The complexity of sparse Hensel lifting and sparse polynomial factorization

Michael Monagan

Department of Mathematics,
Simon Fraser University,
Burnaby, British Columbia, V5A 1S6, CANADA

Baris Tuncer

Department of Mathematics,
Simon Fraser University,
Burnaby, British Columbia, V5A 1S6, CANADA

Abstract

The standard approach to factor a multivariate polynomial in $\mathbb{Z}[x_1, x_2, \ldots, x_n]$ is to factor a univariate image in $\mathbb{Z}[x_1]$ then recover the multivariate factors from their images using a process known as multivariate Hensel lifting. Wang's approach, which recovers the variables one at a time, is currently implemented in many computer algebra systems, including Maple, Magma and Singular.

For the case when the factors are expected to be sparse, sparse Hensel lifting was first introduced by Zippel and then improved by Kaltofen. Recently, Monagan & Tuncer introduced a new approach which uses sparse polynomial interpolation to solve the multivariate polynomial diophantine equations that arise inside Hensel lifting. This approach is shown to be practical and faster than Wang's, Zippel's and Kaltofen's algorithms for the non-zero ideal case.

In this work we study what happens to the sparsity of multivariate polynomials when the variables are successively evaluated at numbers. We determine the expected number of remaining terms and the variance. We use these results to revisit and correct the complexity analysis of Zippel's original sparse interpolation. Next we present an average case complexity analysis of the sparse Hensel lifting introduced by Monagan & Tuncer. Finally we present some experimental data comparing our sparse method with Wang's method.

Key words: Polynomial Factorization, Sparse Polynomial Interpolation, Multiviarate Hensel Lifting, Polynomial Diophantine Equations.

Email addresses: `mmonagan@cecm.sfu.ca` (Michael Monagan), `ytuncer@sfu.ca` (Baris Tuncer).

1. Introduction

Suppose we seek to factor a multivariate polynomial $a \in R = \mathbb{Z}[x_1, \ldots, x_n]$. The first step chooses a main variable, say x_1, then computes the content of a in x_1 and removes it from a. Here, if $a = \sum_{i=0}^{d} a_i(x_2, \ldots, x_n)x_1^i$, the content of a is $\gcd(a_0, a_1, \ldots, a_n)$, a polynomial in one fewer variables which is factored recursively. Let us assume this has been done.

The second step identifies any repeated factors in a by doing a *square-free factorization*. See Ch. 8 of (GCL). In this step one obtains the factorization $a = b_1 b_2^2 b_3^3 \cdots b_k^k$ such that each factor b_i has no repeated factors and $\gcd(b_i, b_j) = 1$. Let us assume this has also been done. So let $a = f_1 f_2 \ldots f_m$ be the irreducible factorization of a over $\mathbb{Z}$

The multivariate Hensel lifting algorithm (MHL) developed by Yun (Yun74) and improved by Wang (Wan78; Wan75) chooses an evaluation point $\alpha = (\alpha_2, \alpha_3, \ldots, \alpha_n) \in \mathbb{Z}^{n-1}$ and factors $a(x_1, \alpha_2, \ldots, \alpha_n)$ over $\mathbb{Z}$. The evaluation point α must satisfy

(i) $L(\alpha_2, \ldots, \alpha_n) \neq 0$ where L is the leading coefficient of a in x_1,
(ii) $a(x_1, \alpha_2, \ldots, \alpha_n)$ must have no repeated factors in x_1 and
(iii) $f_i(x_1, \alpha_2, \ldots, \alpha_n)$ must be irreducible over $\mathbb{Q}$.

If any condition is not satisfied the algorithm must restart with a new evaluation point. Conditions (i) and (ii) may be imposed in advance of the next step. To ensure that condition (iii) is true with high probability Maple picks a second evaluation point $\beta = (\beta_2, \ldots, \beta_n) \in \mathbb{Z}^{n-1}$, factor $a(x_1, \beta_2, \ldots, \beta_n)$ over $\mathbb{Z}$ and check that the two factorizations have the same degree pattern before proceeding.

For simplicity let us assume $a = fg$ where f, g in R and a is monic in x_1. Suppose we have obtained the monic factors $f(x_1, \alpha_2, \ldots, \alpha_n)$ and $g(x_1, \alpha_2, \ldots, \alpha_n)$ in $\mathbb{Z}[x_1]$. Next the algorithm picks a prime p for Hensel lifting. [1] For a given polynomial $h \in R$, let us use the notation

$$h_j := h(x_1, \ldots, x_j, x_{j+1} = \alpha_{j+1}, \ldots, x_n = \alpha_n) \bmod p$$

so that $a_1 = a(x_1, \alpha_2, \ldots, \alpha_n) \bmod p$. The input to MHL is a, α, f_1, g_1 and p such that $a_1 = f_1 g_1$ and $\gcd(f_1, g_1) = 1$ in $\mathbb{Z}_p[x_1]$. If the condition $\gcd(f_1, g_1) = 1$ is not satisfied, the algorithm chooses a new prime p until it is.

Wang's MHL lifts the factors f_1, g_1 to f_2, g_2 then to f_3, g_3, until we obtain f_n, g_n. After MHL we have $f_n = f \bmod p$ and $g_n = g \bmod p$. Therefore, for p sufficiently large, we recover the factorization of a over $\mathbb{Z}$.

We give a brief description of the j^{th} step of the MHL (see Algorithm 1) for $j > 1$. Here the input is a_j, f_{j-1}, g_{j-1} and the output is f_j, g_j satisfying $a_j = f_j g_j$. There are two main subroutines in the design of MHL. For details see Ch. 6 of (GCL). The first one is the leading coefficient correction algorithm (LCC). The most well-known is the Wang's heuristic LCC (Wan75) which works well in practice and is the one Maple currently uses. There are other approaches by Kaltofen (Kal85) and most recently by Lee (Lee13). In our implementation we use Wang's LCC.

In a typical application of Wang's LCC, one first factors the leading coefficient of a, a polynomial in $\mathbb{Z}[x_2, \ldots, x_n]$, then one applies LCC before the j^{th} step of MHL. Then the total cost of the factorization is given by the cost of LCC + the cost of factoring $a(x_1, \alpha_2, \ldots, \alpha_n)$ over $\mathbb{Z}$ + the cost of MHL. One can easily construct examples where LCC

[1] One may also perform Hensel lifting modulo a power of a prime instead of a large prime – see Ch. 6 of (GCL). Our methods may also be done modulo a power of a prime but we omit the details here.

or factoring $a(x_1, \alpha_2, \ldots, \alpha_n)$ dominates the cost. However this is not typical. Typically, MHL dominates the cost.

The second main subroutine solves a multivariate polynomial diophantine problem (MDP). In MHL, for each j with $2 \le j \le n$, Wang's design of MHL must solve many instances of MDP in $\mathbb{Z}_p[x_1, \ldots, x_{j-1}]$. In the Maple timings in section 8, often 80% or more is spent solving MDPs. Wang's method for solving an MDP (see Algorithm 2) is recursive. It is exponential in n for sparse factors when the evaluation points $\alpha_2, \ldots, \alpha_n$ are non-zero. See Tables 10 and 11. To solve this problem, Sparse Hensel Lifting (SHL) was first introduced by Zippel (Zip81) and then improved by Kaltofen (Kal85). In (MT16b) we proposed various approaches of sparse interpolation to solve MDP and presented our version of SHL. We also compared our SHL with Kaltofen's SHL in (Kal85).

In this paper we assume a, f, g are monic in x_1 so as not to complicate the presentation of SHL algorithm with LCC. In section 2 we define the MDP in detail and show that interpolation is an option to solve the MDP. If the factors to be computed are sparse then the solutions to the MDP are also sparse. Then we show how to use Zippel's sparse interpolation from (Zip79) to solve the MDP. Section 2 also describes the multivariate polynomial evaluation method that we use because evaluation is often the most expensive part of our SHL.

In Section 3 we will give the idea of SHL and our organization of SHL which is presented as Algorithm 5 MTSHL. In Appendix A we give a concrete example of how the j^{th} step of MTSHL works.

The cost of the $j'th$ step of MHL (Algorithm 1) depends on the degree and number of terms of a_j and the factors f_j and g_j being computed. Let us use $\#h$ for the number of terms of a polynomial h. In Section 4 we study what happens to a when we successively evaluate it at non-zero numbers, that is, we are interested in the sequence $\#a_n, \#a_{n-1}, \ldots, \#a_1$. A simple experiment will show what happens when a is sparse. The Maple command `randpoly` below constructs a polynomial with 10,000 terms where the monomials are chosen uniformly at random from the set of all monomials in $(x_1, \ldots, x_{12})$ with total degree at most 15 and with integer coefficients chosen from $[1, 99]$ uniformly at random.

```
> X := [x1,x2,x3,x4,x,x6,x7,x8,x9,x10,x11,x12];
> a := randpoly(X,degree=15,terms=10000,coeffs=rand(1..99)):
```

We obtain the following data using $(\alpha_2, \alpha_3, \ldots, \alpha_{12}) = (3, 5, 2, 7, 9, 1, 6, 2, 4, 5, 7)$.

j	12	11	10	9	8	7	6	5	4	3	2	1
$\#a_j$	10000	9996	9954	9802	9207	7550	4837	2478	978	304	68	12

The reader should observe that the number of terms does not drop significantly until we have evaluated about half of the variables. This means that the cost of recovering $x_7, x_8, \ldots, x_{11}$ with Hensel lifting is not significantly cheaper than recovering x_{12}. In Section 4 we determine the expected value and variance of $\#a_i$ and make this observation precise. The observations in Section 4 show that one of the assumptions made by Zippel in his complexity analysis of sparse interpolation in (Zip79) is false. We will revise this assumption and correct his analysis.

In the j'th step of MHL we recover x_j in f_j and g_j. Let us write the factors f_j and g_j as

$$f_j = \sum_{i=0}^{\deg_{x_j} f_j} \sigma_i (x_j - \alpha_j)^i \quad \text{and} \quad g_j = \sum_{i=0}^{\deg_{x_j} g_j} \tau_i (x_j - \alpha_j)^i$$

where $\sigma_i, \tau_i \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$. These are just the Taylor series for f_j and g_j expanded about $x_j = \alpha_j$. Let us use $\mathrm{Supp}(f)$ to denote the support of f, that is, the set of monomials of f. Algorithm 1 (MHL) and Algorithm 5 first compute $\sigma_0 := f_{j-1} = f(x_1, \ldots, x_{n-1}, \alpha_n)$ and $\tau_0 := g_{j-1} = g(x_1, \ldots, x_{n-1}, \alpha_n)$ recursively. Then they compute (σ_k, τ_k) for $k = 1, 2, \ldots$ by solving the MDP $\sigma_k g_{j-1} + \tau_k f_{j-1} = c_k$ where c_k is the Taylor coefficient of the polynomial

$$a_j - \left(\sum_{i=0}^{k-1} \sigma_i (x_j - \alpha_j)^i\right) \left(\sum_{i=0}^{k-1} \tau_i (x_j - \alpha_j)^i\right)$$

of $(x_j - \alpha_j)^k$. We make three observations about the coefficients σ_i and τ_i. The Taylor coefficient $\sigma_i = f_j^{(i)}(\alpha_j)/i!$ where $f_j^{(i)}$ is the i'th derivative of f_j in x_j. Since $\#f_j^{(i)} \leq \#f_j$ it follows that $\#\sigma_i \leq \#f_j \leq \#f$. This means that if the factors f and g are sparse, then the σ_i and τ_i computed during Hensel lifting remain sparse. The second observation is that if $\alpha_j \neq 0$, normally $\mathrm{Supp}(\sigma_i) \supseteq \mathrm{Supp}(\sigma_{i+1})$ for $0 \leq i < \deg_{x_j} f_j$. In Section 5 we show that this is the case for most α_j (see Lemma 1). Algorithm 5 (MTSHL) exploits this property by using $\mathrm{Supp}(\sigma_{i-1})$ as the support for σ_i to construct a linear system to solve for the coefficients of σ_i.

Because the cost of MHL depends on the size of $\mathrm{Supp}(\sigma_i)$ we are interested in the sequence $\#\sigma_i$ for $i = 0, 1, 2, \ldots$. To see what happens, let f be the polynomial a above with 10,000 terms and let $f = \sum_{i=0}^{11} \sigma_i (x_{12} - 7)^i$. We obtain the following data for $\#\sigma_i$.

i	0	1	2	3	4	5	6	7	8	9	10	11
$\#\sigma_i$	9996	5526	2988	1504	760	343	158	60	28	8	3	1

What is immediately apparent is that the $\#\sigma_i$ are decreasing rapidly and therefore it will be advantageous if the cost of our algorithm depends on $\#\sigma_i$ and not $\#\sigma_0$. In section 5 we will determine the expected value and variance of $\#\sigma_i$ from which we are able to determine the expected cost of Algorithm 5 (MTSHL) in sections 6 and 7.

Finally in section 8 we give some timing data to compare our factorization algorithms with Wang's factorization algorithm as it is implemented in Maple. We also include some timings for Magma and Singular's factorization codes which also use Wang's algorithm to provide some perspective. See Tables 6 and 7.

We end our introduction by explaining why Hensel lifting is done modulo a prime p. If it is run over $\mathbb{Z}$, an expression swell occurs when univariate diophantine equations $\sigma A + \tau B = C$ are solved in $\mathbb{Z}[x_1]$ by first solving $SA + TB = \gcd(A, B) = 1$ for $S, T \in \mathbb{Q}[x_1]$ using the Euclidean algorithm. The denominators in S and T may be as large as the resultant of A and B which is an integer of length approximately $\max(\deg A, \deg B)$ times longer than the integers in A and B. Working modulo a prime p trivially eliminates this expression swell.

Let $\| \cdot \|$ denote the height (maximum of the absolute value of the coefficients) of a polynomial. If f is any factor of a over $\mathbb{Z}$, not necessarily irreducible, then the prime p used in MHL must satisfy $p > 2\|f\|$ so that MHL recovers the integer (positive and negative) coefficients of f. It is possible that $\|f\| > \|a\|$. For example, the polynomial $x^{105} - y^{105}$ has height 1 but it has a degree 60 factor with height 74. For this purpose the following bound may be derived from Lemma II on page 135 of (Gel52): $\|f\| < e^{d_1 + d_2 + \cdots + d_n} \|a\|$ where $e = 2.7818$ and $d_i = \deg_{x_i} a$.

2. The Multivariate Diophantine Problem (MDP)

Following the notation in section 1, let $u, w, c \in \mathbb{Z}_p[x_1, \ldots, x_j]$ with u and w monic with respect to the variable x_1 and let $I_j = \langle x_2 - \alpha_2, \ldots, x_j - \alpha_j \rangle$ be an ideal of $\mathbb{Z}_p[x_1, \ldots, x_j]$ with $\alpha_i \in \mathbb{Z}$. The MDP is to find multivariate polynomials $\sigma, \tau \in \mathbb{Z}_p[x_1, \ldots, x_j]$ that satisfy

$$\sigma u + \tau w = c \mod I_j^{d_j+1} \tag{1}$$

with $\deg_{x_1}(\sigma) < \deg_{x_1}(w)$ where d_j is the maximal degree of σ and τ with respect to the variables $x_2, \ldots, x_j$ and it is given that

$$\mathrm{GCD}\,(u \bmod I_j, w \bmod I_j) = 1 \text{ in } \mathbb{Z}_p[x_1].$$

It can be shown that the solution (σ, τ) exists and is unique and independent of the choice of the ideal I_j. For $j = 1$ the MDP is in $\mathbb{Z}_p[x_1]$ and can be solved with the extended Euclidean algorithm (see Chapter 2 of (GCL)).

Algorithm 1 j^{th} step of Multivariate Hensel Lifting for $j > 1$.

Input : $\alpha_j \in \mathbb{Z}_p$, $a_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$, $f_{j-1}, g_{j-1} \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$ where a_j, f_{j-1}, g_{j-1} are monic in x_1 and $a_j(x_j = \alpha_j) = f_{j-1} g_{j-1}$.
Output : $f_j, g_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$ such that $a_j = f_j g_j$ or FAIL.

```
f_j ← f_{j-1};  g_j ← g_{j-1}.
error ← a_j − f_{j-1} g_{j-1}.
for i from 1 while error ≠ 0 and deg_{x_j} f_j + deg_{x_j} g_j < deg_{x_j} a_j do
    c_i ← Taylor coefficient of (x_j − α_j)^i of error at x_j = α_j
    if c ≠ 0 then
        Solve the MDP σ_i g_{j-1} + τ_i f_{j-1} = c_i in Z_p[x_1, ..., x_{j-1}] for σ_i and τ_i.
        (f_j, g_j) ← (f_j + σ_i × (x_j − α_j)^i, g_j ← f_j + τ_i ×, (x_j − α_j)^i)
        error ← a_j − f_j g_j
    end if
end for
if error = 0 then return f_j, g_j else return FAIL end if
```

To solve the MDP in Algorithm 1, for $j > 1$, Wang uses the same approach as for Hensel Lifting, that is, an ideal-adic approach which we present as Algorithm 2.

In general, if $\alpha_j \neq 0$ then the Taylor series expansion of σ and τ about $x_j = \alpha_j$ in Algorithm 2 is dense so the $c_i \neq 0$. If we let $d_j = \deg_{x_j} a_j$ and $M(x_j)$ be the number of calls to the Euclidean algorithm in Step 1 of Algorithm 2, then $M(x_1) = 1$ and $M(x_j) \leq (d_j - 1)M(x_{j-1})$ thus $M(x_{n-1}) \leq \prod_{i=2}^{n-1}(d_i - 1)$. If all $\alpha_j \neq 0$ for $2 \leq j \leq n-1$ then $M(x_{n-1})$ becomes exponential in n. It is this exponential behaviour that sparse multivariate diophantine solvers eliminate. On the other hand, if MHL can choose some α_j to be 0, for example, if the input polynomial $a(x_1, \ldots, x_n)$ is monic in x_1 then this exponential behaviour may not occur for sparse factors f and g.

2.1. Solution to the MDP via Interpolation

We consider whether we can interpolate $x_2, \ldots, x_j$ in σ and τ in (1) using sparse interpolation methods. If $\beta \in \mathbb{Z}_p$ with $\beta \neq \alpha_j$, then

$$\sigma(x_j = \beta)u(x_j = \beta) + \tau(x_j = \beta)w(x_j = \beta) \;=\; c(x_j = \beta) \bmod I_{j-1}^{d_{j-1}+1}.$$

Algorithm 2 WMDS (Wang's multivariate diophantine solver)

Input Polynomials $u, w, c \in \mathbb{Z}_p[x_1, \ldots, x_j]$ and an ideal $I = \langle x_2 - \alpha_2, \ldots, x_n - \alpha_n \rangle$ with $n \geq j$ where $\gcd(u \mod I, w \mod I) = 1$ in $\mathbb{Z}_p[x_1]$ and degree bounds $d_2, \ldots, d_n$ satisfying $d_i \geq \max(\deg_{x_i} \sigma, \deg_{x_i} \tau)$ for $2 \leq i \leq n$. (One may use $d_i = \deg_{x_i} a_i$)

Output $(\sigma, \tau) \in \mathbb{Z}_p[x_1, \ldots, x_j]$ satisfying $\sigma u + \tau w = c$ and $\deg_{x_1} \sigma < \deg_{x_1} w$ or FAIL if no such solution exists.

if $j = 1$ **then** use the extended Euclidean algorithm **end if**
$(\sigma_0, \tau_0) \leftarrow$ WMDS($u(x_j = \alpha_j),\ w(x_j = \alpha_j),\ c(x_j = \alpha_j),\ I$)
if WMDS output FAIL **then** return FAIL **end if**
$(\sigma, \tau) \leftarrow (\sigma_0, \tau_0)$; $\;error \leftarrow c - \sigma u - \tau w$
for $i = 1, 2, \ldots, d_j$ **while** $error \neq 0$ **do**
  $c_i \leftarrow$ Taylor coeff($error,\ (x_j - \alpha_j)^i$)
  if $c_i \neq 0$ **then**
    $(s, t) \leftarrow$ WMDS($\sigma_0,\ \tau_0,\ c_i,\ I$)
    if WMDS output FAIL **then** output FAIL **end if**
    $(s, t) \leftarrow (s \times (x_j - \alpha_j)^i, t \times (x_j - \alpha_j)^i))$
    $(\sigma, \tau) \leftarrow (\sigma + s, \tau + t)$
    $error \leftarrow error - su - tw$
  end if
end for
if $error = 0$ **then** return (σ, τ) **else** return FAIL **end if**

For $K_j = \langle x_2 - \alpha_2, \ldots, x_{j-1} - \alpha_{j-1}, x_j - \beta \rangle$ and $G_j = \mathrm{GCD}(u \bmod K_j,\ w \bmod K_j)$, we obtain a unique solution $\sigma(x_j = \beta)$ iff $G_j = 1$. However $G_j \neq 1$ is possible. Let $R = \mathrm{res}_{x_1}(u, w)$ be the Sylvester resultant of u and w taken in x_1. Since u, w are monic in x_1 one has

$$G_j \neq 1 \iff \mathrm{res}_{x_1}(u \bmod K_j,\ w \bmod K_j) = 0 \iff R(\alpha_2, \ldots, \alpha_{j-1}, \beta) = 0.$$ [2]

Let $r = R(\alpha_2, \ldots, \alpha_{j-1}, x_j) \in \mathbb{Z}_p[x_j]$ so that $R(\alpha_2, \ldots, \alpha_{j-1}, \beta) = 0 \iff r(\beta) = 0$. Also $\deg(R) \leq \deg(u)\deg(w)$ (CLO). Now if β is chosen at random from $\mathbb{Z}_p$ and $\beta \neq \alpha_j$ then

$$\Pr[G_j \neq 1] = \Pr[r(\beta) = 0] \leq \frac{\deg(r, x_j)}{p - 1} \leq \frac{\deg(u)\deg(w)}{p - 1}.$$

This bound for $\Pr(G_j \neq 1)$ is a worst case bound. In (MT16a) we show that the average probability for $\Pr[G_j \neq 1] = 1/(p-1)$. Thus if p is large, the probability that $G_j = 1$ is high. Interpolation is thus an option to solve the MDP. If $G_j \neq 1$, we could choose another β but our implementation simply returns FAIL and restarts by choosing new $\alpha_2, \ldots, \alpha_n$.

2.2. Solution to the MDP via Sparse Interpolation

Following the sparse interpolation idea of Zippel in (Zip90), given a sub-solution $\sigma_j(x_j = \alpha_j)$ we use this information to create a sub-solution form σ_f and compute $\sigma_j(x_j = \beta_j)$ for other random $\beta_j's \in \mathbb{Z}_p$ with high probability if p is big. We could interpolate $x_2, \ldots, x_{j-1}$ in $\sigma_j(x_j = \beta_j)$ from univariate images in $\mathbb{Z}_p[x_1]$. Instead we use bivariate images in $\mathbb{Z}_p[x_1, x_2]$ because for polynomials in many variables with many terms bivariate images will likely be

[2] This argument also works for the non-monic case if the leading coefficients of u and w w.r.t. x_1 do not vanish at $(\alpha_2, \ldots, \alpha_n)$ modulo p, conditions which we note are imposed by Wang's LCC.

dense so it makes sense to use a dense solver for the bivariate case. This improvement likely reduces the number of images required which reduces the size of the linear systems and the evaluation cost (see (MT16b)).

Suppose the form of σ_j is

$$\sigma_f = \sum_{i+k \le d} c_{ik}(x_3, ..., x_j) x_1^i x_2^k \text{ where } c_{ik} = \sum_{l=0}^{s_{ik}} c_{ikl} x_3^{\gamma_{3l}} \cdots x_j^{\gamma_{jl}} \text{ with } c_{ikl} \in \mathbb{Z}_p \backslash \{0\}$$

where the total degree of σ_j in x_1, x_2 is bounded by d. Let $s = \max s_{ik}$ be the maximum number of terms in the coefficients of σ_f. We obtain each c_{ik} by solving $\mathcal{O}(d^2)$ linear systems of size at most $s \times s$. As explained in (Zip90), each linear system can be solved in $\mathcal{O}(s^2)$ arithmetic operations in $\mathbb{Z}_p$ and using $\mathcal{O}(s)$ space. We then interpolate x_j in σ_j from $\sigma_j(x_j = \beta_k)$ for $k = 0, \ldots, \deg_{x_j}(\sigma_j)$. Finally we compute $\tau_j = (c_j - \sigma_j u_j)/w_j$. If this division fails it means that σ_f is wrong; we must restart the factorization with a new $\alpha_2, \ldots, \alpha_n$.

To solve the MDP in $\mathbb{Z}_p[x_1, x_2]$ we have implemented an efficient dense bivariate Diophantine solver (BDP) in C. The algorithm incrementally interpolates x_2 in both σ and τ from univariate images in $\mathbb{Z}_p[x_1]$. When σ and τ stabilize we test whether $\sigma(x_1, x_2)u(x_1, x_2) + \tau(x_1, x_2)w(x_1, x_2) = c(x_1, x_2)$ using sufficiently many evaluations to prove the correctness of the solution. The cost is $\mathcal{O}(d^3)$ arithmetic operations in $\mathbb{Z}_p$ where d bounds the total degree of c, u, w, σ and τ in x_1 and x_2. We do not compute τ using division because that would cost $\mathcal{O}(d^4)$ arithmetic operations.

This multivariate MDP solving algorithm is presented as Algorithm MDSolver (MultivariateDiophantSolver) below. Following this we present the sparse interpolation algorithm used in Step 11.

Algorithm 3 MDSolver

Input A big prime p and $u, w, c \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ where the MDP conditions (see section 1) are satisfied.

Output $(\sigma, \tau) \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ such that $\sigma u + \tau w = c \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$.

1: **if** $n = 2$ **then** call BDP to **return** $(\sigma, \tau) \in \mathbb{Z}_p[x_1, x_2]^2$ or FAIL **end if**.
2: Pick $\beta_1 \in \mathbb{Z}_p$ at random
3: $(u_{\beta_1}, w_{\beta_1}, c_{\beta_1}) \leftarrow (u(x_j = \beta_1), w(x_j = \beta_1), c(x_j = \beta_1))$.
4: $(\sigma_1, \tau_1) \leftarrow$ **MDSolver**$(u_{\beta_1}, w_{\beta_1}, c_{\beta_1}, p)$.
5: **if** $\sigma_1 =$ FAIL **then return** FAIL **end if**
6: $k \leftarrow 1$, $\sigma \leftarrow \sigma_1$, $q \leftarrow (x_j - \beta_1)$ and $\sigma_f \leftarrow \sigma_1$.
7: **repeat**
8:     $h \leftarrow \sigma$
9:     Set $k \leftarrow k + 1$ and pick $\beta_k \in \mathbb{Z}_p$ at random distinct from $\beta_1, \ldots, \beta_{k-1}$
10:     $(u_{\beta_k}, w_{\beta_k}, c_{\beta_k}) \leftarrow (u(x_j = \beta_k), w(x_j = \beta_k), c(x_j = \beta_k))$.
11:     $(\sigma_k, \tau_k) \leftarrow$**SparseInt**$(u_{\beta_k}, w_{\beta_k}, c_{\beta_k}, \sigma_f)$ (Algorithm 4)
12:     **if** $\sigma_k =$ FAIL **then return** FAIL **end if**
13:     Solve $\{\sigma = h \bmod q$ and $\sigma = \sigma_k \bmod (x_j - \beta_k)\}$ for $\sigma \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$.
14:     $q \leftarrow q \cdot (x_j - \beta_k)$
15: **until** $\sigma = h$ and $w | (c - \sigma u)$
16: Set $\tau \leftarrow (c - \sigma u)/w$ and **return** (σ, τ).

Algorithm 4 SparseInt : solve an MDP using a sparse interpolation

Input: Polynomials $u, w, c, \sigma_f \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ for u, w monic in x_1 and p a prime.
Output: The solution (σ, τ) to the MDP $\sigma u + \tau w = c \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ or FAIL

1: Let $\sigma = \sum_{i,k} c_{ik}(x_3, ..., x_j) x_1^i x_2^k$ where $c_{ik} = \sum_{l=1}^{s_{ik}} c_{ikl} M_{ikl}$ with c_{ikl} unknown coefficients to be solved for and $x_1^i x_2^j M_{ikl}$ are the monomials in $\mathrm{Supp}(\sigma_f)$.
2: Let $s = \max s_{ik} = \max \#c_{ik}$.
3: Pick $(\alpha_3, \ldots \alpha_j) \in (\mathbb{Z}_p \backslash \{0\})^{j-2}$ at random.
4: Compute monomial evaluation sets

$$\{S_{ik} = \{m_{ikl} = M_{ikl}(\alpha_3, \ldots, \alpha_j) : 1 \le l \le s_{ik}\} \text{ for each } i, k\}\,.$$

If $|S_{ik}| \neq s_{ik}$ for some ik try a different choice for $(\alpha_3, \ldots, \alpha_j)$. If this fails repeatedly, **return** FAIL. (*p is not big enough*)
5: **for** i from 1 to s **do** (*Compute the bivariate images of σ*)
6: Let $Y_i = (x_3 = \alpha_3^i, \ldots, x_j = \alpha_j^i)$.
7: Evaluate $u(x_1, x_2, Y_i), w(x_1, x_2, Y_i), c(x_1, x_2, Y_i)$ by the method in section 2.3.
8: Solve $\sigma_i(x_1, x_2) u(x_1, x_2, Y_i) + \tau_i(x_1, x_2) w(x_1, x_2, Y_i) = c(x_1, x_2, Y_i)$ in $\mathbb{Z}_p[x_1, x_2]$ for $\sigma_i(x_1, x_2)$ using BDP (see section 2.2).
9: **if** BDP returns FAIL **then**
10: **return** FAIL *(Y_i is unlucky or there is no solution to the BDP).*
11: **end if**
12: **end for**
13: **for** each i, k **do**
14: Construct and solve the $s_{ik} \times s_{ik}$ linear system

$$\left\{ \sum_{l=1}^{s_{ik}} c_{ikl}\, m_{ikl}^n \;=\; \text{coefficient of } x_1^i x_2^k \text{ in } \sigma_n(x_1, x_2) \;\; \text{for } 1 \le n \le s_{ik} \right\}$$

for the coefficients c_{ikl} of $c_{ik}(x_3, \ldots, x_j)$. Because it is a Vandermonde system in m_{ikl} which are distinct by Step 4 it has a unique solution.
15: **end for**
16: Substitute the solutions for c_{ikl} into σ
17: **if** $w \,|\, (c - \sigma u)$ **then**
18: Set $\tau = (c - \sigma u)/w$ and **return** (σ, τ)
19: **else**
20: **return** FAIL *(σ_f is wrong)*
21: **end if**

Algorithm SparseInt interpolates σ only and obtains τ by division in Step 18. The division also detects an incorrect σ_f. An alternative organization of SparseInt could interpolate both σ and τ and then test if $c - \sigma u - \tau w = 0$. The reason we interpolate only σ is that this is faster if σ is significantly smaller than τ, more precisely if s the number of bivariate images needed to interpolate σ is significantly fewer than the number needed to interpolate τ.

2.3. The evaluation cost

Suppose $f = \sum_{i=1}^{s} c_i X_i Y_i$ where X_i is a monomial in x_1, x_2 and Y_i is a monomial in $x_3, \ldots, x_n$ and $0 \neq c_i \in \mathbb{Z}_p$. In sparse interpolation we want to compute

$$f_j := f(x_1, x_2, x_3 = \alpha_3^j, \ldots, x_n = \alpha_n^j), \text{ for } j = 1, \ldots, t.$$

We can take advantage of the form of the evaluation points. As an example suppose that

$$f = x_1^{22} + 72x_1^3x_2^4x_4x_5 + 37x_1x_2^5x_3^2x_4 - 92x_1x_2^5x_5^2 + 6x_1x_2^3x_3x_4^2.$$

Before combining and sorting, we write the terms of each f_j as

$$f_j = x_1^{22} + 72(\alpha_4\alpha_5)^j x_1^3x_2^4 + 37(\alpha_3^2\alpha_4)^j x_1x_2^5 - 92(\alpha_5^2)^j x_1x_2^5 + 6(\alpha_3\alpha_4^2)^j x_1x_2^3.$$

Now let

$$c^{(0)} := [1, 72, 37, -92, 6] \quad \text{and} \quad \theta := [1, \alpha_4\alpha_5, \alpha_3^2\alpha_4, \alpha_5^2, \alpha_3\alpha_4^2].$$

Then in a for loop $j = 1, \ldots, t$ we can update the coefficient array $c^{(0)}$ by the monomial array θ by defining $c_i^{(j)} = c_i^{(j-1)}\theta_i$ for $i = 1, \ldots, t$ so that each iteration computes the coefficient array

$$c^{(j)} = [1, 72(\alpha_4\alpha_5)^j, 37(\alpha_3^2\alpha_4)^j, -92(\alpha_5^2)^j, 6(\alpha_3\alpha_4^2)^j].$$

of the unsorted f_j using $\#f$ multiplications in the coefficient field. Then combining and sorting the monomials to get

$$f_j = x_1^{22} + 74\alpha_3^j(\alpha_4^j)^5x_1^3x_2^4 + \left(37(\alpha_3^j)^2\alpha_4^j - 92(\alpha_5^j)^2\right)x_1x_2^5 + 6\alpha_3^j(\alpha_4^j)^2x_1x_2^3.$$

Note that the sorting is a time consuming step too. So we should do the sorting once at the beginning. Then compute the arrays $c^{(j)}$ and then combine according to the sorting rule. In the example above by looking at the terms of f we know that after the evaluation first and the second, also the third and the forth terms of f will collide. Hence after computing each $c^{(j)}$ we know that the sum of the first and the second, also the third and the fourth terms of each array will correspond to the coefficients of f_j, so we won't spend time to sort the terms of each unsorted f_j.

With the organization described above one evaluates Y_i at $(\alpha_3, \ldots, \alpha_n)$ in $(n-3)$ multiplications using tables. The cost of $n-2$ tables of powers is $\leq (n-2)d$. Then at the first step cost of (creating monomial array) is $\leq s(n-3)+(n-2)d$. After that cost of each t evaluations is st multiplications. Hence the total cost is bounded above by $C_N = st + s(n-3) + (n-2)d$.

3. Sparse Hensel Lifting

3.1. The main Idea of SHL

Factoring multivariate polynomials via Sparse Hensel Lifting (SHL) uses the same idea of the sparse interpolation. Following the same notation introduced in section 1, at $(j-1)^{\text{th}}$ step we have

$$f_{j-1} = x_1^{df} + c_{j1}M_1 + \cdots + c_{jt_j}M_{t_j}$$

where t_j is the number of non-zero terms that appear in f_{j-1}, M_k's are the distinct monomials in $x_1, \ldots, x_{j-1}$ and $c_{jk} \in \mathbb{Z}_p$ for $1 \leq k \leq t_j$. Then at the j^{th} step SHL assumes

$$f_j = x_1^{df} + \Lambda_{j1}M_1 + \cdots + \Lambda_{jt_j}M_{t_j}$$

where for $1 \leq k \leq t_j$,

$$\Lambda_{jk} = c_{jk}^{(0)} + c_{jk}^{(1)}(x_j - \alpha_j) + c_{jk}^{(2)}(x_j - \alpha_j)^2 + \cdots + c_{jk}^{(d_{j_k})}(x_j - \alpha_j)^{d_{j_k}}$$

with $c_{jk}^{(0)} := c_{jk}$ and where $df = \deg_{x_1}(f)$, $d_{j_k} = \deg_{x_n}(\Lambda_{jk})$ with $c_{jk}^{(i)} \in \mathbb{Z}_p$ for $0 \leq i \leq d_{j_k}$. We will call this assumption the weak SHL assumption. The assumption is the same for the factor g_{j-1}.

To recover f_j from f_{j-1} and g_j from g_{j-1}, during the j^{th} step of MHL (see Algorithm 1) one starts with $\sigma_0 = f_{j-1}$, $\tau_0 = g_{j-1}$, then in a for loop starting from $i = 1$ and incrementing it while the error term and its i^{th} Taylor coefficient is non-zero, by solving MDPs $\sigma_i\tau_0 + \tau_i\sigma_0 = e_i$ for $1 \le i \le \max(\deg_{x_j}(f_j), \deg_{x_j}(g_j))$. After the loop terminates we have $f_j = \sum_{k=0}^{d_j} \sigma_k (x_j - \alpha_j)^k$. On the other hand if the weak SHL assumption is true then we also have

$$\begin{aligned} f_j &= x_1^{df} + (\sum_{i=0}^{d_j} c_{j1}^{(i)} (x_j - \alpha_j)^i) M_1 + \cdots + (\sum_{i=0}^{d_j} c_{jt_j}^{(i)} (x_j - \alpha_j)^i) M_{t_j} \\ &= x_1^{df} + \sum_{i=0}^{d_j} (c_{j1}^{(i)} M_1 + \cdots + c_{jt_j}^{(i)} M_{t_j})(x_j - \alpha_j)^i. \end{aligned}$$

Similarly for g_j. Hence if the weak SHL assumption is true then the support of each σ_k will be a subset of support of f_{j-1}. Therefore we can use f_{j-1} as a skeleton of the solution of each σ_k. The same is true for τ_k. Although it is not stated explicitly in (Kal85), this is one of the underlying ideas of Kaltofen's SHL.

3.2. Our SHL organization

Before explaining our SHL organization (MTSHL), we make the following observation which has been proven in (MT16b):

Lemma 1. *Let $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$ and let α be a randomly chosen element in $\mathbb{Z}_p$ and $f = \sum_{i=0}^{d_n} b_i(x_1, \ldots, x_{n-1})(x_n - \alpha)^i$ where $d_n = \deg_{x_n} f$. Then*

$$\Pr[\operatorname{Supp}(b_{j+1}) \not\subseteq \operatorname{Supp}(b_j)] \;\le\; |\operatorname{Supp}(b_{j+1})| \, \frac{d_n - j}{p - d_n + j + 1} \quad \text{for } 0 \le j < d_n.$$

Lemma 1 shows that for the sparse case, if p is big enough then the probability of $\operatorname{Supp}(b_{j+1}) \subseteq \operatorname{Supp}(b_j)$ is high. This observation suggests we use σ_{i-1} (or τ_{i-1}) as a form of the solution of σ_i (or τ_i). We call this assumption $\operatorname{Supp}(\sigma_i) \subseteq \operatorname{Supp}(\sigma_{i-1})$ for all $i > 0$ the strong SHL assumption. Based on this observation the j^{th} step of our SHL organization is summarized in Algorithm 5. See Appendix A for a concrete example.

4. The expected number of terms after evaluation

The complexity of MTSHL depends on the number of terms in the factors, and the number of terms of each factor is expected to decrease from the step $j+1$ to j after evaluation. To make our complexity analysis as precise as possible, we need to give an upper bound for the expected sizes of the factors in each step j. In this section we will compute these bounds and confirm our theoretical bounds with experimental data.

Let p be a big prime and $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$ be a randomly chosen multivariate polynomial of degree $\le d$ that has T non-zero terms. Let $s = \binom{n+d}{n}$ be the number of all possible monomials of degree $\le d$. An upper bound for T is then given by $T \le s$. By randomly chosen we mean that the probability of occurrence of each monomial is the same and equal to $1/s$. So we think of choosing a random polynomial of degree $\le d$ that has T terms

Algorithm 5 j^{th} step of MTSHL for $j > 1$.

Input : $a_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$, $f_{j-1}, g_{j-1} \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$ and $\alpha_j \in \mathbb{Z}_p$ where a_j, f_{j-1}, g_{j-1} are monic in x_1. Also, $a_j(x_1, \ldots, x_{j-1}, x_j = \alpha_j) = f_{j-1}g_{j-1}$.

Output : $f_j, g_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$ such that $a_j = f_j g_j$ or FAIL

if $\#f_{j-1} > \#g_{j-1}$ **then** interchange f_{j-1} with g_{j-1} **end if**
$(\sigma_0, \tau_0) \leftarrow (f_{j-1}, g_{j-1})$.
$(f_j, g_j) \leftarrow (f_{j-1}, g_{j-1})$.
$\mathit{error} \leftarrow a_j - f_j\, g_j$; $\mathit{monomial} \leftarrow 1$.
for $i = 1, 2, 3, \ldots$ **while** $\mathit{error} \neq 0$ and $\deg(f_j, x_j) + \deg(g_j, x_j) < \deg(a_j, x_j)$ **do**
    $\mathit{monomial} \leftarrow \mathit{monomial} \times (x_j - \alpha_j)$.
    $c \leftarrow$ Taylor coefficient of $(x_j - \alpha_j)^i$ of *error* at $x_j = \alpha_j$
    if $c \neq 0$ **then**
        Solve the MDP $\sigma_i\, g_{j-1} + \tau_i\, f_{j-1} = c$ for σ_i and τ_i in $\mathbb{Z}_p[x_1, \ldots, x_{j-1}]$:
        $\sigma_g \leftarrow \sigma_{i-1}$. (assume $\mathrm{Supp}(\sigma_i) \subseteq \mathrm{Supp}(\sigma_{i-1})$)
        $(\sigma_i, \tau_i) \leftarrow$ **SparseInt**(g_{j-1}, f_{j-1}, c, σ_g) (see Algorithm 4)
        if (σ_i, τ_i)=FAIL **then** $(\sigma_i, \tau_i) \leftarrow$ **MDSolver**(f_{j-1}, g_{j-1}, c, p) **end if**
        if (σ_i, τ_i)=FAIL **then** restart SHL with a different evaluation point α **end if**
        $(f_j, g_j) \leftarrow (f_j + \sigma_i \times \mathit{monomial},\ g_j + \tau_i \times \mathit{monomial})$.
        $\mathit{error} \leftarrow a_j - f_j\, g_j$.
    end if
end for
if $\mathit{error} = 0$ **return** (f_j, g_j) **end if**
return FAIL *(No such factorization exists)*

as choosing T distinct monomials out of s choices and choosing coefficients uniformly at random from $[1, p-1]$.

Let also $g = f(x_n = \alpha_n)$ for a randomly chosen non-zero element $\alpha_n \in \mathbb{Z}_p$. Before evaluation let $f = \sum_{i=1}^{t} \mathbf{m}_i c_i(x_n)$ where $\mathbf{m}_i$ are monomials in the variables $x_1, \ldots, x_{n-1}$. Then $T = \sum_{i=1}^{t} \#c_i(x_n)$ and $g = \sum_{i=1}^{t} \mathbf{m}_i c_i(\alpha_n)$. One has

$$\Pr[c_i(\alpha_n) = 0] \leq \frac{\deg c_i(x_n)}{p-1} \leq \frac{d}{p-1}$$

for each i. If p is much bigger than d and α_n is random, we expect $c_i(\alpha_n) \neq 0$ for $1 \leq i \leq t$. In the following first we will assume that $c_i(\alpha_n) \neq 0$ for $1 \leq i \leq t$ and compute the expected value of number of terms T_g of g, in terms of T, d and n. Then the upper bounds on the expected value of T_g will be valid even for the case in which some of the $c_i(\alpha_n) = 0$.

Let Y_k be a random variable that counts the number of terms of the k^{th} homogeneous component f_k of f which is homogeneous of degree k in the variables $x_1, \ldots, x_{n-1}$. Then $f = \sum_{k=0}^{d} f_k$ and $\deg_{x_n} f_k \leq d - k$ for $0 \leq k \leq d$.

Example 2. Let $n = 3, d = 6$ and let

$$f = \underbrace{1 + x_3^5}_{f_0} + \underbrace{2x_1x_3^4}_{f_1} + \underbrace{x_1x_2^2(3x_3 + 4x_3^2) + 5x_1^2x_2x_3^3}_{f_3} + \underbrace{x_1^2x_2^2(6x_3 + 7x_3^2)}_{f_4} + \underbrace{8x_1^3x_2^3}_{f_6}$$

with $f_2 = f_5 = 0$. Then $Y_0 = 2$, $Y_1 = 1$, $Y_3 = 3$, $Y_4 = 2$, $Y_6 = 1$, and $Y_2 = Y_5 = 0$.

Let s_k be the number of all possible monomials in the variables $x_1, \ldots, x_{n-1}$ with homogeneous degree k, i.e. $s_k = \binom{n-2+k}{n-2}$ and let $d_k = d - k + 1$ for $0 \le k \le d$. Since the number of all possible monomials in $x_1, \ldots, x_n$ up to degree d which are homogeneous of degree k in the variables $x_1, \ldots, x_{n-1}$ is $s_k d_k$, we have $\sum_{k=0}^{d} s_k d_k = \binom{n+d}{n} = s$ and

$$\Pr[Y_k = j] = \frac{1}{\binom{s}{T}} \binom{s_k d_k}{j} \binom{s - s_k d_k}{T - j}.$$

This is a hypergeometric distribution with expected value and variance (see 26.1.21 in (AS72) with $n = T$ and $p = s_k d_k / s$) where

$$E[Y_k] = T \frac{s_k d_k}{s} \text{ and } \operatorname{Var}[Y_k] = \frac{(s - s_k d_k)(s - T) T s_k d_k}{s^2 (s-1)}.$$

Let X_k be a random variable that counts the number of terms of the k^{th} homogeneous component g_k of g which is homogeneous of degree k in the variables $x_1, \ldots, x_{n-1}$. Let us define the random variable $X = \sum_{k=0}^{d} X_k$ which counts the number of terms T_g of g.

Example 3. Let $n = 3, d = 6$ and $\alpha_3 = 1$. Below $g = f(x_3 = 1)$.

$$\begin{array}{rcccccc}
f = & \underbrace{1 + x_3^5}_{Y_0=2} & + \underbrace{2x_1x_3^4}_{Y_1=1} & + \underbrace{x_1x_2^2(3x_3 + 4x_3^2) + 5x_1^2x_2x_3^3}_{Y_3=3} & + \underbrace{x_1^2x_2^2(6x_3 + 7x_3^2)}_{Y_4=2} & + \underbrace{8x_1^3x_2^3}_{Y_6=1} \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
g = & \underbrace{2}_{X_0=1} & + \underbrace{2x_1}_{X_1=1} & + \underbrace{7x_1x_2^2 + 5x_1^2x_2}_{X_3=2} & + \underbrace{13x_1^2x_2^2}_{X_4=1} & + \underbrace{8x_1^3x_2^3}_{X_6=1}
\end{array}$$

So $Y = \sum_{k=0}^{6} Y_k = 9$ and $X = \sum_{k=0}^{6} X_k = 6$.

The expected number of terms of g is $E[X] = \sum_{k=0}^{d} E[X_k]$. We have

$$\begin{aligned}
E[X_k] = \sum_{i=0}^{s_k} i \Pr[X_k = i] &= \sum_{i=0}^{s_k} i \sum_{j=0}^{s_k d_k} \Pr[X_k = i \,|\, Y_k = j] \Pr[Y_k = j] \\
&= \sum_{j=0}^{s_k d_k} \Pr[Y_k = j] \sum_{i=0}^{s_k} i \Pr[X_k = i \,|\, Y_k = j].
\end{aligned}$$

Our first aim is to find the conditional expectation

$$E[X_k \,|\, Y_k = j] = \sum_{i=0}^{s_k} i \Pr[X_k = i \,|\, Y_k = j].$$

To this end, let $M_k = \{\mathbf{m}_1, \ldots, \mathbf{m}_{s_k}\}$ be the set of all monomials in $x_1, \ldots, x_{n-1}$ with homogeneous degree k. We have $|M_k| = s_k$. For $1 \le i \le s_k$ and $j > 0$, let A_i be the set of all non-zero polynomials in $\mathbb{Z}_p[x_1, \ldots, x_n]$ with j terms that are homogenous of total degree k in the variables $x_1, \ldots, x_{n-1}$, have degree $< d_k$ in the variable x_n and do not include a term of the form $c\mathbf{m}_i x_n^r$ for any $0 \le r < d_k$ for some non-zero $c \in \mathbb{Z}_p$. So using the table below, A_i is the set of all non-zero polynomials whose support does not contain any monomial from the i^{th} row. Note that if $f_k \in A_i$ then $\#f(x_n = \alpha_n) \le s_k - 1$.

	1	x_n	$\dots$	$x_n^{d_k-1}$
$\mathbf{m_1}$	$\mathbf{m_1}$	$\mathbf{m_1}x_n$	$\dots$	$\mathbf{m_1}x_n^{d_k-1}$
$\mathbf{m_2}$	$\mathbf{m_2}$	$\mathbf{m_2}x_n$	$\dots$	$\mathbf{m_2}x_n^{d_k-1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{m_{s_k}}$	$\mathbf{m_{s_k}}$	$\mathbf{m_{s_k}}x_n$	$\dots$	$\mathbf{m_{s_k}}x_n^{d_k-1}$

If f_k is the k^{th} homogeneous component of f in the first $n-1$ variables, then if no zero evaluation occurs in the coefficients of f_k in the variables x_n (as was assumed), we have

$$f_k \in \bigcup_{i=1}^{s_k} A_i \Longleftrightarrow \#f_k(x_1, \dots, x_{n-1}, \alpha) \le s_k - 1.$$

Example 4. Let $n = 3, k = 3, d_3 = 3, j = 4$. We have

$$M_k = \{\mathbf{m}_1 = x_1^3, \mathbf{m}_2 = x_1^2 x_2, \mathbf{m}_3 = x_1 x_2^2, \mathbf{m}_4 = x_2^3\}.$$

with $s_k = 4$. Consider the polynomials

$$F = x_1x_2^2 + x_2^3x_3^2 + x_1^3(x_3 + x_3^2) \in A_2\ ,\ G = x_1x_2^2x_3^2 + x_1^3(1 + x_3 + x_3^2) \in A_2 \cap A_4.$$

So, $\#F(x_n = \alpha_n) \le 4 - 1 = 3$ and $\#G(x_n = \alpha_n) \le 4 - 2 = 2$.

Let us define

$$C_l := \bigcup_{i_1 < \cdots < i_l} (A_{i_1} \cap A_{i_2} \cdots \cap A_{i_l}) \text{ and } B_l := C_l - C_{l+1}$$

for $1 \le l \le s_k - 1$. Then C_l is the union of all possible intersections of the $l-$subsets of the collection $\Gamma = \{A_i, i = 1, \dots, s_k\}$. Observe that $C_l \supseteq C_{l+1}$, so $|B_l| = |C_l| - |C_{l+1}|$. (See Figure 1) Let us also define $B_{s_k} = C_{s_k} = \bigcap_{i=1}^{s_k} A_i$ and

$$b_l := |B_l| \text{ and } m_l := \sum\nolimits_{i_1 < \cdots < i_l} |A_{i_1} \cap A_{i_2} \cdots \cap A_{i_l}| \text{ for } 1 \le l \le s_k.$$

so that $m_1 = \sum_{i=1}^{s_k} |A_i|$ and $m_2 = \sum_{1 \le i < j \le s_k} |A_i \cap A_j|$.

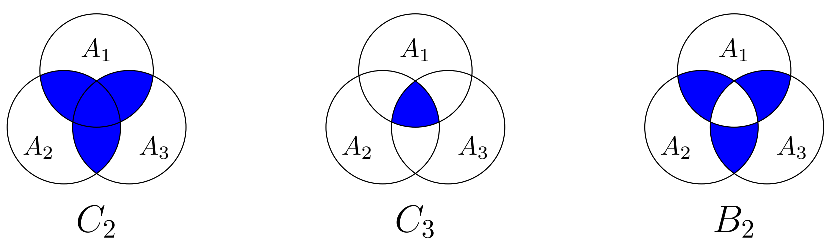

Figure 1. Show sets C_2, C_3, B_2 for three sets A_1, A_2, A_3

With this notation we have

$$f_k \in B_l \Longleftrightarrow \#f_k(x_1, \dots, x_{n-1}, \alpha) = s_k - l.$$

Let $v_j := \binom{s_k d_k}{j}$ and $q := (p-1)$. Assuming that no zero evaluation occurs, we have

$$\Pr[X_k = s_k - l \,|\, Y_k = j] = \frac{|B_l|}{q^j v_j}. \tag{2}$$

It can be seen by counting that

$$\Pr[X_k = l \,|\, Y_k = j] = v_j^{-1} \sum_{i=0}^{l} (-1)^i \tbinom{s_k-(l-i)}{i} \tbinom{s_k}{l-i} \tbinom{d_k(l-i)}{j}.$$

But this formula is not easy to manipulate. So, to compute the expected value and the variance of X, we will follow the easier way described in (MT16a).

We have $|A_i| = (p-q)^j \binom{(s_k-1)d_k}{j}$ and $|A_i \cap A_l| = q^j \binom{(s_k-2)d_k}{j}$ for $1 \le i, l \le s_k$ where $i \neq l$. It has been proven in (MT16a) that $\sum_{i=1}^{s_k} i|B_i| = m_1$ and $\sum_{i=1}^{s_k} i^2|B_i| = m_1 + 2m_2$. To find the expected value and the variance of X, let us first define $w_j := \binom{(s_k-1)d_k}{j}$ and the random variable $Z_k := s_k - X_k$. Then $Z_k = i \iff X_k = s_k - i$. Recall that $v_j := \binom{s_k d_k}{j}$. Then we have

$$\sum_{i=1}^{s_k} i\Pr[Z_k = i \,|\, Y_k = j] \overset{(1)}{=} \sum_{i=1}^{s_k} i\frac{|B_l|}{q^j v_j} = \frac{\sum_{i=1}^{s_k} i|B_l|}{q^j v_j} = \frac{\sum_{i=1}^{s_k} |A_i|}{q^j v_j} = s_k \frac{w_j}{v_j}.$$

Since $E[X_k \,|\, Y_k = j] = E[s_k - Z_k \,|\, Y_k = j] = s_k - E[Z_k \,|\, Y_k = j]$, we have

$$E[X_k \,|\, Y_k = j] = \sum_{i=0}^{s_k} i \Pr[X_k = i \,|\, Y_k = j] = s_k(1 - \frac{w_j}{v_j}).$$

To save some space let $y_{kj} := \Pr[Y_k = j]$. Then

$$\begin{aligned}
E[X] &= \sum_{k=0}^{d} E[X_k] = \sum_{k=0}^{d} \sum_{j=0}^{s_k d_k} y_{kj} \sum_{i=0}^{s_k} i \Pr[X_k = i \,|\, Y_k = j] \\
&= \sum_{k=0}^{d} \sum_{j=0}^{s_k d_k} y_{kj} s_k (1 - \frac{w_j}{v_j}) = \sum_{k=0}^{d} \sum_{j=0}^{s_k d_k} y_{kj} s_k - \sum_{k=0}^{d} \sum_{j=0}^{s_k d_k} y_{kj} s_k \frac{w_j}{v_j} \\
&= \sum_{k=0}^{d} s_k \sum_{j=0}^{s_k d_k} y_{kj} - \sum_{k=0}^{d} \sum_{j=0}^{s_k d_k} \frac{1}{\binom{s}{T}} \binom{s_k d_k}{j} \binom{s - s_k d_k}{T-j} s_k \frac{\binom{(s_k-1)d_k}{j}}{\binom{s_k d_k}{j}} \\
&= \sum_{k=0}^{d} s_k - \frac{1}{\binom{s}{T}} \sum_{k=0}^{d} s_k \sum_{j=0}^{s_k d_k} \binom{s - s_k d_k}{T-j} \binom{(s_k-1)d_k}{j} \\
&= \sum_{k=0}^{d} s_k - \sum_{k=0}^{d} s_k \frac{\binom{s-d_k}{T}}{\binom{s}{T}} = \sum_{k=0}^{d} s_k \left(1 - \frac{\binom{s-d_k}{T}}{\binom{s}{T}}\right).
\end{aligned}$$

Note that what we have done so far can easily be generalized when the number of evaluation points $m > 1$ and redefining s_k and d_k. For $0 < m < n$ let

$$g = f(x_1, \ldots x_{n-m}, x_{n-m+1} = \alpha_{n-m+1}, \ldots, x_n = \alpha_n)$$

for m randomly chosen non-zero elements $\alpha_{n-m+1}, \ldots, \alpha_n \in \mathbb{Z}_p$ and $s = \binom{n+d}{n}$, $s_k = \binom{n-m-1+k}{n-m-1}$ and $d_k = \binom{d-k+m}{m}$. If no zero evaluation occurs at the coefficients of f and we define the random variable $Y = \sum_{k=0}^{d} Y_k$ which counts the number of terms of f, then what we get is the conditional expectation

$$E[X \mid Y = T] = \sum_{k=0}^{d} s_k \left(1 - \frac{\binom{s-d_k}{T}}{\binom{s}{T}} \right). \tag{3}$$

From now on, to save some space, when it is clear from the context, we will use the notation $E[X]$ instead of $E[X \mid Y = T]$. Although it is not difficult to compute, this formula is not useful. In order to have a smooth formulation in our complexity analysis, we want a good approximation. First, we observe

$$\binom{s-d_k}{T} / \binom{s}{T} = (1 - \frac{T}{s})(1 - \frac{T}{s-1}) \cdots (1 - \frac{T}{s-d_k+1}).$$

For $0 \le i < d_k$ we have $(1 - \frac{T}{s}) - (1 - \frac{T}{s-i}) = \frac{iT}{s(s-i)} < \frac{d_k T}{s(s-d_k)}$. Let $\gamma_i := \frac{iT/s}{s-i}$. Then

$$\prod_{i=0}^{d_k-1} (1 - \frac{T}{s-i}) = \prod_{i=0}^{d_k-1} (1 - \frac{T}{s} - \gamma_i) = (1 - \frac{T}{s})^{d_k} + Er$$

where

$$Er = \sum_{l=1}^{d_k} (-1)^l \left(\sum_{0 \le i_1 < \cdots < i_l \le d_k - 1} \prod_{j=1}^{l} \gamma_{i_j} \right) (1 - \frac{T}{s})^{d_k - l}.$$

Since

$$\prod_{i=1}^{l} \gamma_i < \left(\frac{d_k T/s}{s-d} \right)^l = \left(\frac{d_k}{s} \right)^l \left(\frac{T/s}{1 - d_k/s} \right)^l \to 0 \text{ as } \frac{d_k}{s} \to 0$$

we see that $Er \to 0$ and hence the ratio $\binom{s-d_k}{T} / \binom{s}{T} \to (1 - \frac{T}{s})^{d_k}$ as $\frac{d_k}{s} \to 0$.

Remark 5. From now on, unless indicated, whenever we use the symbol $\approx$ or $\lesssim$ we mean that in the calculation the approximation

$$\binom{s-d_k}{T} / \binom{s}{T} \approx (1 - t_f)^{d_k} \tag{4}$$

is used (with error Er) where $t_f = T/s$ is the density ratio of f. When m is not close to n, since $s \in \mathcal{O}(n^d)$ and in the sparse case t_f is relatively small, the error Er is very close to zero not only asymptotically but also for practical values for n and d (see example 6).

For a single evaluation, that is, $m = 1$, we expect

$$E[X] \approx \sum_{k=0}^{d} s_k \left(1 - (1 - t_f)^{d_k} \right). \tag{5}$$

So, in the dense case where t_f is very close to 1, we expect approximately $\sum_{k=0}^{d} s_k$ many terms, i.e. most of the possible monomials in the variables $x_1, \ldots, x_{n-1}$ up to degree d. In the very sparse case where t_f is very close to zero, using the approximation $(1-t_f)^{d_k} \approx 1 - d_k t_f$, we expect approximately $t_f \sum_{k=0}^{d} s_k d_k = t_f s = T$ terms as we intuitively expect.

Eqn (2) above is the expected number of terms $E[T_g]$ of g. Let γ be the number of all possible monomials in the variables $x_1, \ldots, x_{n-1}$ up to degree $d - 1$, i.e. $\gamma = \binom{n+d-1}{n-1}$.

Dividing the both sides of the equation (2) by γ we get the expected density ratio

$$E[T_g]/\gamma = E[T_g/\gamma] \Rightarrow E[t_g] = 1 - \gamma^{-1} \sum_{k=0}^{d} s_k \frac{\binom{s-d_k}{st_f}}{\binom{s}{st_f}}. \tag{6}$$

So, we have an induced function $e_t : t_f \mapsto E[t_g]$. Using eqn (4)

$$E[t_g] \approx 1 - \gamma^{-1} \sum_{k=0}^{d} s_k (1 - t_f)^{d_k}. \tag{7}$$

Example 6. Table 1 below presents the results of experiments with 4 random polynomials f_1, f_2, f_3, f_4 with $n = 7$ variables and degree $d = 15$. T_{g_i} and t_{g_i} are the actual number of terms and the density ratio of each $g_i = f_i(x_n = \alpha_i)$. $E[t_{g_i}]$ and eT_{g_i} are the expected number of terms of g_i based on Eqns (3) and (5) resp. $E[t_{g_i}]$ and et_{g_i} are the expected density ratio of g_i based on Eqns (6) and (7) resp.

	T_{f_i}	t_{f_i}	T_{g_i}	$E[T_{g_i}]$	eT_{g_i}	t_{g_i}	$E[t_{g_i}]$	et_{g_i}
f_1	17161	.100625	14356	14370.47	14370.36	.264558	.264825	.264823
f_2	19887	.116609	16196	16221.84	16221.73	.298466	.298943	.298941
f_3	29845	.174998	22303	22211.09	22210.96	.411009	.409315	.409313
f_4	39823	.233505	27244	27199.53	27199.41	.502063	.501244	.501242

Table 1. Expected number of terms after evaluation

Note that the polynomial function $e_t(y) = 1 - \gamma^{-1} \sum_{k=0}^{d} s_k (1-y)^{d_k}$ is strictly increasing on the interval $[0, 1]$, since $e_t'(y) > 0$ on the interval [0,1]. Also $e_t(0) = 0$ and $e_t(1) = 1$. For a given $0 \leq y_0 \leq 1$, consider the function $h(y) := e_t(y) - y_0$. We have $h(0) \leq 0$ and $h' > 0$ on [0,1]. Hence $h(y)$ has only 1 real root in $[0, 1]$. This helps us to estimate T_f when we're given only T_g. Here is one example in the reverse direction:

Example 7. We call Maple's `randpoly` command to give a random sparse polynomial of degree 15 in 7 variables. It gives us a polynomial f with $T_f = 25050$. Suppose we don't know T_f. Then we choose a random point α and evaluate $g = f(x_n = \alpha)$. We compute $T_g = 19395$. Then we compute $t_g = 19395/\binom{7-1+15}{15} \approx 0.3574192835$ and $\gamma = \binom{7+15-1}{15}$. Using (6.6) we seek for the solutions of the polynomial equation

$$0.3574192835 = 1 - \gamma^{-1} \sum_{k=0}^{15} \binom{5+k}{k} (1-y)^{16-k}.$$

This polynomial equation has only one real root $y = 0.1461603065$ in [0,1]. So we guess $E[t_f] \approx 0.1461603065$. The actual density ratio is $t_f = 25050/\binom{7+15}{15} = 0.1461089220$. Our guess implies $E[T_f] \approx t_f \binom{7+15}{7} = 24926$, whereas $T_f = 25050$. We repeated this example with 4 random polynomials f_i where $g_i = f_i(x_n = \alpha_i)$. The results of the experiments are in Table 2.

	Tg_i	t_{g_i}	$E[t_{f_i}]$	t_{f_i}	$E[T_{f_i}]$	T_{f_i}
f_1	14967	.2758182220	.1056824733	.1052983394	18023	17958
f_2	14597	.2689997051	.1025359262	.1020792288	17486	17409
f_3	14439	.2660880142	.1012024458	.1008713294	17259	17203
f_4	14375	.2649085950	.1006640188	.1005605592	17167	17150

Table 2. Expected number of terms before evaluation

Finally, following the notation in the beginning of the section, if some of the $c_i(\alpha_n) = 0$ for $1 \le i \le t$, then we consider $\tilde{f} = \sum_{k=0}^{t_{i_k}} c_{i_k}(x_n)\mathbf{m}_{i_k}$ where $\mathrm{Supp}(\tilde{f}) \subseteq \mathrm{Supp}(f)$ and $c_{i_k}(\alpha_n) = 0$. Then

$$E[T_f] = E[T_{f-\tilde{f}}] \lesssim \sum_{k=0}^{d} s_k \left(1 - (1 - t_f)^{d_k}\right)$$

So what we have found is an upper bound for $E[X]$. On the other hand since we choose $\alpha_n \neq 0$ a non-zero monomial does not evaluate to zero, that is, if $\#c_i(x_n) = 1$ then $\mathbf{m}_i c_i(\alpha_n) \neq 0$. So we should apply the zero evaluation probability only to the terms $\mathbf{m}_i c_i(x_n)$ with $\#c_i(x_n) \ge 2$. Then, for given $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$ of degree $\le d$ with T terms, the probability that zero evaluation occurs for a randomly chosen non-zero $\alpha_n \in \mathbb{Z}_p$ is $\le \frac{dT}{2(p-1)}$. Note that this is also true when the number of evaluations are more than one by using the Swartz-Zippel Lemma.

To see how spread out the distribution from the mean is, we must compute the variance. First consider the sum $A = \sum_{i=0}^{s_k} i^2 \Pr[Z_k = i \,|\, Y_k = j]$

$$\begin{aligned}
A &= \sum_{i=0}^{s_k} (s_k - i)^2 \Pr[Z_k = s_k - i \,|\, Y_k = j] \\
&= s_k^2 - 2s_k \sum_{i=0}^{s_k} i \Pr[X_k = i \,|\, Y_k = j] + \sum_{i=0}^{s_k} i^2 \Pr[X_k = i \,|\, Y_k = j] \\
&= s_k^2 - 2s_k(s_k - s_k \frac{w_j}{v_j}) + \sum_{i=0}^{s_k} i^2 \Pr[X_k = i \,|\, Y_k = j] \\
&= -s_k^2(1 - 2\frac{w_j}{v_j}) + \sum_{i=0}^{s_k} i^2 \Pr[X_k = i \,|\, Y_k = j]
\end{aligned}$$

Then

$$\sum_{i=0}^{s_k} i^2 \Pr[X_k = i \,|\, Y_k = j] = \sum_{i=0}^{s_k} i^2 \Pr[Z_k = i \,|\, Y_k = j] + s_k^2(1 - 2\frac{w_j}{v_j}).$$

Let $r_j := \binom{(s_k-2)d_k}{j}$. Recall that $q = p - 1$, $|A_i| = q^j w_j$ and $|A_i \cap A_l| = q^j v_j$ for $i \neq l$. So

$$\begin{aligned}
\sum_{i=1}^{s_k} i^2 \Pr[Z_k = i \,|\, Y_k = j] &= \sum_{i=1}^{s_k} i^2 \frac{|B_l|}{q^j v_j} = \frac{\sum_{i=1}^{s_k} i^2 |B_l|}{q^j v_j} \\
&= \frac{\sum_{i=1}^{s_k} |A_i|}{q^j v_j} + 2\frac{\sum_{1 \le i,l \le s_k} |A_i \cap A_l|}{q^j v_j} = s_k \frac{w_j}{v_j} + 2\binom{s_k}{2}\frac{r_j}{v_j}.
\end{aligned}$$

Then

$$\sum_{i=0}^{s_k} i^2 \Pr[X_k = i \mid Y_k = j] = s_k \frac{w_j}{v_j} + 2\binom{s_k}{2}\frac{r_j}{v_j} + s_k^2(1 - 2\frac{w_j}{v_j}).$$

Hence

$$\begin{aligned}\mathrm{Var}[X_k] = E[X_k^2] - E[X_k]^2 &= s_k \frac{w_j}{v_j} + 2\binom{s_k}{2}\frac{r_j}{v_j} + s_k^2(1 - 2\frac{w_j}{v_j}) - s_k^2(1 - \frac{w_j}{v_j})^2 \\ &= s_k \frac{w_j}{v_j} + s_k(s_k - 1)\frac{r_j}{v_j} - \left(s_k \frac{w_j}{v_j}\right)^2.\end{aligned}$$

It follows that

$$\begin{aligned}\sum_{k=0}^{d} \mathrm{Var}[X_k] &\approx \sum_{k=0}^{d} s_k (1-t_f)^{d_k} + \sum_{k=0}^{d} s_k(s_k - 1)(1-t_f)^{2d_k} - \sum_{k=0}^{d} s_k^2 (1-t_f)^{2d_k} \\ &= \sum_{k=0}^{d} s_k \left((1-t_f)^{d_k} - (1-t_f)^{2d_k}\right).\end{aligned}$$

Note that $\sum_{k=0}^{d} \mathrm{Var}[X_k]$ is not not equal to $\mathrm{Var}[X] = \sum_{k=0}^{d} \mathrm{Var}[X_k] + \sum_{k \neq l}^{d} \mathrm{Covar}[X_k, X_l]$. For the sparse case where t_f is close to zero, using the approximation $(1-t_f)^{d_k} \approx 1 - d_k t_f$, we expect

$$\begin{aligned}\sum_{k=0}^{d} s_k \left((1-t_f)^{d_k} - (1-t_f)^{2d_k}\right) &\approx \sum_{k=0}^{d} s_k \left((1 - d_k t_f) - (1 - 2 d_k t_f)\right) \\ &= t_f \sum_{k=0}^{d} s_k d_k = t_f s = T_f.\end{aligned}$$

So, the sum of the squares of the deviations of each X_k from $E[X_k]$ is T_f. As an experiment, for $n = 7$, $d = 15$ we generated 1000 random polynomials with $T_f = 1716$ for each of them. So each has density ratio $t_f = 0.01$. Then the expected number of terms after evaluation at a random non-zero point is 1684.14 and $\sum_{k=0}^{d} \mathrm{Var}[X_k] = 1606.30$ which is close to T_f.

Lemma 8. *Let p be a big prime and $f \in \mathbb{Z}_p[x_1, \dots, x_n]$ be a random multivariate polynomial of degree of at most d that has T_f non-zero terms. Also let $0 < m < n$ and*

$$g = f(x_1, \dots x_{n-m}, x_{n-m+1} = \alpha_{n-m+1}, \dots, x_n = \alpha_n)$$

for m randomly chosen non-zero elements $\alpha_{n-m+1}, \dots, \alpha_n \in \mathbb{Z}_p$. Let T_g be the expected number of terms of g and T_{g_k} be the number of monomials in g that are homogeneous of degree k in the variables $x_1, \dots x_{n-m}$. Also let $s = \binom{n+d}{n}$, $s_k = \binom{n-m-1+k}{n-m-1}$, $\gamma = \binom{n-m+d}{n-m}$ and $d_k = \binom{d-k+m}{m}$. Then

$$E[T_g] \leq \sum_{k=0}^{d} s_k \left(1 - \frac{\binom{s-d_k}{T_f}}{\binom{s}{T_f}}\right). \tag{8}$$

Let the density of f, $t_f = T_f/s$. Equation (7) implies that if $\frac{d_k}{s} \to 0$, then with probability $\geq 1 - \frac{dT_f}{2(p-1)}$ one has

$$E[T_g] \approx \sum_{k=0}^{d} s_k \left(1 - (1-t_f)^{d_k}\right). \tag{9}$$

Equation (8) implies that with probability $\geq 1 - \frac{dT_f}{2(p-1)}$

$$E[t_g] \approx 1 - \gamma^{-1} \sum_{k=0}^{d} s_k (1-t_f)^{d_k} \tag{10}$$

$$\text{and } \operatorname{Var}[T_{g_k}] \approx s_k \left((1-t_f)^{d_k} - (1-t_f)^{2d_k}\right). \tag{11}$$

4.1. On Zippel's assumption

The sparse interpolation idea and the first gcd algorithm to use sparse interpolation were introduced and analyzed by Zippel in his research paper (Zip79). The goal polynomial (the gcd) is $P(x_1, \ldots, x_n)$ and the starting point is $\vec{a} = (a_1, \ldots, a_n)$. According to our notation, $T_{f_{n-i}}$ denotes the number of terms of the polynomial $P(x_1, \ldots, x_i, a_{i+1}, \ldots, a_n)$. In subsection 3.2 of (Zip79) after computing a sum which depends on the values $T_{f_{n-i}}$, Zippel claims "We need to make some assumptions about the structure of $T_{f_{n-i}}$ to get anything meaningful out of this. We will assume that the ratio of the terms $T_{f_{n-i}}/T_{f_{n-i+1}}$ is a constant k." [3]

Our observations in this section show that this assumption is wrong. To see a more accurate bound on the expected ratio of the subsequent number of terms, let us denote by $d_k^{(i)} = \binom{d-k+i}{d-k}$, $s^{(i)} = \binom{n-i+d}{n-i}$, $s_k^{(i)} = \binom{n-i+k-1}{n-i-1}$, $r_k^{(i)} = \binom{s-d_k^{(i)}}{T}/\binom{s}{T}$ and $\beta_k^{(i)} = 1 - r_k^{(i)}$. Let

$$f_i = f(x_1, \ldots x_{n-i}, x_{n-i+1} = \alpha_{n-i+1}, \ldots, x_n = \alpha_n)$$

be the polynomial in $n-i$ variables after i random evaluations. According to Lemma 8 $E[T_{f_i}] = \sum_{k=0}^{d} s_k^{(i)} \beta_k^{(i)}$. Our aim is to find an upper and lower bound for

$$\frac{E[T_{f_i}]}{E[T_{f_{i+1}}]} = \frac{\sum_{k=0}^{d} s_k^{(i)} \beta_k^{(i)}}{\sum_{k=0}^{d} s_k^{(i+1)} \beta_k^{(i+1)}}.$$

Observe that

$$d_k^{(i+1)} > d_k^{(i)} \Rightarrow s - d_k^{(i+1)} < s - d_k^{(i)} \Rightarrow r_k^{(i+1)} < r_k^{(i)} \Rightarrow \beta_k^{(i+1)} > \beta_k^{(i)}.$$

Then

$$\frac{E[T_{f_i}]}{E[T_{f_{i+1}}]} = \frac{\sum_{k=0}^{d} s_k^{(i)} \beta_k^{(i)}}{\sum_{k=0}^{d} s_k^{(i+1)} \beta_k^{(i+1)}} < \frac{\sum_{k=0}^{d} s_k^{(i)} \beta_k^{(i)}}{\sum_{k=0}^{d} s_k^{(i+1)} \beta_k^{(i)}} \leq \max_k \left\{ \frac{s_k^{(i)}}{s_k^{(i+1)}} \right\}.$$

We have

$$\frac{s_k^{(i)}}{s_k^{(i+1)}} = \frac{n-i+k-1}{n-i-1} = 1 + \frac{k}{n-i-1} \leq 1 + \frac{d}{n-i-1}.$$

[3] In (Zip79) Zippel uses the notation t_i for $T_{f_{n-i}}$. Since we use the symbol t for the density, we use our notation as to not confuse the reader.

Our next aim is to show that $E[t_{f_{i+1}}] \geq E[t_{f_i}]$: we have

$$E[t_{f_i}] = 1 - \sum_{k=0}^{d} \frac{s_k^{(i)}}{s^{(i)}} r_k^{(i)} = 1 - \sum_{k=0}^{d} \frac{\binom{n-i+k-1}{n-i-1}}{\binom{n-i+d}{n-i}} \frac{\binom{s-d_k^{(i)}}{T}}{\binom{s}{T}} = 1 - \sum_{k=0}^{d} \frac{\binom{n-i+k-1}{n-i-1}}{\binom{s}{T}} \frac{\binom{s-d_k^{(i)}}{T}}{\binom{n-i+d}{n-i}}$$

Let $v_k^{(i)} = \binom{n-i+k-1}{n-i-1}/\binom{s}{T}$ and $w_k^{(i)} = \binom{s-d_k^{(i)}}{T}/\binom{n-i+d}{n-i}$. Then $v_k^{(i+1)}/v_k^{(i)} < 1$ and $w_k^{(i+1)}/w_k^{(i)} < 1$. So $\sum_{k=0}^{d} v_k^{(i)} w_k^{(i)}$ decreases and hence $E[t_{f_i}]$ increases as i increases, i.e., we expect an increase in the density ratio after each evaluation. On the other hand,

$$E[T_{f_i}] = E[t_{f_i}]\binom{n-i+d}{n-i} = E[t_{f_i}]\tfrac{n-i+d}{n-i}\binom{n-(i+1)+d}{n-(i+1)} \geq E[t_{f_i}]\tfrac{n-i+d}{n-i}E[T_{f_{i+1}}].$$

It follows that

$$\tfrac{E[T_{f_i}]}{E[T_{f_{i+1}}]} \geq E[t_{f_i}]\left(1 + \tfrac{d}{n-i}\right).$$

Hence

$$E[t_{f_i}]\left(1 + \tfrac{d}{n-i}\right) \leq \tfrac{E[T_{f_i}]}{E[T_{f_{i+1}}]} \leq 1 + \tfrac{d}{n-i-1}.$$

Now, since each of $E[t_{f_i}], 1 + \frac{d}{n-i}, 1 + \frac{d}{n-i-1}$ increases as i increases we expect an increase in the ratio $E[T_{f_i}]/E[T_{f_{i+1}}]$.

This means we expect that the ratio $T_{f_{n-i}}/T_{f_{n-i+1}}$ increases as i decreases, i.e., after each evaluation we expect an increase in the ratio of subsequent number of terms. See Example 9 for a sparse case and Example 10 for a (relatively) dense case.

Example 9. (**Sparse case with $\mathbf{t_f = 0.000044}$**). Table 3 below shows the result of a random experiment where $p = 2^{31} - 1$, $n = 12, d = 20, T_f = 10^4, t_f = 0.000044$ and $f_i := f_{i-1}(x_{n-i+1} = \alpha_{n-i+1})$ with $f_0 = f$, for randomly chosen non-zero α_i's in $\mathbb{Z}_p$. Observe that when we evaluate the first 4 variables the number of terms didn't drop significantly.

i	T_{f_i}	$E[T_{f_i}]$	t_{f_i}	$t_{f_i}(1+\frac{d}{n-i})$	$T_{f_i}/T_{f_{i+1}}$	$1+\frac{d}{n-i-1}$
0	10000	10000	0.00004	0.00012	1.0000	2.8182
1	10000	9999.32	0.00012	0.00033	1.0006	3.0000
2	9994	9995.19	0.00033	0.00100	1.0025	3.2222
3	9969	9971.08	0.00100	0.00321	1.0135	3.5000
4	9836	9837.31	0.00316	0.01108	1.0686	3.8571
5	9205	9200.70	0.01037	0.03998	1.2767	4.3333
6	7210	7219.37	0.03132	0.13570	1.7470	5.0000
7	4127	4144.61	0.09710	0.48548	2.4435	6.0000
8	1689	1690.52	0.23090	1.38540	3.3512	7.6667
9	504	497.93	0.37895	2.90530	4.8932	11.000
10	103	104.50	0.54210	5.96310	7.3571	21.000

Table 3. The ratio of subsequent number of terms (sparse case): T_{f_i} decreases slowly until $i = 5$.

Example 10. (**Dense case with $\mathbf{t_f = 0.1}$**). Table 4 below shows the result of a random experiment where $p = 2^{31}-1$, $n = 7$, $d = 13$, $T_f = 7752$, $t_f = 0.1$ where $f_i := f_{i-1}(x_{n-i+1} = \alpha_{n-i+1})$ with $f_0 = f$, for randomly chosen non-zero α_i's in $\mathbb{Z}_p$. Observe that the number of terms dropped by about 10% after the first evaluation.

i	T_{f_i}	$E[T_{f_i}]$	t_{f_i}	$t_{f_i}(1+\frac{d}{n-i})$	$T_{f_i}/T_{f_{i+1}}$	$1+\frac{d}{n-i-1}$
0	7752	7752	0.10	0.28	1.17	3.16
1	6670	6643.44	0.24	0.77	1.76	3.60
2	3774	3800.14	0.44	1.58	2.70	4.25
3	1398	1409.83	0.58	2.49	3.65	5.33
4	383	394.03	0.68	3.64	4.78	7.50
5	80	81.14	0.76	5.71	6.66	14

Table 4. The ratio of subsequent number of terms (dense case): T_{f_i} decreases immediately.

The dominating term of the complexity analysis of Zippel is $\sum_{i=1}^{n} c_1 dT^3_{f_{n-i+1}}$ where c_1 is a constant. He assumes $T_{f_{n-i}}/T_{f_{n-i+1}} = k \Rightarrow T_{f_{n-i}} = T_{f_n} k^i$. It follows that

$$\sum_{i=1}^{n} c_1 dT^3_{f_{n-i+1}} = c_1 d \sum_{i=1}^{n} T^3_{f_{n-i+1}} = c_1 dT^3_{f_n} \sum_{i=1}^{n} k^{3i} = c_1 dk^3 \frac{(k^{3n}-1)T^3_{f_n}}{k^3-1}.$$

Since $T_f = T_{f_0} = T_{f_n} k^n$, if k is large in comparison to 1 then this sum approaches $c_1 dT_f^3$ but if k is very close to 1 then this sum approaches to $c_1 ndT_f^3$. Then he concludes that if $T_f \gg d$ or n, the dominant behaviour is $\mathcal{O}(T_f^3)$.

The case where k is very close to 1 (where Zippel has a very sparse case in mind) is the worst case analysis. One can construct such an example. More precisely, Zippel's analysis shows that the complexity for the worst case is $\mathcal{O}(ndT_f^3)$ and for the average case (where he assumes that "k is large in comparison to 1") the complexity is $\mathcal{O}(dT_f^3)$. Below we will show that this is not true and prove that even for the average case the expected complexity is $\mathcal{O}(ndT_f^3)$ for the sparse gcd computation.

Example 9 (the sparse case) shows that after 5 evaluations the number of terms is still 90% of T_f (See column T_f). This means the cost of each interpolation of the last few variables in Zippel's algorithm is the same and equal to $\mathcal{O}(dT_f^3)$. However in Example 10 (the relatively dense case) the number of terms starts to drop right away. Proposition 11 below qualifies this behaviour and makes the observation of Example 9 more precise.

Proposition 11. *Following the notation above, let* $\mathcal{I} = \left\{ i \in \mathbb{N} \mid t_f \leq 1/\binom{i+d}{d} \text{ and } i \leq n \right\}$. *Then with probability* $\geq 1 - \frac{dT_f}{2(p-1)}$, *one has*

(i) $\frac{E[T_{f_i}]}{T_f} \geq 1 - t_f^2$ for $i \in \mathcal{I}$,

(ii) if $n \leq d$, *then* $|\mathcal{I}| \geq \max\{1, \lceil n - \log_r T_f \rceil\}$ *where* $r = 1 + \frac{d}{n}$.

Proof. To save some space we will use the notation $T = T_f$. With probability $\geq 1 - \frac{dT}{2(p-1)}$, one has

$$E[T_{f_i}] = \sum_{k=0}^{d} s_k^{(i)} \left(1 - \frac{\binom{s-d_k^{(i)}}{s}}{\binom{s}{T}}\right)$$

where $d_k^{(i)} = \binom{d-k+i}{i}$. Note that

$$t_f \leq \frac{1}{\binom{i+d}{d}} \leq \frac{1}{d_k^{(i)}} \Rightarrow t_f d_k^{(i)} \leq 1. \tag{12}$$

It follows that

$$\frac{d_k^{(i)}}{s} = \frac{d_k^{(i)} t_f}{T} \leq \frac{1}{T}. \tag{13}$$

We have

$$\frac{\binom{s-d_k^{(i)}}{s}}{\binom{s}{T}} \leq (1-t_f)^{d_k^{(i)}}. \tag{14}$$

Then

$$\begin{aligned}
E[T_{f_i}] &= \sum_{k=0}^{d} s_k^{(i)} \left(1 - \frac{\binom{s-d_k^{(i)}}{s}}{\binom{s}{T}}\right) \overbrace{\geq}^{\text{by } (14)} \sum_{k=0}^{d} s_k^{(i)} \left(1-(1-t_f)^{d_k^{(i)}}\right) \\
&= \sum_{k=0}^{d} s_k^{(i)} \left(1 - \sum_{j=0}^{d_k^{(i)}} (-1)^j \binom{d_k^{(i)}}{j} t_f^j\right) \\
&= \sum_{k=0}^{d} s_k^{(i)} \left(1 - 1 + d_k^{(i)} t_f - \sum_{j=2}^{d_k^{(i)}} (-1)^j \binom{d_k^{(i)}}{j} t_f^j\right) \\
&= \sum_{k=0}^{d} s_k^{(i)} d_k^{(i)} t_f - \sum_{k=0}^{d} \sum_{j=2}^{d_k^{(i)}} (-1)^j s_k^{(i)} \binom{d_k^{(i)}}{j} t_f^j \\
&= t_f s - t_f^2 \sum_{k=0}^{d} \sum_{j=2}^{d_k^{(i)}} (-1)^j s_k^{(i)} \binom{d_k^{(i)}}{j} t_f^{j-2} \\
&= T - t_f^2 \sum_{k=0}^{d} \sum_{j=2}^{d_k^{(i)}} (-1)^j s_k^{(i)} \binom{d_k^{(i)}}{j} t_f^{j-2}.
\end{aligned}$$

The expected decrease in the number of terms is determined by the quantity

$$A = t_f^2 \sum_{k=0}^{d} \sum_{j=2}^{d_k^{(i)}} (-1)^j s_k^{(i)} \binom{d_k^{(i)}}{j} t_f^{j-2}.$$

We have

$$s_k^{(i)} \binom{d_k^{(i)}}{j} t_f^{j-2} < s_k^{(i)} \frac{\left(d_k^{(i)}\right)^j}{j!} t_f^{j-2} \overbrace{\leq}^{\text{by } (12)} \frac{s_k^{(i)} \left(d_k^{(i)}\right)^2}{j!}.$$

Then the absolute value of A is

$$|A| \leq t_f^2 \sum_{k=0}^{d} \sum_{j=2}^{d_k^{(i)}} \frac{s_k^{(i)} \left(d_k^{(i)}\right)^2}{j!} = t_f^2 \sum_{k=0}^{d} s_k^{(i)} \left(d_k^{(i)}\right)^2 \underbrace{\sum_{j=2}^{d_k^{(i)}} \frac{1}{j!}}_{\leq 1} \leq t_f^2 \sum_{k=0}^{d} s_k^{(i)} \left(d_k^{(i)}\right)^2.$$

So we see that

$$\frac{|A|}{T} \leq \frac{t_f^2 \sum_{k=0}^{d} s_k^{(i)} \left(d_k^{(i)}\right)^2}{t_f s} = t_f \sum_{k=0}^{d} s_k^{(i)} d_k^{(i)} \frac{d_k^{(i)}}{s} \overbrace{\leq}^{\text{by (13)}} t_f \frac{1}{T} s = t_f^2. \tag{15}$$

Then

$$\frac{E[T_{f_i}]}{T} \geq \frac{T - A}{T} \geq \frac{T - |A|}{T} = 1 - \frac{|A|}{T} \overbrace{\geq}^{\text{by(15)}} 1 - t_f^2.$$

This proves the first part. For the second part, note that

$$t_f \leq \frac{1}{\binom{i+d}{d}} \iff T \leq \frac{\binom{n+d}{d}}{\binom{i+d}{d}} \iff \frac{1}{T} \geq \frac{\binom{i+d}{d}}{\binom{n+d}{d}}$$

and that $\frac{n-i}{n+d} = \frac{n}{n+d} - \frac{i}{n+d} = \frac{1}{1+d/n} - \frac{i}{n+d}$. So if $n \leq d$ then $\frac{n-i}{n+d}$ is small and the following bound is useful

$$\frac{\binom{i+d}{d}}{\binom{n+d}{d}} = \frac{\binom{n+d-(n-i)}{d}}{\binom{n+d}{d}} \leq \left(1 - \frac{d}{n+d}\right)^{n-i}.$$

Now if

$$T \leq \left(1 - \frac{d}{n+d}\right)^{i-n} \iff \frac{1}{T} \geq \left(1 - \frac{d}{n+d}\right)^{n-i} \Rightarrow i \in \mathcal{I}$$

and if we define $w = \frac{n}{n+d}$, then $w^n T \leq w^i \Rightarrow i \geq n + \log_w T$. Finally, if $r = \frac{1}{w} = 1 + \frac{d}{n}$, we have $i \geq n - \log_r T$. Hence if $n \leq d$, then $|\mathcal{I}| \geq \lceil n - \log_r T \rceil$. $\square$

For Example 9, $\max\{1, \lceil n - \log_r T_f \rceil\} = \max\{1, \lceil 12 - \log_{2.6} 10^4 \rceil\} = \max\{1, 3\} = 3$ whereas $|\mathcal{I}| = 5$. For Example 10, $|\mathcal{I}| = 1$ whereas $\max\{1, \lceil n - \log_r T_f \rceil\} = \max\{1, \lceil 7 - \log_{2.86} 7752 \rceil\} = \max\{1, -1\} = 1$.

Corollary 12. *Following the notation above, the average complexity of the sparse gcd algorithm of Zippel is* $\Omega(|I| d T^3)$. *If* $n \leq d$ *then the average complexity is in* $\Omega(\max\{1, \lceil n - \log_r T \rceil\} d T^3)$ *where* $r = 1 + \frac{d}{n}$.

Remark 13. According to Proposition 11 (i), when t_f is small, on average we expect that $T_{f_i} \approx T_f$ for $i \in \mathcal{I}$, i.e. for at least $|\mathcal{I}|$ many steps, we don't expect a significant decrease in the number of terms. Note that as t_f gets smaller, i.e., the inputs gets sparser, $|\mathcal{I}|$ gets closer to n. Also, in practice $n \leq d$ and as T_f get smaller $\log_r T_f$ gets smaller and according to Proposition (ii) $|\mathcal{I}| \geq \lceil n - \log_r T_f \rceil$ gets closer to n. This means for the sparse case the average complexity is in $\mathcal{O}(n d T_f^3)$. Thus, the common perception that in the sparse interpolation most of the work is done when recovering the last variable x_n is not true: For most sparse examples, the work done to recover $x_{n/2}, x_{n/2+1}, \ldots, x_{n-1}$ is the same as x_n.

5. The expected number of terms of Taylor coefficients

Consider the Taylor series expansion of $f_j, g_j, e_j \in \mathbb{Z}_p[x_1, \ldots, x_n]$ about $x_n = \alpha_n$ for $0 \neq \alpha_n \in \mathbb{Z}_p$. Let $f_j = \sum_{i=0}^{d_n} f_{ji}(x_n - \alpha_n)^i, g_j = \sum_{i=0}^{d_n} g_{ji}(x_n - \alpha_n)^i$ and $e_j = \sum_{i=0}^{d_n} c_{ji}(x_n - \alpha_n)^i$ where $f_{ji}, g_{ji}, c_{ji} \in \mathbb{Z}_p[x_1, \ldots, x_{n-1}]$. In the i^{th} iteration of the for loop of the j^{th} step of MTSHL (Algorithm 5) one solves the MDP $f_{ji} g_{j0} + g_{ji} f_{j0} = c_{ji}$ to compute f_{ji} and g_{ji} for $1 \leq i \leq d_n$. The cost of MDP depends on the sizes of the polynomials f_{ji}, g_{ji} and c_{ji}.

To make our complexity analysis as precise as possible, in this section we will compute theoretical estimations for the expected sizes of the Taylor coefficients f_j and the upper bounds for $E[T_{f_j}]$ of a randomly chosen $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$ where $f = \sum_{i=0}^{d_n} f_j(x_n - \alpha_n)^j$ for a randomly chosen non-zero element $\alpha_n \in \mathbb{Z}_p$ and p is a big prime. We will confirm our theoretical estimations by experimental data.

In the sequel we will use the notation $\#f$ and T_f interchangeably. For a random non-zero $\alpha_n \in \mathbb{Z}_p$, consider $f = \sum_{j=0}^{d_n} f_j(x_n - \alpha_n)^j$ where each $f_j \in \mathbb{Z}_p[x_1, \ldots, x_{n-1}]$. We expect $T_{f_{j+1}} \leq T_{f_j}$, that is, the size of the Taylor coefficients f_j decrease as j increases.

As a first step to find a upper bound on $E[T_{f_j}]$, we have the following Lemma.

Lemma 14. *Let $0 < d < p$ and $n > 0$. Then following the notation above, the probability of occurrence of each monomial in the support of $f' = \frac{\partial}{\partial x_n} f$ is the same.*

Proof. Let $\mathbf{m}' = cx_1^{\beta_1} \cdots x_n^{\beta_n} \in \mathrm{Supp}(f')$ where $\beta_1 + \cdots + \beta_n \leq d - 1$. Then any monomial of the form $\mathbf{m} = (\beta_n + 1)^{-1} c x_1^{\beta_1} \cdots x_n^{\beta_n+1} + \mathbf{n}$ where $\mathbf{n}$ is a monomial of degree $\leq d$ which does not contain the variable x_n lies over $\mathbf{m}'$. We have $\beta_1 + \cdots + (\beta_n + 1) \leq d$. On the other hand, for $\mathbf{m} = cx_1^{\beta_1} \cdots x_n^{\beta_n} \in \mathrm{Supp}(f)$, one has $\frac{\partial}{\partial x_n}\mathbf{m} = c\beta_n x_1^{\beta_1} \cdots x_n^{\beta_n - 1} = 0 \iff p \,|\, c\beta_n \iff p \,|\, \beta_n$. So if $d < p$ we have $\frac{\partial}{\partial x_n}\mathbf{m} = 0 \iff \mathbf{m}$ does not contain the variable x_n, i.e., $\beta_n = 0$. Therefore the number of monomials lying over each distinct monomial in the support of f' is the same and equal to $(p-1)^\gamma + 1$ where $\gamma = \binom{n+d-1}{n}$. Since f is random, this implies that the probability of occurrence of each monomial in the support of f' is the same. $\square$

After differentiation, monomials which do not contain the variable x_n in f will disappear. Since the expected number of them is $= t_f \binom{n-1+d}{n-1}$, we expect

$$E[\#f'] = T_f - t_f \binom{n-1+d}{n-1}.$$

What about the density ratio $t_{f'}$ of f'? We have

Lemma 15. *Following the notation above $E[t_{f'}] = t_f$.*

Proof.

$$\begin{aligned}
E[t_{f'}] &= \left(T_f - t_f \tbinom{n-1+d}{n-1}\right) / \tbinom{n+d-1}{n} \\
&= t_f \left(\tbinom{n+d}{n} - \tbinom{n-1+d}{n-1}\right) / \tbinom{n+d-1}{n} \\
&= t_f \left(\tfrac{n+d}{n} \tbinom{n+d-1}{n-1} - \tbinom{n-1+d}{n-1}\right) / \tbinom{n+d-1}{n} \\
&= t_f \tbinom{n+d-1}{n-1} / \tfrac{n}{d} \tbinom{n+d-1}{d-1} = t_f \tbinom{n+d-1}{n-1} / \tbinom{n+d-1}{n-1} = t_f.
\end{aligned}$$

$\square$

For simplicity let us assume that $p > j$, otherwise we will need to introduce Hasse derivatives, but the idea will be the same. We have $f_j = \frac{1}{j!} \frac{\partial}{\partial x_n^j} f(x_n = \alpha)$. Also $f^{(j)} := \frac{\partial}{\partial x_n^j} f$ is of degree $\leq d_j := d - j$. Using Lemma 14 and 15 repeatedly,

$$E[f_j] \leq E[\#f^{(j)}] = E[t_{f^{(j)}}] \tbinom{n+d-j}{n} = t_f \tbinom{n+d-j}{n}.$$

It follows that

$$E[t_{f_j}] = E[f_j]/\binom{n+d-j}{n} \leq t_f.$$

We sum up the observations of this section in such a way that it will be helpful for the next sections.

Lemma 16. *Let p be a big prime and $f_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$ be a multivariate polynomial of total degree $\leq d_j$ that has T_{f_j} non-zero terms. For a randomly chosen non-zero element $\alpha_j \in \mathbb{Z}_p$, consider $f_j = \sum_{i=0}^{d_j} f_{ji}(x_j - \alpha_j)^i$ where $f_{ji} \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$. Let $t_{f_j} = T_{f_j}/\binom{j+d_j}{j}$. Then*

$$E[T_{f_{ji}}] \lesssim t_{f_j}\binom{j+d_j-i}{j} \text{ and } E[t_{f_{ji}}] \lesssim t_{f_j}\frac{j+d_j-i}{j}. \tag{16}$$

Example 17. Table 5 below shows the result of a random experiment where $p = 2^{31} - 1$, $j = 7$, $d_j = 13$, $T_{f_j} = 7752$. In the Table 5, $T_{f_{ji}}, t_{f_{ji}}$ and $t_{f_j^{(i)}}$ are the actual values. Also the expected number of terms $E[T_{f_{ji}}]$ of f_{ji}, the bound $bT_{f_{ji}}$ on the expected number of terms of f_{ji}, and the bound $bt_{f_{ji}}$ on the density ratio of f_{ji}, are based on (5) and (16) resp.

i	$T_{f_{ji}}$	$\frac{T_{f_{ji-1}}}{T_{f_{ji}}}$	$E[T_{f_{ji}}]$	$bT_{f_{ji}}$	$t_{f_j^{(i)}}$	$t_{f_{ji}}$	$bt_{f_{ji}}$
0	6651	–	6643.345	7752	0.09	0.085	0.285
1	4343	1.53	4366.828	5038.8	0.1	0.086	0.271
2	2773	1.57	2789.364	3182.4	0.09	0.088	0.257
3	1722	1.61	1724.183	1944.8	0.10	0.088	0.242
4	977	1.76	1025.981	1144.0	0.10	0.092	0.228
5	564	1.73	583.867	643.5	0.10	0.093	0.214
6	306	1.84	315.075	343.2	0.10	0.094	0.200
7	150	2.04	159.417	171.6	0.10	0.103	0.185
8	68	2.21	74.463	79.2	0.10	0.104	0.171
9	26	2.62	31.403	33.0	0.10	0.127	0.157
10	12	2.17	11.559	12.0	0.05	0.125	0.142
11	3	4.00	3.511	3.6	0.12	0.111	0.128
12	1	3.00	0.790	0.8	1	0.112	0.114

Table 5. The bounds on the expected number of terms and the density ratio.

6. The complexity of the MDP

Let p be a big prime and $u, w, h \in \mathbb{Z}_p[x_1, \ldots, x_n]$ where u, w are monic in x_1. Suppose we are trying to solve the MDP (which satisfies the MDP conditions)

$$D := fu + gw = h \tag{17}$$

to find the unique solution pair (f, g) via sparse interpolation as described in section 2. Let d be a total degree bound for f, g, u, w, h. Our aim in this section is to estimate the expected complexity of solving (17). Since the calculations of the complexity evaluation in the next section are somewhat tedious, this section is intended to help the reader follow it easily.

Suppose the solution-form of f is

$$\sigma_f = \sum_{i+j\leq d} c_{ij}(x_3, ..., x_n) x_1^i x_2^j \text{ where } c_{ij} = \sum_{l=1}^{m_{ij}} c_{ijl} x_3^{\gamma_{3l}} \cdots x_n^{\gamma_{jl}} \text{ with } c_{ijl} \in \mathbb{Z}_p \backslash \{0\}.$$

Let $m = \max m_{ij}$. Then in sparse interpolation the first step is to choose a random $(\beta_3, \ldots, \beta_n) \in (\mathbb{Z}_p\backslash\{0\})^{n-2}$ and solve bivariate MDP's

$$D_r := \tilde{f} \cdot u(x_1, x_2, \beta_3^r, \ldots, \beta_n^r) + \tilde{g} \cdot w(x_1, x_2, \beta_3^r, \ldots, \beta_n^r) = h(x_1, x_2, \beta_3^r, \ldots, \beta_n^r)$$

for $r = 1, \ldots, m$ where $(\tilde{f}, \tilde{g}) \in \mathbb{Z}_p[x_1, x_2]^2$ is to be solved.

As before let $s = \binom{n+d}{n}$, $r = \frac{n}{n+d}$, $s_k = \binom{n-2+k}{n-2}$ and $d_k = d - k + 1$ for $0 \leq k \leq d$. Suppose that the solution form σ_f of f is correct. Then the expected number of monomials of the form $x_3^{\alpha_3} \cdots x_n^{\alpha_n} x_1^i x_2^j$ in $\mathrm{Supp}(f)$ is $t_f \binom{n-2+d-k}{n-2} = t_f s_{d-k}$ when $i + j = k$. So we expect $\#c_{ij} = t_f s_{d-k}$. When $i = j = 0$, i.e. the number of monomials that are in the variables $x_3, \ldots, x_n$ in $\mathrm{Supp}(f)$ is expected to be greatest, therefore the expected number of evaluations is $m = t_f s_d$. At this point we remark that

$$t_f s_d = T_f \frac{n(n-1)}{(n+d)(n+d-1)} \leq T_f \left(\frac{n}{n+d}\right)^2.$$

So, if $T_f \left(\frac{n}{n+d}\right)^2 < 1$, our theoretical expectation of $m = t_f s_d$ will be less than 1, which in practice means that there won't be any evaluation. If d is big and T_f is small, this inequality may occur, however the algorithm makes at least one evaluation and hence calls BDP at least once, so we should have $m = \lceil t_f s_d \rceil$.

Let $T = (T_f + T_u + T_w + T_h)$. (We are evaluating σ_f too, to get the linear system of equations in c_{ijl}). Then according to subsection 2.3, the total cost C_E of the consecutive evaluations is bounded above by

$$\begin{aligned} C_E &\leq \#\text{of terms} \times (\#\text{of evaluations} + n - 3) + (n-2)d \\ &\approx (T_f + T_u + T_w + T_h)\,(T_f \frac{s_d}{s} + 1 + n - 3) + (n-2)d \\ &\leq T(\lceil T_f r^2 \rceil + n) + nd \end{aligned}$$

If d is huge and T_f is small so that $T_f r^2 < n$ then nT or nd can dominate the sum above. So we obtain

$$C_E \in \mathcal{O}(T\lceil T_f r^2 \rceil + nT + nd). \tag{18}$$

After evaluation, the sparse interpolation routine calls BDP to solve the bivariate diophantine equations D_r. For a given D_r, BDP solves it in $\mathcal{O}(d_2^3)$ arithmetic operations in $\mathbb{Z}_p$ via dense interpolation where d_2 is a bound for the total degrees of f, g, u, w, h in x_1, x_2 above. In our case $d_2 \leq d$. Hence the expected cost C_B of solving D_r's is

$$C_B \in \#\text{of evaluations} \times \mathcal{O}(d^3) = \mathcal{O}(\lceil T_f r^2 \rceil d^3). \tag{19}$$

BDP gives the unique solution for D_r iff the MDP conditions for D_r are satisfied. To have a unique solution for D_r, for $1 \leq t \leq m$, the first condition is

$$\gcd\big(u(x_1, x_2, \beta_3^t, \ldots, \beta_n^t), w(x_1, x_2, \beta_3^t, \ldots, \beta_n^t)\big) \mid h(x_1, x_2, \beta_3^t, \ldots, \beta_n^t).$$

which is the case if D has a solution. The second condition is, when BDP chooses a random $\gamma \in \mathbb{Z}_p$ while it is interpolating, it must be the case that

$$\gcd\left(u(x_1, \gamma, \beta_3^t, \ldots, \beta_n^t), w(x_1, \gamma, \beta_3^t, \ldots, \beta_n^t)\right) = 1 \,\text{in}\, \mathbb{Z}_p[x_1].$$

The probability that the second condition fails is $\leq m \deg(u) \deg(w)/p \leq md^2/p$ (MT16b). This is a worst case upper bound. On average, the expected number of failures is only m out of p trials (MT16a). In this case then the expected probability of failure is $\leq \lceil T_f r^2 \rceil / p$.

As we have seen in Section 4, we expect that the density ratio increases after each evaluation of f. Hence after evaluations, we expect dense polynomials over $\mathbb{Z}_p[x_1, x_2]$ and this is why BDP uses the dense interpolation to solve bivariate MDP's. While solving the bivariate MDP's, BDP chooses a random $\gamma \in \mathbb{Z}_p$ and solves the univariate MDP over $\mathbb{Z}_p$. This is done by using the Euclidean algorithm (see (GCL)). To do that it computes the univariate gcd and if it is not equal to 1 it detects it. So if such an unlucky evaluation occurs then BDP detects it and the algorithm terminates.

If $i + j = k$, then to recover c_{ijl}'s one needs to solve a linear system which corresponds to a Vandermonde matrix of expected size $t_f s_{d-k}$. The cost of this operation is $\mathcal{O}\left(t_f^2 s_{d-k}^2\right)$(Zip90). We have $k+1$ monomials of the form $x_1^i x_2^j$ with $i + j = k$. So, the expected total cost C_V for the solution is in

$$C_V \in \mathcal{O}\left(\sum_{k=0}^{d} (k+1) t_f^2 s_{d-k}^2\right) = \mathcal{O}\left(T_f^2 \sum_{k=0}^{d} (k+1) \left(\frac{s_{d-k}}{s}\right)^2\right).$$

First, note that

$$s = \binom{n+d}{n} = \frac{(n+d)(n+d-1)}{n(n-1)} \binom{n+d-2}{n-2} > r^{-2} \binom{n+d-2}{n-2} = r^{-2} s_{d-2}.$$

Then, as we did in the first section, we obtain

$$\frac{s_{d-k}}{s} < r^2 \frac{s_{d-k}}{s_{d-2}} < r^2 \left(1 - \tfrac{n-2}{n+d-2}\right)^k = r^2 (1-\theta)^k$$

where $\theta := \frac{n-2}{n+d-2}$. Then, we get

$$T_f^2 \sum_{k=0}^{d} (k+1) \left(\frac{s_{d-k}}{s}\right)^2 < T_f^2 r^4 \sum_{k=0}^{d} (k+1)(1-\theta)^{2k}.$$

By using the summation formula, a straightforward (but a bit tedious) calculation shows that if $n > 2$ (which is infact the case),

$$r^4 \sum_{k=0}^{d} (k+1)(1-\theta)^{2k} \leq r^4 \frac{(n+d-2)^4}{(n-2)^2(n+2d-2)^2} < r^2 \left(\frac{n}{n-2}\right)^2 \leq 9r^2.$$

Hence we see that the expected cost of solving linear systems is $(r = n/(n+d))$

$$C_V \in \mathcal{O}(T_f^2 r^2). \tag{20}$$

After computing f, the next step is the multivariate division $(h - fu)/w$ to get g. The expected cost C_M of the sparse multiplication and sparse multivariate division C_D is

$$C_M \in \mathcal{O}\left(T_f T_u\right) \,\text{and}\, C_D \in \mathcal{O}\left(T_w T_g\right) \tag{21}$$

arithmetic operations in $\mathbb{Z}_p$ ignoring the sorting cost.

Combining equations (18), (19), (20) and (21) above wee see that the expected cost of solving the MDP is in

$$\mathcal{O}(\underbrace{\underbrace{T\lceil T_f r^2\rceil + nT + nd}_{C_E} + \underbrace{\lceil T_f r^2\rceil d^3}_{C_B} + \underbrace{T_f^2 r^2}_{C_V}}_{\text{to recover } f} + \underbrace{\underbrace{T_f T_u}_{C_M} + \underbrace{T_w T_g}_{C_D}}_{\text{to recover } g})$$

with the failure probability $\leq \lceil T_f r^2\rceil d^3/p$ where $r = \frac{n}{n+d}$ and $T = (T_f + T_u + T_w + T_h)$.

Finally suppose that the guessed solution-form σ_f of f is wrong. Then the solution to f that the sparse interpolation routine computes will be wrong. Since the solution to the MDP is unique as long as the MDP conditions are satisfied, then we will have $w \nmid h - fu$. So, in the sparse interpolation a possible failure, i.e. a possible false assumption is detected. In this case the cost of sparse division may increase (we don't consider this).

Theorem 18. *Let p be a big prime and $u, w, h \in \mathbb{Z}_p[x_1, \ldots, x_n]$ where u, w are monic in x_1. If the solution-form σ_f is true, then the number of arithmetic operations in $\mathbb{Z}_p$ for solving the MDP $fu + gw = h$ (which satisfies the MDP conditions) to find the unique solution pair (f, g) via sparse interpolation as described in section 2 is in*

$$\mathcal{O}\left(T\lceil T_f r^2\rceil + nT + nd + \lceil T_f r^2\rceil d^3 + T_f^2 r^2 + T_f T_u + T_w T_g\right)$$

where d is a total degree bound for f, g, u, w, h, $r = \frac{n}{n+d}$, and $T = T_f + T_u + T_w + T_h$. Moreover the probability of success is $> 1 - \lceil T_f r^2\rceil d^3/p$.

7. The Complexity of MTSHL

For $j \geq 3$, during the j^{th} step of the MTSHL, one aims to reach the factorization $a_j = f_j g_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$ from the knowledge of $a_j, f_{j-1}, g_{j-1} \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$ satisfying $a_{j-1} = f_{j-1} g_{j-1}$. Let $f_j = \sum_{i=0}^{d_j} f_{ji}(x_j - \alpha_j)^i, g_j = \sum_{i=0}^{d_j} g_{ji}(x_j - \alpha_j)^i$ for a randomly chosen non-zero element α_j in $\mathbb{Z}_p$ and where $f_{j0} = f_{j-1}$, $g_{j0} = g_{j-1}$ and $\deg_{x_j}(a_j) = d_j$.

For $1 \leq i \leq d_j$ one recovers each f_{ji}, g_{ji} by solving the MDP problems

$$f_{ji} g_{j0} + g_{ji} f_{j0} = c_{ji} \quad \text{in} \quad \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$$

via sparse interpolation in a for loop where c_{ji} is the i^{th} Taylor coefficient of $error$. Our aim in this section is first to estimate the complexity of this lifting process at the j^{th} step of the MTSHL algorithm, that is, finding f_j and g_j, and then estimate the expected complexity of the multivariate factorization via MTSHL. To this end, let t_h, T_h denote the expected density ratio and the expected number of non-zero elements of a polynomial $h \in \mathbb{Z}_p[x_1, \ldots, x_j]$. By $(\#k)$ we will refer to the k^{th} line in Algorithm 5. Before continuing, we suggest the reader read the concrete example in the Appendix A to be able to follow the rest of discussion below easily.

7.1. Before we go into the details

Before we go into the details of tedious calculations, we want to make a guess what we will get, based on our observations from Section 6:

Suppose that the smallest factor is f and $T_f \gg \max\{n, d\}$. Based on Theorem 18 we may guess that the evaluation cost will be the most expensive part. Now, at the j^{th} step of MTSHL, suppose that $T_{f_{j-1}} \leq T_{g_{j-1}}$. Then based on Lemma 8, MTSHL makes a probabilistic guess (#1) that T_{f_j} will be smaller than T_{g_j} and (in the for loop), for $1 \leq i \leq d_j$, in the sparse interpolation routine, it first computes f_{ji} and then recovers g_{ji} via the multivariate division. Since we expect $T_{f_{ji}} \leq T_{f_j}$ the expected number of evaluations for each i in the loop will be $\leq T_{f_j}\left(\frac{j}{j+d_j}\right)^2$(see Section 6). So at the j^{th} step the expected evaluation cost will be in $d_j\mathcal{O}(T_{a_j}T_{f_j}\left(\frac{j}{j+d_j}\right)^2) = d_j\mathcal{O}(T_{a_j}T_{f_j}\left(\frac{j}{d_j}\right)^2) = \mathcal{O}(\frac{j^2}{d_j}T_{a_j}T_{f_j})$. Then, running j from 1 to n, based on our observations in section 4.1, we expect that the average complexity will be close to or less than $n\mathcal{O}(\frac{n^2}{d}T_aT_f) = \mathcal{O}(\frac{n^3}{d}T_aT_f)$. Finally, since $T_f \leq T_g$ and $T_a \in \mathcal{O}(T_fT_g)$, our guess is that the average complexity will be close to or less than $\mathcal{O}(\frac{n^3}{d}T_g^3)$.

In the following we will make this guess more precise and prove that if $T_f \leq T_g$ and $T_g > nd^2$ the expected cost of MTSHL is in fact quadratic in n and is in $\mathcal{O}(\frac{n^2}{d}T_g^3)$.

7.2. *In detail*

Suppose that $T_{f_{j-1}} \leq T_{g_{j-1}}$. Then based on Lemma 8, MTSHL makes a probabilistic guess (#1) that T_{f_j} will be smaller than T_{g_j} and for $1 \leq i \leq d_j$, in the sparse interpolation routine, it first computes f_{ji} and then recovers g_{ji} via the multivariate division $(c_{ji} - g_{j0}f_{ji})/f_{j0}$.

The cost of (#4) is the cost of subtraction since we are given $a_{j-1} = f_{j-1}g_{j-1}$. In the sparse case, this cost is for sorting the monomials, which we will ignore for the rest of the discussion.

In the i^{th} iteration, updating the monomial (#6) has cost linear in i which is negligible. Then, the algorithm computes the i^{th} Taylor coefficient c_{ji} of the error at $x_j = \alpha_j$ (#7). Since this is linear in $\#error$ and thus dominated by the computation of the error in (#4) and (#15), it can be ignored.

Then to solve the MDP, it comes to the sparse interpolation (#11). Suppose that $f_{ji} = \sum c_{jikl}x_1^kx_2^l \in \mathbb{Z}_p[x_3, \ldots, x_{j-1}][x_1, x_2]$. Let $d_{ji} := d_j - i$ and $ev_{ji} := t_{f_{ji}}\binom{j-1-2+d_{ji}}{j-1-2}$. We have $\deg(f_{ji}) \leq d_{ji}$ and, as explained in Section 6, we expect that $\#c_{ji00} = ev_{ji}$. Then by Lemma 16, we expect

$$ev_{ji} := t_{f_{ji}}\binom{j-3+d_{ji}}{j-3} \leq t_{f_j}\frac{j+d_{ji}}{j}\binom{j-3+d_{ji}}{j-3}.$$

Based on the Lemma 1, Algorithm 5 makes a probabilistic guess (#10) and assumes that in sparse interpolation the solution form $\sigma_{f_{ji}} = f_{j,i-1}$, so the expected number of evaluations at the i^{th} step is $\lceil ev_{j,i-1}\rceil$. According to subsection 2.3 (see also Section 6), the expected cost $C_{Ev_{ji}}$ of evaluation at the i^{th} step is bounded above by

$$C_{Ev_{ji}} < \left(T_{g_{j0}} + T_{f_{j0}} + T_{f_{j,i-1}} + T_{c_{ji}}\right)\left(\lceil ev_{j,i-1}\rceil + j - 4\right) + (j-3)\,d_{j,i}$$

After evaluation, the sparse interpolation routine calls BDP. For a given bivariate diophantine equation the cost is $\mathcal{O}(d_{ji}^3)$. Hence the expected cost $C_{B_{ji}}$ of solving the bivariate diophantine equations via BDP in the i^{th} iteration is

$$C_{B_{ji}} \in \mathcal{O}\left(\lceil ev_{j,i-1}\rceil d_{ji}^3\right).$$

Note again that the sparse interpolation routine first computes f_{ji} and then recovers g_{ji} via a multivariate division. The linear systems to be solved to recover f_{ji} corresponds to Vandermonde matrices and they are constructed by the unknown coefficients of the solution form $\sigma_{f_{ji}}$ of f_{ji}. Hence, if we define $r_{ji} = \frac{j-1}{j-1+d_{ji}}$, then the expected cost $C_{V_{ji}}$ of solving the linear system in the i^{th} iteration is (see section 6 equation (19))

$$C_{V_{ji}} \in \mathcal{O}\left(T^2_{f_{j,i-1}} r^2_{j,i-1}\right).$$

By Lemma 16, $E[\#g_{j0}] \le t_{g_j}\binom{j+d_j}{j}$ and $E[\#f_{ji}] \le t_{f_j}\binom{j+d_{ji}}{j}$. So, after computing f_{ji}, the expected cost $C_{M_{ji}}$ of sparse multiplication $g_{j0}f_{ji}$ and the expected cost $C_{D_{ji}}$ of sparse division $(c_{ji} - g_{j0}f_{ji})/f_{j0}$ are both in

$$C_{M_{ji}} \text{ and } C_{D_{ji}} \in \mathcal{O}\left(t_{f_j} t_{g_j} \binom{j+d_j}{j}\binom{j+d_{ji}}{j}\right).$$

So far we have covered the (dominating) costs in sparse interpolation at the i^{th} iteration. Next we consider (#14). The cost of updating, $C_{U_{ji}}$, i.e computing $f_{ji}(x_j - \alpha_j)^i$ and $g_{ji}(x_j - \alpha_j)^i$ is in

$$C_{U_{ji}} \in \mathcal{O}\left(i(t_{f_j} + t_{g_j})\binom{j+d_{ji}}{j}\right).$$

Finally, (#15) the cost of updating *error* is in

$$C_{Er_{ji}} \in \mathcal{O}\left(T_{f_j} T_{g_j}\right).$$

Let C_{Ev_j} be the expected total evaluation cost (in sparse interpolation) at the j^{th} step. Then $C_{Ev_j} = \sum_{i=1}^{d_j} C_{Ev_{ji}}$. To compute it we'll split the sum

$$\sum_{i=1}^{d_j} \left(T_{g_{j0}} + T_{f_{j0}} + T_{f_{j,i-1}} + T_{c_{ji}}\right)\left(\lceil ev_{j,i-1}\rceil + j - 4\right) + (j-3)\, d_{j,i-1}$$

and consider the parts separately: We first consider the sum

$$\begin{aligned}
\sum_{i=1}^{d_j} \lceil ev_{j,i-1}\rceil = \sum_{i=0}^{d_j-1} \lceil ev_{ji}\rceil \le \sum_{i=0}^{d_j} \lceil t_{f_j} \frac{j+d_{ji}}{j}\binom{j-3+d_{ji}}{j-3}\rceil \\
\le \sum_{i=0}^{d_j}\left(t_{f_j}\frac{j+d_{ji}}{j}\binom{j-3+d_{ji}}{j-3} + 1\right) = \sum_{i=0}^{d_j} t_{f_j}\frac{j+d_{ji}}{j}\binom{j-3+d_{ji}}{j-3} + \sum_{i=0}^{d_j} 1 \\
\le t_{f_j}\frac{j+d_j}{j}\sum_{i=0}^{d_j}\binom{j-3+d_{ji}}{j-3} + d_j = t_{f_j}\frac{j+d_j}{j}\binom{j-2+d_j}{j-2} + d_j \\
= t_{f_j}\frac{j+d_j}{j}\frac{j-1}{j-1+d_j}\binom{j-1+d_j}{j-1} + d_j \le t_{f_j}\binom{j-1+d_j}{j-1} + d_j \\
= t_{f_j}\frac{j}{j+d_j}\binom{j+d_j}{j} + d_j = \frac{j}{j+d_j} T_{f_j} + d_j.
\end{aligned}$$

As a next step, since we expect $T_{f_j} \le T_{g_j}$, $T_{f_{j0}} \le T_{f_j}$ and $T_{g_{j0}} \le T_{g_j}$,

$$\sum_{i=1}^{d_j} \left(T_{g_{j0}} + T_{f_{j0}}\right) \lceil ev_{j,i-1} \rceil \le (T_{f_j} + T_{g_j}) \left(\frac{j}{j+d_j} T_{f_j} + d_j \right) \in \mathcal{O}\left(\frac{j}{j+d_j} T_{f_j} T_{g_j} + d_j T_{g_j} \right). \tag{22}$$

On the other hand, we expect $T_{c_{ji}} \le T_{a_{j-1}} \le T_{a_j}$. Then

$$\sum_{i=1}^{d_j} T_{c_{ji}} \lceil ev_{j,i-1} \rceil \le T_{a_j} \sum_{i=1}^{d_j} \lceil ev_{j,i-1} \rceil \le \frac{j}{j+d_j} T_{f_j} T_{a_j} + d_j T_{a_j}.$$

So, since we expect $T_{f_{j,i-1}} \le T_{f_{j0}}$, we see that

$$\sum_{i=1}^{d_j} \left(T_{g_{j0}} + T_{f_{j0}} + T_{f_{j,i-1}} + T_{c_{ji}}\right) \left(\lceil ev_{j,i-1} \rceil\right) \in \mathcal{O}\left(\frac{j}{j+d_j} (T_{f_j} T_{g_j} + T_{f_j} T_{a_j}) + d_j (T_{g_j} + T_{a_j})\right). \tag{23}$$

Also,

$$\sum_{i=0}^{d_j-1} (j-3) d_{j,i} \in \mathcal{O}(j d_j^2). \tag{24}$$

Now we need to consider the sum

$$\begin{aligned}
\sum_{i=0}^{d_j} T_{c_{ji}} j &\le \sum_{i=0}^{d_j} t_{a_j} j \binom{j+d_{ji}}{j} = \sum_{i=0}^{d_j} t_{a_j} j \frac{j+d_{ji}}{j} \binom{j-1+d_{ji}}{j-1} \\
&\le t_{a_j} (j+d_j) \sum_{i=0}^{d_j} \binom{j-1+d_{ji}}{j-1} = t_{a_j} (j+d_j) \binom{j+d_j}{j} \;=\; (j+d_j) T_{a_j}.
\end{aligned}$$

Using the same idea we see that $\sum_{i=0}^{d_j} T_{f_{j,i-1}} j \le (j+d_j) T_{f_j}$.

$$\sum_{i=0}^{d_j} (T_{c_{ji}} + T_{f_{j,i-1}}) j \le (j+d_j)(T_{f_j} + T_{a_j}).$$

Also,

$$\sum_{i=1}^{d_j-1} \left(T_{g_{j0}} + T_{f_{j0}}\right) j \le (T_{f_j} + T_{g_j}) \sum_{i=1}^{d_j-1} j \le (T_{f_j} + T_{g_j}) j d_j.$$

So we get

$$\sum_{i=1}^{d_j-1} \left(T_{g_{j0}} + T_{f_{j0}} + T_{f_{j,i-1}} + T_{c_{ji}}\right) j \in \mathcal{O}\left(j d_j (T_{f_j} + T_{g_j}) + (j+d_j)(T_{f_j} + T_{a_j}) \right). \tag{25}$$

Let us consider the terms appearing in Eqns (23), (24) and (25),

$$\underbrace{\frac{j}{j+d_j} (T_{f_j} T_{g_j} + T_{f_j} T_{a_j}) + d_j (T_{g_j} + T_{a_j})}_{(22)} + \underbrace{j d_j^2}_{(23)} + \underbrace{j d_j (T_{f_j} + T_{g_j}) + (j+d_j)(T_{f_j} + T_{a_j})}_{(24)}.$$

We have $d_jT_{g_j} \leq jd_jT_{g_j}$ and, since $T_{f_j} \leq T_{g_j}$, we get $jd_j(T_{f_j}+T_{g_j}) \in \mathcal{O}(jd_jT_{g_j})$. So by Eqns (23), (24) and (25) the expected cost $C_{Ev_j} = \sum_{i=1}^{d_j} C_{Ev_{ji}}$ of evaluation at the j^{th} step is in

$$C_{Ev_j} \in \mathcal{O}\left(\frac{j}{j+d_j}T_{a_j}T_{f_j} + \frac{j}{j+d_j}T_{f_j}T_{g_j} + jd_jT_{g_j} + (j+d_j)(T_{f_j}+T_{a_j}) + jd_j^2\right). \qquad (26)$$

Let C_{B_j} be the expected cost of BDP at the j^{th} step. Then $C_{B_j} = \sum_{i=1}^{d_j} C_{B_{ji}} = \sum_{i=1}^{d_j} \mathcal{O}\left(\lceil ev_{j,i-1}\rceil d_{ji}^3\right)$. First, we consider

$$\sum_{i=1}^{d_j-1} \lceil ev_{j,i-1}\rceil d_{ji}^3 \leq d_j^3(d_j + \frac{j}{j+d_j}T_{f_j}) \leq d_j^4 + jd_j^2T_{f_j}.$$

Hence

$$C_{B_j} \in \mathcal{O}\left(d_j^4 + jd_j^2T_{f_j}\right). \qquad (27)$$

Let C_{V_j} be the expected cost of solving linear systems (in sparse interpolation) at the j^{th} step. Then $C_{V_j} = \sum_{i=1}^{d_j} C_{V_{ji}} = \sum_{i=1}^{d_j} \mathcal{O}\left(T_{f_{j,i-1}}^2 r_{j,i-1}^2\right)$. We consider

$$\begin{aligned}
\sum_{i=0}^{d_j-1} T_{f_{ji}}^2 r_{ji}^2 &= \sum_{i=0}^{d_j-1}\left(t_{f_j}\binom{j+d_{ji}}{j}\frac{j-1}{j-1+d_{ji}}\right)^2 \\
&= \sum_{i=0}^{d_j-1}\left(t_{f_j}\frac{j-1}{j-1+d_{ji}}\frac{j+d_{ji}}{j}\binom{j-1+d_{ji}}{j-1}\right)^2 \\
&\leq t_{f_j}^2\sum_{i=0}^{d_j}\binom{j-1+d_{ji}}{j-1}^2 \leq t_{f_j}^2\left(\sum_{i=0}^{d_j}\binom{j-1+d_{ji}}{j-1}\right)^2 \\
&= t_{f_j}^2\binom{j+d_j}{j}^2 = T_{f_j}^2.
\end{aligned}$$

Hence, the expected cost C_{V_j} is in

$$C_{V_j} \in \mathcal{O}\left(T_{f_j}^2\right). \qquad (28)$$

Let the expected cost of multiplication and division at the j^{th} step be C_{D_j} and C_{M_j} resp. Then $C_{M_j} = \sum_{i=1}^{d_j} C_{M_{ji}} = \sum_{i=1}^{d_j} \mathcal{O}\left(t_{f_j}t_{g_j}\binom{j+d_j}{j}\binom{j+d_{ji}}{j}\right)$, and similarly for C_{D_j}. Note that

$$\begin{aligned}
\sum_{i=1}^{d_j-1} t_{f_j}t_{g_j}\binom{j+d_j}{j}\binom{j+d_{ji}}{j} &= t_{f_j}t_{g_j}\binom{j+d_j}{j}\sum_{i=1}^{d_j-1}\binom{j-1+d_{ji}}{j-1} \\
&\leq t_{g_j}t_{f_j}\binom{j+d_j}{j}^2 = T_{f_j}T_{g_j}.
\end{aligned}$$

So, the expected cost $C_{M_j}$of sparse multiplication and C_{D_j} of sparse division (in sparse interpolation) at the j^{th} step is

$$C_{M_j} \in \mathcal{O}\left(T_{f_j}T_{g_j}\right) \text{ and } C_{D_j} \in \mathcal{O}\left(T_{f_j}T_{g_j}\right). \qquad (29)$$

Let C_{U_j} be the cost of updating the factors at the the j^{th} step. Then $C_{U_j} = \sum_{i=1}^{d_j} C_{U_{ji}} = \sum_{i=1}^{d_j} \mathcal{O}\left(i(t_{f_j} + t_{g_j})\binom{j+d_{ji}}{j}\right)$. We have

$$\begin{aligned}\sum_{i=1}^{d_j} i t_{g_j}\binom{j+d_{ji}}{j} &= t_{g_j}\sum_{i=1}^{d_j} i\binom{j+d_{ji}}{j} = t_{g_j}\frac{d_j(j+d_j+1)}{(j+1)(j+2)}\binom{j+d_j}{j} \\ &\leq \left(\frac{d_j(j+1)+d_j^2}{j^2}\right) t_{g_j}\binom{j+d_j}{j} = \left(\frac{d_j(j+1)+d_j^2}{j^2}\right) T_{g_j}\end{aligned}$$

Since $T_{f_j} \leq T_{g_j}$, we get

$$C_{U_j} \in \mathcal{O}\left(\frac{d_j}{j}T_{g_j} + \frac{d_j^2}{j^2}T_{g_j}\right). \tag{30}$$

Let C_{Er_j} be the cost of updating *error* at the j^{th} step. Then $C_{Er_j} = \sum_{i=1}^{d_j} C_{Er_{ji}} = \sum_{i=1}^{d_j} \mathcal{O}\left(T_{f_j}T_{g_j}\right)$ satisfies

$$C_{Er_j} \in \mathcal{O}\left(d_j T_{f_j} T_{g_j}\right) \tag{31}$$

According to equations (26) to (31), we shall consider the dominating terms

$$\underbrace{\frac{j}{j+d_j}T_{a_j}T_{f_j} + \frac{j}{j+d_j}T_{f_j}T_{g_j} + jd_jT_{g_j} + (j+d_j)(T_{f_j}+T_{a_j}) + jd_j^2}_{C_{Ev_j}}$$

$$\underbrace{d_j^4 + jd_j^2T_{f_j}}_{C_{B_j}} + \underbrace{T_{f_j}^2}_{C_{V_j}} + \underbrace{T_{f_j}T_{g_j}}_{C_{M_j}\text{ and }C_{D_j}} + \underbrace{(\frac{d_j}{j} + \frac{d_j^2}{j^2})T_{g_j}}_{C_{U_j}} + \underbrace{d_jT_{f_j}T_{g_j}}_{C_{Er_j}}.$$

The terms $T_{f_j}^2$, $\frac{j}{j+d_j}T_{f_j}T_{g_j}$, $\frac{d_j}{j}T_{g_j}$, $T_{f_j}T_{g_j}$ are dominated by the term $d_jT_{f_j}T_{g_j}$. The term jd_j^2 is dominated by $jd_j^2T_{f_j}$. Hence the expected complexity at the j^{th} step is in

$$\mathcal{O}(\underbrace{\frac{j}{j+d_j}T_{a_j}T_{f_j} + jd_jT_{g_j} + (j+d_j)(T_{a_j}+T_{f_j})}_{C_{Ev_j}} + \underbrace{d_j^4 + jd_j^2T_{f_j}}_{C_{B_j}} + \underbrace{\frac{d_j^2}{j^2}T_{g_j}}_{C_{U_j}} + \underbrace{d_jT_{f_j}T_{g_j}}_{C_{Er_j}}).$$

Recall that $f_j := f(x_1, \ldots, x_j, x_{j+1} = \alpha_{j+1}, \ldots, x_n = \alpha_n) \bmod p$. Similarly for a and g. Let $\mathcal{J} = \left\{ j \in \mathbb{N} \mid \max\{t_a, t_f, t_g\} \leq 1/\binom{n-j+d}{d}\right\}$. Then as it was explained in Section 4.1, we expect $T_{f_j}, T_{g_j}, T_{a_j}$ to be very close to T_f, T_g, T_a resp. for $j \in \mathcal{J}$. Then

$$\sum_{j=3}^{n} T_{a_j}T_{f_j} \in \Omega(|\mathcal{J}|T_aT_f).$$

According to Remark 13, in the sparse examples $|\mathcal{J}| \in \mathcal{O}(n)$. Then

$$\sum_{j=3}^{n}\frac{j}{j+d_j}T_{a_j}T_{f_j} < \sum_{j=3}^{n}\frac{j}{d_j}T_{a_j}T_{f_j} < \frac{n}{d}\sum_{j=3}^{n}T_{a_j}T_{f_j} \in \mathcal{O}(\frac{n^2}{d}T_aT_f)$$

On the other hand, we have $\sum_{j=3}^{n} j d_j T_{g_j} \leq dT_g \sum_{j=3}^{n} j \in \mathcal{O}(n^2 dT_g)$, $\sum_{j=3}^{n}(j+d_j)(T_{a_j} + T_{f_j}) \leq (T_a + T_f)\sum_{j=3}^{n}(j+d_j) \leq (T_a+T_f)(n^2+nd)$.

So assuming that the inputs are sparse while running the index j from 3 to n, the expected complexity of MTSHL is in

$$\mathcal{O}(\underbrace{\frac{n^2}{d}T_aT_f + n^2dT_g + (n^2+nd)(T_a+T_f)}_{\text{Evaluation}} + \underbrace{nd^4 + n^2d^2T_f}_{\text{BDP}} + \underbrace{d^2T_g}_{\text{Update}} + \underbrace{ndT_fT_g}_{\text{Er}}).$$

Since $T_f \leq T_g$, the expected complexity is in

$$\mathcal{O}\left(\frac{n^2}{d}T_aT_g + n^2dT_g + (n^2+nd)(T_a+T_g) + nd^4 + n^2d^2T_g + ndT_g^2\right).$$

$$= \mathcal{O}\left(\frac{n^2}{d}T_aT_g + n^2T_a + ndT_a + n^2d^2T_g + ndT_g^2 + nd^4\right). \tag{32}$$

Finally since $T_a \in \mathcal{O}(T_fT_g)$, we have $T_a \in \mathcal{O}(T_g^2)$, and hence the expected complexity is in

$$\mathcal{O}\left(\frac{n^2}{d}T_g^3 + n^2T_g^2 + n^2d^2T_g + ndT_g^2 + nd^4\right). \tag{33}$$

Note that if T is big enough, for example $T_g > nd^2$ (which is the case for the most our experiments in the final section), then

$$ndT_g^2 > n^2d^3T_g > n^2d^2T_g \text{ and } n^2d^2T_g > n^3d^4 > nd^4, \text{ and}$$

$$ndT_g^2 < nd^2T_g^2 < T_g^3 \text{ and } nT_g^3 > n^2d^2T_g^2 > n^2T_g^2$$

Hence the expected complexity is in

$$\mathcal{O}\left(\frac{n^2}{d}T_g^3\right).$$

The cubic term is coming from evaluation and suggests the evaluation is the most time dominating step. This is what we have expected and will confirm by experimental data. Now according to Lemma 1

$$\Pr[\mathrm{Supp}(f_{j,i+1}) \not\subseteq \mathrm{Supp}(f_{ji})] \leq T_{f_{j,i+1}}\frac{d_{ji}}{p-d_{ji}+1} \leq T_{f_j}\frac{d-i}{p-2d+1}.$$

Then the probability that there is a fallacy on one of the assumptions of (#10) at the j^{th} step is $\leq \sum_{i=0}^{d_j-1} T_{f_{j,i+1}}\frac{d_{ji}}{p-d_{ji}+1} \leq \frac{d^2}{2(p-2d+1)}T_{f_j}$. Hence throughout the whole MHL process the probability of failure of MHL because of a false assumption at (#10) is $\leq \frac{(n-2)d^2}{2(p-2d+1)}T_{f_j}$. Also note that at the j^{th} step the algorithm used $\leq d_jT_{f_j}$ many evaluations and made an assumption on the gcd of univariate polynomials in BDP based on Schwartz-Zippel's lemma. Then the probability of a failure because of a false assumption on the gcd of polynomials in BDP is then $\leq \frac{d}{p-1}\sum_{j=3}^{n} d_jT_{f_j} \leq \frac{(n-2)(T_fd^2)}{p-1}$. This implies the probability of failure of MTSHL is

$$\leq (n-2)d^2T_f\left(\frac{1}{p-1} + \frac{1}{2(p-2d+1)}\right).$$

This is a very generous bound. In our experiments to construct the data in the final section, we have used $p = 2^{31}-1$ and MTSHL has never failed.

As a final note, we consider the probabilistic assumption in Step 4 of the sparse interpolation routine described in detail in Algorithm 4.

In the notation of Algorithm 4, consider the polynomials

$$\Delta_{ik} = \prod_{1 \le a < b \le s_{ik}} (M_{ika} - M_{ikb}) \in \mathbb{Z}_p[x_3, \ldots, x_j].$$

In Step 4, $|S_{ik}| = s_{ik}$ means the monomial evaluations are distinct, and if this is not the case, then at least one of the Vandermonde matrices constructed in Step 14 is not invertible. In that case, $\Delta_{ik}(\alpha_3, \ldots, \alpha_j) = 0$. We want to bound the probability that this may happen for any Δ_{ik}. Let $\Delta = \prod \Delta_{ik}$. Then $\Delta(\alpha_3, \ldots, \alpha_j) = 0$ means one or more monomial sets are not distinct. Since $\alpha_3, \ldots, \alpha_j$ were chosen at random from $[1, p-1]$, we have by Schwartz-Zippel

$$\Pr[\Delta(\alpha_3, \ldots, \alpha_j) = 0] \le \frac{\deg(\Delta)}{p-1}.$$

We have $\deg M_{ikl} \le d$ and $\deg \Delta \le \sum_{0 \le i+k \le d} d\binom{s_{ik}}{2}$. Note that $\sum s_{ik} = T_{f_j}$ and $\deg \Delta$ is maximized when one of the coefficients has all T_{f_j} terms, that is, some $s_{ik} = T_{f_j} \le T_f$. Thus, $\deg \Delta \le d\binom{T_f}{2}$ and we obtain

$$\Pr[\Delta(\alpha_3, \ldots, \alpha_j) = 0] \le \frac{dT_f^2}{2(p-1)}.$$

So for $p \gg 1 + dT_f^2/2$, we expect that a different choice of $(\alpha_3, \ldots, \alpha_j)$ will satisfy the condition. This is again a very generous bound. In our experiments we have used $p = 2^{31} - 1$ and Algorithm **SparseInt** has never failed at Step 4.

Theorem 19. *Let $a = fg \in \mathbb{Z}_p[x_1, \ldots, x_n]$ with f, g monic in x_1 and $T_f \le T_g$. Let $d = \deg(a)$, $r = \frac{n}{n+d}$, and p be a big prime with $p \gg d$. Then with the probability of failure $\le (n-2)d^2 T_f \left(\frac{1}{p-1} + \frac{1}{2(p-2d+1)}\right)$, the expected complexity of MHL to recover the factors f, g via MTSHL algorithm is in*

$$\mathcal{O}\left(\frac{n^2}{d} T_g^3 + n^2 T_g^2 + n^2 d^2 T_g + n d T_g^2 + n d^4\right).$$

8. Some Timing Data

To compare our algorithms to Wang's, we first factored the determinants of Toeplitz and Cyclic matrices of different sizes as concrete examples. These are dense problems where Wang's algorithm should fare well in comparison with our sparse algorithm. Then we generated random sparse examples.

We have also included a comparison of Maple's factorization timings with Singular and Magma to be sure that the main gain by MTSHL is independent of implementation of Wang's algorithm. For multivariate factorization Maple, Singular and Magma all use Wang's MHL following the presentation in (GCL).

In the tables that follow all timings are in CPU seconds and were obtained on an Intel Core i5–4670 CPU running at 3.40GHz with 16 gigabytes of RAM. For all Maple timings, we set `kernelopts(numcpus=1);` to restrict Maple to use only one core as otherwise it will do polynomial multiplications and divisions in parallel whereas Singular and Magma have only serial codes.

8.1. *Factoring determinants of Toeplitz and Cyclic matrices*

Let C_n denote the $n \times n$ cyclic matrix and let T_n denote the $n \times n$ symmetric Toeplitz. See Figure 2.

$$C_n = \begin{pmatrix} x_1 & x_2 & \dots & x_{n-1} & x_n \\ x_n & x_1 & \dots & x_{n-2} & x_{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_3 & x_4 & \dots & x_1 & x_2 \\ x_2 & x_3 & \dots & x_n & x_1 \end{pmatrix} \quad \text{and} \quad T_n = \begin{pmatrix} x_1 & x_2 & \cdots & x_{n-1} & x_n \\ x_2 & x_1 & \cdots & x_{n-2} & x_{n-1} \\ & \ddots & \ddots & \ddots & \\ x_{n-1} & x_{n-2} & \cdots & x_1 & x_2 \\ x_n & x_{n-1} & \cdots & x_2 & x_1 \end{pmatrix}$$

Figure 2. The determinants of C_n and T_n are homogeneous polynomials in $x_1, x_2, \ldots, x_n$.

We implemented MTSHL in Maple with two key parts: BDP (see Section 2.2) and the evaluation routine (see Section 2.3) coded in C. The data in Tables 6 and 7 are for factoring the determinants of C_n and T_n. It compares Maple 2017 with Magma 2.22–5 and Singular 3–1–6. $\det T_n$ and $\det C_n$ are homogeneous. Maple, Magma, Singular all check for the homogeneity before factorization and if it is the case they first de-homogenize the polynomial to be factored. After factoring the de-homogenized polynomial, they homogenize the factors to obtain the actual factors.

To have a fair comparison, we de-homogenized the determinants by fixing $x_n = 1$ for all three systems. That is, for the determinant $\det T_n$ (or $\det C_n$), we time the factorization of $d_n = \det T_n(x_n = 1)$. In this way, we eliminate the de-homogenization and homogenization time which can be significant, as the determinants and the factors to be computed are huge for large n. Also d_n is monic in x_1, so we eliminate the leading coefficient correction timing for MTSHL. Finally by any choice of ideal, the univariate factorization time of the projection of d_n into $\mathbb{Z}[x_1]$ is negligible, that is, the timings simply represent multivariate Hensel lifting time for all three systems.

The column (MDP) shows the number of calls (including recursive calls) to Maple's MDP algorithm and the percentage of time in Hensel lifting spent solving MDPs. Data for the number of terms of $\det T_n$ and $\det C_n$ and the number of terms of their factors is also given in Tables 6 and 7. In Table 7 notice that the second factor for $n = 7, 11, 13$ has more terms than $\det C_n$. Also in Table 7 NA means we could neither compute the determinant in Singular using Singular's determinant command nor read the determinant nor it's factors into Singular to time Singular's factorize command.

The data confirms the data from (MP14) that Maple's multivariate factorization code is relatively fast. This is mainly because the underlying polynomial arithmetic is fast (MP14). We note here that Maple, Magma (see (Ste)) and Singular (see (Lee13)) are all doing Hensel lifting one variable at a time (see Algorithm 6.4 of (GCL)) and lifting solutions to MDP equations one variable at a time (see Algorithm 6.2 of (GCL)). All coefficient arithmetic is done modulo a prime p or prime power p^l which bounds the size the coefficients of any factors of the input. Singular (see (Lee13)) differs from Maple and Magma in that it first factors a bivariate image $f(x_1, x_2, \alpha_3, \ldots, \alpha_n)$ over $\mathbb{Z}$ then starts the Hensel lifting from bivariate images of the factors.

n	$\#d_n$	#factors	Maple	(MDP)	Magma	Singular
7	427	30,56	0.035	161,30%	0.01	0.02
8	1628	167,167	0.065	383,43%	0.04	0.05
9	6090	153,294	0.166	1034,73%	0.10	0.28
10	23797	931,931	0.610	2338,76%	0.89	1.77
11	90296	849,1730	2.570	6508,74%	1.96	8.01
12	350726	5579,5579	19.45	15902,80%	72.17	84.04
13	1338076	4983,10611	84.08	45094,84%	181.0	607.99
14	5165957	34937,34937	637.8	103591,77%	6039.0	20333.45
15	19732508	30458,66684	4153.2	286979,84%	12899.2	–

Table 6. Factorization timings in CPU seconds for factoring $d_n = \det(T_n)(x_n = 1)$, the determinant of the n by n Toeplitz matrix T_n evaluated at $x_n = 1$

n	$\#d_n$	#factors	Maple	(MDP)	Magma	Singular
7	246	7,924	0.045	330,90%	0.01	0.02
8	810	8,8,20,86	0.059	218,46%	0.07	0.06
9	2704	9,45,1005	0.225	1810,74%	0.74	0.24
10	7492	10,10,715,715	0.853	1284,62%	8.44	2.02
11	32066	11,184756	7.160	75582,91%	104.3	11.39
12	86500	12,12,42,78,78,621	19.76	1884,76%	7575.1	30.27
13	400024	13, 2704156	263.4	1790701,92%	30871.90	NA
14	1366500	14,14,27132,27132	1664.4	50381,77%	$> 10^6$	288463.17
15	4614524	15,120,3060,303645	18432.	477882,82%	–	NA

Table 7. Factorization data and timings in CPU seconds for factoring $d_n = \det(C_n)(x_n = 1)$, the determinant of the n by n Cyclic matrix C_n evaluated at $x_n = 1$

8.2. Factoring Toeplitz and Cyclic matrices with MTSHL

For MTSHL, it is important that α_i's in the ideal $I = \langle x_2 - \alpha_1, x_3 - \alpha_2, \cdots, x_n - \alpha_n \rangle$ are chosen from a large interval. For these we chose α_i's randomly from $[1, 65520]$. On the other hand, note that when we factored d_n with Maple, Magma and Singular to form tables 6 and 7, Wang's algorithm chose its own ideal. It can include some zeros, although it is not possible to choose all zero for these examples.

Tables 8 and 9 presents timings for Hensel lifting to factor d_n, the de-homogenized $\det T_n$ and $\det C_n$ with MTSHL. The column notation used in the tables is explained in Figure 3.

The density ratio of factors of d_n can be seen in Table 8. For example, the de-homogenization of d_n is of total degree 15 which has 2 factors. The first factor computed is in 14 variables of total degree 8, has 66684 terms and density ratio 0.208537. The second factor is in 14 variables of total degree 7, has 30458 terms and density ratio 0.261937.

Factoring d_n is a challenging problem. They are huge and can be considered as dense polynomials. The factors have total degree small and less than their total number of variables. Our natural expectation in this case is that Wang's approach is preferable to sparse approaches.

tW	is the time for Wang's algorithm which Maple currently uses (see(GCL)),
tBS	is the time for the factoring algorithm based on Algorithm 3; it uses Zippel's variable at a time sparse interpolation,
tMTSHL	is the time where factoring algorithm is based on Algorithm 5 ,
tX(tY)	means factoring time tX with tY seconds spent on solving MDP,
t_{mul}	means time spent on multiplication in MTSHL,
t_{eval}	means time spent on evaluation in MTSHL,
T_{f_i}	denotes the number of terms of a factor
t_{f_i}	denotes the density ratio of a factor

Figure 3. Notation for Tables 8–12

As can be seen from Table 6, if we factor d_{14} and d_{15} using Maple that uses Wang's algorithm for multivariate factorization, the calculation will take 637 s. and 4153 s. resp. MTSHL factors d_{14} and d_{15} in 250s. and 1650 s. resp.

Table 9 presents timings for Hensel liftings to factor $d_n = \det C_n$ with MTSHL. **For C_n the density ratio is 1** for all factors except for $n = 12$, in which out of 6 factors one has $t_f = 0.53$ and one has $t_f = 0.45$ and for $n = 15$, one of the 4 factors has density ratio 0.95.

The timings in the columns tW and tMTSHL show that in general the most time dominating step of MHL is solving MDP and it confirms that even for complicated examples MTSHL is quicker than Wang's algorithm, because it spends less time to solve MDP. Also, the values in the columns t_{mul} and t_{eval} confirm the theoretical complexity analysis that evaluation and multiplication are the most time dominating operations in MTSHL. We have not reported the time spent solving Vandermonde systems because it was always $< 10\%$. Also, except for C_{15} the time spent in trial divisions is $< 10\%$. For C_{15} it was 3362 seconds.

n	t_{f_1}	t_{f_2}	tW	tMTSHL	t_{mul}	t_{eval}
7	0.27	0.36	0.035 (0.015)	0.046 (0.037)	0.001	0.003
8	0.50	0.50	0.065 (0.028)	0.073 (0.059)	0.007	0.005
9	0.31	0.23	0.166 (0.121)	0.122 (0.075)	0.018	0.001
10	0.47	0.47	0.610 (0.467)	0.418 (0.251)	0.099	0.024
11	0.22	0.28	2.570 (1.902)	1.138 (0.458)	0.339	0.053
12	0.45	0.45	19.45 (15.56)	13.165 (5.445)	3.779	0.897
13	0.27	0.21	84.08 (70.623)	21.769 (11.064)	6.904	4.361
14	0.45	0.45	637.8 (491.106)	249.961 (160.04)	71.351	102.918
15	0.21	0.26	4153.2 (1771.54)	1651.68 (689.634)	674.356	405.016

Table 8. Timings for factoring $\det(T_n)(x_n = 1)$.

8.3. Factoring random sparse polynomials with MTSHL

To compare MTSHL with Wang's algorithm on randomly generated examples, we created random sparse multivariate polynomials A, B in Maple and computed $C = AB$. We used $p = 2^{31} - 1$ and **two ideal types to factor C:**

ideal type 1: $I = \langle x_2 - 0, x_3 - 0, \cdots, x_n - 0 \rangle$ and

ideal type 2: $I = \langle x_2 - \alpha_1, x_3 - \alpha_2, \cdots, x_n - \alpha_n \rangle$

n	tW	tMTSHL	t_{mul}	t_{eval}
7	0.041 (0.012)	0.026 (0.015)	0.002	0.001
8	0.057 (0.025)	0.063 (0.046)	0.010	0.003
9	0.209 (0.152)	0.12 (0.042)	0.024	0.002
10	0.845 (0.642)	0.5 (0.22)	0.20	0.01
11	6.6 (4.884)	0.945 (0.094)	0.386	0.003
12	19.76 (15.808)	5.121 (1.385)	3.108	0.048
13	252.2 (211.848)	27.689 (1.474)	9.362	0.093
14	1861.8 (1563.912)	523.073 (85.326)	346.067	38.399
15	18432.0 (14929.2)	7496.94 (426.014)	3602.739	19.231

Table 9. Timings for factoring $\det(C_n)(x_n = 1)$.

For Wang's algorithm, the first attempt should be to try an ideal of type 1, because a sparse polynomial remains sparse in this case and hence the number of MDP to be solved significantly decreases. If it does not work, for ideal type 2 , the α_i's are chosen from a small interval including zero.

As noted it is not always possible to use ideal type 1 because the leading coefficient of C must not vanish at $\alpha_2, \ldots, \alpha_n$ and also, the factors A and B must be relatively prime at $\alpha_2, \ldots, \alpha_n$. To generate random examples where ideal type 1 cannot be used, we chose A and B of the form

$$(i)\ x_1^d + (\prod_{i=1}^{n} x_i) \cdot \texttt{randpoly}([x_1, .., x_n], \texttt{degree} = d - n, \texttt{terms} = T, \texttt{coeffs} = \texttt{rand}(1..99))$$

$$(ii)\ (\sum_{i=1}^{n} x_i \cdot \texttt{randpoly}([x_1, .., x_n], \texttt{degree} = d - 1, \texttt{terms} = T/n, \texttt{coeffs} = \texttt{rand}(1..99)) + c$$

where c is small positive integer and the Maple command `randpoly` again is used to generate random polynomials. So, for (i), one must choose all ideal points to be non-zero and for (ii) one cannot choose ideal type 1 but one can choose some of the evaluation points to be zero for Wang's algorithm.

Tables 10 and 11 present timings for the randomly generated data of the form (i) and (ii). They show that for the both cases MTSHL is significantly faster than Wang's algorithm.

We also included the timings for ideal type 1 case, as according to our experiments it is the only case where Wang's algorithm is quicker. In this case, the evaluation cost of sparse interpolation becomes dominant which is not the case for Wang's algorithm for the ideal type 1. To generate random examples where ideal type 1 can be used, we chose A and B in the form

$$x_1^d + \texttt{randpoly}([x_1, .., x_n], \texttt{degree} = d - 1, \texttt{terms} = T) + c$$

where c is a small positive integer.

Table 12 presents timings for the random data for which ideal type 1 is used. For the ideal type 1 case MTSHL was not used, since the zero evaluation probability is large for the sparse case. According to our experiments Wang's algorithm is faster for $t_f < 0.2$. For $t_f \geq 0.2$ the performance of Wang's algorithm and Algorithm 2 (which uses sparse interpolation without SHL assumption) are almost the same.

$n/d/T$	tW	tMTSHL	$n/d/T$	tW	tMTSHL
3/35/100	1.846 (1.343)	0.724 (0.043)	7/35/100	567.965 (566.607)	2.269 (0.527)
3/35/500	3.359 (1.796)	1.704 (0.056)	7/35/500	> 3000. stopped	43.139 (6.839)
5/35/100	50.900 (49.594)	2.102 (0.308)	8/35/100	1859.862 (1858.204)	2.309 (0.547)
5/35/500	237.844 (217.310)	32.121 (2.882)	8/35/500	> 3000. stopped	47.446 (7.385)
6/35/100	174.220 (174.205)	2.031 (0.415)	9/35/100	> 3000. stopped	2.937 (0.558)
6/35/500	923.003 (897.277)	38.997 (5.096)	9/35/500	> 3000. stopped	79.585 (9.715)

Table 10. The timing table for random data with ideal type 2, (i)

$n/d/T$	tW	tMTSHL
3/20/100	0.264 (0.204)	0.15 (0.007)
3/20/500	0.443 (0.247)	0.288 (0.01)
5/20/100	4.131 (3.682)	0.581 (0.126)
5/20/500	21.922 (17.635)	5.028 (1.009)
7/20/100	30.442 (29.654)	0.958 (0.29)
7/20/500	138.128 (129.054)	14.285 (3.081)
9/20/100	113.421 (112.632)	1.09 (0.359)
9/20/500	1088.882 (1073.387)	20.528 (5.6)

Table 11. The timing table for random data with ideal type 2, (ii)

$n/d/T$	t_f	tW	tBS
5/20/5000	0.1	7.08 (2.605)	11.804 (7.376)
5/15/3000	0.2	4.25 (1.554)	4.963 (2.29)
5/15/5000	0.3	6.882 (2.988)	6.471 (2.99)
4/20/5000	0.4	2.709 (1.211)	2.704 (1.267)
5/20/30000	0.56	86.224 (18.394)	90.111 (21.134)

Table 12. The timing table for random data with ideal type 1

9. Conclusion

We have shown that solving the multivariate polynomial diophantine equations that arise in Wang's multivarite Hensel lifting algorithm can be improved by using sparse interpolation. This leads to an overall improvement in multivariate polynomial factorization. Our experiments show that the improvement is practical.

In the paper we have attempted an average case complexity analysis for our sparse Hensel lifting algorithm. In order to do this we needed to know how many terms appear in a sparse polynomial $f(x_1, \ldots, x_n)$ when we successively evaluate its variables at integers one at a time. We also needed to know how many terms appear in the coefficients of a Taylor series expansion of a sparse polynomial expanded about a non-zero point. These results (Proposition 11 and Lemma 16) may be useful elsewhere.

We only presented algorithms for the case when $a(x_1, \ldots, x_n)$ has two irreducible factors f and g. Let $a_{j-1} = f_1 f_2 \cdots f_m$ with $m \geq 2$ be the factorization of $a(x_1, \ldots, x_{j-1}, \alpha_j, \ldots, \alpha_n)$ from step $j-1$. At the j'th step of Hensel lifting, Algorithm 5 must solve multivariate polynomial diophantine equations of the form

$$\sigma_1 \frac{a_{j-1}}{f_1} + \sigma_2 \frac{a_{j-1}}{f_2} + \cdots + \sigma_m \frac{a_{j-1}}{f_m} = c$$

for $\sigma_i \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$ with $\deg(\sigma_i, x_1) < \deg(f_i, x_1)$ for $1 \le i \le m$. Currently our implementation follows (GCL) and solves this m term MDP by solving $m-1$ two term MDPs. It first solves $\sigma_1 \frac{a_{j-1}}{f_1} + \tau_1 f_1 = c$ for σ_1 and τ_1 where $\tau_1 = \sum_{i=2}^{m} \sigma_i \frac{a_j}{f_1 f_i}$. Then it solves $\sum_{i=2}^{m} \sigma_i \frac{a_j}{f_1 f_i} = \tau_1$ for the σ_i recursively. We are experimenting with using sparse interpolation to simultaneously interpolate all σ_i from bivariate images.

References

[AS72] Milton Abramowitz and Irene Stegun. *Handbook of Mathematical Functions.* 9th printing, Dover, New York. (1970).

[CLO] David Cox, John Little, Donal O'Shea. *Ideals, Varieties and Algorithms.* Springer-Verlag, 3rd ed., (2007).

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

[Gel52] A.O. Gelfond, *Transcendental and Algebraic Numbers*, GITTL, Moscow, 1952; English translation by Leo F. Boron, Dover, New York, 1960.

[Kal85] Kaltofen, E., Sparse Hensel lifting. Proc. EUROCAL '85, Springer Verlag LNCS, vol 204, pp. 4–17, (1985).

[Lee13] Martin M. Lee. Factorization of multivariate polynomials. Ph.D. Thesis. (2013).

[MY74] Miola A., Yun D. Y. Y. Computational Aspects of Hensel-type Univariate Polynomial Greatest Common Divisor Algorithms. *Proceedings of EUROSAM '74*, ACM, pp. 46–54, (1974).

[MT16a] Michael Monagan and Baris Tuncer. Some results on counting roots of polynomials and the Sylvester resultant. *Proceedings of FPSAC 2016*, DMTCS. pp. 887–898, (2016).

[MT16b] Michael Monagan and Baris Tuncer. Using Sparse Interpolation in Hensel Lifting. Proceedings of CASC 2016, Springer-Verlag LNCS **9890**, 381–400 (2016).

[MP14] Michael Monagan and Roman Pearce. POLY. A New Polynomial Data Structure for Maple 17. *Computer Mathematics*, Springer Verlag, 325–348, (2014).

[Sch80] Schwartz, Jack . Fast probabilistic algorithms for verification of polynomial identities. *Journal of* the ACM 27:701–717, (1980).

[Ste] Steel, Allan. Private Communication.

[TE51] F.G. Tricomi, A. Erdelyi, The asymptotic expansion of a ratio of gamma functions, Pacific Journal of Mathematics, Nov 1951, Vol:1, No:1.

[Wan78] Wang, P.S. An improved Multivariate Polynomial Factoring Algorithm, *Mathematics of Computation*, **32**, (1978).

[Wan75] Wang, P.S., Rothschild, L.P. Factoring multivariate polynomials over the integers. *Mathematics of Computation*, vol 29, NUMBER 131, pp. 935–950, (1975).

[Yun74] Yun, D.Y.Y. The Hensel Lemma in algebraic manipulation. Ph.D. Thesis. (1974)

[Zip79] Zippel, R.E. Probabilistic algorithms for sparse polynomials. *Proc. EUROSAM '79*, Springer Lec. Notes Comp. Sci., vol. 72, pp. 216–226, (1979).

[Zip81] Zippel, R.E. Newton's iteration and the sparse Hensel algorithm. *Proc.* ACM Symp. Symbolic Algebraic Comp., 68–72, (1981).

[Zip90] Zippel, R.E. Interpolating polynomials from their values. *J. Symbolic Comput.*, **9**(3):375–403, (1990).

Appendix MTSHL

We give an example of our SHL. Suppose we seek to factor $a = fg$ where

$$f = {x_1}^8 + 2\,x_1{x_2}^2{x_4}^3x_5 + 4\,x_1{x_2}^2{x_3}^3 + 3\,x_1{x_2}^2x_4{x_5}^2 + {x_2}^2x_3x_4 - 5$$
$$g = {x_1}^8 + 3\,{x_1}^2x_2x_3{x_4}^2x_5 + 5\,{x_1}^2x_2{x_3}^2x_4 - 3\,{x_4}^2{x_5}^2 + 4\,x_5$$

Let $\alpha_3 = 1$ and $p = 2^{31} - 1$. Before lifting x_5 we have

$$f^{(0)} := f(x_5 = 1) = {x_1}^8 + 4\,x_1{x_2}^2{x_3}^3 + 2\,x_1{x_2}^2{x_4}^3 + 3\,x_1{x_2}^2x_4 + {x_2}^2x_3x_4 - 5$$
$$g^{(0)} := g(x_5 = 1) = {x_1}^8 + 5\,{x_1}^2x_2{x_3}^2x_4 + 3\,{x_1}^2x_2x_3{x_4}^2 - 3\,{x_4}^2 + 4$$

satisfying $a(x_5 = \alpha_5) = f^{(0)}g^{(0)}$. If the SHL assumption is true then at the first step we assume $f = \sum_{i=0}^{\deg_{x_5} f} f_i(x_5 - 1)^i$ and $g = \sum_{i=0}^{\deg_{x_5} g} g_i(x_5 - 1)^i$ where f_1 and g_1 are in the form

$$f_1 = \left(c_1{x_3}^3 + c_2{x_4}^3 + c_3x_4\right) x_1{x_2}^2 + c_4{x_2}^2x_3x_4 + c_5$$
$$g_1 = \left(c_6{x_3}^2x_4 + c_7x_3{x_4}^2\right) {x_1}^2x_2 + c_8{x_4}^2 + c_9$$

for unknowns $\{c_1, \ldots, c_9\}$. In the following $e_5^{(k)}$ denotes the coefficient of $(x_5 - 1)^k$ in the Taylor expansion of the *error* about $x_5 = 1$. Let also $f_0 := f^{(0)}$, $g_0 := g^{(0)}$, $f^{(k)} := \sum_{i=0}^{k} f_i(x_5 - 1)^i$ and $g^{(k)} := \sum_{i=0}^{k} g_i(x_5 - 1)^i$. We start by computing the first error term $e_5^{(1)} = a - f^{(0)}g^{(0)}$. We obtain

$$\begin{aligned}e_5^{(1)} &= 3\,{x_1}^{10}x_2x_3{x_4}^2 + 2\,{x_1}^9{x_2}^2{x_4}^3 + 6\,{x_1}^9{x_2}^2x_4 + 12\,{x_1}^3{x_2}^3{x_3}^4{x_4}^2 + 10\,{x_1}^3{x_2}^3{x_3}^2{x_4}^4 \\ &+ 12\,{x_1}^3{x_2}^3x_3{x_4}^5 - 6\,{x_1}^8{x_4}^2 + 30\,{x_1}^3{x_2}^3{x_3}^2{x_4}^2 + 27\,{x_1}^3{x_2}^3x_3{x_4}^3 + 3\,{x_1}^2{x_2}^3{x_3}^2{x_4}^3 \\ &+ 4\,{x_1}^8 - 24\,x_1{x_2}^2{x_3}^3{x_4}^2 - 18\,x_1{x_2}^2{x_4}^5 - 15\,{x_1}^2x_2x_3{x_4}^2 + 16\,x_1{x_2}^2{x_3}^3 \\ &- 20\,x_1{x_2}^2{x_4}^3 - 6\,{x_2}^2x_3{x_4}^3 + 36\,x_1{x_2}^2x_4 + 4\,{x_2}^2x_3x_4 + 30\,{x_4}^2 - 20\end{aligned}$$

The MDP to be solved is $D := g_1f_0 + f_1g_0 = e_5^{(1)}$. Since f_1 has three terms in $x_1x_2^2$ and g_1 has two terms in $x_1^2x_2$ we will interpolate g_1 using two evaluations then obtain f_1 by division. We choose $(x_3 = 2, x_4 = 3)$ and $(x_3 = 2^2, x_4 = 3^2)$ and compute $D(x_3 = 2, x_4 = 3)$:

$$\begin{aligned}&\left({x_1}^8 + 95\,x_1{x_2}^2 + 6\,{x_2}^2 - 5\right)\left((12\,c_6 + 18\,c_7)\,{x_1}^2x_2 + 9\,c_8 + c_9\right) \\ &\quad + \left({x_1}^8 + 114\,{x_1}^2x_2 - 23\right)\left((8\,c_1 + 27\,c_2 + 3\,c_3)\,x_1{x_2}^2 + 6\,c_4{x_2}^2 + c_5\right) \\ &= 54\,{x_1}^{10}x_2 + 72\,{x_1}^9{x_2}^2 - 50\,{x_1}^8 + 13338\,{x_1}^3{x_2}^3 + 324\,{x_1}^2{x_2}^3 - 270\,{x_1}^2x_2 \\ &\quad - 6406\,x_1{x_2}^2 - 300\,{x_2}^2 + 250\end{aligned}$$

and similarly $D(x_3 = 4, x_4 = 9)$. Calling BDP to solve these bivariate Diophantine equations we obtain the solutions $[\sigma_1, \tau_1] = [54\,{x_1}^2x_2 - 50, 72\,x_1{x_2}^2]$ and $[\sigma_2, \tau_2] = [972\,{x_1}^2x_2 - 482, 1512\,x_1{x_2}^2]$. Hence we have

$$(12\,c_6 + 18\,c_7)\,{x_1}^2x_2 + 9\,c_8 + c_9 = 54\,{x_1}^2x_2 - 50$$
$$(144\,c_6 + 324\,c_7)\,{x_1}^2x_2 + 81\,c_8 + c_9 = 972\,{x_1}^2x_2 - 482$$

Then we solve the Vandermonde linear systems

$$\begin{bmatrix} 12 & 18 \\ 144 & 324 \end{bmatrix} \begin{bmatrix} c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 54 \\ 972 \end{bmatrix} \text{ and } \begin{bmatrix} 9 & 1 \\ 81 & 1 \end{bmatrix} \begin{bmatrix} c_8 \\ c_9 \end{bmatrix} = \begin{bmatrix} -50 \\ -482 \end{bmatrix}$$

to obtain $c_6 = 0, c_7 = 3, c_8 = -6, c_9 = 4$. So $g_1 = 3\,{x_1}^2 x_2 x_3 {x_4}^2 - 6\,{x_4}^2 + 4$. Then by division we get $f_1 = (e_5^{(1)} - f_0 g_1)/g_0 = 2\,x_1 x_2 {x_4}^3 + 8\,x_2 {x_3}^4$. Hence

$$\begin{aligned} f^{(1)} &= f_0 + \left(2\,x_1 {x_2}^2 {x_4}^3 + 6\,x_1 {x_2}^2 x_4\right)(x_5 - 1) \\ g^{(1)} &= g_0 + \left(3\,{x_1}^2 x_2 x_3 {x_4}^2 - 6\,{x_4}^2 + 4\right)(x_5 - 1)\,. \end{aligned}$$

Note that we use the division step above also as a check for the correctness of the SHL assumption that $\mathrm{Support}(g_1) \subseteq \mathrm{Support}(g_0)$. Since the solution to the MDP is unique, we would have $g_0 \nmid (e_5^{(1)} - f_0 g_1)$, if this assumption were wrong.

Now following Lemma 1 by looking at the monomials of f_1 and g_1, we assume that the form of the f_2 and g_2 are

$$\begin{aligned} f_2 &= c_1 x_1 {x_2}^2 {x_4}^3 + c_2 x_1 {x_2}^2 x_4 + c_3 \\ g_2 &= c_4 {x_1}^2 x_2 x_3 {x_4}^2 + c_5 {x_4}^2 + c_6 \end{aligned}$$

for some unknowns $\{c_1, \ldots, c_6\}$. After computing the next error $a - f^{(1)} g^{(1)}$ we compute $e_5^{(2)}$ and the MDP to be solved is $D := f_0 g_2 + g_0 f_2 = e_5^{(2)}$. We need 2 evaluations again to solve for g_2. Choose $(x_3 = 5, x_4 = 6)$ and $(x_3 = 5^2, x_4 = 6^2)$ and compute

$$\begin{aligned} D(x_3 = 5, x_4 = 6) := {} & \left({x_1}^8 + 950\,x_1 {x_2}^2 + 30\,{x_2}^2 - 5\right)\left(180\,c_4 {x_1}^2 x_2 + 36\,c_5 + c_6\right) \\ & + \left({x_1}^8 + 1290\,{x_1}^2 x_2 - 104\right)\left(216\,c_1 x_1 {x_2}^2 + 6\,c_2 x_1 {x_2}^2 + c_3\right) \\ & = 18\,{x_1}^9 {x_2}^2 - 108\,{x_1}^8 + 23220\,{x_1}^3 {x_2}^3 - 104472\,x_1 {x_2}^2 - 3240\,{x_2}^2 + 540 \end{aligned}$$

and similarly for $D(x_3 = 25, x_4 = 36)$. Calling BDP we obtain the solutions to these bivariate Diophantine equations $[\sigma_1, \tau_1] = [-108, 18\,x_1 {x_2}^2]$ and $[\sigma_2, \tau_2] = [-3888, 108\,x_1 {x_2}^2]$ respectively. Hence we have $180\,c_4 {x_1}^2 x_2 + 36\,c_5 + c_6 = -108$ and $32400\,c_4 {x_1}^2 x_2 + 1296\,c_5 + c_6 = -3888$ respectively. Then we solve the Vandermonde linear systems

$$[180]\,[c_4] = [0] \text{ and } \begin{bmatrix} 36 & 1 \\ 1296 & 1 \end{bmatrix} \begin{bmatrix} c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} -108 \\ -3888 \end{bmatrix}$$

to obtain $c_4 = 0, c_5 = -3, c_6 = 0$. So $g_2 = -3\,{x_4}^2$. Then by division we get $f_2 = (e_5^{(2)} - f_0 g_2)/g_0 = 3\,x_1 {x_2}^2 x_4\,(x_5 - 1)^2$. Hence

$$\begin{aligned} f^{(2)} &= f^{(1)} + 3\,x_1 {x_2}^2 x_4\,(x_5 - 1)^2 \\ &= {x_1}^8 + 2\,x_1 {x_2}^2 {x_4}^3 x_5 + 4\,x_1 {x_2}^2 {x_3}^3 + 3\,x_1 {x_2}^2 x_4 {x_5}^2 + {x_2}^2 x_3 x_4 - 5 \\ g^{(2)} &= g^{(1)} + \left(-3\,{x_4}^2\right)(x_5 - 1)^2 \\ &= {x_1}^8 + 3\,{x_1}^2 x_2 x_3 {x_4}^2 x_5 + 5\,{x_1}^2 x_2 {x_3}^2 x_4 - 3\,{x_4}^2 {x_5}^2 + 4\,x_5. \end{aligned}$$

Since the next error $a - f^{(2)} g^{(2)} = 0$ we have found the factors of a.