde{\mathcal{O}}$ notation ignores logarithmic factors. As in [19, Chapter 25], we write $f \in \tilde{\mathcal{O}}(g)$ if there exist two positive integer c and N such that $f(n) \leqslant g(n)(\log(3 + g(n)))^c$ for all $n \geqslant N$.

1.2 Overview of the algorithm and contributions

Under Hypothesis (H), we consider the following three-stage bivariate polynomial factorization algorithm:

1. *Factorize Newton polynomial.* Factorize the Newton polynomial $f^{(0)}(x, y)$ in $\mathbb{Z}[x, \hat{y}]$, denoted by $f^{(0)} = g_1 \cdots g_s$, where $\hat{y} = y^{-\hat{\delta}/\hat{d}}$.
2. *Combine.* Combine $g_1, \ldots, g_s$ to the initial factors $G_1^{(0)}, \ldots, G_r^{(0)}$.
3. *Lift.* Lift the initial factors to the irreducible factors of f over $\mathbb{Z}$.

Obviously, this algorithm is different from the traditional lifting and recombination strategy, since the recombination is performed before lifting. The first stage is essentially a univariate factorization (see Proposition 3.1). Therefore we focus on the last two stages.

Our first contribution is the generalized Hensel lifting. As mentioned in Subsection 1.1.1, some concepts are motivated by the *extended Hensel construction* in [25–28, 49–51, 53]. Besides the similarities, there are two main differences between ours and Sasaki et al.'s. The one is that we introduce a parameter n in the definition of $I_k(n)$, which can be considered as a generalization of that in the work of Sasaki et al. [49–54]. More importantly, we correct [51, (3.11)], which is crucial in the lifting stage (see Remark 2.8 for detail). Thus the generalized Hensel lifting in this article is not only a simple generalization, but also a modification of the work of Sasaki et al. [49–54]. Meanwhile, we redefine the initial factors of f. The advantage is that if the lifting stage starts with the initial factors (see Definition 1.5), then, during the lifting stage, all the lifted factors are exactly in $\mathbb{Z}[x, y]$, not in a power series algebra. Moreover, the sparsity is preserved (see Section 2), in the sense that no extra term appears during lifting. Therefore, there does not exist the expression swell problem during the generalized Hensel lifting.

For computing the initial factors, our seccond contribution is a novel recombination method which is based on the use of numerical linear algebra. As in many common recombination methods in factorization, such as [5, 8, 23, 39, 43], we employ logarithm to linearize the problem. Since $f^{(0)} = G_1^{(0)} \cdots G_r^{(0)}$ (see Lemma 2.6), there must exist a unique vector $\mu_i = (\mu_{j,i}) \in \{0, 1\}^s$ such that $G_i^{(0)} = \prod_{j=1}^{s} g_j^{\mu_{j,i}}$ for $i = 1, \ldots, r$. After taking natural logarithm, we have

$$\mathrm{Ln}\, G_i^{(0)} = \sum_{j=1}^{s} \mu_{j,i} \mathrm{Ln}\, g_j.$$

Here $\mathrm{Ln} z$ represents the complex natural logarithm function for $z \in \mathbb{C}$. Differently from existing methods, we use neither trace recombination nor logarithmic derivative recombination to construct the linear system. Instead, we simply construct a linear system with full rank Jacobian by evaluating the equations at some appropriate points. Thus, the solution μ_i can be easily obtained by numerical computation. We prove the combination method is correct under certain error controls.

Here we need to point out that this combination method is not deterministic, since its validity depends on an effective version of the Hilbert irreducibility theorem for bivariate polynomials. This is the only one reason that causes our algorithm non-deterministic. More precisely, suppose $g(x, y) \in \mathbb{Z}[x, y]$ is irreducible and let S be a subset of $\mathbb{Q}$ with $|S| = N$. We need an explicit bound on

$$\frac{|\{b \in S : g(x, b) \text{ is irreducible in } \mathbb{Q}[x]\}|}{N},$$

which is the probability that $g(x, b)$ is irreducible when b is picked randomly from S. Finding a good bound for this ratio in terms of N and the height of g is well-known open; see for example [19, pp. 469 –472]. However, experiments indicate that this probability is near to 1 when N is large enough. Totally, our algorithm is not deterministic, but with a very highly successful probability in practice.

We have the following theorem to summarize the main result of this article, and its proof will be given in Section 4.

Theorem 1.7. *Assume that there exists an effective version of the Hilbert irreducibility theorem for bivariate polynomials over $\mathbb{Z}$. Given a bivariate polynomial f over $\mathbb{Q}$ satisfying Hypothesis* (H)*, there exists an algorithm which reduces the computation of the irreducible factors of f over $\mathbb{Q}$ to univariate polynomial factorization with degree at most d_x over $\mathbb{Q}$ in $\tilde{O}(T d_x d_y + s^{\omega})$ arithmetic operations in $\mathbb{Q}$,* *where T is the number of non-zero terms of f, d_x and d_y are the degrees of f in x and y, respectively, and s $(\leqslant d_x)$ is the number of the irreducible factors in $\mathbb{Z}[x, \hat{y}]$ of the Newton polynomial of f.*

To the authors' knowledge, the best complexities for the reduction from bivariate polynomial factorization over $\mathbb{Q}$ to univariate polynomial factorization are due to Lecerf [43], $\tilde{\mathcal{O}}((d_xd_y)^{(\omega+1)/2})$ in the deterministic case and $\tilde{\mathcal{O}}((d_xd_y)^{1.5})$ in the probabilistic case. We note that Lecerf's algorithm is for dense polynomials. The main algorithm in this article also works in dense case, but we have $T = \mathcal{O}(d_xd_y)$. However, in sparse case, especially for those polynomials satisfying $T < (d_xd_y)^{1/2}$, we have $\tilde{\mathcal{O}}(Td_xd_y + s^\omega) \subset \tilde{\mathcal{O}}((d_xd_y)^{1.5})$, i.e., the present algorithm outperforms that in [43]. In [58], a sparse version of a Lecerf's algorithm was presented, which is based on polytope method. The complexity of the reduction is $\mathcal{O}(\mathrm{Vol}(N_f)^\omega)$, where $\mathrm{Vol}(N_f)$ is the volume of the Newton polytope of f. However, $\mathrm{Vol}(N_f)$ is near to d_xd_y in the worst case, even though the input polynomial is sparse. Moreover, Hypothesis (H1) in [58] is that the Newton polytope must contain $(0, 0)$, $(1, 0)$, and $(0, 1)$, which is somewhat strict and limits its applicability.

Our last contribution is the Maple implementation of our algorithm. The experimental results show that our algorithm is efficient in practice (see Section 5).

1.3 Related works

We refer to [11, 18, 33–35, 38] and references therein for details on existing algorithms and the history of polynomial factorization. We only discuss related methods.

The work of Sasaki et al. [52–54] not only introduced the idea of extended Hensel construction, but also initiated the trace recombination technique. The extended Hensel construction was then used for solving multivariate algebraic equation and Puiseux series factorization of multivariate polynomials in [25–28, 49–51]. The generalized Hensel lifting in this article is a generalization and also a modification of the extended Hensel construction. It preserves the sparsity of the input polynomial during the lifting stage.

The idea of trace recombination has been successfully applied to polynomial factorization, such as van Hoeij's trace recombination [23] for univariate polynomial factorization in $\mathbb{Z}[x]$, the logarithmic derivative recombination of Belabas et al. [5] for factoring polynomials over global fields, etc. We refer to [8] for more historical notes in this area. The combination method in this article commonly employs logarithm to linearize the problem, but uses a different method to construct the linear system, which is then solved by numerical linear algebra.

For multivariate polynomial factorization over number fields, many effective approaches, which are various Hensel lifting based algorithms, have been developed over the past few decades. These sophisticated techniques and their complexity analysis have been studied extensively in [8, 16, 24, 33, 34, 40–43, 47, 56, 57]. However, it is well known that the Hensel lifting usually destroys the sparsity and leads to expression swell. Although Bernardin worked on bivariate Hensel lifting and its parallelization [6], his work also focuses on the dense case. The first attempt to this problem was addressed in [61, 62]. The author presented a probabilistic method to detect zero coefficient terms, namely the sparse structure, then convert the lifting stage to solve linear systems with much smaller size by using sparsity. Von zur Gathen and Kaltofen [17, 20, 32] substantially contributed to this subject and gave rigorous proofs for the probability and the complexity of this scheme based on effective Hilbert Irreducibility Theorem.

In 1999, a polynomial-time approach to find factors of fixed degree, especially linear and quadratic factors, in an algebraic extension of fixed degree, in particular to compute rational roots of univariate supersparse polynomials, was given in [45, 46] based on the result in [13]. The further study generalizing to bivariate supersparse polynomials has been made by other researchers in [4, 36, 37]. We refer to [10] for the most recent progress in this direction.

Another important direction for sparse polynomial factorization is by using Newton polytopes to obtain the information about the supports. Gao's work [15] initiated such a first step. Later on a new type of factorization method appeared in [1, 3]. Abu Salem [2] presented a sparse adaptation of the polytope method to factorize bivariate polynomials over finite fields, which also can be adapted to fields of arbitrary characteristic. Such polytope methods can perform well for sparse polynomials when their Newton polytopes have very few Minkowski decompositions.

As mentioned previously, the algorithm having the currently best complexity for dense bivariate polynomial factorization is due to Lecerf [43]. Weimann [58] presented a sparse version of Lecerf's algorithm and showed that sparse bivariate polynomial factorization has a polynomial complexity in the volume of the Newton polytope of the input. But Weimman's algorithm requires that the input must have the constant term and the linear terms. The most recent advance [7] improves Weimman's results and gives a reduction algorithm which reduces the volume of the Newton polytope based on the use of unimodular transformation. Recently, Weimann [59] presented an interesting method for bivariate polynomial factorization without using Hensel lifting.

Note. The abstract of this paper was presented as a poster at the 37th International Symposium on Symbolic and Algebraic Computation [60].

2 Generalized Hensel lifting

Let $f \in \mathbb{Z}[x,y]$ satisfy Hypothesis (H) and $G_1,\dots,G_r$ be its irreducible factors over $\mathbb{Q}$. In this section, we assume that initial factors $G_1^{(0)},\dots,G_r^{(0)}$ are given. Under these assumptions, we give the generalized Hensel lifting algorithm, which preserves the sparsity of the input polynomial. Moreover, the lifted factors are always in $\mathbb{Z}[x,y]$.

2.1 Moses-Yun polynomial

Lemma 2.1. *Let $h_i(x,y)$ $(i=1,\dots,r)$ be homogeneous polynomials in x and y with $r \geqslant 2$ and $\deg_x(h_i)=d_i \geqslant 1$ such that*

$$\gcd(h_i,h_j)=1 \quad \textit{for any} \quad i \neq j. \tag{2.1}$$

Then for each $l \in \{0,\dots,d_x-1\}$, where $d_x=\sum_{i=1}^r d_i$, there exists a unique set of polynomials $\{W_i^{(l)}(x,y)\}$ satisfying

$$\begin{aligned} &W_1^{(l)}[h_1\cdots h_r/h_1]+\cdots+W_r^{(l)}[h_1\cdots h_r/h_r]=x^l y^{d_x-l},\\ &\deg_x(W_i^{(l)})<\deg_x(h_i), \quad i=1,\dots,r. \end{aligned} \tag{2.2}$$

Moreover, each $W_i^{(l)}$ is a homogeneous polynomial in x and y of total degree d_i.

Following [49], we call $W_i^{(l)}$ $(i=1,\dots,r,\ l=0,\dots,d_x-1)$ Moses-Yun (interpolation) polynomials, which are used to represent the monomials lying on Newton line by initial factors. This lemma was presented in [51] and its proof can be found there. In addition, its univariate version was presented in [53, 54]. From the proof of this lemma, we can get the following algorithm to compute Moses-Yun polynomials.

Algorithm 2.2 (Moses-Yun polynomials). Input: $h_i(x,y)$ $(i=1,\dots,r)$ satisfying the conditions in Lemma 2.1. Output: The Moses-Yun polynomials $W_i^{(l)}$.

Step 1 (Compute $W_i^{(l)}(x,1)$). For i from 1 to r do the following: Let $G:=h_i(x,1)$ and

$$H:=\prod_{j=1}^r h_j(x,1)/G.$$

Call extended Euclidean algorithm to compute V and W such that $VG+WH=1$, $\deg(V)<\deg(H)$ and $\deg(W)<\deg(G)$; for l from 1 to d_x-1, let $W_i^{(l)}(x,1):=\mathrm{rem}(xW_i^{(l-1)},G,x)$, where $W_i^{(0)}(x,1)=W$.

Step 2 (Homogenize $W_i^{(l)}(x,1)$). Homogenize each $W_i^{(l)}(x,1)$ such that $W_i^{(l)}(x,y)$ is homogeneous polynomial with total degree d_i with respect to x and y. Return $W_i^{(l)}(x,y)$.

Remark 2.3. If $G, H \in \mathbb{Z}[x]$, then $\|W\|_\infty \leqslant \max\{\|G\|_\infty, \|H\|_\infty\}^{\deg G+\deg H-1}$ by the resultant theory (see e.g., [12, Subsection 3.5, Proposition 9]). Therefore, $\|W_i^{(l)}\|_\infty \leqslant \max\{\|G\|_\infty, \|H\|_\infty\}^{\deg G+\deg H}$. Here $\|\cdot\|_\infty$ represents the ∞-norm for a polynomial, i.e., the maximum of the absolute values of the coefficients of a polynomial.

Proposition 2.4. *Algorithm* 2.2 *works correctly and requires* $\mathcal{O}(r\log d_x\, \boldsymbol{M}(d_x) + d_x^2)$ *arithmetic operations.*

Proof. The correctness of Algorithm 2.2 follows from Lemma 2.1 and its proof. In Step 1, for computing H we can first compute $\prod h_i(x,1)$. The product of r univariate polynomials whose degree sum is d_x takes $\mathcal{O}(\boldsymbol{M}(d_x)\log r)$ operations [19, Chapter 10]. We then compute H for i from 1 to r with at most $r\boldsymbol{M}(d_x)$ operations. From [19, Chapter 11], the extended Euclidean algorithm costs $\mathcal{O}(\boldsymbol{M}(d_x)\log d_x)$ operations. Here we call extended Euclidean algorithm r times. This needs $\mathcal{O}(r\boldsymbol{M}(d_x)\log d_x)$ operations. Furthermore, computing all $W_i^{(l)}(x,1)$ by division with remainder costs $\mathcal{O}(r\boldsymbol{M}(d_x))$. From Lemma 2.1, $W_i^{(l)}(x,y)$ is homogeneous polynomial of total degree d_i with respect to x and y. Thus for each l, it costs at most $d_1+\cdots+d_r = d_x$ operations to homogenize $W_i^{(l)}(x,y)$ for $i=1,\ldots,r$, and hence Step 2 of the above algorithm costs at most $\mathcal{O}(d_x^2)$ operations. Thus Algorithm 2.2 uses in total $\mathcal{O}(r\log d_x\, \boldsymbol{M}(d_x)+d_x^2)$ operations. □

2.2 Lifting

Recall the mod operation (Definition 1.4) and

$$I_k(n) := \{x^j y^{-(n-j)\hat{\delta}/\hat{d}} \cdot y^{k/\hat{d}} : j = 0,\ldots,n\}, \tag{2.3}$$

where $\hat{\delta}$ and $\hat{d} > 0$ are two integers such that $\gcd(\hat{\delta},\hat{d}) = 1$ and $\hat{\delta}/\hat{d}$ is the slope of the Newton line of f; if the slope is 0 then $\hat{\delta} = 0$ and $\hat{d} = 1$.

In what follows, let $\hat{y} = y^{-\hat{\delta}/\hat{d}}$. Then each element in $I_k(n)$ can be seen as a product of $y^{k/\hat{d}}$ and an n degree homogeneous part $x^j\hat{y}^{n-j}$.

If $n = d_x$, then we denote I_k by $I_k(d_x)$, where d_x is the degree of f in x. From Definition 1.3, it follows that any monomial of f is contained in I_k for some integer k. For instance, each monomial of $f^{(0)}(x,y)$ lies in I_0, i.e.,

$$f^{(0)} \equiv 0 \mod I_0 \quad \text{and} \quad f^{(0)} \equiv f \mod I_1. \tag{2.4}$$

From the geometric point of view, each I_k corresponds to a line parallel to the Newton line, and $f \equiv g \mod I_k$ means all monomials of $f-g$ lie above the line corresponding to I_{k-1} (see Example 1.6).

Lemma 2.5. *Let all notations be as above. Let* $\hat{G}_i$ $(i = 1,\ldots,r)$ *be polynomials in* x *and* $\hat{y}$ *with* $r \geqslant 2$ *and* $\deg_x(\hat{G}_i) = d_i \geqslant 1$. *For a fixed integer* $2 \leqslant m \leqslant r$, *if each monomial of* $\hat{G}_i$ *lies in* $I_k(d_i)$ *for* $i = 1,\ldots,m$, *and* $\hat{G}_j \equiv G_j^{(0)} \bmod I_1(d_j)$ *for* $j = m+1,\ldots,r$, *then*

$$\prod_{i=1}^{m}\hat{G}_i \prod_{j=m+1}^{r}\hat{G}_j \equiv 0 \mod I_{k+1}(d_x), \tag{2.5}$$

where $d_x = \deg_x(\hat{G}_1\cdots\hat{G}_r)$ *and* $k \geqslant 1$.

Proof. Denote the product in (2.5) by $P = P_1\cdot P_2$. We consider the low degree in y of the coefficient of x^j of P, denoted by $\mathrm{ldeg}(\mathrm{coeff}(P,x^j),y)$. Then

$$\begin{aligned}\mathrm{ldeg}(\mathrm{coeff}(P,x^j),y) &= \mathrm{ldeg}\Bigg(\sum_{\ell_1+\ell_2=j}\mathrm{coeff}(P_1,x^{\ell_1})\cdot\mathrm{coeff}(P_2,x^{\ell_2}),y\Bigg)\\ &\geqslant \frac{(d_1+\cdots+d_m-\ell_1)\cdot(-\hat{\delta})}{\hat{d}} + \frac{mk}{\hat{d}} + \frac{(d_{m+1}+\cdots+d_r-\ell_2)\cdot(-\hat{\delta})}{\hat{d}}\end{aligned}$$

$$\geqslant \frac{(2k-\hat{\delta}(d_x-j))}{\hat{d}} \geqslant \frac{(k+1-\hat{\delta}(d_x-j))}{\hat{d}},$$

where the first inequality is from that each monomial of $\hat{G}_i$ lies in $I_k(d_i)$ for $i=1,\ldots,m$ and that $\hat{G}_j \equiv G_j^{(0)} \mod I_1(d_j)$ for $j=m+1,\ldots,r$, the second inequality is from $m \geqslant 2$, and the last one is from $k \geqslant 1$. Then this lemma follows from Definition 1.4. □

This lemma plays an important role during the lifting stage. Although the proof of Lemma 2.5 does not work for $k=0$, we still have the following lemma about the relation between the Newton polynomial and the initial factors of f.

Lemma 2.6. *Let f be a bivariate polynomial in $\mathbb{Z}[x,y]$, $f^{(0)}$ its Newton polynomial, G_i its irreducible factors, and $G_i^{(0)}$ its initial factors for $i=1,\ldots,r$. Then*

$$f^{(0)} = G_1^{(0)} \cdots G_r^{(0)}.$$

Proof. From Definition 1.5, we have

$$G_i^{(0)} \equiv 0 \mod I_0(d_i),$$
$$G_i^{(0)} \equiv G_i \mod I_1(d_i).$$

Then, by Definition 1.4, it is easy to check

$$G_1^{(0)} \cdots G_r^{(0)} \equiv 0 \mod I_0,$$
$$G_1^{(0)} \cdots G_r^{(0)} \equiv G_1 \cdots G_r \mod I_1.$$

Then this lemma follows from (2.4). □

We now present the generalized Hensel lifting, in which we use I_k ($k=1,2,\ldots$) as moduli. Geometrically speaking, we lift all factors by $y^{1/\hat{d}}$, along the positive direction of e_y axis (see Figure 2), in each lifting step.

Theorem 2.7 (Generalized Hensel lifting). *Let $f \in \mathbb{Z}[x,y]$ satisfy Hypothesis* (H) *and $G_1,\ldots,G_r$ be its irreducible factors. Let $f^{(0)}$ be the Newton polynomial of f, $G_1^{(0)},\ldots,G_r^{(0)}$ ($r \geqslant 2$) the initial factors of f with $\deg_x G_i^{(0)} = d_i \geqslant 1$, and I_k ($k=0,1,2,\ldots$) as in* (2.3)*. Then, for any nonnegative integer k, we can construct $\Delta G_i^{(k)} \in \mathbb{Z}[x,y]$, with each monomial lying in $I_k(d_i)$, satisfying*

$$G_i \equiv G_i^{(0)} + \sum_{j=0}^{k} \Delta G_i^{(j)} \mod I_{k+1}(d_i), \tag{2.6}$$

$$f \equiv \prod_{i=1}^{r} \left(G_i^{(0)} + \sum_{j=0}^{k} \Delta G_i^{(j)} \right) \mod I_{k+1}.$$

Moreover, this construction is unique.

Proof. By induction on k. When $k=0$, let $\Delta G_i^{(0)} = 0$ for $i=1,\ldots,r$. Then this theorem follows from Lemma 2.6. Suppose this theorem is true up to $k-1$ ($k \geqslant 1$). Let

$$G_i^{(k-1)} = G_i^{(0)} + \sum_{j=0}^{k-1} \Delta G_i^{(j)}.$$

By induction assumptions,

$$f(x,y) \equiv \prod_{i=1}^{r} G_i^{(k-1)}(x,y) \mod I_k.$$

Let

$$\Delta f^{(k)}(x,y) = f(x,y) - \prod_{i=1}^{r} G_i^{(k-1)}(x,y) \mod I_{k+1}. \tag{2.7}$$

Then $\Delta f^{(k)}(x,y) \in \mathbb{Z}[x,y]$ can be expressed as

$$\begin{aligned} &\Delta f^{(k)}(x,y) = c_{d_x-1}^{(k)} \cdot x^{d_x-1}\hat{y} + \cdots + c_0^{(k)} \cdot \hat{y}^{d_x}, \\ &c_l^{(k)} = a_l^{(k)} y^{k/\hat{d}}, \quad a_l^{(k)} \in \mathbb{Z}, \quad l = 0, \dots, d_x - 1. \end{aligned} \tag{2.8}$$

From Lemma 2.1, we compute the corresponding $W_i^{(l)}$ for $G_i^{(0)}$, where $i = 1, \dots, r$ and $l = 0, \dots, d_x - 1$. Then we construct $\Delta G_i^{(k)}$ as

$$\Delta G_i^{(k)}(x,y) = \sum_{l=0}^{d_x-1} W_i^{(l)}(x,\hat{y}) c_l^{(k)}, \tag{2.9}$$

whose monomials are obviously in $I_k(d_i)$. Now, let $G_i^{(k)}(x,y) = G_i^{(k-1)}(x,y) + \Delta G_i^{(k)}(x,y)$. Then

$$\begin{aligned} f - \prod_{i=1}^r G_i^{(k)} &= f - \prod_{i=1}^r (G_i^{(k-1)} + \Delta G_i^{(k)}) \bmod I_{k+1} \\ &\equiv f - \Bigg(\prod_{i=1}^r G_i^{(k-1)} + \sum_{i=1}^r \frac{\Delta G_i^{(k)}}{G_i^{(0)}} \prod_{j'=1}^r G_{j'}^{(0)} \\ &\quad + \sum_{i=1}^r \sum_{j=1}^r \frac{\Delta G_i^{(k)} \Delta G_j^{(k)}}{G_i^{(0)} G_j^{(0)}} \prod_{j'=1}^r G_{j'}^{(0)} + \cdots \Bigg) \bmod I_{k+1}. \end{aligned}$$

Since each monomial of $\Delta G_i^{(k)}(x,y)$ is in $I_k(d_i)$, from Lemma 2.5 we have

$$\sum_{i=1}^r \sum_{j=1}^r \frac{\Delta G_i^{(k)} \Delta G_j^{(k)}}{G_i^{(0)} G_j^{(0)}} \prod_{j'=1}^r G_{j'}^{(0)} + \cdots \equiv 0 \bmod I_{k+1}.$$

Hence,

$$f - \prod_{i=1}^r G_i^{(k)} \equiv \Delta f^{(k)} - \sum_{i=1}^r \frac{\Delta G_i^{(k)}}{G_i^{(0)}} \prod_{j'=1}^r G_{j'}^{(0)} \bmod I_{k+1},$$

and then substituting $\Delta G_i^{(k)}$ by the expression in (2.9) to this equation gives

$$\begin{aligned} f - \prod_{i=1}^r G_i^{(k)} &\equiv \Delta f^{(k)} - \sum_{i=1}^r \frac{\sum_{l=0}^{d_x-1} c_l^{(k)} W_i^{(l)}}{G_i^{(0)}} \prod_{j'=1}^r G_{j'}^{(0)} \bmod I_{k+1} \\ &\equiv \Delta f^{(k)} - \sum_{l=0}^{d_x-1} c_l^{(k)} \sum_{i=1}^r W_i^{(l)} \frac{\prod_{j'=1}^r G_{j'}^{(0)}}{G_i^{(0)}} \bmod I_{k+1} \\ &\equiv \Delta f^{(k)} - \sum_{l=0}^{d_x-1} c_l^{(k)} x^l \hat{y}^{d_x-l} \bmod I_{k+1}, \end{aligned}$$

where the last equality is from the property of Moses-Yun polynomials in Lemma 2.1. From (2.8) we have

$$f(x,y) \equiv \prod_{i=1}^r G_i^{(k)}(x,y) \bmod I_{k+1}.$$

Moreover, the uniqueness of $\Delta G_i^{(k)}$ follows directly from the uniqueness of Moses-Yun polynomials. Meanwhile, the uniqueness of the construction of $\Delta G_i^{(k)}$ implies (2.6) holds, and hence $\Delta G_i^{(k)} \in \mathbb{Z}[x,y]$. □

Remark 2.8. The basic idea of Theorem 2.7 is similar to the extended Hensel construction in the work of Sasaki et al., such as [51, Theorem 1], but they are different in the following aspects. Firstly, our initial factors of f are redefined to be $G_1^{(0)}, \dots, G_r^{(0)}$ in this article. In [51], the initial factors are the irreducible factors in $\mathbb{Z}[x,\hat{y}]$ of the Newton polynomial $f^{(0)}$. Secondly, [51, Theorem 1, (3.11)] claims that $G_i^{(k)} \equiv G_i^{(0)} \bmod I_1$, which is not true. From Theorem 2.7, we can only infer that $G_i^{(k)} \equiv G_i^{(0)}$

mod $I_1(d_i)$ holds for $i = 1, \dots, r$. Thirdly, this theorem implies that all $G_i^{(k)} = G_i^{(0)} + \Delta G_i^{(1)} + \cdots + \Delta G_i^{(k)}$ are in always $\mathbb{Z}[x, y]$, while $G_i^{(k)}$ lies in $\mathbb{C}\{y^{1/\hat{d}}\}[x]$ in the work of Sasaki et al. The reason is the new definition of initial factors, which makes the results of each lifting step be always in $\mathbb{Z}[x, y]$. From these points of view, the generalized Hensel lifting is not only a simple generalization, but also a modification of the extended Hensel construction in the work of Sasaki et al.

Remark 2.9. From (2.6), we have $G_i \equiv G_i^{(k)} \mod I_{k+1}(d_i)$, i.e., each lifted factor $G_i^{(k)}$ is always a part of G_i and tends to G_i when k increases. We then have some benefits. It is obvious but important that the lifting process do not impact on the sparsity of $\Delta f^{(k)}$, since it follows from (2.6) that $\Delta f^{(k)}$ is always a part of f, too. Moreover, the number of terms of $\Delta f^{(k)}$ will be smaller and smaller when k tends to be larger and larger. Therefore, the lifting process terminates as soon as $\Delta f^{(k)} = 0$. This property is different from the classical Hensel lifting and the extended Hensel construction, in both of which the expression swell may occur.

Remark 2.10. The uniqueness of the Moses-Yun polynomial is a very important part in the proof of Theorem 2.7. From Lemma 2.1, the uniqueness is a direct consequence of Hypothesis ($\mathrm{H_c}$. In other words, Hypothesis ($\mathrm{H_c}$) guarantees the uniqueness of the generalized Hensel lifting.

As a by-product of Theorem 2.7, we can prove the following interesting result, which will be used in Section 3.

Corollary 2.11. *Let $f \in \mathbb{Q}[x, y]$ satisfy Hypothesis* (H) *and $G_1, \dots, G_r$ be its irreducible factors over $\mathbb{Q}$. Then the number of non-zero terms of G_i is not more than $(T + 1)(d_i + 1)$, where T is the number of non-zero terms of f, and d_i is the degree of G_i with respect to x.*

Proof. From the generalized Hensel lifting, there exists some positive integer k such that $G_i = G_i^{(0)} + \Delta G_i^{(1)} + \cdots + \Delta G_i^{(k)}$. Here $k \leqslant T$ as the number of layers of G_i is always less than that of f (see Figures 1 and 2), and the number of layers of f is less than T. Furthermore, since $G_i^{(0)}$ is homogeneous with degree d_i in x and y, the number of non-zero terms is not more than $d_i + 1$. By (2.9), each $\Delta G_i^{(j)}$ has at most $d_i + 1$ non-zero terms, which completes the proof. □

Based on Theorem 2.7, we present the following algorithm to lift all factors by $y^{1/\hat{d}}$.

Algorithm 2.12 (Lifting). Input: A bivariate polynomial $f \in \mathbb{Z}[x, y]$ satisfying Hypothesis (H), Moses-Yun polynomials $W_i^{(l)}(x, \hat{y})$ of its initial factors for $i = 1, \dots, r$, $l = 0, \dots, d_x - 1$ and $G_1^{(k-1)}, \dots, G_r^{(k-1)}$ such that $f \equiv G_1^{(k-1)} \cdots G_r^{(k-1)} \mod I_k$. Output: $G_1^{(k)}, \dots, G_r^{(k)}$ such that $f \equiv G_1^{(k)} \cdots G_r^{(k)} \mod I_{k+1}$.

Step 1 (Compute $\Delta f^{(k)}$). Compute $\Delta f^{(k)} := f - G_1^{(k-1)} \cdots G_r^{(k-1)} \mod I_{k+1}$.

Step 2 (Produce $\Delta G_i^{(k)}$). Express $\Delta f^{(k)}$ as

$$\begin{aligned} \Delta f^{(k)}(x, y) &= c_{d_x-1}^{(k)} \cdot x^{d_x-1}\hat{y} + \cdots + c_0^{(k)} \cdot \hat{y}^{d_x}, \\ c_l^{(k)} &= a_l^{(k)} y^{k/\hat{d}}, \quad a_l^{(k)} \in \mathbb{Z}, \quad l = 0, \dots, d_x - 1. \end{aligned}$$

For i from 1 to r, let $\Delta G_i^{(k)}(x, y) := \sum_{l=0}^{d_x-1} W_i^{(l)} c_l^{(k)}$, and $G_i^{(k)} := G_i^{(k-1)} + \Delta G_i^{(k)}$. Return $G_i^{(k)}$.

Proposition 2.13. *Algorithm* 2.12 *works correctly. It requires $\mathcal{O}(\boldsymbol{M}(d_x d_y) + r d_x)$ arithmetic operations.*

Proof. The correction follows from Theorem 2.7. By Kronecker substitution, two polynomials in $\mathbb{Z}[x, y]$ of degree less than n in x and d in y can be multiplied using $\mathcal{O}(\boldsymbol{M}(nd))$ operations. Since the degrees of $G_1^{(k-1)} \cdots G_r^{(k-1)}$ in x and y are not more than d_x and d_y, respectively, Step 1 can be done by means of the sub-product tree technique in [19, Chapter 10] with $\mathcal{O}(\boldsymbol{M}(d_x d_y))$ operations. Since each $W_i^{(l)}$ is a homogeneous polynomial in x and $\hat{y}$ of total degree d_i, it has at most $d_i + 1$ non-zero terms. Moreover, each $c_l^{(k)}$ is monomial. So all $G_i^{(k)}$ can be computed with $\mathcal{O}(r d_x)$ operations. Totally, Algorithm 2.12 uses $\mathcal{O}(\boldsymbol{M}(d_x d_y) + r d_x)$ operations. □

So far, we have proposed an efficient method to lift the initial factors towards the irreducible factors. The lifting method preserves the sparsity of the input polynomial, and hence no intermediate expression swell happens. In next section, we will give a recombination strategy to obtain the correct initial factors.

3 Recombination for initial factors

Recall that $\hat{\delta}$ and $\hat{d} > 0$ are two integers such that $\gcd(\hat{\delta}, \hat{d}) = 1$ and $\hat{\delta}/\hat{d}$ is the slope of the Newton line of f. If the slope is 0 then $\hat{\delta} = 0$ and $\hat{d} = 1$. If $\hat{\delta} = 0$ then the Newton polynomial $f^{(0)}$ is a univariate polynomial in x (because of Hypothesis $(\mathrm{H_a})$, f has no univariate factors, including power of y), else $f^{(0)}$ is a homogeneous polynomial with respect to x and $\hat{y} = y^{-\hat{\delta}/\hat{d}}$. Therefore, the irreducible factorization of the Newton polynomial can be deduced from that of the univariate polynomial $f^{(0)}(x, 1)$, denoted by $f^{(0)}(x, y) = f^{(0)}(x, \hat{y}) = g_1(x, \hat{y}) \cdots g_s(x, \hat{y})$, where $g_i(x, \hat{y})$ for $i = 1, \dots, s$ are in $\mathbb{Z}[x, \hat{y}]$. Consequently we have the following result.

Proposition 3.1. *Factoring the Newton polynomial $f^{(0)}(x, y)$ in $\mathbb{Z}[x, \hat{y}]$ is equivalent to factoring a univariate polynomial over $\mathbb{Z}$ whose degree is at most d_x.*

However, it is not always the case that $g_1, \dots, g_s$ are exactly the initial factors.

Example 3.2. For the polynomial in Example 1.6, we have $\hat{y} = y^{1/2}$ and $f^{(0)}(x, \hat{y}) = (x^4 + \hat{y}^4)(x^2 + \hat{y}^2)(x^2 - 2\hat{y}^2) \triangleq g_1 g_2 g_3$, but the initial factors are $G_1^{(0)} = x^4 + y^2 = g_1$ and $G_2^{(0)} = x^4 - 4y^2 = g_2 g_3$.

This is somewhat similar to the extraneous factors problem. Here it is unavoidable, too. That is to say, $f^{(0)}(x, y)$ may have more factors than $f(x, y)$ does, i.e., $r \leqslant s$. In this section, we introduce a method, based on approximation theory and linear algebra, which combines $g_1, \dots, g_s$ to the initial factors $G_1^{(0)}, \dots, G_r^{(0)}$ of f when $r < s$.

3.1 Recombination method

For simplicity, we only discuss the case that the slope of the Newton line of f is -1, i.e., $\hat{\delta} = -1$, $\hat{d} = 1$ and $\hat{y} = y$. The same strategy works for any other case. Then the factorization of the Newton polynomial is now

$$f^{(0)}(x, y) = g_1(x, y) \cdots g_s(x, y). \tag{3.1}$$

We now describe the idea behind the combination method. From Lemma 2.6, the product of the initial factors is the Newton polynomial. Thus there must exist a unique vector $\mu_i = (\mu_{j,i}) \in \{0, 1\}^s$ such that $G_i^{(0)} = \prod_{j=1}^s g_j^{\mu_{j,i}}$ for $i = 1, \dots, r$. After taking natural logarithm, we have

$$\operatorname{Ln} G_i^{(0)} = \sum_{j=1}^{s} \mu_{j,i} \operatorname{Ln} g_j, \tag{3.2}$$

where $\operatorname{Ln} z$ represents the complex natural logarithm function for $z \in \mathbb{C}$. Thus if we can evaluate (3.2) at some points (x_i, y_i), then we can construct a linear system, from which it is possible to solve for the 0-1 vector μ_i. However, we do not know $G_i^{(0)}$ in advance, but fortunately, we can approximate it.

Let f have r irreducible rational factors $G_1, \dots, G_r$. By Theorem 2.7, there exists an integer k_i such that $G_i = G_i^{(0)} + \sum_{k=1}^{k_i} \Delta G_i^{(k)}$ for $i = 1, \dots, r$, where each monomial of $\Delta G_i^{(k)}$ lies in $I_k(d_i)$ and $d_i = \deg_x G_i^{(0)}$. By the definition of Newton line and Hypothesis $(\mathrm{H_a})$, we have $\deg_x G_i = d_i$. Let $w = y/x$. Then G_i/x^{d_i} can be expressed as

$$\frac{G_i}{x^{d_i}} = \frac{G_i^{(0)}}{x^{d_i}} + \sum_{k=1}^{k_i} \frac{\Delta G_i^{(k)}}{x^{d_i}} = G_i^{(0)}(w) + g_i^{(1)}(w) \cdot y + \cdots + g_i^{(k_i)}(w) \cdot y^{k_i}, \tag{3.3}$$

where $g_i^{(k)} \in \mathbb{Z}[w]$ satisfies $x^{d_i} y^k \cdot g_i^{(k)} = \Delta G_i^{(k)}$ for $k \geqslant 1$. Thus if we choose an evaluation point (x, y) such that $w \sim \mathcal{O}(1)$ and $y \sim o(1)$, then

$$\left| \frac{G_i}{x^{d_i}} - \frac{G_i^{(0)}}{x^{d_i}} \right| \sim o(1), \tag{3.4}$$

where $y \sim o(1)$ means that y is sufficiently small.

Now, we give the following combination algorithm.

Algorithm 3.3 (Combine). Input: A bivariate polynomial $f \in \mathbb{Z}[x,y]$ satisfying Hypothesis (H) with degree d_x in x, $g_1,\dots,g_s$, the factors of $f^{(0)}$, and an arbitrary positive number M. Output: $G_1^{(0)},\dots,G_r^{(0)}$, i.e., the initial factors of f.

Step 1 (Construct coefficient matrix). Let $z = y/x$. Then $g_j(x,y)/x^{\deg_x g_j} = g_j(1,z)$ by the homogeneity of the Newton line. Choose s random numbers $z_1,\dots, z_s \in \mathbb{C}$ uniformly with absolute value $\leqslant M$, set $A := (a_{i,j})_{s\times s}$ with $a_{i,j} := \mathrm{Re}\,(\mathrm{Ln}\,(g_j(1,z_i)))$, and let $\tilde{A} = (\tilde{a}_{i,j}) = A + \delta A$ with $\tilde{a}_{i,j} \in \mathbb{Q}$ and

$$\|\delta A\|_\infty \leqslant \frac{1}{10\|A^{-1}\|_\infty}. \tag{3.5}$$

Compute $\tilde{A}^{-1}$ and its infinity norm $\|\tilde{A}^{-1}\|_\infty$.

Step 2 (Determine evaluation points). Let T be the number of nonzero terms of f. Set

$$\delta_i = \prod_{j=1}^{s} \min\{|\tilde{a}_{i,j}|, 1\}.$$

Let y_0 be a random rational number such that

$$|y_0| < \min\Big\{\min_{1\leqslant i\leqslant s}\{1/\delta_i\}(4\|\tilde{A}^{-1}\|_\infty (T+1)(d_x+1)M^{d_x}N)^{-1}, 1\Big\}, \tag{3.6}$$

where $N = 2^{d_x^2}(d_x+1)^{d_x/2}\|f\|_\infty^{d_x+1}$. Call a symbolic algorithm to factorize $f(x,y_0)$ over $\mathbb{Q}[x]$ and denote by the factors $\tilde{G}_1(x),\dots,\tilde{G}_r(x)$. If $r = s$ then return $g_1,\dots,g_s$. Set $x_i := y_0/z_i$ for $i = 1,\dots,s$ and $B := (b_{i,j})_{s\times r}$, where $b_{i,j} := \mathrm{Re}(\mathrm{Ln}(\tilde{G}_j(x_i)/x_i^{d_j}))$.

Step 3 (Solve linear system). Compute $U := \tilde{A}^{-1}B = (u_{i,j})_{s\times r}$, let $\mu_{i,j} := \lfloor u_{i,j} + 0.5\rfloor$ and $G_i^{(0)} := \prod_{j=1}^{s} g_j^{\mu_{j,i}}$. Return $G_1^{(0)},\dots,G_r^{(0)}$.

3.2 Analysis of recombination

Lemma 3.4. *The matrix A in Step 1 of Algorithm 3.3 is invertible with probability 1.*

Proof. Denote $g_j(z) = g_j(1,z)$ for $j = 1,\dots,s$. From complex analysis, $\mathrm{Ln} z = \ln|z| + I\cdot \mathrm{Arg}\, z$, where $I = \sqrt{-1}$ and $\mathrm{Arg}\, z$ is the argument of $z \in \mathbb{C}$. Then $a_{i,j} = \ln|g_j(z_i)|$ in Step 1 of Algorithm 3.3. Assume the rank of

$$A = \begin{pmatrix} \ln|g_1(z_1)| & \ln|g_2(z_1)| & \cdots & \ln|g_s(z_1)| \\ \ln|g_1(z_2)| & \ln|g_2(z_2)| & \cdots & \ln|g_s(z_2)| \\ \cdots & \cdots & \ddots & \cdots \\ \ln|g_1(z_s)| & \ln|g_2(z_s)| & \cdots & \ln|g_s(z_s)| \end{pmatrix}$$

is $k < s$ for $z_1,\dots,z_s$ in Step 1 of Algorithm 3.3. Without loss of generality, we assume that the $k\times k$ principal minor is not singular. We now consider the following equation system with respect to $\lambda_j \in \mathbb{R}$,

$$\lambda_1 \ln|g_1(z_i)| + \cdots + \lambda_{s-1}\ln|g_{s-1}(z_i)| = \ln|g_s(z_i)|, \tag{3.7}$$

where $i = 1,\dots,s-1$ and $j = 1,\dots,s-1$. From linear algebra, we know the system (3.7) has at least one solution. Let $\lambda_1,\dots,\lambda_{s-1}$ be an arbitrary solution of the system. Then for $z_1,\dots,z_{s-1}$, we have $|g_1(z_i)^{\lambda_1}\dots g_{s-1}(z_i)^{\lambda_{s-1}}| = |g_s(z_i)|$.

For $z = x + I\,y \in \mathbb{C}$ with x, $y \in \mathbb{R}$, let $C(z) = |c(z)| - |g_s(z)|$, where $c(z) = g_1(z)^{\lambda_1}\cdots g_{s-1}(z)^{\lambda_{s-1}}$. Let $z_s = x_s + I\,y_s$ be the s-th complex number selected in Step 1 of Algorithm 3.3. By the assumption on the rank of A we have $C(z_s) = 0$.

However, $C(z)$ can be considered as a bivariate real function with respect to the real and the imaginary part of z, written $C(x,y)$. We assert that $C(x,y)$ is not a zero function. In fact, if $z_0 = x_0 + I\,y_0$ is a zero point of $c(z)$, then $g_s(z_0)$ is not zero since the initial factors of the input polynomial are co-prime, and hence $C(x_0, y_0) \neq 0$. Note that the selection of z_s is independent of $z_1, \ldots, z_{s-1}$, and the selection of x_s is independent of y_s. Thus for a random $x_s \in \mathbb{R}$, $C(x_s, y)$ is also not a zero function, and hence for a random $y_s \in \mathbb{R}$, $C(x_s, y_s) \neq 0$ holds with probability 1. This contradicts $C(z_s) = 0$. □

Remark 3.5. From the analysis above, the positive integer M is to ensure the matrix A is invertible with probability 1. The proof works for any $M > 0$.

Proposition 3.6. *Algorithm* 3.3 *works correctly if the random* y_0 *satisfies that*

$$f(x, y_0) = \tilde{G}_1(x) \cdots \tilde{G}_r(x),$$

where $G_i(x, y_0) = \tilde{G}_i(x)$ *for* $i = 1, \ldots, r$.

Proof. From Lemma 3.4, the matrix A is invertible with probability 1, and hence the infinity norm of A^{-1} is finite. From the condition in this proposition, we have $G_i(x, y_0) = \tilde{G}_i(x)$ for $i = 1, \ldots, r$.

From another aspect, we have to control the error of the solutions of $AU = B$ such that we can get the correct $\mu_{i,j}$'s by rounding the solutions. For simplicity, we only consider the first initial factor $G_1^{(0)}$. Denote $Au = b$ for the exact case and $\tilde{A} \cdot (u + \delta u) = b + \delta b$ for the perturbed case, where $\tilde{A} = A + \delta A$, u and b are the first column of U and B, respectively, and δA, δu and δb are the perturbations.

We have $\|A^{-1} \cdot \delta A\| < 1$ from (3.5), and hence by [22, Theorem 2.3.4], $\tilde{A}$ is also invertible, which means $\|\tilde{A}^{-1}\|_\infty$ is computable. We now analyze

$$\begin{aligned}\|\delta b\|_\infty &= \max_{1 \leqslant j \leqslant s} \{|\mathrm{Re}(\mathrm{Ln}(G_1(x_j, y_0)/x_j^{d_1})) - \mathrm{Re}(\mathrm{Ln}(G_1^{(0)}(x_j, y_0)/x_j^{d_1}))|\} \\ &= \max_{1 \leqslant j \leqslant s} \{|\ln|G_1(x_j, y_0)/x_j^{d_1}| - \ln|G_1^{(0)}(x_j, y_0)/x_j^{d_1}||\}.\end{aligned}$$

Let us recall (3.3),

$$\frac{G_1}{x^{d_1}} = \frac{G_1^{(0)}}{x^{d_1}} + \sum_{k=1}^{k_1} \frac{\Delta G_1^{(k)}}{x^{d_1}} = G_1^{(0)}(w) + g_1^{(1)}(w) \cdot y + \cdots + g_1^{(k_1)}(w) \cdot y^{k_1}.$$

Denote $G_1(x_j, y_0)/x_j^{d_1} = G_1^{(0)}(z_j) + \bar{G}_1(z_j, y_0) \cdot y_0$, where $\bar{G}_1(z_j, y_0) = \sum_{k=1}^{k_1} g_1^{(k)}(z_j) \cdot y_0^{k-1}$. So we have

$$\begin{aligned}\|\delta b\|_\infty &= \max_{1 \leqslant j \leqslant s} \left\{\left|\ln\left|\frac{G_1^{(0)}(z_j) + \bar{G}_1(z_j, y_0) \cdot y_0}{G_1^{(0)}(z_j)}\right|\right|\right\} \\ &= \max_{1 \leqslant j \leqslant s} \left\{\left|\ln\left|1 + \frac{\bar{G}_1(z_j, y_0) \cdot y_0}{G_1^{(0)}(z_j)}\right|\right|\right\} \\ &\leqslant \max_{1 \leqslant j \leqslant s} \left\{\ln\left(1 + \left|\frac{\bar{G}_1(z_j, y_0) \cdot y_0}{G_1^{(0)}(z_j)}\right|\right)\right\} \\ &\leqslant \max_{1 \leqslant j \leqslant s} \left\{\left|\frac{\bar{G}_1(z_j, y_0) \cdot y_0}{G_1^{(0)}(z_j)}\right|\right\} \\ &\leqslant \max_{1 \leqslant j \leqslant s} \left\{|\bar{G}_1(z_j, y_0) \cdot y_0| \cdot \min_{1 \leqslant i \leqslant s} \{\delta_i\}\right\} \\ &\leqslant (T+1) \cdot (d_x + 1) \cdot M^{d_x} \cdot N \cdot \delta_1 \cdot |y_0|.\end{aligned}$$

The first inequality is from $|1 + z| \leqslant 1 + |z|$ for $z \in \mathbb{C}$; the second inequality is from $\ln(1 + x) \leqslant x$ for $x > -1$; the third one is from $|G_1^{(0)}(z_j)| \geqslant \delta_1 \geqslant \min_{1 \leqslant i \leqslant s} \{\delta_i\}$ and $G_1^{(0)} = \prod_{j=1}^{s} g_j^{\mu_{j,1}}$. Additionally, for each monomial $z_j^a y_0^b$ in $\bar{G}_1(z_j, y_0)$, we have $|z_j^a y_0^b| \leqslant |z_j^{d_1}| \leqslant M^{d_x}$ since $|y_0| < 1$ and $|z_j| \leqslant M$. From Remark 2.3 and (2.9), we have $\|G_1\|_\infty \leqslant N$, so that the last inequality is from the number of nonzero terms of factors is not more than $(T+1) \cdot (d_x + 1)$, by Corollary 2.11. Therefore if y_0 satisfies (3.6), we have

$$\|\tilde{A}^{-1}\|_\infty \cdot \|\delta b\|_\infty < \frac{1}{4}. \tag{3.8}$$

Furthermore, from [22, Theorem 2.3.4] and (3.5),

$$\begin{aligned}\|\tilde{A}^{-1}\|_\infty \cdot \|\delta A\|_\infty &\leqslant \left(\|A^{-1}\|_\infty + \|\delta A\|_\infty \frac{\|A^{-1}\|_\infty^2}{1-\|A^{-1}\delta A\|_\infty}\right)\cdot \|\delta A\|_\infty \\ &\leqslant 2(\|A^{-1}\|_\infty \|\delta A\|_\infty)^2 + (\|A^{-1}\|_\infty \|\delta A\|_\infty) < \frac{1}{4}.\end{aligned} \tag{3.9}$$

Now we have $\|\delta u\|_\infty = \|\tilde{A}^{-1}(\delta b + \delta A \cdot u)\|_\infty \leqslant \|\tilde{A}^{-1}\|_\infty(\|\delta b\|_\infty + \|\delta A\|_\infty) < 1/2$, where the first inequality is from $\|u\|_\infty = 1$, and the second one is from (3.8) and (3.9). This completes the proof. □

Proposition 3.7. *Algorithm* 3.3 *requires* $\mathcal{O}(s^\omega + \boldsymbol{M}(d_x)\log s)$ *operations to reduce the combination to the computation of univariate polynomial factorization over* $\mathbb{Q}$ *whose degree is at most* d_x.

Proof. For $i \leqslant s$, we first evaluate $g_j(1, z_i)$ by fast multipoint evaluation. This requires $\mathcal{O}(\boldsymbol{M}(d_j)\log s)$ operations, and hence constructing A has cost $\mathcal{O}(\boldsymbol{M}(d_1)\log s + \cdots + \boldsymbol{M}(d_s)\log s) \subset \mathcal{O}(\boldsymbol{M}(d_x)\log s)$ by the super-additivity of $\boldsymbol{M}$. Similarly, constructing B requires the same operations. Computing A^{-1} and multiplying A^{-1} and B have cost $\mathcal{O}(s^\omega)$. By Hypothesis ($\mathrm{H_b}$), f is monic, and hence the degree of $f(x, y_0)$ is d_x. □

Our combination algorithm is in fact based on the idea of "linearization", i.e., reducing the problem of combination to solving a system of linear equations. The basic idea is the use of logarithms, which has been successfully applied to polynomial factorization, such as trace recombination [23, 52–54] and logarithmic derivative recombination [5, 8, 39, 43]. However, we use a novel strategy to construct the linear system after taking logarithm. We choose some different points to evaluate the equation (3.2). Although we only get some approximate equations, we analyze the error control. If we choose the evaluation points satisfying the error controlling conditions, we can solve correctly for these 0–1 vectors. This is different from existing methods in literature.

4 Main algorithm and analysis

In this section, we describe our factorization algorithm, give a typical example to better illustrate the key steps of the algorithm, and analyze its complexity.

Algorithm 4.1 (BiFactor). Input: A bivariate polynomial $f \in \mathbb{Z}[x, y]$ satisfying Hypothesis (H). Output: A factorization of f over $\mathbb{Z}$.

Step 1 (Determine Newton polynomial). Compute $L = (l_1, l_2)$ and $R = (r_1, r_2)$ the two endpoints of the Newton line, $f^{(0)}(x, y)$ the Newton polynomial of f. Set $d_x := \deg_x f$ and $d_y := \deg_y f$.

Step 2 (Factorize Newton polynomial). If $l_2 = r_2 = 0$ then $f^{(0)}$ is a univariate polynomial, else $f^{(0)}$ is a homogeneous polynomial of total degree $\frac{r_2 l_1 - r_1 l_2}{r_2 - l_2}$ with respect to x and $\hat{y}$, where $\hat{y} = y^{-\frac{l_2 - r_2}{l_1 - r_1}}$. Let $\hat{d}$ and $\hat{\delta}$ be nonnegative integers such that $\frac{\hat{\delta}}{\hat{d}} = |\frac{l_2 - r_2}{l_1 - r_1}|$ is an irreducible fraction. If $|\frac{l_2 - r_2}{l_1 - r_1}| = 0$ then let $\hat{\delta} := 0$ and $\hat{d} := 1$. Let $I_k := \{x^j \hat{y}^{d_x - j} \cdot y^{k/\hat{d}} : j = 0, \ldots, d_x\}$. Factorize $f^{(0)}(x, 1)$ using a univariate polynomial factorization algorithm and then get the factorization in $\mathbb{Z}[x, \hat{y}]$: $f^{(0)}(x, y) = g_1(x, \hat{y}) \cdots g_s(x, \hat{y})$.

Step 3 (Combine). Call Algorithm 3.3 with $g_1(x, \hat{y}), \ldots, g_s(x, \hat{y})$ and f. Let the combined factors be $G_1(x, y), \ldots, G_r(x, y)$. Let $\Delta f := f - f^{(0)}$. If $\Delta f = 0$ then return $G_1, \ldots, G_r$.

Step 4 (Moses-Yun). Call Algorithm 2.2 feeding with initial factors to compute $W_i^{(l)}(x, \hat{y})$.

Step 5 (Determine moduli). Let k be the maximum integer such that $\Delta f \equiv 0 \mod I_k$ and use I_{k+1} as the modulus of the generalized Hensel lifting.

Step 6 (Lifting). Compute $\Delta f^{(k)} := \Delta f \mod I_{k+1}$. For i from 1 to r, let

$$\Delta G_i(x, y) := \sum_{l=0}^{d_x - 1} W_i^{(l)} c_l^{(k)}$$

and update $G_i := G_i + \Delta G_i$.

Step 7 (Finished). Set $\Delta f := f - G_1 \cdots G_r$. If $\Delta f = 0$ then return $G_1, \dots, G_r$, else go to Step 5.

Remark 4.2. In practice, we use one technique to deal with the case that the evaluation point is not good: If Δf in the step 7 of Algorithm 4.1 includes some terms that do not appear in the input polynomial $f(x, y)$, we then stop lifting and choose another evaluation point for redoing recombination. Thus, if the algorithm returns a factorization, then it must be the irreducible factors of the input bivariate polynomial. In fact, the number of irreducible factors of the image of the evaluation homomorphism will never be less than the exact number of irreducible factors of the original polynomial. So, a bad evaluation must imply that r, in Step 3 of Algorithm 4.1, is strictly larger than the exact number of irreducible factors, and then during the lifting step, there will be at least one extra term that does not appear in the input bivariate polynomial. This phenomenon is easy to check in practice. Once this happens, we choose another evaluation point and redo recombination.

Here, we use the following simple example to illustrate the main steps of our factorization algorithm.

Example 4.3. We consider again the polynomial in Example 1.6: $f = x^8 - 3\,x^4y^2 + 5\,x^4y^5 - 4\,y^4 + 5\,y^7 + 2\,y^3x^4 - 8\,y^5 + 10\,y^8$. The Newton polynomial of f is

$$f^{(0)}(x, y) = x^8 - 3\,x^4y^2 - 4\,y^4 = (x^2 + \hat{y}^2)(x^2 + 2\,\hat{y}^2)(x^2 - 2\,\hat{y}^2) \triangleq g_1g_2g_3,$$

where $\hat{\delta} = -1$, $\hat{d} = 2$ and hence $\hat{y} = y^{\frac{1}{2}}$. Thus

$$I_0 = \{y^4, xy^{\frac{7}{2}}, x^2y^3, x^3y^{\frac{5}{2}}, x^4y^2, x^5y^{\frac{3}{2}}, x^6y, x^7y^{\frac{1}{2}}, x^8\}.$$

In the combination stage, we construct a linear system by specialization. First, letting $z_1 = \frac{2}{5}\mathrm{e}^{\frac{\pi}{11}I}$, $z_2 = \frac{7}{5}\mathrm{e}^{\frac{27\pi}{55}I}$, $z_3 = \mathrm{e}^{\frac{49\pi}{55}I}$ and $y_0 = 1/223$, we can get the following linear system

$$\begin{pmatrix} \frac{200466}{6353} & \frac{355429}{21663} & \frac{120385}{7398} \\ \frac{88198}{3365} & \frac{47978}{4353} & \frac{185008}{12743} \\ \frac{120647}{4175} & \frac{79463}{5347} & \frac{100672}{7121} \end{pmatrix} \begin{pmatrix} \mu_{1,1} & \mu_{1,2} \\ \mu_{2,1} & \mu_{2,2} \\ \mu_{3,1} & \mu_{3,2} \end{pmatrix} = \begin{pmatrix} \frac{272848}{8645} & \frac{117484}{3595} \\ \frac{68722}{2621} & \frac{296190}{11597} \\ \frac{615298}{21291} & \frac{60868}{2099} \end{pmatrix}.$$

Solving this linear system and rounding the solutions give two 0-1 vectors $(1, 0, 0)$ and $(0, 1, 1)$, which means the initial factors are $G_1^{(0)} = g_1 = x^4 + y^2$ and $G_2^{(0)} = g_2g_3 = x^4 - 4\,y^2$. Now, $\Delta f = 5\,x^4y^5 + 5\,y^7 + 2\,y^3x^4 - 8\,y^5 + 10\,y^8$. Since only x^0 and x^4 appear in Δf, we only need to compute $W_i^{(l)}$ for $l = 0, 4$ instead of l from 0 to 7 $(= d_x - 1)$:

$$\begin{aligned} W_1^{(0)} &= \frac{1}{5}\,y^2, \quad W_2^{(0)} = -\frac{1}{5}\,y^2, \\ W_1^{(4)} &= \frac{4}{5}\,y^2, \quad W_2^{(4)} = \frac{1}{5}\,y^2. \end{aligned}$$

Since $\Delta f = x^4y^2 \cdot (5y^{6/\hat{d}}) + y^4 \cdot (5y^{6/\hat{d}}) + x^4y^2 \cdot (2y^{2/\hat{d}}) + y^4 \cdot (-8y^{2/\hat{d}}) + y^4 \cdot (10y^{8/\hat{d}})$, we have $k = 2$ is the maximum integer satisfying $\Delta f \equiv 0 \bmod I_k$. Then compute $\Delta f^{(2)} \equiv \Delta f \bmod I_3 \equiv x^4y^2 \cdot (2y^{2/\hat{d}}) + y^4 \cdot (-8y^{2/\hat{d}})$. Hence, $c_7^{(2)} = c_6^{(2)} = c_5^{(2)} = 0$, $c_4^{(2)} = 2y$, $c_3^{(2)} = c_2^{(2)} = c_1^{(2)} = 0$, $c_0^{(14)} = -8y$. Thus, we have

$$\begin{aligned} G_1 &:= G_1 + \Delta G_1 = G_1 + (W_1^{(0)}c_0^{(2)} + W_1^{(4)}c_4^{(2)}) = x^4 + 2\,x^2y + y^2 + 2\,y^3, \\ G_2 &:= G_2 + \Delta G_2 = G_2 + (W_2^{(0)}c_0^{(2)} + W_2^{(4)}c_4^{(2)}) = x^4 - 4\,y^2. \end{aligned}$$

Update $\Delta f = 5\,x^4y^5 + 10\,x^2y^6 + 5\,y^7 + 10\,y^8 \neq 0$. Then go to Step 5. After one more lifting step, the algorithm returns the irreducible factors of f in $\mathbb{Z}[x, y]$: $G_1 = x^4 + 2\,x^2y + y^2 + 2\,y^3$ and $G_2 = x^4 - 4\,y^2 + 5\,y^5$.

Remark 4.4. Note that the condition for y_0 in (3.6) is a little bit heavy. But it is just sufficient to make sure that $\|\delta u\| < \frac{1}{2}$, and may not be necessary in practice. When implementing, one can try y_0 with small size first, and if necessary, then use y_0 with larger size, as in the above example.

From this example, we get an intuition on how the BiFactor algorithm takes advantage of sparsity. On the one hand, it may not always hold that all degrees from 0 to d_x with respect to x appear for a generic sparse polynomial. Thus we need not to compute all $W_i^{(l)}$, but only to compute those $W_i^{(l)}$ such that x^l appears in Δf. If the number of nonzero terms of the input polynomial T is less than d_x, then Step 4 requires at most $\mathcal{O}\left(r \log d_x \boldsymbol{M}(d_x) + T d_x\right)$ operations, instead of $\mathcal{O}\left(r \log d_x \boldsymbol{M}(d_x) + d_x^2\right)$ operations in Proposition 2.4. On the other hand, when the polynomial is dense, each lifting stage lifts the factors only by $y^{1/\hat{d}}$. However, for a generic sparse polynomial, it usually holds that there exists no term lying in I_k for some k. That's to say for those k we have $\Delta f^{(k)} = 0$, which means $\Delta G_i^{(k)} = 0$ for $i = 1, \ldots, r$. We thus can directly handle with the smallest k such that $\Delta f^{(k)} \neq 0$.

Proposition 4.5. *Assume that there exists an effective version of the Hilbert irreducibility theorem for bivariate polynomials over* $\mathbb{Z}$. *Given a bivariate polynomial* f *over* $\mathbb{Q}$ *satisfying Hypothesis* (H), *the BiFactor algorithm reduces the computation of the irreducible factors of* f *over* $\mathbb{Q}$ *to factoring univariate polynomials with degree* d_x *over* $\mathbb{Q}$ *in* $\mathcal{O}(Trd_x + T \log r\boldsymbol{M}(d_x d_y) + d_x^2 + s^\omega + r \log d_x \boldsymbol{M}(d_x))$ *or* $\tilde{\mathcal{O}}(Trd_x + Td_x d_y + d_x^2 + s^\omega)$ *arithmetic operations in* $\mathbb{Q}$, *where* T *is the number of non-zero terms of* f, d_x *and* d_y *are the degree of* f *with respect to* x *and* y *respectively,* r *is the number of irreducible factors of* f, *and* s *is the number of factors in* $\mathbb{Z}[x, \hat{y}]$ *of the Newton polynomial of* f.

Proof. By the assumption, Proposition 3.6 holds. Hypothesis ($\mathrm{H_a}$) implies that $d_i = \deg_x G_i^{(0)} \geqslant 1$. Then the correctness of Algorithm 4.1 follows from Propositions 2.13, 3.1 and 3.6. We consider its complexity in the following.

From Definition 1.2, it costs at most $\mathcal{O}(T)$ operations to determine the Newton line of f. By Corollary 3.1 and Proposition 3.7, it requires $\mathcal{O}(s^\omega + \boldsymbol{M}(d_x) \log s)$ operations to reduce Steps 2 and 3 to factoring univariate polynomials whose degree is at most d_x. By Proposition 2.4, it has cost $\mathcal{O}\left(r \log d_x \boldsymbol{M}(d_x) + d_x^2\right)$ to compute the Moses-Yun interpolation polynomials, where r is the number of irreducible factors of f over $\mathbb{Q}$. Step 6 has cost $\mathcal{O}(rd_x)$ to compute ΔG_i by Proposition 2.13, and Step 7 has cost $\mathcal{O}(\boldsymbol{M}(d_x d_y))$ according to the proof of Proposition 2.13. Furthermore, the number of lifting is not larger than T since the number of non-zero terms of Δf decreases at least by 1 in each lifting. At last Step 5 totally has cost $\mathcal{O}(T)$.

Therefore, Algorithm 4.1 needs $\mathcal{O}(Trd_x + T\boldsymbol{M}(d_x d_y) + d_x^2 + s^\omega + r \log d_x \boldsymbol{M}(d_x))$ or $\tilde{\mathcal{O}}(Trd_x + Td_x d_y + d_x^2 + s^\omega)$ operations to reduce the computation of a bivariate polynomial factorization under Hypotheses (H) to factoring univariate polynomials with degree at most d_x. □

Proof of Theorem 1.7. Without loss of generality we can assume that $d_x \leqslant d_y$. Then, by Proposition 4.5 the computation of the irreducible factorization of f satisfying Hypothesis (H) reduces to the univariate polynomials factorization over $\mathbb{Q}$ whose degree is at most d_x, plus $\tilde{\mathcal{O}}(Trd_x + Td_x d_y + d_x^2 + s^\omega) \subset \tilde{\mathcal{O}}(Td_x d_y + s^\omega)$ arithmetic operations, where $r \leqslant d_x$ and $2 < \omega \leqslant 3$ are used. □

Again, we note that the main algorithm is not deterministic, since its validity depends on a good specialization y_0. Actually, if one has such a good choice, one can use directly the classical Hensel lifting along the fiber $y = y_0$, and its total complexity of the reduction from bivariate polynomial factorization to univariate case would be in $\tilde{\mathcal{O}}(d_x d_y)$ rather than in $\tilde{\mathcal{O}}(Td_x d_y + s^\omega)$. But, as explained before, the classical Hensel lifting does not use the sparsity of the input polynomial, which usually leads to expression swell. Moreover, the experimental results in next section show the main algorithm in this article is efficient in practice.

5 Experimental results

We have implemented the BiFactor algorithm in the computer algebra system Maple. In this section, we give some details of our implementation and report some experimental results which show the efficiency of our algorithm. All the test polynomials and the Maple package are available from http://sites.google.com/site/jingweichen84.

Table 1 Experimental results when recombination happening

no.	d_x	d_y	T	s	$\text{time}_{\text{Maple}}$	$\text{time}_{\text{BiFactor}}$
1	16	27	56	5	0.064	0.113
2	32	55	88	6	0.266	0.160
3	64	113	135	7	3.273	0.417
4	128	230	194	8	36.071	0.663
5	256	478	253	9	685.706	2.433

Table 2 Experimental results for Maple's factor and BiFactor

no.	d	d_x	d_y	T	$\text{time}_{\text{Maple}}$	$\text{time}_{\text{BiFactor}}$
1	50	46	29	135	0.281	0.157
2	100	95	89	143	2.125	0.109
3	200	156	152	144	7.578	0.235
4	400	335	305	144	87.297	0.218
5	800	580	595	144	943.094	0.422
6	50	39	50	705	0.532	0.437
7	100	100	65	827	4.282	1.468
8	200	193	179	890	50.750	2.156
9	400	306	302	746	208.515	2.110
10	800	784	613	912	10301.844	4.578

It is well known that there exists the phenomenon of intermediate expression swell in many applications, for example computing gcd of two polynomials with rational coefficients. Thus we use modular arithmetic to compute Moses-Yun polynomials. As seen from previous sections, the total efficiency of our algorithm heavily depends on the factorization of univariate polynomials over $\mathbb{Q}$. For sparse case, there exists many efficient algorithm, such as [14, 45, 46]. In our implementation, we call Maple's built-in function `factor` to compute all univariate polynomials. Moreover, for a polynomial f which does not satisfy Hypothesis ($\mathrm{H_c}$), we introduce the following kind of affine transformations $\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right) := \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)\left(\begin{smallmatrix} X \\ Y \end{smallmatrix}\right)$, where $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ is an element of the group generated by $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, $\left(\begin{smallmatrix} -1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$ under matrix multiplication, and then compute the factorization of $F(X,Y) = X^mY^nf(X,Y)$ if $F(X,Y)$ satisfies Hypothesis ($\mathrm{H_c}$), where m and n are such integers that $F(X,Y)$ satisfies Hypothesis ($\mathrm{H_a}$) and ($\mathrm{H_b}$). Then the factorization of $f(x,y)$ can be inferred from that of $F(X,Y)$. Even though the affine transformation can be used to extend the applicability of the BiFactor algorithm, the algorithm remains incomplete. The reason is one can construct some polynomials, such as $f = (xy + x^3 + x^2y^5 + y^4)(xy + x^4 + x^2y^6 + y^3)$, which do not satisfy Hypothesis ($\mathrm{H_c}$) under any above affine transformation.

We now report some experimental results. All of these tests were run on an AMD Athlon$^{\text{TM}}$ 7750 processor (2.70 GHz) with 2GB memory. In the following tables, $\text{time}_{\text{Maple}}$ represents the running time of Maple's built-in function factor and $\text{time}_{\text{BiFactor}}$ represents the running time of the BiFactor algorithm. Time is shown in seconds. All univariate polynomial factorizations are computed directly by calling Maple's factor.

In Table 1, we investigate the efficiency of the BiFactor algorithm when the combination step is designed to happen, i.e., $s > r = 2$. Each polynomial in Table 1 has only two irreducible factors, and its Newton polynomial is $x^{d_x} - y^{d_x}$ which can be factorized as s factors in $\mathbb{Z}[x, \hat{y}]$. Hence we have to go into the combination step. We note that the factorization of the Newton polynomial is easy to compute, however, we have to factorize a univariate polynomial with degree d_x during the combination stage. In spite of this fact, we can learn from Table 1 that the BiFactor algorithm is efficient in this case.

The purpose of the trials in Table 2 is to compare the performances of Maple's built-in function factor with the BiFactor algorithm for some random polynomials. All test polynomials in Table 2 are of total degree d and constructed by multiplying two random polynomials with total degree $d/2$. Both of them

Table 3 Experiments for polynomials with more than two irreducible factors

r	d	d_x	d_y	T	$\text{time}_{\text{Maple}}$	$\text{time}_{\text{BiFactor}}$
3	94	90	49	54	0.562	0.483
4	167	160	103	189	3.931	0.811
5	255	250	102	623	16.786	6.271
6	376	360	231	8336	151.711	56.239

are generated from the following Maple code. The first two terms are to ensure the input polynomial is of degree d and has no monomial factors.

[> $a := \text{rand}(0..d/2)()$;

[> $\text{rand}(1..100)()*x\hat{}a*y\hat{}(d/2-a) + \text{rand}(1..100)()$ $+\text{randpoly}([x,y], \text{terms} = t, \text{degree} = d/2)$.

The first 5 polynomials are generated with $t := 10$, i.e., each factor of the polynomials has $10 \sim 12$ non-zero terms. The last 5 polynomials are generated with $t := 4$ for the first factor and $t := 150$ for the second factor. Evidently, our algorithm outperforms the Maple built-in factor.

The last test set in Table 3 is to investigate the performance of BiFactor when the input polynomial has more than two irreducible factors. The number of irreducible factors ranges from 3 to 6. Accordingly, the number of terms ranges from 54 to 8336. For these tests, BiFactor seems slower than that in Tables 1 and 2, because the test polynomials are denser than those in Tables 1 and 2. However, it is still much faster than the Maple built-in function factor.

6 Conclusion

In this article, we present an algorithm, BiFactor, which factorizes bivariate polynomials over $\mathbb{Q}$. It can be seen as a polytope method. Unlike the traditional Hensel lifting based methods, it takes advantage of sparsity because of the generalized Hensel lifting scheme. Another feature is that a novel recombination strategy for computing the initial factors is used before the lifting stage. Although the validity of the algorithm is based on Hypothesis (H), both theoretic analysis and experimental data show its efficiency. Unfortunately, it is non-deterministic. We cannot even give a success probability because of lacking a proven randomized Hilbert irreducibility theorem for bivariate polynomials. It is beyond the scope of this article.

Our future work is to give a deterministic or probabilistic recombination methods for computing the initial factors. The future directions also include: to weaken the Hypothesis (H_c), to analyze the bit-complexity of our algorithm when using floating-point arithmetic operations, and to generalize the method to polynomials with variables more than two.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 91118001 and 11170153), National Key Basic Research Project of China (Grant No. 2011CB302400) and Chongqing Science and Technology Commission Project (Grant No. cstc2013jjys40001). The authors thank anonymous reviewers for many helpful suggestions, which improved this article much.

References

1 Abu Salem F. Factorisation Algorithms for Univariate and Bivariate Polynomials over Finite Fields. PhD thesis. Oxford: Oxford University Computing Laboratory, 2004

2 Abu Salem F. An efficient sparse adaptation of the polytope method over $\mathbb{F}_p$ and a record-high binary bivariate factorisation. J Symbolic Comput, 2008, 43: 311–341

3 Abu Salem F, Gao S, Lauder A G B. Factoring polynomials via polytopes. In: Proceedings of the 2004 international symposium on Symbolic and algebraic computation. Santander: ACM, 2004, 4–11

4 Avendaño M, Krick T, Sombra M. Factoring bivariate sparse (lacunary) polynomials. J Complexity, 2007, 23: 193–216

5 Belabas K, van Hoeij M, Klüners J, et al. Factoring polynomials over global fields. J Théorie Nombres Bordeaux, 21: 15–39, 2009

6 Bernardin L. On bivariate Hensel and its parallelization. In: Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation. New York: ACM, 96–100, 1998

7 Berthomieu J, Lecerf G. Reduction of bivariate polynomials from convex-dense to dense, with application to factorizations. Math Comput, 2012, 81: 1799–1821

8 Bostan A, Lecerf G, Salvy B, et al. Complexity issues in bivariate polynomial factorization. In: Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation. Santander: ACM, 2004, 42–49

9 Bürgisser P, Clausen M, Shokrollahi M A. Algebraic Complexity Theory. New York: Springer, 1997

10 Chattopadhyay A, Grenet B, Koiran P, et al. Factoring bivariate lacunary polynomials without heights. In: Proceedings of the 2013 International Symposium on Symbolic and Algebraic Computation. Boston: ACM, 2013, 141–148

11 Chéze G, Lecerf G. Lifting and recombination techniques for absolute factorization. J Complexity, 2007, 23: 380–420

12 Cox D, Little J, O'Shea D. Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3rd ed. New York: Springer, 2007

13 Cucker F, Koiran P, Smale S. A polynomial time algorithm for diophantine equations in one variable. J Symbolic Comput, 1999, 27: 21–29

14 Davenport J. Factorisation of sparse polynomials. In: van Hulzen J, ed. Computer Algebra, vol. 162. Lecture Notes in Computer Science. Berlin: Springer, 1983, 214–224

15 Gao S. Absolute irreducibility of polynomials via Newton polytopes. J Algebra, 2001, 237: 501–520

16 Gao S. Factoring multivariate polynomials via partial differential equations. Math Comput, 2003, 72: 801–822

17 Von zur Gathen J. Factoring sparse multivariate polynomials. In: 24th Annual Symposium on Foundations of Computer Science. Tucson: IEEE, 1983, 172–179

18 Von zur Gathen J. Who was who in polynomial factorization. In: Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation. Genova: ACM, 2006, 1–2

19 Von zur Gathen J, Gerhard J. Modern Computer Algebra. London: Cambridge University Press, 1999

20 Von zur Gathen J, Kaltofen E. Factoring sparse multivariate polynomials. J Comput Syst Sci, 1985, 31: 265–287

21 Geddes K O, Czapor S R, Labahn G. Algorithms for Computer Algebra. Boston: Kluwer Academic Publishers, 1992

22 Golub G H, van Loan C. Matrix Computations, 3rd ed. London: The John Hopkins University Press, 1996

23 Van Hoeij M. Factoring polynomials and the knapsack problem. J Number Theory, 2002, 95: 167–189

24 Hoppen C, Rodrigues V M, Trevisan V. A note on Gao's algorithm for polynomial factorization. Theo Comput Sci, 2011, 412: 1508–1522

25 Inaba D. Factorization of multivariate polynomials by extended Hensel construction. ACM SIGSAM Bulletin, 2005, 39: 2–14

26 Inaba D, Sasaki T. A numerical study of extended Hensel series. In: Proceedings of the 2007 International Workshop on Symbolic-numeric Computation. London-Canada: ACM, 2007, 103–109

27 Iwami M. Analytic factorization of the multivariate polynomial. In: Proceedings of the 6th International Workshop on Computer Algebra in Scientific Computing. Passau, 2003, 213–226

28 Iwami M. Extension of expansion base algorithm to multivariate analytic factorization. In: Proceedings of the 7th International Workshop on Computer Algebra in Scientific Computing. Petersburg, 2004, 269–281

29 Kaltofen E. A polynomial reduction from multivariate to bivariate integral polynomial factorization. In: Proceedings of the 14th Annual ACM Symposium on Theory of Computing. New York: ACM, 1982, 261–266

30 Kaltofen E. A polynomial-time reduction from bivariate to univariate integral polynomial factorization. In: 23rd Annual Symposium on Foundations of Computer Science. Chicago: IEEE, 1982, 57–64

31 Kaltofen E. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. SIAM J Comput, 1985, 14: 469–489

32 Kaltofen E. Sparse Hensel lifting. In: Caviness B, ed. EUROCAL '85. Lecture Notes in Computer Science, vol. 204. New York: Springer, 1985, 4–17

33 Kaltofen E. Polynomial factorization 1982–1986. In: Computers and Mathematics. Lecture Notes in Pure and Applied Mathematics, vol. 125. New York: Springer, 1990, 285–309

34 Kaltofen E. Polynomial factorization 1987–1991. In: Simon I, ed. LATIN '92, Lecture Notes in Computer Science, vol. 583. New York: Springer, 1992, 294–313

35 Kaltofen E. Polynomial factorization: A success story. In: Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation. Philadelphia: ACM, 2003, 3–4

36 Kaltofen E, Koiran P. On the complexity of factoring bivariate supersparse (lacunary) polynomials. In: Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation. Beijing: ACM, 2005, 208–215

37 Kaltofen E, Koiran P. Finding small degree factors of multivariate supersparse (lacunary) polynomials over algebraic number fields. In: Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation. Genoa: ACM, 2006, 162–168

38 Kaltofen E, Lecerf G. Factorization of multivairate polynomials. In: Mullen G L, Panario D, eds. Handbook of Finite Fields. Boca Raton: CRC Press, 2013, 382–392

39 Klüners J. The van Hoeij algorithm for factoring polynomials. In: Nguyen P Q, Vallée B, eds. The LLL Algorithm:

Survey and Applications. New York: Springer, 2010, 283–291

40 Lecerf G. Sharp precision in Hensel lifting for bivariate polynomial factorization. Math Comput, 2006, 75: 921–934

41 Lecerf G. Improved dense multivariate polynomial factorization algorithms. J Symbolic Comput, 2007, 42: 477–494

42 Lecerf G. Fast separable factorization and applications. Appl Algebra Engrg Comm Comput, 2008, 19: 135–160

43 Lecerf G. New recombination algorithms for bivariate polynomial factorization based on Hensel lifting. Appl Algebra Engrg Comm Comput, 2010, 21: 151–176

44 Lenstra A K, Lenstra H W, Lovász L. Factoring polynomials with rational coefficients. Math Ann, 1982, 261: 515–534

45 Lenstra H W. Finding small degree factors of lacunary polynomials. Number Theory, 1999, 1: 267–276

46 Lenstra H W. On the factorization of lacunary polynomials. Number Theory, 1999, 1: 277–291

47 Musser D R. Multivariate polynomial factorization. J ACM, 1975, 22: 291–308

48 Press W H, Teukolsky S A, Vetterling W T, et al. Numerical Recipes: The Art of Scientific Computing, 3rd ed. New York: Cambridge University Press, 2007

49 Sasaki T, Inaba D. Hensel construction of $f(x, u_1, \dots, u_l)$, $l \geqslant 2$, at a singular point and its applications. ACM SIGSAM Bulletin, 2000, 34: 9–17

50 Sasaki T, Inaba D. Convergence and many-valuedness of Hensel series near the expansion point. In: Proceedings of the 2009 Conference on Symbolic Numeric Computation. Kyoto: ACM, 2009, 159–168

51 Sasaki T, Kako K. Solving multivariate algebraic equation by Hensel construction. Japan J Indust Appl Math, 1999, 16: 257–285

52 Sasaki T, Saito T, Hilano T. Analysis of approximate factorization algorithm I. Japan J Indust Appl Math, 1992, 9: 351–368

53 Sasaki T, Sasaki M. A unified method for multivariate polynomial factorizations. Japan J Indust Appl Math, 1993, 10: 21–39

54 Sasaki T, Suzuki M, Kolář M, et al. Approximate factorization of multivariate polynomials and absolute irreducibility testing. Japan J Indust Appl Math, 1991, 8: 357–375

55 Wang P S. Preserving sparseness in multivariate polynominal factorization. In: Proceedings of 1977 MACSYMA Users' Conference (NASA). Boston: MIT, 1977, 55–64

56 Wang P S. An improved multivariate polynomial factoring algorithm. Math Comput, 1978, 32: 1215–1231

57 Wang P S, Rothschild L P. Factoring multivariate polynomials over the integers. Math Comput, 1975, 29: 935–950

58 Weimann M. A lifting and recombination algorithm for rational factorization of sparse polynomials. J Complexity, 2010, 26: 608–628

59 Weimann M. Factoring bivariate polynomials using adjoints. J Symbolic Comput, 2013, 58: 77–98

60 Wu W, Chen J, Feng Y. An efficient algorithm to factorize sparse bivariate polynomials over the rationals. ACM Commun Comput Algebra, 2012, 46: 125–126

61 Zippel R. Probabilistic algorithms for sparse polynomials. In: Ng E, ed. Symbolic and Algebraic Computation. Lecture Notes in Computer Science, vol. 72. London: Springer, 1979, 216–226

62 Zippel R. Newton's iteration and the sparse Hensel algorithm (extended abstract). In: Proceedings of the 4th ACM Symposium on Symbolic and Algebraic Computation. Snowbird: ACM, 1981, 68–72