The Complexity Analysis of the MutantXL Family

Mohamed Saied Emam Mohamed[1], Jintai Ding[2], and Johannes Buchmann[1]

[1] TU Darmstadt, FB Informatik
Hochschulstrasse 10, 64289 Darmstadt, Germany
{mohamed,buchmann}@cdc.informatik.tu-darmstadt.de
[2] Department of Mathematical Sciences, University of Cincinnati,
South China University of Technology
jintai.ding@uc.edu

Abstract. Algebraic attacks are based on the problem of solving systems of multivariate polynomial equations. The complexity of algorithms for solving this problem is essentially affected by the method of enlarging these multivariate systems. The MutantXL algorithm was presented as an efficient algorithm for solving multivariate systems. In this paper, we study the complexity of the MutantXL algorithm and give an upper bound to the number of mutants necessary for terminating the computations of the algorithm. At the ePrint archive, E. Thomae and C. Wolf recently published a new paper and presented two formulas for an upper bound on the number of linearly independent equations generated by MutantXL. We have noticed that these two formulas are not accurate. The first one leads to, indeed, a large upper bound. However, the second one does not take into account a lot of linearly independent computations. We indicate the errors in these formulas and propose new ones. Our formulas based on a theoretical evaluation of the maximum number of linearly independent equations produced by MutantXL. We have verified the new formulas with a large number of experiments on some HFE and random generated systems. Furthermore, we study the complexity of the MGB algorithm for computing Gröbner basis. The analysis of MGB suggested a new upper bound for the complexity of computing Gröbner bases.

Keywords: Solving Multivariate System, Complexity, Matrix-Based Algorithms, XL, MutantXL, MGB.

1 Introduction

Solving systems of multivariate non-linear polynomial equations is one of the important research problems in symbolic computation. The standard way to solve this problem is to compute a Gröbner basis of the polynomial ideals generated by such systems.

The Buchberger algorithm [4] was the first algorithm for computing Gröbner bases. It is based on the computation of Gröbner bases using the so-called s-polynomials. F_4 [9] and F_5 [10] algorithms are two advanced algorithms that use linear algebra and Buchberger's s-polynomial techniques to compute Gröbner bases. The XL algorithm was introduced in [6] as an algorithm for solving polynomial equations in case only a single solution exists.

The complexity of these algorithms is based on the size of a large linear system. In [1, 19, 16], complexity bounds for Gröbner bases computations and the XL algorithm were given. Furthermore, a comparison between XL and (F_4) was presented in [11, 17]. It was concluded that F_4 computes a Gröbner basis faster and uses fewer memory resources than XL.

The MutantXL algorithm was proposed as a variant of XL that is based on the mutant strategy [8, 7]. The MXL_3 algorithm [14] is an improvement to MutantXL that uses an optimized mutant strategy, the partial enlargement technique [15] and a mutant based termination criterion [14]. In [5] the MGB algorithm, a new variant of MutantXL, was introduced. It uses a flexible partial enlargement technique to extend the system.

In this paper, we investigate the complexity of the MutantXL algorithm and present new upper bounds for computing Gröbner basis and solving polynomial systems that emphasize the efficiency of MutantXL. The MGB algorithm is an improvement for both MXL_2 and MXL_3 algorithms, so we sufficiently study the complexity of MGB as a variant of MutantXL. Our study based on random systems (semi-regular systems) and some HFE systems as an application from cryptography.

This paper is organized as follows. In Section 2 we revisit the current boundaries of complexity and generalize XL for computing Gröbner basis. In Section 3 we describe the MutantXL algorithm and study its complexity in Section 4. A complexity analysis of the MGB algorithm is discussed in Section 5. Finally, we conclude the paper in Section 6. Before continuing let us introduce the necessary notation.

1.1 Notation

Let $X := \{x_1, \ldots, x_n\}$ be a set of variables, upon which we impose the following order: $x_1 > x_2 > \ldots > x_n$. We consider the boolean ring

$$R = F[x_1, \ldots, x_n]/\langle x_1^q - x_1, ..., x_n^q - x_n\rangle$$

of polynomials over R in X over a finite field F. Here $x_i^q - x_i = 0$, $1 \leq i \leq n$ are polynomials of the so-called field equations. In R, each element is uniquely expressed as a polynomial where each x_i has a power less than q.

A term in the indeterminate $x_1, \ldots, x_n$ is a power product of the form $x_1^{e_1} \cdots x_n^{e_n}$ with $e_i \in \mathbb{N}$. We denote by T the set of all terms. A monomial is any product of a field element and a term.

Let $<$ denote a term order on T. The degree of $t = x_1^{e_1} \cdots x_n^{e_n} \in T$ is defined by $\deg(t) := \sum_{i=1}^{n} e_i$. For $f = \sum_{t \in T} c_t t \in R$, where $c_t \in F$ is the coefficient of t

in f. We define the terms of f by $T(f) := \{t \in T \mid c_t \neq 0\}$, the degree of f by $\deg(f) := \max\{\deg(t) \mid t \in T(f)\}$, the leading term of f by $\mathrm{LT}(f) := \max_{\leq} \mathrm{T}(f)$. We denote the set of exactly degree d terms in R by T_d

In this paper, we assume that the polynomials are ordered by a total-degree order. In the total-degree orderings, we compare total degree first. In case of the equality, there are many different orderings that can break the ties. The most commonly used are the graded lexicographic and the graded reverse lexicographic orderings.

Let $P = \{p_1, \ldots, p_m\}$ be a set of generic randomly generated polynomials in R, we assume that P is quadratic and the ideal generated by P is zero-dimensional ideal. We define the set H_d of degree $\leq d$ polynomials as follows

$$H_d := \{t \cdot p \mid t \text{ is a term in } R,\ p \in P,\ \deg(tp) \leq d\} \tag{1}$$

and the set of exactly degree d, $H_d^+ \subset H_d$, as

$$H_d^+ := \{h \in H_d \mid \ \deg(h) = d\}. \tag{2}$$

A row echelon form $\widetilde{H}_d$ of H_d is a basis for $\mathrm{span}(H_d)$ with pairwise distinct leading terms. Moreover, the set of leading terms $\mathrm{LT}(\widetilde{H}_d)$ is uniquely defined as

$$\mathrm{LT}(\widetilde{H}_d) = \mathrm{LT}(\mathrm{span}(\widetilde{H}_d)) = \mathrm{LT}(\mathrm{span}(H_d)).$$

In this context, the maximum number of linearly independent polynomials in $\mathrm{span}(H_d)$ can be defined as

$$I_d = \mathrm{LT}(\widetilde{H}_d).$$

2 The Complexity of XL and Gröbner Basis Algorithms

In this section, we review the complexity of XL and Gröbner basis algorithms for solving polynomial systems. Since it is common in algebraic computation that the complexity of solving a finite set of polynomial equations is essentially based on computing Gröbner basis. We have been adapted XL for computing Gröbner bases.

For this paper, we are more interesting in the so called matrix-based algorithms especially the algorithms that relay on the maximal degree appeared during the computations. A matrix-based algorithm A for computing a Gröbner basis of the ideal generated by P can be described as in Algorithm 1.

The main difference that affects the complexity of these algorithms is the method, they used to enlarge the system. They generate additional polynomials of not a very high degree in the ideal generated by the elements of P, $Enlarge(H)$.

The XL algorithm [6] is a version of the algorithm described in Algorithm 1 that enlarges the system P up to a certain degree d_{xl} by constructing all the elements of $H_{d_{xl}}$. However, XL works under the condition that the set of equations $(P = 0)$ has only one solution, which means that it computes a Gröbner

Algorithm 1 A(P)

$H \rightarrow P$
loop
  $G \rightarrow Echelonize(H)$
  if Termination criterion satisfied **then**
    return G
  else
    $Enlarge(H)$
  end if
end loop

basis under that condition. The authors of [14] have presented and proved a new criterion for computing Gröbner bases that they used with some variant of XL. Based on that, XL can compute a Gröbner basis in more general case. In this context, we can describe the XL algorithm as in Algorithm 2.

Algorithm 2 XL(P)

Require: P is a quadratic finite polynomials of R
$d \rightarrow 2$
$H_d \rightarrow P$
loop
  $G \rightarrow Echelonize(H_d)$
  if Termination criterion satisfied on d **then**
    return G
  else
    $d \rightarrow d + 1$
    $H_d \rightarrow H_d^+ \cup G$
  end if
end loop

Now we are going to estimate the highest degree d and justify the termination condition of XL. The complexity of the XL algorithm has been used to estimate the hardness of solving polynomial systems in many studies. In [19] Yang et al, introduced a formula to compute the number of linearly independent equations produced by XL of a maximal degree d. This formula has been verified by Rønjem and Raddum in [16].

Let we extend P up to degree d, the number of such linearly independent polynomial equations is computed as follows:

$$\mathrm{LT}(\mathrm{span}(H_d) = I_d = \sum_{i=0}^{\lfloor \frac{d_k}{d_e} \rfloor} (-1) \binom{m+i}{i+1} \sum_{j=0}^{d_k - i \cdot d_e} \binom{n}{j}, \tag{3}$$

where each polynomial in $P = (p_1, \ldots, p_m)$ has degree d_e and $d_k = d - d_e$ is the highest degree of monomials used to extend P.

However, to give an upper bound to the complexity of the XL algorithm, we have to estimate the degree of its extended system. Let d_{xl} be the highest degree of the polynomials generated by XL. In the worst case, the constructed set of polynomials by XL, $\widetilde{H}_{d_{xl}}$, satisfies

$$\forall t \in \bigcup_{i=d}^{d_x} T_i,\ \exists p \in \widetilde{H}_{d_{xl}} \ \text{ and } \ \mathrm{LT}(p) = t. \tag{4}$$

Using (3) and (4), an upper bound for the maximal degree of XL computations (d_{xl}) is given by

$$I_{d_{xl}} \geq (\sum_{i=1}^{d_{xl}} \mid T_i \mid) \tag{5}$$

Assuming that (5) is satisfied. The termination criterion of the XL version that computes a Gröbner basis of the ideal generated by P can be given in the following proposition.

Proposition 1. *Let P be a finite subset of R. The XL algorithm terminates at a maximal degree d_{xl} and returns a set $G = \widetilde{H}_{d_{xl}}$. If d_{xl} is the minimum degree that satisfies (5), then G is a Gröbner basis of the ideal generated by P, $\langle P \rangle$.*

Remark 1. Let $d \leq d_{xl}$ and $G \subset \widetilde{H}_{d_{xl}}$ such that

$$G = \{g \in \widetilde{H}_{d_{xl}} \mid \ \deg(g) \leq d\}.$$

G is a Gröbner basis if it satisfies the following condition

$$\mid T_d \mid = \mid \{g \in \widetilde{H}_{d_{xl}} \mid \deg(g) = d\} \mid .$$

The XL algorithm described in Algorithm 2 has the same complexity as the original XL algorithm. Based on Remark 1, let the system $P = 0$ have only one solution, then XL terminates and can return only the set G of highest degree one.

The complexity of computing Gröbner bases of a zero dimensional random system was studied in [1]. The authors of [1] used the concept of regularity and defined the degree of regularity (d_{reg}) of the ideal generated by the homogenous set of polynomials F as follows

$$d_{reg} = \min\{d \geq 0 \mid \ \forall t \in T_d\ \exists f \in \langle P \rangle \ \text{ and } \ \deg(f) = d\} \tag{6}$$

It is well known that the degree of regularity can be used as the basic parameter for the complexity of computing Gröbner bases. Assuming the affine generic system P, the degree of regularity (d_{reg}) can be defined in terms of our notation as

$$d_{reg} = \min\{d \geq 0 \mid \ \forall t \in T_d\ \exists p \in \widetilde{H}_d,\ \deg(p) = d\} \tag{7}$$

Remark 2. Using Proposition 1 and the definition of the degree of regularity in (7), XL can be terminated at a highest degree d_{xl}, $d_{reg} \leq d_{xl}$.

3 The MutantXL Algorithm

The MutantXL algorithm was given in [8] as an improved version of XL. Let the elements of the ideal be sorted by some degree ordering. In many experiments with the XL algorithm, we have obtained after echelonizing the enlarged system certain polynomials of degree lower than expected. Let the highest degree of the enlarged system be d. We can define and construct the coefficient matrix of the resulting degree d polynomials H_d as

$$H_d = \begin{pmatrix} \times & \times & \dots & \times & \times & \times & \dots & \times \\ \times & \times & \dots & \times & \times & \times & \dots & \times \\ \times & \times & \dots & \times & \times & \times & \dots & \times \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \times & \times & \dots & \times & \times & \times & \dots & \times \\ \times & \times & \dots & \times & \times & \times & \dots & \times \end{pmatrix}$$

where m_d is the number of degree d terms. Let the echelonization matrix $\widetilde{H}_d$ of $\mathcal{M}_d$ have some rows such that all degree d terms were canceled during the elimination process as follows.

$$\widetilde{H}_d = \begin{pmatrix} \times & \times & \dots & \times & \times & \times & \dots & \times \\ 0 & \times & \dots & \times & \times & \times & \dots & \times \\ 0 & 0 & \dots & \times & \times & \times & \dots & \times \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \times & \times & \dots & \times \\ 0 & 0 & \dots & 0 & 0 & \times & \dots & \times \\ & & & & & & & \end{pmatrix}$$

$\underbrace{\qquad\qquad}_{degree\ d}$ $\underbrace{\qquad\qquad\qquad}_{degree\ <d}$

In this case, these rows represent new polynomials of degree less than d (the degree before *Echelonize*).

The mutant strategy [7, 8] is to distinguish lower degree polynomials from the rest of polynomials and give them a predominant role in the process of solving the system. It calls those polynomials mutants. Similar strategies used to improve Buchberger's algorithm [4] was explained in [13]. The precise mathematical definition of mutants is as follows.

Definition 1. *Let I be the ideal generated by the finite set of polynomials P. An element f in I can be written as*

$$f = \sum_{p \in P} f_p p \tag{8}$$

where $f_p \in K[\underline{x}]$. *The maximum degree of* $f_p p$, $p \in P$, *is the level of this representation. The level of* f *is the minimum level of all of its representations. The polynomial* f *is called mutant with respect to* P *if* $\deg(f)$ *is less than its level.*

Another important definition for mutants that explains how mutants can be distinguished during the process of any matrix-based algorithm from the rest of polynomials.

Definition 2. *Without loss of generality, let* P *be a finite set of degree two polynomials. For* $d \geq 2$ *consider the set*

$$H_d := \{t \cdot p \mid t \text{ is a term}, p \in P, \deg(tp) \leq d\} \tag{9}$$

a polynomial f *is called mutant, if* $\mathrm{LT}(f) \in \mathrm{LT}(\mathrm{span}(H_d)) \setminus \mathrm{LT}((\mathrm{span}(H_{d-1}))$ *and its degree is less than* d.

As we explained in the previous section, the complexity of computing a Gröbner basis of the ideal $\langle P \rangle$ by the XL algorithm is based on the maximal degree (d_{xl}) appeared during the computations. We have shown that $d_{XL} \geq d_{reg}$. However, Bardet, Faugère, et al. in [1] explained that the maximal degree in the Gröbner basis computation is d_{reg}. The disadvantage of XL is that, it does not use mutants. Let $f \in \mathrm{span}(H_d)$ such that $\mathrm{LT}(f)$ is an element in

$$\mathrm{LT}(\mathrm{span}(H_d)) \setminus \mathrm{LT}((\mathrm{span}(H_{d-1}))$$

and its degree is less than d. Then f is mutant. If x is a variable, then the polynomial xf belongs to $\mathrm{span}(H_{d+1})$, however, since its degree is less than $d+1$ then we can reduce it using only elements from H_d. So if we have such mutants, we can enlarge them one degree more, include them to H_d and echlonize H_d again to reduce them instead of generating the hole H_{d+1} set. The MutantXL algorithm is described in Algorithm 3 (MutantXL).

In step 7, polynomials of newly generated leading term that has a degree less than the maximal degree d is mutants. Steps 12-15, MutantXL extends mutants (if any) instead of enlarging the system to the next maximal degree $d + 1$.

We use a HFE system of size $n = 25$ with an univariate degree equals to 96 as an example from our experiments. We solve this system using XL and MutantXL algorithms. Table 1 reports the number of lineally independent polynomials generated by XL and MutantXL at each degree level. For this example, MutantXL can solve the system with a maximal degree equals to 4 while XL goes to degree 6.

The correctness of MutantXL that is defined in Algorithm 3 is guaranteed by the following proposition, first stated and proved in [14].

Proposition 2. *Let* G *be the generated set of polynomials by MutantXL that has a maximal degree* d_G. *the set* G *is a Gröbner basis of the ideal* $\langle G \rangle$, *if* $\exists d < d_G$ *such that*

$$\forall t \in \bigcup_{k=d}^{d_G} T_k, \exists g \in G : \mathrm{LT}(g) = t.$$

Algorithm 3 MutantXL(P)

Require: P is a quadratic finite polynomials of R
1: $d \to 2$
2: $H_d \to P$
3: $L \to \emptyset$
4: $M \to \emptyset$
5: **loop**
6: $G \to Echelonize(H_d)$
7: $M \to \{p \in G \mid \deg(p) < d \text{ and } \mathrm{LT}(p) \notin L\}$
8: $L \to \{\mathrm{LT}(\widetilde{H}) \deg(p) < d\}$
9: **if** Proposition 1 is satisfied on d **then**
10: **return** G
11: **else**
12: **if** $M \neq \emptyset$ **then**
13: $H_d \to G \cup \{tp \mid \deg(t) = 1 \text{ and } p \in M\}$
14: $M \to \emptyset$
15: **end if**
16: **else**
17: $d \to d + 1$
18: $H \to \widetilde{H} \cup \{t \cdot p \mid t \text{ is a term}, p \in P, \deg(tp) = d\})$
19: **end if**
20: **end loop**

	XL						MutantXL			
step	d_1	d_2	d_3	d_4	d_5	d_6	d_1	d_2	d_3	d_4
1	0	25	0	0	0	0	0	25	0	0
2	0	25	625	0	0	0	0	25	625	0
3	0	25	675	7125	0	0	0	25	675	7125
4	0	224	1474	9025	46450	0	0	25	875	8175
5	25	300	2300	12650	53130	177100	20	274	2275	11150
6							25	300	2300	12650

Table 1. Performance of MutantXL versus XL on HFE-96 system of size $n = 25$

4 Complexity Boundaries of MutantXL

We give complexity boundaries for computing Gröbner bases of zero-dimensional ideals using MutantXL. Since MutantXL is an improvement to XL, we use formula 3 to estimate the maximal number of linearly independent polynomials produced by MutantXL without using any mutants.

Let us start with understanding the results that we have obtained in Table 1. The XL algorithm constructs the set of polynomials H_6 to solve the system. In other words, XL cannot find the 25 different leading terms of degree one in $\mathrm{LT}(\mathrm{span}(H_5))$. While, MutantXL finds these 25 leading terms when it enlarges the system only up to degree 4 and using mutants. However, it investigates a subset of $\mathrm{span}(H_6) \backslash \mathrm{span}(H_5)$. Since at step 3, MutantXL finds 50 mutants of de-

gree 3 with leading terms belong to the difference $\mathrm{LT}(\mathrm{span}(H_4)) \setminus \mathrm{LT}(\mathrm{span}(H_3))$. MutantXL enlarges these mutants by multiplying them with terms of degree one and reduces the resulting polynomials using the elements of H_4. It finds new 200 mutants of degree 3. The leading terms of them are elements in $\mathrm{LT}(\mathrm{span}(H_5)) \setminus \mathrm{LT}(\mathrm{span}(H_4))$. Enlarging those mutants means that MutantXL investigates some elements of $\mathrm{span}(H_6) \setminus \mathrm{span}(H_5)$ and so on. This leads us to the following proposition.

Proposition 3. *Let P be a set of finite elements in R and H_d be defined as in (9), then MutantXL must investigate* $\mathrm{span}(H_{d_{XL}}) \setminus \mathrm{span}(H_{d_{XL}-1})$*, where d_{XL} is the highest degree of the system generated by XL.*

Proof. In case of MutantXL does not find mutants, it solves as well as XL and generates the set $H_{d_{XL}}$.

In case of MutantXL gets some mutants at degree $d < d_{XL}$ and computes a Gröbner basis at this degree. Without loss of generality, we assume that P has only one solution and $d_{XL} \geq d + 2$.

Let f be a mutant of degree less than d. By the definition of mutants, $LT(f) \in \mathrm{LT}(\mathrm{span}(H_d)) \setminus \mathrm{LT}(\mathrm{span}(H_{d-1}))$ and $(tf) \in (\mathrm{span}(H_{d+1}) \setminus \mathrm{span}(H_d))$ where $\deg(t) = 1$.

Assuming that $MutantXL$ terminates after multiplying those mutants, it can compute the n linearly independent degree one equations by constructing only a subset of terms in $\mathrm{LT}(\mathrm{span}(H_{d+1}))$ as leading terms. Since the set of leading terms generated by XL at degree $d + 1$ is $\mathrm{LT}(\mathrm{span}(H_{d+1}))$, then it can also find those linearly independent n equations and solves at this degree. This is a contradiction. Then $MutantXL$ cannot solve the system by using only those mutants. So in terms of solving the system at this degree, MutantXL generates many mutants, which lead to investigate some elements from $\mathrm{span}(H_{d+2}) \setminus \mathrm{span}(H_{d+1})$. and so on.

Remark 3. Let P be a set of finite elements in R and H_d^+ be defined as in (2). The necessary condition for computing a Gröbner basis of $\langle P \rangle$ is to investigate some elements of the span of $(H^+_{d_{XL}})$, where d_{XL} is the highest degree of the system generated by XL.

The following lemma come as a result from Proposition 1.

Since MutantXL, as described in Algorithm 3, enlarges the system using XL's method up to d_m and extends all low degree relations $\leq d_m$ (mutants) obtained during its computations, then based on Proposition1 any other matrix-based algorithm should at least a subset of the constructing leading terms of MutantXL.

Remark 4. Based on the assumption that the degree of regularity is the maximal degree appeared during Gröbner basis Computations [3, 1], MutantXL can terminate with maximal degree d_reg.

Figure 1 compares the maximal degree in the Gröbner basis computation by XL (d_{xl}) and by MutantXL (d_m) of a random system. In this case, the maximal

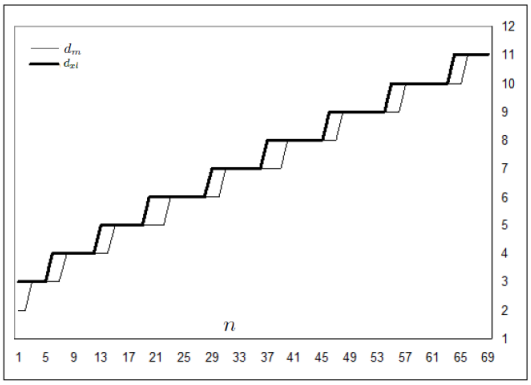


Fig. 1. Experimental results compared the XL maximal degree and the degree of regularity for general random systems with n variables and n different equations.

degree of MutantXL d_m is equal to the degree of regularity. We used Formula (3) to estimate the number of linearly independent polynomials computed by XL at a certain degree d. The maximal degree in the XL computations is such a degree that satisfies (11). The degree of regularity has been computed as defined in (7). Figure 1 confirms Remark 2.

Now, we are going to give an upper bound to the number of linearly independent polynomials produced by mutants.

As we have explained above, MutantXL proceeds like XL up to the degree of finding mutants. Without loss of generality, we let MutantXL start to generate mutants at the same degree it terminates. In this context, we assume that MutantXL terminates at a degree d_m. We can use the formula (3) to compute the number of linearly independent polynomials generated by MutantXL at degree d_m, $I_{d_m} = |\operatorname{LT}(\operatorname{span}(H_{d_m}))|$ before using mutants.

Let I be the number of linearly independent polynomials generated by mutants. From Proposition 3, MutantXL must investigate some elements from $\bigcup_{i=d_m+1}^{d_{xl}} H_i$, where d_{xl} is the termination degree of XL. Since MutantXL terminates at the highest degree d_m, then all elements generated by it from $\bigcup_{i=d_m+1}^{d_{xl}} H_i$ should be of degree $\leq d_m$. So, the number of linearly independent polynomials generated by mutants is upper bounded by the number of those elements.

Let I_d^m is the maximum number of linearly independent polynomials investigated by MutantXL from $(\operatorname{span}(H_d) \setminus \operatorname{span}(H_{d-1})$ for $d_m < d \leq d_{xl}$. Then

$$I_{d_m+1}^m = I_{d_m+1} - I_{d_m} - |\, T_{d_m+1} \,|$$

and

$$I_{d_m+2}^m = I_{d_m+2} - I_{d_m+1} - |\, T_{d_m+2} \,| \,.$$

This leads to

$$I_{d_m+2}^m + I_{d_m+1}^m = I_{d_m+2} - I_{d_m} - (|\, T_{d_m+2} \,| + |\, T_{d_m+1} \,|)$$

and so on up to $d = d_{xl}$. Then the total number of linearly independent polynomials generated by mutants I is bounded as in the following formula

$$I \leq (I_{d_{xl}} - I_{d_m}) - \sum_{i=d_m+1}^{d_{xl}} \mid T_i \mid \tag{10}$$

Another trivial upper bound can be obtained as follows. The maximum number of linearly independent polynomials that MutantXL needs to terminate at a degree (d_m) is to compute all possible linear independents of degree less or equal than d_m. Since MutantXL has already I_{d_m} from them, then the upper bound of linear independents produced by MutantXL is

$$I \leq (\sum_{i=1}^{d_m} \mid T_i \mid) - I_{d_m} \tag{11}$$

Since we multiply mutants by all terms of degree 1 that are equal to n, then the necessary number of mutants needed for computing Gröbner basis is

$$N_m = \lceil I/n \rceil, \tag{12}$$

As an example, let the HFE-96 system of size $n = 29$, XL solves this system at degree $d_{xl} = 6$ and MutantXL solves it at degree $d_m = 4$. The exact linearly independent polynomials generated by mutants is $I = 15628$. By computing the upper bound, we have obtained $I < 16240$ by (10) and $I < 15631$ by (11). Formula (11) is very close to the exact number of linearly independent polynomials generated by mutants for MutantXL described in Algorithm 3.

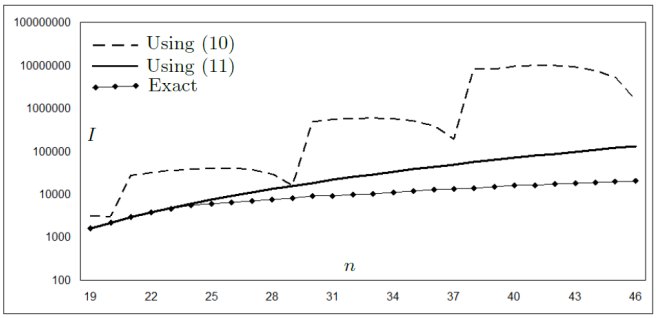


Fig. 2. Compare upper bounds of I using (10) and (11) to the exact value of I for HFE-96 systems solved by MutantXL at degree 4.

Another efficient version of MutantXL that extends only the lowest degree mutants instead of extends all mutants [8]. In this case, the number of linearly independent polynomials produced by mutants is based on the degrees of mutants.

Figure 2 compares the maximum number of linearly independent polynomials generated by mutants in the Gröbner basis computation by MutantXL. We used Formula (11) and (10) to estimate the number of linearly independent polynomials generated by mutants and the exact value when we use only the lowest degree mutants. For these experiments, we use HFE systems with univariate degree 96. These systems are solved at degree 4 as explained by Faugère and Joux in [12]. We used Formula (3) to estimate the number of linearly independent polynomials computed by XL at a certain degree d. XL solves these systems as it solves random, while MutantXL uses mutants appeared at degree 4 and terminates the computations at this degree.

E. Thomae and C. Wolf published a new paper to the ePrint archive [18]. In this paper. They presented two upper bound formulas for I. The first one is trivially an upper bound to I, since they assume that all lower degree polynomials are mutants. This leads to a very large number.

The second formula is also incorrect in many cases. In this formula, they have computed $I^m_{d_m+2}$ as follows

$$I^m_{d_m+2} = I_{d_m+2} - (I_{d_m} + \mid T_{d_m+1} \mid) - \mid T_{d_m+2} \mid .$$

So, they assume that

$$I_{d_m+1} = I_{d_m} + \mid T_{d_m+1} \mid .$$

This means that XL generated all linearly independent polynomials of degree $d_m + 1$, which is not valid in all cases. The accurate relation should be

$$I_{d_m+1} \leq I_{d_m} + \mid T_{d_m+1} \mid .$$

Back to our example, if we compute an upper bound to I using their formula

$$I \leq (\sum_{k=d_m+1}^{d_{xl}} (I_k - I_{d_m} - \sum_{i=d_m+1}^{d_k} \mid T_i \mid),$$

we have $I < -9164$.

5 The MGB algorithm

In this section, we study the complexity of the MGB algorithm. We consider the ring

$$R = \mathbb{F}_2[x_1, \ldots, x_n]/\langle x_1^2 - x_1, ..., x_n^2 - x_n \rangle$$

of polynomials in X with the terms of R ordered by the graded lexicographical order $<_{glex}$. The MGB algorithm is a variant of MutantXL that based on the partial enlargement concept [15, 15]. The idea of this concept is as follows.

Let the set H_d^+ of degree d polynomials in R be defined as in (2). When we enlarge the system to a highest degree d, MutantXL as well as XL constructs the elements of H_d^+. Instead of constructing all elements of H_d^+, MGB generates only a subset of it. By this method, MGB can explore the elements of H_d^+ one

step at a time without being forced to store the whole set at once. MGB used the so-call leading variable to partition the constructing polynomials. The leading variable of $p \in R$ is defined according to the order defined on the set variables X as

$$\mathrm{LV}(p) := \max\{x \mid \ x \in \mathrm{LT}(p)\}.$$

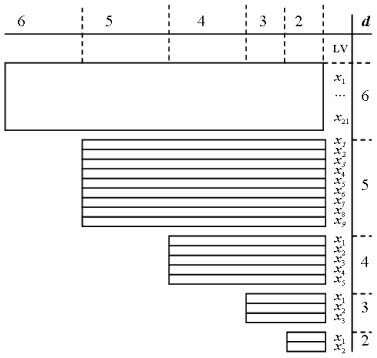


Fig. 3. Variable partitions of enlargement for random system of size $n = m = 26$. Horizontal stripes represent variable-partitions.

Figure 3 shows the structure of the generated system by XL for a random system of size $n = 26$. For each degree d, we assume that H_d^+ is in row echelon form and divided into subsets based on the leading variable of the polynomials. Since the polynomials of H_d^+ is ordered using the graded lexicographical order, then H_d^+ are partitioned from up to down by $x_1, x_2, \ldots, x_n$ partitions. Only some of these partitions are not empty.

MGB works on the assumption that the partitions of H_d^+ at certain degree d are filled from up to down. More precisely, The partition of x_1 is filled first, then the partition of x_2, and so on. MGB enlarges one non-empty variable partition at a time, i.e, for $x_n, \ldots, x_1$ (in that order), up to the first *full* partition. The variable partition of leading variable x and degree d is called *full* if all terms of degree d and leading variable x are represented as leading terms in that partition.

Let $\mathrm{LVT}(d, k) \subset R$ be the set of terms of degree d that have been leading variables in $\{x_1, \ldots, x_k\}$. The number of the elements of $\mathrm{LVT}(d, k)$ can be computed as

$$|\,\mathrm{LVT}(d, k)| = \sum_{i=1}^{k} \binom{n-i}{d-1}, k \leq n \tag{13}$$

Let k satisfy the following

$$|\,\mathrm{LVT}(d, k-1)| < I_d < |\,\mathrm{LVT}(d, k)|. \tag{14}$$

From many experiments with XL over Semi-Regular systems (random) we have noticed that the number of linearly independent polynomials of degree d is sufficient to fill k partitions with leading variables $(x_1, \ldots, x_k)$. In the Semi-Regular case, at least $k-1$ of them are *full*.

Table 2 shows the number of linearly independent polynomials (Num_polys) and the partitions generated by XL at degree 5 for random systems of size $24 \ldots 34$. All partitions are *full* except the last one for each system.

n	Num_polys	partitions	*full*
24	41376	$x_1, \ldots, x_{12}$	11
25	49375	$x_1, \ldots, x_{10}$	9
26	58474	$x_1, \ldots, x_9$	8
27	68769	$x_1, \ldots, x_8$	7
28	80360	$x_1, \ldots, x_8$	7
29	93351	$x_1, \ldots, x_7$	6
30	107850	$x_1, \ldots, x_7$	6
31	123969	$x_1, \ldots, x_7$	6
32	141824	$x_1, \ldots, x_7$	6
33	161535	$x_1, \ldots, x_7$	6
34	183226	$x_1, \ldots, x_7$	6

Table 2. Partitions generated by XL at degree 5 for Random systems

Now we are going to study the complexity of the MGB algorithm in case of solving random systems. We need the following definition

Definition 3. *Let H be a set of polynomials such that all elements in H have degree d. Let*

$$H_x = \{h \in H \mid \mathrm{LV}(h) = x\}$$

and F_x be the set produced by enlarging H_x one degree more as follows

$$F_x = \{y \cdot p : p \in H_x, y \leq x\}.$$

*We called H_x a leading variable reduced partition (*LVR-partition*), if there is $q \in \widetilde{F_x}$ such that $LV(q) < x$, where $\widetilde{F_x}$ is a row echelon form of F_x.*

Let the set H_d defined as in 9 be the set of polynomials generated by XL at degree d. Let x_j be the smallest leading variable with non empty partition $(\widetilde{H}d-1)_{x_j}$ that is *full*. Experimentally, we found that only the partitions that have leading variable $\leq x_j$ are $LVR-partitions$. The MGB algorithm enlarges only that $LVR-partitions$ of degree $d-1$ to generate the degree d polynomials of the extended system, while XL enlarges all partitions. We have noticed that, without using any mutants MGB generates all partitions of degree d polynomials with leading variable $\leq x_j$ as well as XL generates. This leads to solving the system by MGB at the same degree as XL with fewer equations and fewer terms.

Table 3 compares the partitions generated by XL and MGB. For example, in the case of $n = 24$, at degree 4 XL has $x_1, \ldots, x_5$ partitions. It enlarges all of them up to degree 5. We obtain 12 partitions that are filled with 41376 polynomials. MGB has $x_2, \ldots, x_5$ partitions at degree 4 only x_4, x_5 are $LVR - partitions$. So MGB enlarges only these two partitions. It generates 9 partitions at degree 5, which means that MGB skip x_1, x_2, x_3 from the degree 5. MGB fills the 9 partitions by 19221 polynomials. By adding the number of skipped polynomials to them, we have exactly the same as XL generated at degree 5 (41376). Which explains that MGB generate all new partitions as well as XL.

	XL			MGB		
n	Num_polys	partitions	*full*	Num_polys	partitions	*full*
24	41376	$x_1, \ldots, x_{12}$	11	19221	$x_4, \ldots, x_{12}$	8
25	49375	$x_1, \ldots, x_{10}$	9	22579	$x_4, \ldots, x_{10}$	6
26	58474	$x_1, \ldots, x_9$	8	26343	$x_4, \ldots, x_9$	5
27	68769	$x_1, \ldots, x_8$	7	30543	$x_4, \ldots, x_8$	4
28	80360	$x_1, \ldots, x_8$	7	35210	$x_4, \ldots, x_8$	4
29	93351	$x_1, \ldots, x_7$	6	40376	$x_4, \ldots, x_7$	3
30	107850	$x_1, \ldots, x_7$	6	46074	$x_4, \ldots, x_7$	3
31	123969	$x_1, \ldots, x_7$	6	52338	$x_4, \ldots, x_7$	3
32	141824	$x_1, \ldots, x_7$	6	82954	$x_3, \ldots, x_7$	4
33	161535	$x_1, \ldots, x_7$	6	94110	$x_3, \ldots, x_7$	4
34	183226	$x_1, \ldots, x_7$	6	106346	$x_3, \ldots, x_7$	4

Table 3. Compare Partitions generated by XL and MGB at degree 5 for Random systems

Let us use 3 to compute the number of linearly independent polynomials by XL at certain degree d. By filling the partitions $x_1, x_2, \ldots, x_n$ (in this order). In case of the Semi-Regular systems, we fill all possible monomials that has leading variable x_1 first, followed by the x_2 partition and so on. In this way, we can compute the number of partitions generated by XL at this degree. Also, in this way we can determine the number of $full$ partitions and the $LVR - partitions$. Then we can estimate the avoided partitions at each degree level.

Figure 4 explains the number of avoided variable partitions during the enlargement process (the number of unexpanded partitions). MGB enlarges different systems with size $n \in (5, \ldots, 65)$. For example, in case of $n = 50$, MGB avoids $(1, 2, 4, 6, 9, 17)$ variable partitions from the enlarged system at degree 9. This affects the size of the matrix generated by MGB, which is much smaller than the XL algorithm, even if we could represent only the linearly independent rows.

Based on the results in [2], the degree of regularity is the maximal degree appeared in the computation of Gröbner bases. In this case, the size of the system can be upper bounded by the dimensional matrix that has a number of columns

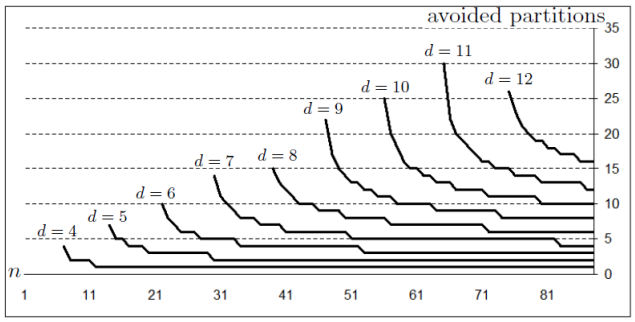


Fig. 4. MGB different degree level number of variables.

is equal to the number of terms up to the degree of regularity. The number of terms in this case is computed as follows.

$$\sum_{d=0}^{d_{reg}} \binom{n+d-1}{d}$$

The MGB algorithm puts in question the sharpness of this bound. As we show in Figure 4, MGB avoids the full partitions from the enlarging step. For example, in case of $n = 50$, MGB avoids $(1, 2, 4, 6, 9, 17)$ variable partitions from the enlarged system for the degree respectively. The complexity of MGB depends not only on the highest degree d of the system and the number of variables n, but also on the sequence $n_1, n_2, \ldots, n_d$ of variables omitted at each degree d. The number of columns of the largest matrix of MGB is given by

$$\sum_{d=0}^{d} \binom{n-n_d+d-1}{d}$$

Figure 5 shows the number of variables that are used during the process at each degree level d. We estimate the number of omitted partitions as explained above of a random system with size $n = 77$.

Another kind of comparison to explain the improvement, Figure 6 shows the number of variables that are used during the process at the degree of regularity d_{reg}. This analysis can be used to estimate the complexity the MXL_3 algorithm, since we explain only the improvements at the final degree which is typically MXL_3.

To have more explanations about the efficiency of the MGB algorithm, we need to define an algorithm called FXL_5 algorithm. This algorithm is a variant of XL that generates only the linearly independent polynomials during the enlargement step using the advanced criteria of the F_5 algorithm [10]. We assume that the maximal degree by FXL_5 and MGB is the degree of regularity. Figure 7 compares the total number of terms generated in the system up to the degree of regularity between MGB and FXL_5 for random systems of size

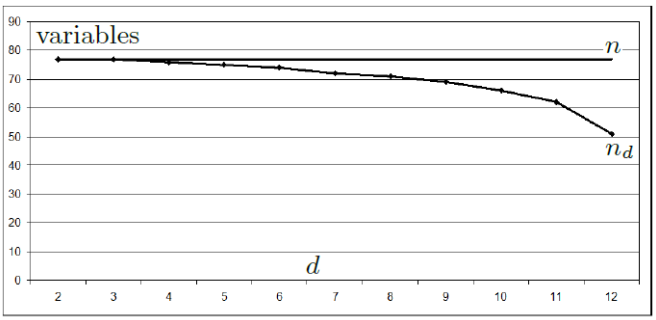


Fig. 5. Compares n to n_d for a random system of size $n = 77$.

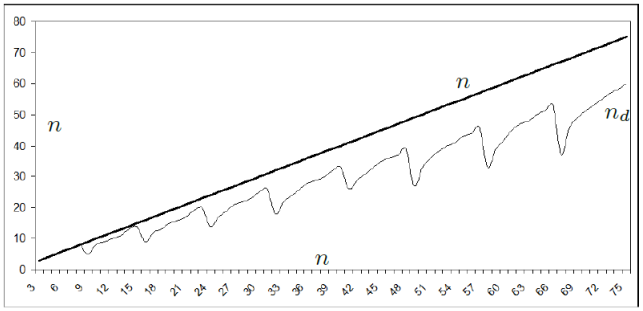


Fig. 6. Compares n to n_d for a random system of sizes $n = 3 \ldots 77$ at the degree of regularity.

$n \in (10, 15, \ldots, 65)$. We compare these results with the total number of terms generated by XL.

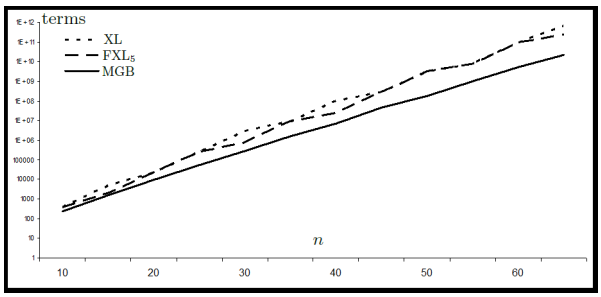


Fig. 7. Experimental results compared with number of terms at the degree of regularity.

6 Conclusion

In this paper, we present new upper bounds of the complexity of computing Gröbner basis and solving polynomial systems. Our new formulas based on the MutantXL algorithm. We compare it with the complexity of the XL algorithm. Furthermore, we identify the error in the formula of Thomae and Wolf for setting an upper bound to the number of linearly independent polynomials generated by mutants. In addition, we experimentally explain the complexity of the MGB algorithm. The complexity analysis of MGB suggests that this new strategy may change substantially our thinking of the hardness of computing Gröbner bases.

Acknowledgments

We acknowledge the useful comments from Martin Albrecht and Stanislav Bulygin on this paper and their valuable suggestions.

References

1. M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *MEGA'05*, 2005. Eighth International Symposium on Effective Methods in Algebraic Geometry, Porto Conte, Alghero, Sardinia (Italy), May 27th – June 1st.
2. M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *MEGA'05*, 2005. Eighth International Symposium on Effective Methods in Algebraic Geometry, Porto Conte, Alghero, Sardinia (Italy), May 27th – June 1st.

3. Bardet, M. and Faugre, J.C and Salvy, B. On the complexity of grbner basis computation of semi-regular overdetermined algebraic equations. In *International Conference on Polynomial System Solving - ICPSS*, pages 71 –75, Nov 2004.
4. B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal)*. PhD thesis, Mathematical Institute, University of Innsbruck, Austria, 1965. (English translation in Journal of Symbolic Computation, 2004).
5. J. Buchmann, D. Cabarcas, J. Ding, and M. S. E. Mohamed. Flexible Partial Enlargement to Accelerate Gröbner Basis Computation over $\mathbb{F}_2$. In *Proceedings of the 3rd Africacrypt Conference, (Africacrypt 2010)*, Lecture Notes in Computer Science, pages 69–81. Springer-Verlag, Berlin, May 2010.
6. N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *Proceedings of International Conference on the Theory and Application of Cryptographic Techniques(EUROCRYPT)*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407, Bruges, Belgium, May 2000. Springer.
7. J. Ding. Mutants and its impact on polynomial solving strategies and algorithms. Privately distributed research note, University of Cincinnati and Technical University of Darmstadt, 2006.
8. J. Ding, J. Buchmann, M. S. E. Mohamed, W. S. A. Moahmed, and R.-P. Weinmann. MutantXL. In *Proceedings of the 1st international conference on Symbolic Computation and Cryptography (SCC08)*, pages 16 – 22, Beijing, China, April 2008. LMIB.
9. J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Pure and Applied Algebra*, 139(1-3):61–88, June 1999.
10. J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation (ISSAC)*, pages 75 – 83, Lille, France, July 2002. ACM.
11. J.-C. Faugère and G. Ars. Comparison of XL and Gröbner basis algorithms over Finite Fields. Research Report RR-5251, Institut National de Recherche en Informatique et en Automatique (INRIA), 2004.
12. J.-C. Faugère and A. Joux. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In *Proceedings of the 23rd Annual International Cryptology Conference Advances in Cryptology - CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60, Santa Barbara, California, USA, August 2003. Springer.
13. A. Giovini, T. Mora, G. Niesi, L. Robbiano, and C. Traverso. One sugar cube, please or selection strategies in the buchberger algorithm. In *Proceedings of the 1991 international symposium on Symbolic and algebraic computation*, ISSAC '91, pages 49–54, New York, NY, USA, 1991. ACM.
14. M. S. E. Mohamed, D. Cabarcas, J. Ding, J. Buchmann, and S. Bulygin. MXL3: An efficient algorithm for computing gröbner bases of zero-dimensional ideals. In *Proceedings of The 12th international Conference on Information Security and Cryptology, (ICISC 2009)*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, December 2009. Accepted for publication.
15. M. S. E. Mohamed, W. S. A. E. Mohamed, J. Ding, and J. Buchmann. MXL2: Solving polynomial equations over GF(2) using an improved mutant strategy. In

Proceedings of The Second international Workshop on Post-Quantum Cryptography, (PQCrypto08), Lecture Notes in Computer Science, pages 203–215, Cincinnati, USA, October 2008. Springer-Verlag, Berlin.
16. S. Rønjom and H. Raddum. On the Number of Linearly Independent Equations Generated by XL. In *Sequences and Their Applications - SETA 2008*, pages 239–251, Lexington, Kentucky, USA, 2008. Springer-Verlag, Berlin.
17. M. Sugita, M. Kawazoe, and H. Imai. Relation between the XL Algorithm and Gröbner Basis Algorithms. *Transactions on Fundamentals of Electronics, Communications and Computer Sciences (IEICE)*, E89-A(1):11–18, January 2006.
18. E. Thomae and C. Wolf. Unravel xl and its variants. Cryptology ePrint Archive, Report 2010/596, 2010. `http://eprint.iacr.org/`.
19. B. yin Yang and J. ming Chen. Theoretical analysis of xl over small fields. In *In Proceedings of the 9th Australasian Conference on Information Security and Privacy*, pages 277–288. Springer, 2004.