Efficient Integer-Linear Decomposition of Multivariate Polynomials

Mark Giesbrecht
Symbolic Computation Group, University of Waterloo
Waterloo, Ontario, Canada
mwg@uwaterloo.ca

Hui Huang
Symbolic Computation Group, University of Waterloo
Waterloo, Ontario, Canada
h2huang@uwaterloo.ca

George Labahn
Symbolic Computation Group, University of Waterloo
Waterloo, Ontario, Canada
glabahn@uwaterloo.ca

Eugene Zima
Physics and Computer Science,
Wilfrid Laurier University
Waterloo, Ontario, Canada
ezima@wlu.ca

ABSTRACT

We present a new algorithm for the computation of the integer-linear decomposition of a multivariate polynomial. Such a decomposition is used in Ore-Sato theory and discrete creative telescoping, for example to detect applicability of Zeilberger's algorithm to a hypergeometric term. Our algorithm is quite straightforward, requiring only basic polynomial arithmetic along with efficient rational root finding. We present complete complexity analyses for both our and previous algorithms in the case of bivariate integer polynomials, and show that our method has a better theoretical performance. We also provide a Maple implementation which shows that our method is faster in practice than previous algorithms.

1 INTRODUCTION

An irreducible polynomial $p \in \mathrm{R}[x_1, \ldots, x_n]$ over a ring R of characteristic zero is said to be *integer-linear* if there exists a univariate polynomial $P \in \mathrm{R}[z]$ and integers $\lambda_1, \ldots, \lambda_n$ such that

$$p(x_1, \ldots, x_n) = P(\lambda_1 x_1 + \cdots + \lambda_n x_n). \tag{1}$$

Since a common factor of the integers $\lambda_1, \ldots, \lambda_n$ can be pulled out and absorbed into P, we assume that the integers $\lambda_1, \ldots, \lambda_n$ have no common divisor, that the last integer $\lambda_n \geq 0$ and that $\lambda_i = 0$ whenever $\deg_{x_i}(p) = 0$. Such an n-tuple $(\lambda_1, \ldots, \lambda_n)$ is unique and is called the *integer-linear type* of p. A polynomial in $\mathrm{R}[x_1, \ldots, x_n]$ is called *integer-linear* (over R) if all its irreducible factors are integer-linear, possibly with different integer-linear types. We similarly define the notion of the *integer-linear decomposition* of a polynomial p which factors into irreducible integer-linear or non-integer-linear polynomials and groups irreducible factors having common types.

The notion of integer-linear polynomials appears frequently in symbolic summation and has wide applications both in theory and in practice. In theory, it is used in the Ore-Sato theorem [18, 20] to describe the structure of multivariate hypergeometric terms, which in turn is applied by several authors [3, 4, 6, 19] to prove Wilf-Zeilberger's conjecture in different cases. In practice, the integer-linearity of a polynomial also plays a crucial role in the applicability of the well-known Zeilberger's algorithm (also known as the method of creative telescoping) for a bivariate hypergeometric term [1] or for a trivariate rational function [5]. Moreover, the integer-linearity of the denominator determines if a given rational sequence is holonomic (cf. [4, Theorem 13]).

The full integer-linear decomposition of a polynomial also has many uses. For example such a decomposition enables one to compute the Ore-Sato decomposition of a given hypergeometric term [19]. In 2003, Le [15] used the integer-linear decomposition to develop a direct creative telescoping algorithm in the case of discrete bivariate rational functions, which is improved later by the authors in the preparing work [13] to obtain a faster creative telescoping algorithm. Therefore, the efficiency of the computation of the integer-linear decomposition directly affects the utility of the above algorithms.

There are currently two algorithms available to compute the integer-linear decomposition of a polynomial. The first algorithm, developed by Abramov and Le [2] in 2002, is a method based on using resultants. Its main idea is to first find candidates for integer-linear types via resultant and then, for each candidate, extract the corresponding univariate polynomial by a suitable substitution and a content computation. In that paper, their algorithm mainly serves as a terminating condition for Zeilberger's algorithm applied to bivariate rational functions, and thus has only been worked out for bivariate polynomials.

A second algorithm, by Li and Zhang [16], was developed for their implementation of the multivariate Ore-Sato theorem. The algorithm is based on full irreducible factorization of polynomials, and works for polynomials in any number of variables. The core of the algorithm lies in the fact that any nonzero homogeneous component of an integer-linear polynomial of only one type, say a polynomial p of the form (1), can always be written as some power of the same linear factor $\lambda_1 x_1 + \cdots + \lambda_n x_n$. Complexity comparisons between these two algorithms were not provided before but empirical tests suggest that the second algorithm is considerably faster than the first one especially for polynomials of high degrees.

ISSAC '19, July 15–18, 2019, Beijing, China
© 2019 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

As one of our contributions we give a number of improvements to the algorithm of Abramov-Le including the generalization to handle more than two variables, taking into account the reordering of coefficients and avoiding polynomial arithmetic with rational coefficients. We also include a complexity analysis of the above two algorithms in the case of bivariate integer polynomials.

An alternative way to look at integer-linear polynomials is through functional decomposition. More specifically, for a polynomial $p \in \mathrm{R}[x_1, \ldots, x_n]$ of the form (1) we have $p = g \circ h = g(h)$ where $g = P(z)$ and $h = \lambda_1 x_1 + \cdots + \lambda_n x_n$. In this sense, computing the decomposition (1) of p is essentially the same as computing the functional decomposition of p when restricted to the case where g has the same total degree as p. The latter is in fact a special case of the general problem of multivariate functional decomposition that was considered by von zur Gathen [10] in 1990. In Section 5 of that paper, the author gives a fast algorithm which basically addresses the following problem: given a multivariate polynomial p of total degree n and a nonnegative integer r dividing n, find a univariate polynomial g of degree r and a multivariate polynomial h such that $p = g \circ h$ if such a decomposition exists. Limited to our setup, in which $r = n$, the algorithm simply works as follows. For a given polynomial $p \in \mathrm{R}[x_1, \ldots, x_n]$, compute the possible univariate polynomial $g = p(x_1, 0, \ldots, 0)$ and read out the only possible candidate for h from the non-constant homogeneous component of lowest degree. If it succeeds, say $h = \lambda_1 x_1 + \cdots + \lambda_n x_n$, then check whether $p = g \circ h$. This suggests an alternative method, using a similar pattern as the algorithm of Li-Zhang, to compute the integer-linear decomposition of a given polynomial. Namely, first perform full factorization of the input polynomial, then apply the algorithm of von zur Gathen to each irreducible factor, and finally collect up those factors of same integer-linear types. In certain sense, this method and the algorithm of Li-Zhang actually coincide.

The main goal of this paper is to present a new fast algorithm for computing the integer-linear decomposition of a given multivariate polynomial. Our algorithm combines the main ideas of the algorithms in [2, 16] (in the sense that we follow the pattern of the algorithm of Abramov-Le and also take use of homogeneous polynomials as the algorithm of Li-Zhang), but only requires a (faster) rational root finding of the leading homogeneous component of the input polynomial. In particular our algorithm avoids both the computation of resultants and the full polynomial factorization. In order to do a theoretical comparison we have analyzed the worst-case running time complexity of all three algorithms in the case of bivariate integer polynomials. The analysis shows that our new algorithm is faster by a factor of the total degree of the input (d) than the improved algorithm of Abramov-Le and is at least two orders of magnitude better than the algorithm of Li-Zhang. In addition we also give experimental results which verify our complexity comparisons.

The remainder of the paper proceeds as follows. In Section 2 we provide background and basic notions required in the paper. In Section 3 we present our new algorithm for computing the integer-linear decomposition in the special case of a bivariate polynomial including its complexity costs. The new algorithm for the general multivariate case, along with complexity estimates, is given in Section 4. The following section provides a complexity comparison of our new algorithm and the algorithms of [2] and [16]. The paper ends with an experimental comparison among all three algorithms, along with a conclusion section.

2 PRELIMINARIES

Throughout the paper, we let R be a unique factorization domain (UFD) of characteristic zero with $\mathrm{R}[x_1, \ldots, x_n]$ denoting the ring of polynomials in $x_1, \ldots, x_n$ over R. Note that a domain of characteristic zero always contains the ring of integers $\mathbb{Z}$ as a subdomain.

Let p be a polynomial in $\mathrm{R}[x_1, \ldots, x_n]$. Throughout this paper we will order terms using a pure lexicographic order in $x_1 \prec \cdots \prec x_n$. For this order we let $\mathrm{lc}(p)$, $\mathrm{tc}(p)$ and $\deg(p)$ denote the leading and trailing coefficients and total degree, respectively, of p over R with respect to $x_1, \ldots, x_n$. We follow the convention that $\deg(0) = -\infty$. The content, denoted by $\mathrm{cont}(p)$, of p (over R) is the greatest common divisor (GCD) over R of the coefficients of p with respect to $x_1, \ldots, x_n$ with p being *primitive* if $\mathrm{cont}(p) = 1$. The primitive part $\mathrm{prim}(p)$ of p (over R) is defined to be $p/\mathrm{cont}(p)$. For brevity, we will omit the domain if it is clear from the context. In certain instances, we also need to consider the above notions with respect to a subset of the n variables. In these cases, we will either specify the corresponding domain or emphasize the related variables by indicating them as indices of the corresponding notion. For example, $\mathrm{lc}_{x_1,x_2}(p)$, $\mathrm{tc}_{x_1,x_2}(p)$, $\deg_{x_1,x_2}(p)$, $\mathrm{cont}_{x_1,x_2}(p)$ and $\mathrm{prim}_{x_1,x_2}(p)$ denote each function but applied to a polynomial p viewing it as a polynomial in x_1, x_2 over the domain $\mathrm{R}[x_3, \ldots, x_n]$.

Homogeneous polynomials play a key role in our algorithms. Recall a polynomial in $p \in \mathrm{R}[x_1, \ldots, x_n]$ is called *homogeneous* if all its nonzero terms have the same total degree in the variables $x_1, \ldots, x_n$. By gathering together the nonzero monomials of the same total degree, every polynomial in $\mathrm{R}[x_1, \ldots, x_n]$ can be uniquely decomposed as a sum of homogeneous polynomials. Each of such homogeneous polynomials is called a *homogeneous component* of the given polynomial. Amongst all homogeneous components, the one of maximum total degree, namely the one having the same total degree as the original polynomial, is then called the *leading homogeneous component* of the given polynomial. As described previously, we sometimes only need to consider homogeneous polynomials with respect to part of the variables and these instances will be identified by explicitly listing out the involved variables.

We are interested in finding the following decomposition of a polynomial, something briefly alluded to in the introduction.

Definition 2.1. Let $p \in \mathrm{R}[x_1, \ldots, x_n]$ be a polynomial admitting the decomposition

$$p = c\, P_0(x_1, \ldots, x_n) \prod_{i=1}^{m} P_i(\lambda_{i1} x_1 + \cdots + \lambda_{in} x_n), \tag{2}$$

where $c \in \mathrm{R}$, $m \in \mathbb{N}$, $\lambda_{i1}, \ldots, \lambda_{in} \in \mathbb{Z}$, $P_0 \in \mathrm{R}[x_1, \ldots, x_n]$ and $P_i(z) \in \mathrm{R}[z]$. Then (2) is called the *integer-linear decomposition* of p if and only if

(1) P_0 is primitive over R and none of its non-constant irreducible factors is integer-linear;
(2) each $P_i(z)$ is primitive and of positive degree in z;
(3) each $(\lambda_{i1}, \ldots, \lambda_{in})$ is an integer-linear type, in other words, $\gcd(\lambda_{i1}, \ldots, \lambda_{in}) = 1$ and $\lambda_{in} \geq 0$.
(4) any two n-tuples of the $(\lambda_{i1}, \ldots, \lambda_{in})$ are distinct.

We say that the $(\lambda_{i1}, \ldots, \lambda_{in})$ are *integer-linear types* of p and the $P_i(z)$ are the *corresponding univariate polynomials.*

Clearly, p is integer-linear if and only if $P_0 = 1$ in the decomposition (2). By using a full factorization, we see that every polynomial admits an integer-linear decomposition. Moreover, the decomposition is unique up to the order of the factors and multiplication by units of R, according to the uniqueness of the integer-linear types and the full factorization.

3 THE BIVARIATE CASE

Before turning to the general multivariate case, we first consider the simpler yet important subcase of bivariate polynomials. In this section we develop a fast algorithm to find the integer-linear decomposition for bivariate polynomials, with the intention being to illustrate the main idea of our general algorithm in a concise way. By convention, in this section, we will write (x, y) for the two variables (x_1, x_2).

Let p be a polynomial in $\mathsf{R}[x, y]$. As univariate polynomials in x or y are always integer-linear we may assume without loss of generality that the given polynomial p is non-constant, and it is primitive over $\mathsf{R}[x]$ and over $\mathsf{R}[y]$, respectively. In this case p admits the integer-linear decomposition of the form

$$p = P_0(x, y) \prod_{i=1}^{m} P_i(\lambda_i x + \mu_i y), \tag{3}$$

where $m, \mu_i \in \mathbb{N}$, $\lambda_i \in \mathbb{Z}$, $P_0 \in \mathsf{R}[x, y]$ and $P_i(z) \in \mathsf{R}[z]$ with

- the P_0 being primitive over R and merely having non-integer-linear factors except for constants;
- the $P_i(z)$ being non-constant and primitive over R;
- the (λ_i, μ_i) being distinct integer-linear types and all the λ_i, μ_i are nonzero.

From (3) we immediately have the following proposition.

Proposition 3.1. *The leading homogeneous component of the polynomial p with respect to x, y admits a primitive squarefree part over* R *of the form*

$$\tilde{P}_0(x, y) \prod_{i=1}^{m} (\lambda_i x + \mu_i y) \tag{4}$$

for some squarefree homogeneous polynomial $\tilde{P}_0 \in \mathsf{R}[x, y]$ that is coprime with any of the linear factors $(\lambda_i x + \mu_i y)$. Moreover, if p is integer-linear then $\tilde{P}_0 = 1$.

Proposition 3.1 provides a necessary condition for p to be an integer-linear polynomial, that is, the number of linear factors of the leading homogeneous component of p counting multiplicities must be equal to the total degree of p in x, y. In addition, the formula (4) implies that (the primitive squarefree part of) the leading homogeneous component contains all information needed for knowing (a superset of) integer-linear types of p. With candidates for the (λ_i, μ_i) at hand, we are then able to find the corresponding univariate polynomials $P_i(z)$, making use of the following observation of Abramov and Le [2].

Proposition 3.2. *A pair (λ, μ) is an integer-linear type of p if and only if the polynomial $p(\mu x, z - \lambda x)$ has a nontrivial content with respect to x. Moreover, for any integer-linear type (λ, μ) of p, the corresponding univariate polynomial is equal to the primitive part of $\tilde{P}(z/\mu)$ over* R, *where $\tilde{P} = \mathrm{cont}_x(p(\mu x, z - \lambda x))$.*

Proposition 3.2 also guarantees that any false candidate can easily be recognized by a content computation.

In order to obtain the integer-linear decomposition, it remains to find candidates for the (λ_i, μ_i). Rather than using the resultant method employed by Abramov-Le [2], we choose to study the leading homogeneous component instead. Notice that (4) is a bivariate homogeneous polynomial in x, y of total degree, say, d in x, y. As such, by removing the factor x^d, this homogeneous polynomial can be treated as a univariate polynomial in $w = y/x$ of the form

$$g(w) = \tilde{P}_0(1, w) \prod_{i=1}^{m} (\lambda_i + \mu_i w).$$

It is readily seen from the above equation that all the $-\lambda_i/\mu_i$ appear as (nonzero) rational roots of $g(w)$. Observe that any integer-linear type (λ_i, μ_i) is uniquely determined by the rational number λ_i/μ_i. It then follows that all the integer-linear types of p will be correctly found over a domain R with effective rational root finding, so that we know how to find all rational roots of a given univariate polynomial over R. This essentially requires that the domain R admits effective univariate factorization.

Many domains that we are interested in satisfy such a property. For example, when R is the ring of integers $\mathbb{Z}$ or the ring of integer polynomials, all rational roots of a given univariate polynomial over R can be found in a modular fashion [11, 17]. In fact, for the polynomial $g(w)$ given above, instead of using the general bound $2\deg_w(g)(||g||_\infty^2 + ||g||_\infty)$ for the moduli, with $||g||_\infty$ the max-coefficient-norm of g, it is sufficient to choose a prime greater than $\deg_w(g)$ and $2 \cdot D \cdot N$, where D and N are the minimal absolute values of integer coefficients which appear in $\mathrm{lc}_w(g)$ and $\mathrm{tc}_w(g)$, respectively. Such a prime will make sure every monic linear factor $w + \lambda_i/\mu_i$ has a unique modular image. We have all the $\mu_i \leq D$ and all the $|\lambda_i| \leq N$ by observing that the leading and trailing coefficients of g with respect to w are given by

$$\mathrm{lc}_w(\tilde{P}_0(1, w)) \cdot \prod_{i=1}^{m} \mu_i \quad \text{and} \quad \mathrm{tc}_w(\tilde{P}_0(1, w)) \cdot \prod_{i=1}^{m} \lambda_i.$$

Applying rational number reconstruction (cf. [7, 11]) to the resulting modular linear factors quickly recovers all the integer-linear types.

When R is an algebraic number field, say $\mathsf{R} = \mathbb{Q}(\alpha)$ with α an algebraic number, then this amounts to finding all rational roots of the norm function of a given polynomial, which is just a polynomial over $\mathbb{Q}$ (cf. [21]).

Note that although some false candidates may come from the polynomial $\tilde{P}_0(1, w)$, there are at most $d \leq \deg_{x,y}(p)$ candidates that will be generated, which is linear in the size of the input polynomial.

Proposition 3.3. *Assume that* R *admits effective rational root finding, then a superset of all integer-linear types of p can be found efficiently.*

Combining the information from Proposition 3.3 gives us a new algorithm for computing the integer-linear decomposition of a bivariate polynomial. For compatibility with later algorithms, we take a non-constant and primitive polynomial as input.

BivariateILD. Given a non-constant and primitive polynomial $p \in \mathsf{R}[x, y]$ over R, where R admits effective rational root finding, compute the integer-linear decomposition of p.

```
1. set $f_1(y) = \mathrm{cont}_x(p)$, $P_0 = \mathrm{prim}_x(p)$ and $m = 0$;
   if $f_1(y) \neq 1$ then update $m = m + 1$ and set
         $P_m(z) = f_1(z)$ and $(\lambda_m, \mu_m) = (0, 1)$.
2. set $f_2(x) = \mathrm{cont}_y(P_0)$ and update $P_0 = \mathrm{prim}_y(P_0)$;
   if $f_2(x) \neq 1$ then update $m = m + 1$ and set
         $P_m(z) = f_2(z)$ and $(\lambda_m, \mu_m) = (1, 0)$.
3. if $P_0 = 1$ then return $\prod_{i=1}^m P_i(\lambda_i x + \mu_i y)$.
4. let $\tilde{g}$ be the leading homogeneous component of $P_0$;
   if $\tilde{g} \in \mathsf{R}[x] \cup \mathsf{R}[y]$ then return $P_0 \prod_{i=1}^m P_i(\lambda_i x + \mu_i y)$.
5. let $\Lambda = \{-\lambda/\mu \mid \mu > 0$ and $\gcd(\lambda, \mu) = 1\}$ be all the nonzero
   rational roots of $g(z) = \tilde{g}(1, z)$.
6. for $-\lambda/\mu$ in $\Lambda$ do
  6.1 set $h(z) = \mathrm{cont}_x(P_0(\mu x, z - \lambda x))$.
  6.2 if $\deg_z(h) \neq 0$ then
        update $m = m + 1$, $(\lambda_m, \mu_m) = (\lambda, \mu)$,
        $P_m(z) = \mathrm{prim}_z(h(z/\mu))$ and $P_0 = P_0/P_m(\lambda x + \mu y)$;
7. return $P_0 \prod_{i=1}^m P_i(\lambda_i x + \mu_i y)$.
```

The correctness and complexity of the above algorithm is given in the following two theorems.

Theorem 3.4. *Let* R *and* $p \in \mathsf{R}[x, y]$ *be valid inputs of the algorithm* **BivariateILD**. *Then the algorithm correctly computes the integer-linear decomposition of* p.

Proof. For the correctness of the algorithm, it suffices to show that step 4 is correct, with the rest following from the discussion in the paragraphs preceding the algorithm. Notice that at the stage of step 4, P_0 is a non-constant polynomial in $\mathsf{R}[x, y]$ which is primitive both over $\mathsf{R}[x]$ and over $\mathsf{R}[y]$. Then by Proposition 3.1, we know that the primitive squarefree part of the polynomial $\tilde{g}$ over R has the form (4), in which all the λ_i, μ_i are nonzero. Assume that $\tilde{g} \in \mathsf{R}[x] \cup \mathsf{R}[y]$. Then $\deg_x(\tilde{g}) \deg_y(\tilde{g}) = 0$. It follows that the number m in (4) is equal to zero. This in turn implies that P_0 has no integer-linear factors any more. Hence, $P_0 \prod_{i=1}^m P_i(\lambda_i x + \mu_i y)$ is the integer-linear decomposition of p, proving the correctness of step 4. □

Although our algorithms work in more general UFDs, we confine our complexity analysis to the case of bivariate integer polynomials, that is, when R is the ring of integers $\mathbb{Z}$. The cost is given in terms of number of word operations used so that growth of coefficients comes into play. Recall that the *word length* of a nonzero integer $a \in \mathbb{Z}$ is defined as $O(\log |a|)$. Also recall that the *max-norm* of an integer polynomial $p = \sum_{i,j \geq 0} p_{i,j} x^i y^j \in \mathbb{Z}[x, y]$ is defined as $||p||_\infty = \max_{i,j \geq 0} |p_{i,j}|$. In this paper, all complexity is analyzed in terms of O-estimates (or O~-estimates) for classical and fast arithmetic, where the *soft-Oh notation* "O~" is basically "O" but suppressing logarithmic factors (see [11, Definition 25.8] for a precise definition).

Theorem 3.5. *Let* $p \in \mathbb{Z}[x, y]$ *be a valid input of the algorithm* **BivariateILD**. *Assume that* $\deg_{x,y}(p) \leq d$ *and* $||p||_\infty = \beta$. *Then the algorithm computes the integer-linear decomposition of* p *over* $\mathbb{Z}$ *using* $O(d^4 \log^2 d + d^3 \log^2 \beta)$ *word operations with classical arithmetic and* $O^\sim(d^3 \log \beta)$ *with fast arithmetic.*

Proof. In steps 1–2, the content and the primitive part can be computed with $O(d^4 + d^3 \log^2 \beta)$ word operations with classical arithmetic and $O^\sim(d^3 + d^2 \log \beta)$ with fast arithmetic by [11, Theorem 6.39, 9.6, Corollary 11.11]. Step 3 is the trivial case and takes no word operations, while step 4 takes $O(d^2 \log \beta)$ word operations with classical arithmetic to obtain the leading homogeneous component $\tilde{g}$. Since the polynomial $g \in \mathbb{Z}[z]$ has degree at most d in z and max-norm at most β, finding the set Λ in step 5 takes $O^\sim(d^3 + d^2 \log^2 \beta)$ word operations with classical arithmetic and $O^\sim(d^2 \log \beta)$ with fast arithmetic by [12, Theorem 5.10] and [11, Theorem 15.21]. In step 6, observe that P_0 is primitive over $\mathbb{Z}[x]$ and also primitive over $\mathbb{Z}[y]$, at this stage. For each $-\lambda/\mu \in \Lambda$, a straightforward calculation shows that $P_0(\mu x, z - \lambda x)$ has degree in x at most d and degree in z at most $\deg_y(P_0) \leq d$. Moreover, the word length of its max-norm is $O(d \log d + d \log \beta)$. To compute the content $h(z)$ in Step 6.1 we adapt the algorithm of [8], which randomly reduces the problem to a single GCD of two polynomials of degree d, requiring $O(d^3 \log^2 d + d^2 \log^2 \beta)$ word operations with classical arithmetic and $O^\sim(d^2 \log \beta)$ with fast arithmetic, which dominates the cost for step 6.2. Any errors, happening with provably small probability, will be caught in step 6.2, and step 6.1 can be repeated. Note that we can expand $P_0(\mu x, z - \lambda x)$ with a Horner-like scheme within the allowed costs. Since there are at most d elements in the set Λ, the claimed cost then follows. □

Remark 3.6. *If one is only interested in the integer-linearity of the input polynomial* $p \in \mathsf{R}[x, y]$, *rather than the full integer-linear decomposition, then our algorithm can be adapted to abort early. In this case, any of the following five conditions will trigger the adapted algorithm to terminate early, implying that* p *is not integer-linear:*

(1) *after step 3, we have* $\deg_x(P_0) \neq \deg_y(P_0)$;
(2) *in step 4, the* $\tilde{g}$ *is not primitive over* $\mathsf{R}[x]$ *or over* $\mathsf{R}[y]$;
(3) *in step 5, with multiplicities of each nonzero rational roots recorded, the number of roots counting the multiplicities is less than the degree* $\deg_z(g)$;
(4) *in step 6, we have* $\deg_z(h) \neq e$, *where* e *is the multiplicity of the root* $-\lambda/\mu$ *in* $g(z)$;
(5) *in step 7, we have* $P_0 \neq 1$.

4 THE MULTIVARIATE CASE

In this section, we will deal with multivariate polynomials having more than two variables. The difficulty is that simply following the steps of the bivariate algorithm developed in the previous section does not produce an efficient algorithm.

To understand the difficulties in directly extending the bivariate algorithm we proceed as follows. Let p be a polynomial in $\mathbb{Z}[x_1, \ldots, x_n]$. If p is a constant then we are done. Similarly, if p is not primitive over $\mathbb{Z}[x_i]$ for certain integers i with $1 \leq i \leq n$, then we can iteratively remove the contents for these variables and work with the remaining polynomial as done in steps 1-2 of our bivariate algorithm. Note that the integer-linear decomposition of every removed content, viewed as an $(n-1)$-variate polynomial over $\mathbb{Z}$, can be computed recursively.

Thus, we assume that p is non-constant and primitive over $\mathbb{Z}[x_i]$ for all $1 \leq i \leq n$. This implies that every integer-linear type of p has no zero entries. Following step 4 of the bivariate algorithm, we study the leading homogeneous component $\tilde{g}$ of p with respect to $x_1, \ldots, x_n$. Because finding roots of a given univariate polynomial is essentially computing linear factors of that polynomial, analogously to the bivariate case we can find candidates for the integer-linear types using modular factorization (subject to selection of a suitable prime, together with rational number reconstruction applied to all modular linear factors). As before, a nice bound for the prime can be derived from the leading coefficients $\mathrm{lc}_{x_1}(g), \ldots, \mathrm{lc}_{x_n}(g)$, where g is the primitive squarefree part of the leading homogeneous component $\tilde{g}$. The integer-linear decomposition of p would then follow from a multivariate version of Proposition 3.2 by considering the polynomial

$$p(\lambda_n x_1, \ldots, \lambda_n x_{n-1}, z - \lambda_1 x_1 - \cdots - \lambda_{n-1} x_{n-1})$$

for each candidate $(\lambda_1, \ldots, \lambda_n)$.

Unfortunately this method does not appear to perform well in practice, particularly when the input polynomial is an integer-linear polynomial in more than three variables and of high total degree. Indeed, in this case, empirical tests in Maple suggest that the computation of the squarefree part of the leading homogeneous component takes considerably more percentages of the total timing as the number of variables or the total degree increased.

In order to overcome the above issue, we take a different approach. We recall an important proposition from [4], which, translated into our setting, reads as follows.

Proposition 4.1 ([4, Prop 7]). *Let $p \in \mathsf{R}[x_1, \ldots, x_n]$. Then there exists a univariate polynomial $P(z) \in \mathsf{R}[z]$ and n integers $\lambda_1, \ldots, \lambda_n$, not all zero, such that $p = P(\lambda_1 x_1 + \cdots + \lambda_n x_n)$ if and only if for each pair of indices (i, j) with $1 \le i < j \le n$, there are integers α_{ij}, β_{ij}, not both zero, such that*

$$p = P_{ij}(\alpha_{ij} x_i + \beta_{ij} x_j)$$

for $P_{ij}(z) \in \mathsf{R}[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{j-1}, x_{j+1} \ldots, x_n][z]$.

Using Proposition 4.1 one sees that the problem of multivariate integer-linearity is actually made up of a collection of subproblems of bivariate integer-linearity. This inspires us to propose a new algorithm, namely to iteratively tackle only two variables at a time until all variables are treated. In this way, the GCD computation executed in the algorithm typically involves less variables and also has smaller sizes, particularly in the case of integer-linear polynomials.

MultivariateILD. Given a polynomial $p \in \mathsf{R}[x_1, \ldots, x_n]$, where R admits effective rational root finding, compute the integer-linear decomposition of p.

```
1. if $p \in \mathsf{R}$ then set $c = p$ and return $c$.
2. set $c = \mathrm{cont}_{x_1,\ldots,x_n}(p)$ and $f = \mathrm{prim}_{x_1,\ldots,x_n}(p)$.
3. if $n = 1$ then set $m = 1$, $\lambda_{m1} = 1$, $P_m(z) = f(z)$, and return
   $c \prod_{i=1}^{m} P_i(\lambda_{m1} x_1)$.
4. if $n = 2$ then call the algorithm BivariateILD with input
   $f \in \mathsf{R}[x_1, x_2]$ to compute its integer-linear decomposition
       $$f = P_0 \prod_{i=1}^{m} P_i(\lambda_{i1} x_1 + \lambda_{i2} x_2);$$
   and then return $c\, P_0 \prod_{i=1}^{m} P_i(\lambda_{i1} x_1 + \lambda_{i2} x_2)$.
5. set $m = 0$, $g = \mathrm{cont}_{x_1,x_2}(f)$, $P_0 = 1$, and update $f = \mathrm{prim}_{x_1,x_2}(f)$.
6. if $g \neq 1$ then call the algorithm recursively with input $g \in \mathsf{R}[x_3, \ldots, x_n]$, returning
       $$g = \tilde{P}_0 \prod_{i=1}^{\tilde{m}} \tilde{P}_i(\tilde{\lambda}_{i3} x_3 + \cdots + \tilde{\lambda}_{in} x_n);$$
   then update $P_0 = P_0 \cdot \tilde{P}_0$ and for $i = 1, \ldots, \tilde{m}$ iteratively
   update $m = m + 1$, $(\lambda_{m1}, \ldots, \lambda_{mn}) = (0, 0, \tilde{\lambda}_{i3}, \ldots, \tilde{\lambda}_{in})$ and
   $P_m(z) = \tilde{P}_i(z)$.
7. if $f = 1$ then return $c\, P_0 \prod_{i=1}^{m} P_i(\lambda_{i1} x_1 + \cdots + \lambda_{in} x_n)$.
8. set $\Lambda_1 = \{((1), f(x_0, x_2, \ldots, x_n))\}$ with $x_0$ an indeterminate.
   for $k = 1, \ldots, n-1$ do
   8.1 set $\Lambda_{k+1} = \{\}$.
   8.2 for $((\mu_1, \ldots, \mu_k), h(x_0, x_{k+1}, \ldots, x_n))$ in $\Lambda_k$ do
         call the algorithm BivariateILD with input
         $h \in \mathsf{R}[x_{k+2}, \ldots, x_n][x_0, x_{k+1}]$ to compute its
         integer-linear decomposition
             $$h = P_0' \prod_{i=1}^{m'} P_i'(\lambda_i' x_0 + \mu_i' x_{k+1}, x_{k+2}, \ldots, x_n), \qquad (5)$$
         where $P_0' \in \mathsf{R}[x_0, x_{k+1}, \ldots, x_n]$ and
         $P_i'(z, x_{k+2}, \ldots, x_n) \in \mathsf{R}[x_{k+2}, \ldots, x_n][z]$;
         then update $P_0$ by multiplying the polynomial
             $$P_0'(\mu_1 x_1 + \cdots + \mu_k x_k, x_{k+1}, \ldots, x_n)$$
         and update $\Lambda_{k+1}$ by joining the elements
             $$((\lambda_i' \mu_1, \ldots, \lambda_i' \mu_k, \mu_i'), P_i'(x_0, x_{k+2}, \ldots, x_n))$$
         for $i = 1, \ldots, m'$.
9. for $((\mu_1, \ldots, \mu_n), h(x_0))$ in $\Lambda_n$ do
       update $m = m + 1$, $(\lambda_{m1}, \ldots, \lambda_{mn}) = (\mu_1, \ldots, \mu_n)$ and
       $P_m(z) = h(z)$.
10. return $c\, P_0 \prod_{i=1}^{m} P_i(\lambda_{i1} x_1 + \cdots + \lambda_{in} x_n)$.
```

Theorem 4.2. *Let R and $p \in \mathsf{R}[x_1, \ldots, x_n]$ be valid inputs of the algorithm* **MultivariateILD**. *Then the algorithm correctly computes the integer-linear decomposition of p.*

Proof. The correctness of steps 1-7 is evident by definition and Theorem 3.4. For clarity, with $1 \le k \le n-1$, we denote by $P_0^{(k)}$ and $P_0^{(k+1)}$ the respective polynomials P_0 before and after the kth iteration of the outer loop of step 8. If the algorithm does not terminate after step 7, then at this stage (namely before step 8) we have that $f \neq 1$ and

$$p = c\, P_0^{(1)} \prod_{i=1}^{m} P_i(\lambda_{i1} x_1 + \cdots + \lambda_{in} x_n) \cdot f. \qquad (6)$$

We show by induction on k that the following invariant holds each time before and also after the algorithm passes through steps 8.1-8.2.

$$f = \frac{P_0^{(k)}}{P_0^{(1)}} \prod_{\Lambda_k} h(\mu_1 x_1 + \cdots + \mu_k x_k, x_{k+1}, \ldots, x_n) \qquad (7)$$

for any $1 \le k \le n$, where the product runs through all elements $((\mu_1, \ldots, \mu_k), h(x_0, x_{k+1}, \ldots, x_n))$ in Λ_k. From this the correctness of the algorithm follows, since taking $k = n$ in (7), one sees from (6) that

$$p = c\, P_0^{(n)} \prod_{i=1}^{m} P_i(\lambda_{i1}x_1 + \cdots + \lambda_{in}x_n) \prod_{\Lambda_n} h(\mu_1 x_1 + \cdots + \mu_n x_n),$$

implying that steps 9-10 give the desired integer-linear decomposition of p.

For $k = 1$, we have $\Lambda_1 = \{((1), f(x_0, x_2, \ldots, x_n))\}$ so there is nothing to show. We now assume that the invariant (7) holds for $1 \leq k \leq n-1$ and aim to show that it is also true for the case $k+1$. In the kth iteration, one execution of the inner loop in step 8.2 gives the integer-linear decomposition (5) for each element $((\mu_1, \ldots, \mu_k), h(x_0, x_{k+1}, \ldots, x_n))$ of Λ_k. By substituting $x_0 = \mu_1 x_1 + \cdots + \mu_k x_k$ into (5) and updating P_0, Λ_{k+1} as described in step 8.2, it follows from the induction hypothesis that, after the kth iteration, the invariant (7) holds with k replaced by $k+1$. This completes the proof. □

In terms of complexity, it follows from Theorem 3.5 that our algorithm uses merely polynomial time in the case of integer polynomials.

Theorem 4.3. *Let $p \in \mathbb{Z}[x_1, \ldots, x_n]$ be a valid input of the algorithm* **MultivariateILD**. *Then the algorithm takes $(n + \log ||p||_\infty + \deg(p))^{O(1)}$ word operations.*

Example 4.4. Consider the polynomial $p \in \mathbb{Z}[x_1, x_2, x_3, x_4]$ of the form

$$P_0 \cdot P_1(2x_1 - 4x_2 + 3x_3 + 5x_4) \cdot P_2(-4x_1 + 8x_2 - 6x_3 + 7x_4), \quad (8)$$

where $P_0 = (x_1 - 2x_2)x_3 + x_4$, $P_1(z) = 3z^{30} + z + 1$ and $P_2(z) = 7z^{27} - z + 2$. Note that the decomposed form given here is for readability only. In order to compute the integer-linear decomposition, namely (8) by definition, of p over $\mathbb{Z}$, our algorithm (mainly step 8) proceeds in the following three stages. Firstly, by viewing p as a polynomial in x_1, x_2 over $\mathbb{Z}[x_3, x_4]$, applying the algorithm **BivariateILD** to p gets

$$p = P^{(1)}(-x_1 + 2x_2, x_3, x_4) \quad (9)$$

with $P^{(1)}(z, x_3, x_4)$ equal to

$$(-zx_3 + x_4) \cdot P_1(-2z + 3x_3 + 5x_4) \cdot P_2(4z - 6x_3 + 7x_4).$$

There is only one integer-linear type, namely $(-1, 2)$, of p over $\mathbb{Z}[x_3, x_4]$. Next, with input $P^{(1)}(z, x_3, x_4) \in \mathbb{Z}[x_4][z, x_3]$, calling the algorithm **BivariateILD** again and substituting $z = -x_1 + 2x_2$ yield

$$p = P_0 \cdot P^{(2)}(2x_1 - 4x_2 + 3x_3, x_4) \quad (10)$$

with $P^{(2)}(z, x_4) = P_1(z + 5x_4)P_2(-2z + 7x_4)$. The integer-linear type of p over $\mathbb{Z}[x_4]$ is then given by $(2, -4, 3)$. Finally, the last call to the algorithm **BivariateILD** for $P^{(2)}(z, x_4) \in \mathbb{Z}[z, x_4]$, along with $z = 2x_1 - 4x_2 + 3x_3$, leads to the desired decomposition (8). The two integer-linear types $(2, -4, 3, 5)$ and $(-4, 8, -6, 7)$ of p over $\mathbb{Z}$ have been correctly recovered.

From (9) and (10), one sees that p is integer-linear over $\mathbb{Z}[x_3, x_4]$ but it is not integer-linear over $\mathbb{Z}[x_4]$. The latter in turn implies the non-integer-linearity of p over $\mathbb{Z}$, even before starting the third stage.

Similar to the bivariate algorithm, the above algorithm can be modified so as to only determine the integer-linearity of a given polynomial. This is saying that the algorithm can stop already and return the non-integer-linearity of the input polynomial, provided that one of the following conditions is satisfied.

- In step 4 or step 8.2, one of the triggers of the bivariate algorithm listed in Remark 3.6 is touched.
- In step 6, the g turns out to be non-integer-linear.

5 COMPLEXITY COMPARISON

In this section we provide a complexity analysis of the two known algorithms of [2] and [16] in the case of bivariate integer polynomials. Our complexity model follows that discussed earlier in Section 3.

As mentioned in the introduction, the algorithm in [2] is completely focused on the bivariate case so we will further extend it to also tackle the general multivariate case that the algorithm in [16] can already handle. In order to give complexity comparisons we need to estimate the costs of these previously known algorithms. As such we will first briefly review the algorithms from [2] and [16].

5.1 Algorithm of Abramov-Le

As we do with our bivariate algorithm, the algorithm of Abramov-Le [2] also first finds candidates for all integer-linear types of a given bivariate polynomial and then obtains the univariate polynomials via content computation. The difference is that they use resultants to find the candidates.

In order to state their main idea, let $p \in \mathsf{R}[x, y]$ be non-constant and be respectively primitive over $\mathsf{R}[x]$ and over $\mathsf{R}[y]$. Then p admits the integer-linear decomposition of the form (3) with all the λ_i, μ_i nonzero. Observe that any integer-linear type (λ_i, μ_i) is uniquely determined by the rational number $r_i = \lambda_i/\mu_i$. Thus it amounts to finding candidates for the r_i. Analogous to Proposition 3.2, we have that a rational number r gives rise to an integer-linear type of p if and only if $p(x, y - rx)$ has a nontrivial content with respect to x, or equivalently, all coefficients of $p(x, y - rx)$ with respect to x have a nontrivial GCD. This means that, by writing $g_1, \ldots, g_s \in \mathsf{R}[r, y]$ for those coefficients, all the r_i are nonzero rational roots of any resultant $\mathrm{Res}_y(g_i, g_j)$ for $1 \leq i < j \leq s$. It follows that all nonzero rational roots of any nontrivial $\mathrm{Res}_y(g_i, g_j)$ give candidates for the r_i (and then for the (λ_i, μ_i)). After generating candidates, the algorithm continues to find the corresponding univariate polynomials. To this end, for each candidate r, the original algorithm directly computes the content of $p(x, y - rx)$ with respect to x. This operation actually induces polynomial arithmetic with rational coefficients and thus may take considerably more time than step 6.1 of our bivariate algorithm. In order to improve the performance, we proceed by using step 6 of our bivariate algorithm instead.

In the original work [2], the authors did not make any preference about the order of selection of factors g_i, which actually may make a huge difference about the performance of the algorithm. To stabilize the performance, we reorder the factors g_i in such a way that $\deg_y(g_1) \leq \cdots \leq \deg_y(g_s)$ and then start the computation in increasing order of indices. In this way, the computation of resultant typically involves matrices of much smaller sizes and thus the efficiency is improved.

If one directly generalizes the above idea to a multivariate polynomial, say $p \in \mathsf{R}[x_1, \ldots, x_n]$, then the problem would be reduced to finding nonzero rational roots $(r_1, \ldots, r_{n-1})$ of a $(n-1)$-variate polynomial over R that is deduced from the coefficients of $p(x_1, \ldots, x_{n-1}, z - r_1 x_1 - \cdots - r_{n-1} x_{n-1})$ with respect to $x_1, \ldots, x_{n-1}$. When $\mathsf{R} = \mathbb{Z}$, this is equivalent to the well-known Hilbert's tenth problem and thus is unsolvable for an arbitrary n. Therefore, this direct way will not work. In fact, as indicated by Proposition 4.1, the algorithm of Abramov-Le generalizes to polynomials in any number of variables in the same fashion as the algorithm **BivariateILD**.

The following theorem gives a cost estimate for the algorithm of Abramov-Le when applied to a bivariate integer polynomial.

Theorem 5.1. *Let p be a polynomial in $\mathbb{Z}[x, y]$ of total degree d in x, y and with max-norm β. Then the algorithm of Abramov-Le takes $O(d^5 \log^2 d + d^3 \log^2 \beta)$ word operations with classical arithmetic and $O^{\sim}(d^4 + d^3 \log \beta)$ with fast arithmetic.*

Proof. With a slight abuse of notation, let p be the input polynomial with contents with respect to x and y both being removed. Then the algorithm proceeds to find candidates for the integer-linear types of p by first computing all nonzero coefficients $g_1, \ldots, g_s$ of $p(x, y - rx)$ with respect to x, where r is an indeterminate. This step takes $O(d^5 \log^2 d + d^3 \log^2 \beta)$ word operations with classical arithmetic and $O^{\sim}(d^4 + d^3 \log \beta)$ with fast arithmetic. Assuming that the coefficients are ordered so that $\deg_y(g_1) \le \cdots \le \deg_y(g_s)$. Then one sees that $\deg_y(g_1) = 0$, $0 < \deg_r(g_1) = \deg_y(p) \le d$ and $||g_1||_\infty \le \beta$, provided that $\deg_y(p) > 0$. In the best case where $\deg_y(g_2) = 1$, the resultant $\mathrm{Res}_y(g_1, g_2)$ is exactly equal to $g_1 \neq 0$. Then the algorithm tries to find all nonzero rational roots of g_1, whose cost is outweighed by the subsequent computation of the corresponding univariate polynomials for all integer-linear types. Similar to Theorem 3.5, the latter can be accomplished using $O(d^4 \log^2 d + d^3 \log^2 \beta)$ word operations with classical arithmetic and $O^{\sim}(d^3 \log \beta)$ with fast arithmetic, concluding the total cost of the algorithm. □

5.2 Algorithm of Li-Zhang

Unlike the algorithm of Abramov-Le the method used by Li-Zhang [16] is based on full factorization of polynomials. The key observation of their algorithm is that, for any integer-linear polynomial in $\mathsf{R}[x_1, \ldots, x_n]$ having only one integer-linear type $(\lambda_1, \ldots, \lambda_n)$, every nonzero homogeneous component can be written into the form $c \cdot (\lambda_1 x_1 + \cdots + \lambda_n x_n)^k$ for some $c \in \mathsf{R}$ and $k \in \mathbb{N}$. This allows one to easily determine the integer-linearity of an irreducible polynomial. Now the algorithm of Li-Zhang proceeds as follows. After performing full factorization of an input polynomial, it investigates, for each irreducible factor, whether all nonzero homogeneous components are powers of the same linear factor based on the above key observation. Then regrouping all factors of the same integer-linear type eventually yields the desired integer-linear decomposition of the given polynomial.

A careful study of the algorithm of Li-Zhang leads to the following cost estimates.

Theorem 5.2. *Let p be a polynomial in $\mathbb{Z}[x, y]$ of total degree d in x, y and with max-norm β. Then the algorithm of Li-Zhang takes $O^{\sim}(d^7 \log \beta)$ word operations.*

Proof. Computing a complete factorization of $p(x, y)$ into irreducibles dominates the other costs of the algorithm. While we do not know of an explicit analysis of this complexity of factoring bivariate integer polynomials (beyond being in polynomial-time, since [14]). However, the algorithm of [9] can be applied and analyzed as suggested there, and appears to require $O^{\sim}(d^7 \log \beta)$ word operations. □

6 IMPLEMENTATION AND TIMINGS

We have implemented our algorithm in Maple 18 in the case where the domain R is the ring of integer polynomials or an algebraic number field. The code is available by email request. The main problem we have encountered during the implementation is the following. Given a bivariate polynomial $p \in \mathsf{R}[x, y]$ and a candidate integer-linear type (λ, μ), to find the corresponding univariate polynomial, we need to compute the content of the polynomial $p(\mu x, z - \lambda x)$ with respect to x (cf. step 6.1 of the algorithm). If we directly call the Maple command content to do the job, it would be rather time-consuming and most of time is spent on bringing $p(\mu x, z - \lambda x)$ to an expanded form. Instead, we adopt another solution which takes advantage of properties of homogeneous polynomials. The key idea is that manipulating a bivariate homogeneous polynomial $f(x, y) \in \mathsf{R}[x, y]$ is essentially the same as manipulating the univariate polynomial $f(1, y)$. It is much faster to compute $f(\mu, z - \lambda)$ than to compute $f(\mu x, z - \lambda x)$. Applying this idea to every homogeneous component of the given polynomial p yields the final expanded form of $p(\mu x, z - \lambda x)$, and then calling the content command in Maple efficiently completes the task. Experimental tests indicate that this solution is indeed a significant improvement.

In order to get an idea about the efficiency of our algorithm, we compared its runtime, as well as the memory requirements, to the performance of the Maple implementations of the other two known algorithms. The implementation for the algorithm of Abramov-Le is adapted from the hidden procedure IsFactorable in the built-in Maple command SumTools[Hypergeometric][ZpairDirect]. We incorporate reordering of coefficients, replacing content with the idea mentioned in the previous paragraph and extending to the general multivariate case. The implementation for the algorithm of Li-Zhang uses the code provided by the authors themselves.

The test suite was generated by

$$p = P_0(x_1, \ldots, x_n) \prod_{i=1}^{m} P_i(\lambda_{i1} x_1 + \cdots + \lambda_{in} x_n), \tag{11}$$

where $n, m \in \mathbb{N}$,

- P_0 is a random integer polynomial in $x_1, \ldots, x_n$ of total degree d_0;
- the $(\lambda_{i1}, \ldots, \lambda_{in})$ are random integer n-tuples with $|\lambda_{ij}| \le 20$ (note that they may not be distinct).
- $P_i(z) = f_{i1}(z) f_{i2}(z) f_{i3}(z)$ with $f_{ij}(z) \in \mathbb{Z}[z]$ random polynomials of degrees $j \cdot d$ for some $d \in \mathbb{N}$.

Note that, in all the tests, the algorithms take the expanded forms of examples given above as input. All timings are measured in seconds on a Linux computer with 128GB RAM and fifteen 1.2GHz Dual core processors.

For a selection of random polynomials of the form (11) for different choices of n, m, d_0, d, Table 1 tabulates the timings of the extended algorithm of Abramov-Le (AL), the algorithm of Li-Zhang (LZ) and our algorithm (MILD). The experimental results illustrate that our algorithm indeed outperforms the other two algorithms.

(n, m, d_0, d)	AL	LZ	MILD
(2, 2, 5, 5)	0.28	0.49	0.11
(2, 2, 5, 10)	2.25	3.39	0.77
(2, 2, 5, 15)	9.72	13.80	2.82
(2, 2, 5, 20)	44.20	35.80	6.68
(2, 2, 5, 25)	176.00	80.50	13.50
(2, 3, 5, 10)	9.85	14.00	2.68
(2, 3, 10, 10)	10.80	13.40	3.14
(2, 3, 20, 10)	17.10	16.00	3.80
(2, 3, 30, 10)	19.40	18.00	5.32
(2, 3, 40, 10)	25.50	20.90	5.92
(2, 1, 20, 15)	1.16	2.01	0.39
(2, 2, 20, 15)	15.20	16.00	3.34
(2, 3, 20, 15)	129.00	62.00	14.80
(2, 4, 20, 15)	801.00	181.00	47.40
(2, 5, 20, 15)	3350.00	498.00	114.00
(3, 2, 5, 5)	6.71	10.80	2.52
(4, 2, 5, 5)	710.00	657.00	440.00

Table 1: Comparison of all three algorithms for a collection of polynomials p of the form (11).

7 CONCLUSION

In this paper we have presented a new algorithm for the integer-linear decomposition of a multivariate polynomial. Compared with two previously known algorithms, our algorithm is at least faster by a factor of the total degree of the input (d) in terms of worst-case complexity. In practice, our algorithm is also more efficient than these two algorithms. In addition, we have extended and improved the original contribution of Abramov-Le and provided complexity analysis for the improved version. We remark that our algorithm has much better performance than both algorithms in the case where the coefficient domain contains algebraic numbers.

ACKNOWLEDGMENTS

The authors would like to thank Ziming Li for providing the Maple code on the algorithm of Li-Zhang, and the anonymous referees for their constructive and helpful comments. This research was partly supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Giesbrecht, Labahn and Zima).

REFERENCES

[1] S. A. Abramov. 2003. When does Zeilberger's algorithm succeed? *Adv. in Appl. Math.* 30, 3 (2003), 424–441. https://doi.org/10.1016/S0196-8858(02)00504-3

[2] S. A. Abramov and H. Q. Le. 2002. A criterion for the applicability of Zeilberger's algorithm to rational functions. *Discrete Math.* 259, 1-3 (2002), 1–17. https://doi.org/10.1016/S0012-365X(02)00442-9

[3] S. A. Abramov and M. Petkovšek. 2001. Proof of a conjecture of Wilf and Zeilberger. Preprints Series of the Institute of Mathematics, Physics and Mechanics 39(748)(2001), Ljubljana.

[4] S. A. Abramov and M. Petkovšek. 2002. On the structure of multivariate hypergeometric terms. *Adv. in Appl. Math.* 29, 3 (2002), 386–411. https://doi.org/10.1016/S0196-8858(02)00022-2

[5] S. Chen, Q.-H. Hou, G. Labahn, and R.-H. Wang. 2016. Existence problem of telescopers: beyond the bivariate case. In *Proceedings of ISSAC'16*. ACM, New York, 167–174. https://doi.org/10.1145/2930889.2930895

[6] S. Chen and C. Koutschan. 2019. Proof of the Wilf-Zeilberger conjecture for mixed hypergeometric terms. *J. Symbolic Comput.* 93 (2019), 133–147. https://doi.org/10.1016/j.jsc.2018.06.003

[7] G. E. Collins and M. J. Encarnación. 1995. Efficient rational number reconstruction. *J. Symbolic Comput.* 20, 3 (1995), 287–297. https://doi.org/10.1006/jsco.1995.1051

[8] A. Conflitti. 2003. On computation of the greatest common divisor of several polynomials over a finite field. *Finite Fields Appl.* 9, 4 (2003), 423–431. https://doi.org/10.1016/S1071-5797(03)00022-4

[9] S. Gao. 2003. Factoring multivariate polynomials via partial differential equations. *Math. Comp.* 72, 242 (2003), 801–822. https://doi.org/10.1090/S0025-5718-02-01428-X

[10] J. von zur Gathen. 1990. Functional decomposition of polynomials: the tame case. *J. Symbolic Comput.* 9, 3 (1990), 281–299. https://doi.org/10.1016/S0747-7171(08)80014-4

[11] J. von zur Gathen and J. Gerhard. 2013. *Modern Computer Algebra* (third ed.). Cambridge University Press, Cambridge. xiv+795 pages. https://doi.org/10.1017/CBO9781139856065

[12] J. Gerhard. 2004. *Modular Algorithms in Symbolic Summation and Symbolic Integration (Lecture Notes in Computer Science)*. Springer-Verlag.

[13] M. Giesbrecht, H. Huang, G. Labahn, and E. Zima. 2019. Efficient rational creative telescoping. In preparation.

[14] E. Kaltofen. 1985. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM J. Comput.* 14, 2 (1985), 469–489. https://doi.org/10.1137/0214035

[15] H. Q. Le. 2003. A direct algorithm to construct the minimal Z-pairs for rational functions. *Adv. in Appl. Math.* 30, 1-2 (2003), 137–159. https://doi.org/10.1016/S0196-8858(02)00529-8

[16] Z. Li and Y. Zhang. 2013. An algorithm for decomposing multivariate hypergeometric terms. A contributed talk in CM2013.

[17] R. Loos. 1983. Computing rational zeros of integral polynomials by p-adic expansion. *SIAM J. Comput.* 12, 2 (1983), 286–293. https://doi.org/10.1137/0212017

[18] O. Ore. 1930. Sur la forme des fonctions hypergéométriques de plusieurs variables. *J. Math. Pures Appl. (9)* 9, 4 (1930), 311–326.

[19] G. H. Payne. 1997. *Multivariate Hypergeometric Terms.* Ph.D. Dissertation. Pennsylvania State University, Pennsylvania, USA. Advisor(s) Andrews, George E.

[20] M. Sato. 1990. Theory of prehomogeneous vector spaces (algebraic part)—the English translation of Sato's lecture from Shintani's note. *Nagoya Math. J.* 120 (1990), 1–34. https://doi.org/10.1017/S0027763000003214 Notes by Takuro Shintani, Translated from the Japanese by Masakazu Muro.

[21] B. M. Trager. 1976. Algebraic factoring and rational function integration. In *Proceedings of SYMSAC'76*. ACM, New York, 219–226. https://doi.org/10.1145/800205.806338