On the Relation Between the Mutant Strategy and the Normal Selection Strategy in Gröbner Basis Algorithms

Martin Albrecht[1], Carlos Cid[2], Jean-Charles Faugère[1], and Ludovic Perret[1]

[1] INRIA, Paris-Rocquencourt Center, SALSA Project
UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
CNRS, UMR 7606, LIP6, F-75005, Paris, France
malb@lip6.fr, jean-charles.faugere@inria.fr, ludovic.perret@lip6.fr
[2] Information Security Group, Royal Holloway, University of London, UK
carlos.cid@rhul.ac.uk

Abstract. The computation of Gröbner bases remains one of the most powerful methods for tackling the Polynomial System Solving (PoSSo) problem. The most efficient known algorithms reduce the Gröbner basis computation to Gaussian eliminations on several matrices. However, several degrees of freedom are available to generate these matrices. It is well known that the particular strategies used can drastically affect the efficiency of the computations. In this work we investigate a recently-proposed strategy, the so-called *"Mutant strategy"*, on which a new family of algorithms is based (MXL, MXL_2 and MXL_3). By studying and describing the algorithms based on Gröbner basis concepts, we demonstrate that the Mutant strategy can be understood to be equivalent to the classical Normal Selection strategy currently used in Gröbner basis algorithms. Furthermore, we show that the "partial enlargement" technique can be understood as a strategy for restricting the number of S-polynomials considered in an iteration of the F_4 Gröbner basis algorithm, while the new termination criterion used in MXL_3 does not lead to termination at a lower degree than the classical Gebauer-Möller installation of Buchberger's criteria. We claim that our results map all novel concepts from the MXL family of algorithms to their well-known Gröbner basis equivalents. Using previous results that had shown the relation between the original XL algorithm and F_4, we conclude that the MXL family of algorithms can be fundamentally reduced to redundant variants of F_4.

1 Introduction

Solving systems of multivariate polynomial equations is a fundamental problem arising on a variety of scientific fields, such as cryptography, robotics, biology, error correcting codes, signal theory, among others. One of the most popular and powerful methods for solving systems of equations is the computation of Gröbner bases. A Gröbner basis is a particular set of generators of an ideal in the polynomial ring. Such basis can be used to find the solutions of the system of equations associated with the ideal (i.e. the corresponding *variety*). The concept of Gröbner basis and the first algorithm for their computation were introduced in Buchberger's seminal work [8], giving rise in the past decades to an extraordinarily active area of research in computer algebra.

Although Buchberger's original proposal provided a powerful algorithmic tool for solving systems of equations, it was not particularly suitable for solving large systems arising in applications. Much work in subsequent decades concentrated on the investigation of algorithmic and implementation strategies to speed up the computation of Gröbner bases. The most efficient algorithms currently known, namely the F_4 and F_5 algorithms [20, 21], reduce the Gröbner basis computation to a sequence of Gaussian eliminations on several particular matrices. These methods were motivated by the seminal work of Lazard [27]. He first made the link between the computation of a Gröbner

basis of an ideal and linear algebra operations on the corresponding *Macaulay matrix*. We note however that in all linear algebra-based Gröbner basis methods, several degrees of freedom are available to the designer and implementer of the algorithm to generate these matrices, and it is well known that the particular strategies selected can drastically affect the efficiency and running times of the computations.

The past few years have witnessed a growing interest from the cryptographic community in computational algebra methods, in particular Gröbner basis algorithms. This was motivated by the proposal of algebraic attacks against stream ciphers [12] and block ciphers [16, 26, 1, 2], as well as by the proposal of several public-key schemes based on systems of multivariate polynomial equations (e.g. [33]), and the corresponding cryptanalysis using the F_5 algorithm [24, 34, 25, 22, 6]. One particular algorithm has received considerable attention from the cryptographic community: the XL algorithm [14] (and its several variants, e.g. [15, 16, 13]) was originally proposed to tackle problems arising specifically from cryptology. Although not strictly a Gröbner basis algorithm, it used a similar idea to the one proposed by Lazard: it constructs the Macaulay matrix up to some large degree D and reduces it to obtain the solution of the system. The algorithm was shown to work only under particular conditions [18], while other flaws were also shown in other high-profile variants [11, 28]. Eventually, and perhaps unsurprisingly, it was shown that the XL algorithm could be described essentially as a redundant (and less efficient) variant of the F_4 algorithm [3].

Yet, despite of these well-known results, the XL algorithm continues somehow to attract the attention of researchers working in cryptography [10, 35]. Perhaps because of its simplicity, it remains an attractive method for one to propose improvements and/or implementation tricks and strategies. However, many of (if not all) these proposals can be eventually described based on well-known Gröbner bases concepts and implementation techniques, such as S-polynomials and their selection strategies. These are in turn unsurprisingly often already present in many efficient linear algebra-based Gröbner basis algorithms and implementations.

In this paper we investigate a prominent recent addition to the XL family, namely the MutantXL algorithms [10, 31, 30, 9]. The concept of Mutants was first introduced in [10], giving rise to a family of algorithms and techniques [31, 30, 9], which showed to be particularly efficient against the MQQ multivariate cryptosystem [32]. Unlike the XL algorithm, some of the Mutant algorithms (e.g. MXL_3 [30]) do in fact explicitly compute the Gröbner basis of the corresponding ideal, assuming it is zero-dimensional. Because of the remarkable experimental results reported in [30], a natural question arises: how do we describe mutants in terms of commutative algebra? Are mutants a new concept, or can it be described based on a well-known computational algebra concept? Likewise, are the new *mutant strategies* general enough, so that they can potentially be incorporated to existent Gröbner basis algorithms? To the authors' best knowledge, there has been so far no in-depth study of the mathematical properties of mutants and related strategies, and how they are connected to other Gröbner basis algorithms.

In this work, we undertake this task. In particular, we compare the MXL family with two variants of the F_4 algorithm: first, the so-called simplified F_4 which does not use Buchberger's criteria to avoid useless reductions to zero and second, the full F_4 as specified in [20]. Considering these algorithms, we show that the Mutant strategy can be understood as essentially equivalent to the Normal Selection strategy as used in Gröbner basis algorithms, such as F_4. Based on previous results, which showed the relation between the XL algorithm and F_4 [3], we conclude that MXL can too be described as a redundant variant of F_4. Furthermore, we also study the "partial enlargement" strategy proposed in [31] and demonstrate that it corresponds to selecting a subset of S-polynomials in Gröbner basis algorithms. As a result, we conclude that MXL_2 can also be described as a variant of F_4, although a variant that diverges from known approaches about how to select the number of S-polynomials in each iteration. Finally, we consider the new termination criterion proposed in [30] and demonstrate that it does not lead to a lower degree of termination than using Buchberger's

criteria to remove useless pairs in a Gröbner basis algorithm. As a result, we reach the conclusion that MXL_3 can be reduced to a redundant variant of the full F_4 algorithm.

The remaining of this work is organised as follows. In Section 2 we recall the well-known XL algorithm, and re-state the result showing the relation between Xl and F_4. In Section 3 we review well-known statements from commutative algebra. We place particular emphasis on the concept of S-polynomials and the central role they play in Gröbner bases computations. In particular, we review the fact that in XL-style algorithms any multiplication of polynomials by monomials except for those giving rise to S-polynomials is in essence redundant. In Section 4 we review the definition of Mutants, and present our pseudocode for the MXL_3 algorithm. In Section 5 we state and prove our main result, namely that the Mutant strategy is a redundant variant of the Normal Selection strategy. We also treat partial enlargement and the termination condition of MXL_3 in Section 5. We conclude in Section 6, where we include a brief discussion on what we view as the limitations of using running times as the *sole* basis for *comparison* between Gröbner basis *algorithms*.

2 The XL Algorithm

In this section we briefly recall the well-known XL algorithm. An iterative variant of the algorithm is given in Algorithm 1. We adopt the notation from [30] and, given a set of polynomials S, we denote by $S_{(op)d}$ the subset of S with elements of degree $(op)d$ where $(op) \in \{=, <, \leq, >, \geq\}$.

Input: F – a polynomial system of equations
Input: D – an integer > 0
Result: a D-Gröbner basis for F

```
1  begin
2      G ⟵ ∅;
3      for 1 ≤ d ≤ D do
4          F_{=d} ⟵ ∅;
5          for f ∈ F do
6              if deg(f) = d then
7                  add f to F_d;
8              else if deg(f) < d then
9                  M_{=d−deg(f)} ⟵ all monomials of degree d − deg(f);
10                 for m ∈ M_{=d−deg(f)} do
11                     add m · f to F_{=d};
12         G ⟵ G ∪ the row echelon (or the matrix) form of F_{=d};
13     return G
14 end
```

Algorithm 1: XL

It was shown in [3] that the XL algorithm can be emulated using the F_4 algorithm. In particular, [3] proves that:

Lemma 1. *Algorithm 1 can be simulated using F_4 (described in Algorithm 3) by adding redundant pairs.*

3 Gröbner Bases Basics

In this section we recall some basic results about Gröbner bases. For a more detailed treatment, we refer to, for instance, [17]. Consider a polynomial ring $R = \mathbb{F}[x_0, \ldots, x_{n-1}]$ over some field $\mathbb{F}$. We adopt some admissible ordering on monomials in R. We can then denote by $\mathrm{LM}(f)$ the largest or leading monomial appearing in $f \in R$ and by $\mathrm{LC}(f) \in \mathbb{F}$ the coefficient corresponding to $\mathrm{LM}(f)$ in f. By $\mathrm{LT}(f)$ we denote $\mathrm{LC}(f) \cdot \mathrm{LM}(f)$. In this work $\mathrm{LV}(f)$ denotes the largest variable – ordered w.r.t. the monomial ordering – in the leading monomial $\mathrm{LM}(f)$ of f, and given a set $F \subset R$, we define $\mathrm{LV}(F, x)$ as $\{f \in F \mid \mathrm{LV}(f) = x\}$. The set of leading monomials of F is defined as $\mathrm{LM}(F) = \{\mathrm{LM}(f) \mid f \in F\}$, M denotes the set of all monomials in R, while $M(F)$ is the set of all monomials appearing in any polynomial in F.

The ideal $\mathcal{I}$ generated by $f_0, \ldots, f_{m-1} \in R$, denoted $\langle f_0, \ldots, f_{m-1} \rangle$, is defined as

$$\left\{ \sum_{i=0}^{m-1} h_i f_i \mid h_0, \ldots, h_{m-1} \in R \right\}.$$

It is known that every ideal $\mathcal{I} \subseteq R$ is finitely generated.

A Gröbner basis of an ideal $\mathcal{I}$ is a particular set of generators.

Definition 1 (Gröbner Basis). *Let $\mathcal{I}$ be an ideal of $\mathbb{F}[x_0, \ldots, x_{n-1}]$ and fix a monomial ordering. A finite subset*

$$G = \{g_0, \ldots, g_{m-1}\} \subset \mathcal{I}$$

is said to be a Gröbner basis *of $\mathcal{I}$ if for any $f \in \mathcal{I}$ there exists $g_i \in G$ such that $\mathrm{LM}(g_i) \mid \mathrm{LM}(f)$.*

We note that if a system of polynomials $f_0, \ldots, f_{m-1}$ has a unique root, i.e. the system of equations $f_0 = 0, \ldots, f_{m-1} = 0$ has a unique solution, then computation of the Gröbner basis of the corresponding ideal allows one to solve the system (i.e. the solution can be "read" directly on the Gröbner basis). More generally, if the ideal is zero-dimensional, the solutions of a system can be computed from a Gröbner basis in polynomial-time (in the number of solutions) [23].

Since the notion of Gröbner bases is defined by the existence of *relatively* small leading terms, the task of computing a Gröbner basis is essentially to find new elements in the ideal with smaller leading terms until no more such elements can be found. Buchberger proved in his PhD thesis [8] that Gröbner bases can be computed by considering only S-polynomials. Such polynomials are designed to cancel leading terms and thus potentially produce new elements in the ideal with lower leading terms.

Definition 2 (S-Polynomial). *Let $f, g \in \mathbb{F}[x_0, \ldots, x_{n-1}]$ be non-zero polynomials.*

- *Let $\mathrm{LM}(f) = \prod_{i=0}^{n-1} x_i^{\alpha_i}$ and $\mathrm{LM}(g) = \prod_{i=0}^{n-1} x_i^{\beta_i}$, with $\alpha_i, \beta_i \in \mathbb{N}$, denote the leading monomials of f and g respectively. Set $\gamma_i = \max(\alpha_i, \beta_i)$ for every $0 \leq i < n$, and denote by $x^\gamma = \prod_{i=0}^{n-1} x_i^{\gamma_i}$. It holds that x^γ is the least common multiple of $\mathrm{LM}(f)$ and $\mathrm{LM}(g)$, written as*

$$x^\gamma = \mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g)).$$

- *The S-polynomial of f and g is defined as*

$$S(f, g) = \frac{x^\gamma}{\mathrm{LT}(f)} \cdot f - \frac{x^\gamma}{\mathrm{LT}(g)} \cdot g.$$

Now let $G = \{g_0, \ldots, g_{s-1}\} \subset R$, and $\mathcal{I}$ be the ideal generated by G. We say that a polynomial $f \in \mathcal{I}$ has a *standard representation* w.r.t. G if there exist constants $a_0, \ldots, a_{s-1} \in \mathbb{F}$ and monomials $m_0, \ldots, m_{s-1} \in M$ such that

$$f = \sum_{k=0}^{s-1} a_k t_k g_k,$$

with $\mathrm{LM}(t_k g_k) \leq \mathrm{LM}(f)$. Buchberger's main result stated that G is a Gröbner basis for $\mathcal{I}$ if and only if every S-polynomial $S(g_i, g_j)$ has a *standard representation* w.r.t. G.

Furthermore, Buchberger showed that in the computation of Gröbner bases it is *sufficient* to consider S-polynomials only, since *any* reduction of leading terms can be attributed to S-polynomials. There are many variants of this result in textbooks on commutative algebra; we give below the statement and proof based on [17] since the presentation helps to understand the close connection between XL and Gröbner basis algorithms. The proof is included for the sake of completeness.

Lemma 2. *Let $f_0, \ldots, f_{t-1}$ be nonzero polynomials in R. Given a monomial x^δ, let $x^{\alpha(0)}$, ..., $x^{\alpha(t-1)}$ be monomials in R such that $x^{\alpha(i)}\,\mathrm{LM}(f_i) = x^\delta$ for all $i = 0, \ldots, t-1$. We consider the sum $f = \Sigma_{i=0}^{t-1} c_i x^{\alpha(i)} f_i$, where $c_0, \ldots, c_{t-1} \in \mathbb{F}\backslash\{0\}$. If $\mathrm{LM}(f) < x^\delta$, then there exist constants $b_j \in \mathbb{F}$ such that*

$$f = \sum_{i=0}^{t-1} c_i x^{\alpha(i)} f_i \;=\; \sum_{j=0}^{t-2} b_j x^{\delta-\tau_j}\, S(f_j, f_{j+1}), \tag{1}$$

where $x^{\tau_j} = \mathrm{LCM}(\mathrm{LM}(f_j), \mathrm{LM}(f_{j+1}))$. Furthermore

$$x^{\delta-\tau_j} S(f_j, f_{j+1}) < x^\delta, \text{ for all } j = 0, \ldots, t-2.$$

Proof. We denote by $\mathrm{LM}(f_i) = x^{\beta(i)}$. Thus, $\alpha(i) + \beta(i) = \delta$. Then let $d_i = \mathrm{LC}(f_i)$. It follows that $c_i d_i$ is the leading coefficient of $c_i x^{\alpha(i)} f_i$. Furthermore, let $p_i = \frac{x^{\alpha(i)} f_i}{d_i}$ and thus $\mathrm{LC}(p_i) = 1$. Consider the "telescope sum":

$$\begin{aligned}
f &= \sum_{i=0}^{t-1} c_i x^{\alpha(i)} f_i = \sum_{i=0}^{t-1} c_i d_i \frac{x^{\alpha(i)} f_i}{d_i} = \sum_{i=0}^{t-1} c_i d_i p_i \\
&= \sum_{i=0}^{t-1} \left(\sum_{j=0}^{i} c_j d_j - \sum_{j=0}^{i-1} c_j d_j \right) p_i \\
&= \sum_{i=0}^{t-1} \sum_{j=0}^{i} c_j d_j p_i - \sum_{i=-1}^{t-2} \sum_{j=0}^{i} c_j d_j\, p_{i+1} \\
&= \sum_{j=0}^{t-1} c_j d_j p_{t-1} + \sum_{i=0}^{t-2} \sum_{j=0}^{i} c_j d_j (p_i - p_{i+1}).
\end{aligned}$$

All $c_i x^{\alpha(i)} f_i$ have x^δ as leading monomial. Since their sum has smaller leading monomial, we have that $\Sigma_{i=0}^{t-1} c_i d_i = 0$, leading to:

$$f = \sum_{i=0}^{t-2} \sum_{j=0}^{i} c_j d_j (p_i - p_{i+1}). \tag{2}$$

By assumption $x^{\alpha(i)}\,\mathrm{LM}(f_i) = x^\delta$ for all $i = 0, \ldots, t-1$, and we have:

$$\begin{aligned} x^{\delta-\tau_j} S(f_j, f_{j+1}) &= x^{\delta-\tau_j}\left(\frac{x^{\tau_j}}{\mathrm{LT}(f_j)} f_j - \frac{x^{\tau_j}}{\mathrm{LT}(f_{j+1})} f_{j+1}\right) \\ &= \frac{x^{\alpha(j)}}{d_j} f_j - \frac{x^{\alpha(j+1)}}{d_{j+1}} f_{j+1} \\ &= p_j - p_{j+1}. \end{aligned}$$

This is now plugged into the telescope sum (2) leading to:

$$\begin{aligned} f &= \sum_{i=0}^{t-2} \sum_{j=0}^{i} c_j d_j x^{\delta-\tau_i} S(f_i, f_{i+1}) \\ &= \sum_{i=0}^{t-2} (i+1) c_j d_j x^{\delta-\tau_i} S(f_i, f_{i+1}) \\ &= \sum_{i=0}^{t-2} b_j x^{\delta-\tau_i} S(f_i, f_{i+1}), \end{aligned}$$

with $b_j = (i+1)c_j d_j$. Since the polynomials p_j and p_{j+1} have leading monomial x^δ and leading coefficient 1, the difference $p_j - p_{j+1}$ has a smaller leading monomial. Since we have that $p_j - p_{j+1} = x^{\delta-\tau_j} S(f_j, f_{j+1})$, this claim also holds true for $x^{\delta-\tau_j} S(f_j, f_{j+1})$. Thus the Lemma holds. □

The following corollary is a simple generalisation of Lemma 2 to sums where not all summands have the same leading term.

Corollary 1. *Let $f_0, \ldots, f_{t-1}$ be polynomials in R. Consider the polynomial f as the sum $f = \Sigma_{i=0}^{t-1} c_i x^{\alpha(i)} f_i$, with coefficients $c_0, \ldots, c_{t-1} \in \mathbb{F}\backslash\{0\}$, such that $\mathrm{LM}(f) < x^\delta = \max\{x^{\alpha(i)}\mathrm{LM}(f_i)\}$. Without loss of generality, we can assume that there is a $\tilde{t}$ such that $x^{\alpha(j)}\mathrm{LM}(f_j) = x^\delta$ for $j < \tilde{t}$ and $x^{\alpha(k)}\mathrm{LM}(f_k) < x^\delta$ for $k \geq \tilde{t}$. Then there exist constants $b_j \in \mathbb{F}$ such that*

$$\begin{aligned} f &= \sum_{j=0}^{\tilde{t}-2} b_j x^{\delta-\tau_j} S(f_j, f_{j+1}) + \sum_{k=\tilde{t}}^{t-1} c_k x^{\alpha(k)} f_k \\ &= \sum \tilde{c}_i x^{\tilde{\alpha}(i)} \tilde{f}_i, \end{aligned}$$

where $x^{\tau_j} = \mathrm{LCM}(\mathrm{LM}(f_j), \mathrm{LM}(f_{j+1}))$. Furthermore, for all $0 \leq j \leq \tilde{t} - 2$, we have

$$\mathrm{LM}(x^{\delta-\tau_j} S(f_j, f_{j+1})) < x^\delta$$

and thus

$$x^{\tilde{\alpha}(i)}\mathrm{LM}(\tilde{f}_i) < x^\delta \text{ for all } i.$$

Corollary 1 states that whatever cancellations can be produced by monomial multiplications and $\mathbb{F}$-linear combinations, they can be attributed to S-polynomials. It follows that the only cancellations that need to be considered in a XL-style algorithm are those produced by S-polynomials.

Example 1. Consider the polynomials $f = xy + x + 1$, $g = x + 1$ and $h = z + 1 \in \mathbb{F}_{127}[x, y, z]$. We fix the degree reverse lexicographical term ordering. To compute a Gröbner basis, we start by constructing two S-polynomials of degree two, namely: $f - yg = x - y + 1$ and $zg - yh = -x + z$. In matrix notation, we would thus have to consider the six rows corresponding to f, yg, zg, yh, g and h.

For comparison, XL would consider the following polynomials up to degree two.

$$\begin{aligned} f = xy + x + 1, \quad xg = x^2 + x, \quad yg = xy + y, \\ zg = \quad xz + z, \quad xh = xz + x, \quad yh = yz + y, \\ zh = \quad z^2 + z, \quad g = \quad x + 1, \quad h = \quad z + 1. \end{aligned}$$

In matrix notation we have

$$A = \begin{pmatrix} 0\,1\,0\,0\,0\,1\,0\,0\,1 \\ 1\,0\,0\,0\,0\,1\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,1\,0\,0\,0\,0\,1\,0 \\ 0\,0\,1\,0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,0\,1\,1 \end{pmatrix} \text{ and } E = \begin{pmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,-1 \\ 0\,1\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0\,0\,0\,-1 \\ 0\,0\,0\,1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,-1 \\ 0\,0\,0\,0\,0\,1\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0 \end{pmatrix},$$

where E is the reduced row echelon form of A. Thus, this system has rank eight. Yet, we know from Corollary 1 that six rows (f, yg, zg, yh, g, h) would have been sufficient. Additionally, from Buchberger's first criterion [17] we know that in fact only the four rows f, yg, g, h need to be considered since the leading terms of g and h are coprime. Thus, in addition to the row that reduces to zero (as predicted by Buchberger's first criterion) the matrix constructed by XL contains four rows which are redundant even though they do not reduce to zero. Conversely, any reduction that produced a new lower leading term in the matrix constructed by XL could be attributed to S-polynomials.

Note that Lemma 2 does not state that $\mathrm{LM}(f) = \max\{\mathrm{LM}(S(f_j, f_{j+1}))\}$, but rather that the leading terms of summands decrease once rewritten using S-polynomials. In the following example, we consider the case when $\mathrm{LM}(f) < \max\{\mathrm{LM}(S(f_j, f_{j+1}))\}$. In this case, we can reapply Lemma 2 to $f_i' = S(f_i, f_j)$ as the following example emphasizes.

Example 2. Consider the polynomials $f = xy + a$, $g = yz + b$, and $h = ab + 1$ in the polynomial ring $\mathbb{F}_{127}[x, y, z, a, b]$. We consider the degree reverse lexicographical term ordering. Only one S-polynomial does not reduce to zero: $s_0 = zf - xg = za - xb$. From s_0 we can then construct $s_1 = bs_0 - zh = xb^2 + z$, among others, also at degree 3, which is an element of the reduced Gröbner basis. The XL algorithm at degree 3 will produce

$$\{m \cdot p \mid m \in \{1, x, y, z, a, b\}, p \in \{f, g, h\}\},$$

which reduces to

$$\begin{array}{llll} x^2y + xa, & xy^2 + ya, & xyz + xb, & y^2z + yb, \\ yz^2 + zb, & xya + a^2, & yza - 1, & xyb - 1, \\ yzb + b^2, & xab + x, & yab + y, & zab + z, \\ a^2b + a, & ab^2 + b, & xy + a, & yz + b, \\ za - xb, & \text{and} & ab + 1 & \end{array}$$

by Gaussian elimination. Note that $xb^2 + z$ is not in that list. However, if we increase the degree of XL to 4, the list returned is

$$
\begin{array}{llll}
x^3y + x^2a, & x^2y^2 - a^2, & xy^3 + y^2a, & x^2yz + x^2b, \\
xy^2z + 1, & y^3z + y^2b, & xyz^2 + xzb, & y^2z^2 - b^2, \\
yz^3 + z^2b, & x^2ya + xa^2, & xy^2a + ya^2, & xyza - x, \\
y^2za - y, & yz^2a - z, & xya^2 + a^3, & yza^2 - a, \\
x^2yb - x, & xy^2b - y, & xyzb - z, & y^2zb + yb^2, \\
yz^2b + zb^2, & x^2ab + x^2, & xyab - a, & y^2ab + y^2, \\
xzab + xz. & yzab - b, & z^2ab + z^2, & xa^2b + xa, \\
ya^2b + ya, & za^2b + xb, & a^3b + a^2, & xyb^2 - b, \\
yzb^2 + b^3, & xab^2 + xb, & yab^2 + yb, & zab^2 + zb, \\
a^2b^2 - 1, & ab^3 + b^2, & x^2y + xa, & xy^2 + ya, \\
xyz + xb, & y^2z + yb, & yz^2 + zb, & xya + a^2, \\
xza - x^2b, & yza - 1, & z^2a - xzb, & za^2 + x, \\
xyb - 1, & yzb + b^2, & xab + x, & yab + y, \\
zab + z, & a^2b + a, & \mathbf{xb^2 + z}, & ab^2 + b, \\
xy + a, & yz + b, & za - xb & \text{and } ab + 1,
\end{array}
$$

which does contain $xb^2 + z$. Thus, XL did produce $xb^2 + z$ in one step at degree 4 but it could not produce $xb^2 + z$ at degree 3 since this element corresponds to

$$b(zf - xg) - zh = (bz)f - (bx)g - zh,$$

but we have that $\deg(bzf) = 4$. Note that this behaviour of XL is the motivation for the Mutant concept.

4 Mutants and MXL algorithms

Let $F = \{f_0, \ldots, f_{m-1}\} \subset \mathbb{F}[x_0, \ldots, x_{n-1}]$, and $\mathcal{I} = \langle f_0, \ldots, f_{m-1} \rangle$ be the ideal generated by F. Recall that any element $f \in \mathcal{I}$ can be written as

$$f = \sum_{i=0}^{m-1} h_i \cdot f_i, \text{ with } h_i \in \mathbb{F}[x_0, \ldots, x_{n-1}].$$

Note that this representation is usually not unique. Following the terminology of [10], we call the *level* of the representation $\sum_{f_i \in F} h_i \cdot f_i$ of f the maximum degree of $\{h_i \cdot f_i \mid f_i \in F\}$. We call the *level* of f the minimal level of all its representations. We can then define the concept of a *mutant* [10, 31, 30].

Definition 3. *Given a set of generators F of an ideal $\mathcal{I}$, a polynomial $f \in \mathcal{I}$ is a* mutant *if its total degree is strictly less than its level.*

A mutant corresponds to a "low-degree" relation occurring during XL or more generally during any Gröbner basis computation. It follows from the discussion in Section 3 that, in the language of commutative algebra, a mutant occurs when a S-polynomial has a lower-degree leading monomial after reduction by F and if this new leading monomial was not in the set $\mathrm{LM}(F)$ before reduction.

The concept of mutant has recently motivated the proposal of a family XL-style algorithms [10, 31, 30, 9]. We discuss below the most prominent, namely the MXL_3 algorithm.

4.1 MXL$_3$ Algorithm

The MXL family of algorithms improves the XL algorithm using the mutant concept. In particular, the MXL$_3$ differs from XL in the following respects:

1. Instead of "blindly" increasing the degree in each iteration of the algorithm, the MXL algorithms treat mutants at the lowest possible degree, (cf. line 12 in Algorithm 2). This is the key contribution of the MXL algorithm [10].
2. Instead of considering all elements $F_{=d}$ of the current degree d, MXL$_3$ only considers a subset of elements per iteration. It incrementally adds more elements of the current degree, if the elements of the previous iteration did not suffice to solve the system (cf. line 27 in Algorithm 2). This is called *partial enlargement* in [31, 30]. This is the key contribution of the MXL$_2$ algorithm [31].
3. XL terminates at the user-provided degree D, while MXL$_3$ does not require to fix a degree a priori. Instead, the algorithm will terminate once a Gröbner basis was found using some new criterion (cf. line 21 in Algorithm 2). This is the key contribution of the MXL$_3$ algorithm [30].

In Algorithm 2 we present pseudocode for a slightly simplified variant of the MXL$_3$ algorithm. We use this presentation in Section 5 to compare it with the F_4 algorithm.

Our pseudocode has some minor differences with the pseudocode presented in [30]; we list these below:

Partial enlargement. We disregard any partial enlargement strategy in the case when mutants were found. This matches the pseudocode in [30]. However, the actual implementation of MXL$_3$ does indeed use the partial enlargement when $Mu \neq \varnothing$ (i.e. mutants exist) [29]. We note that our pseudocode and that in [10] are equivalent to MXL [10] in this case. Since our work is mainly concerned with the concept of mutants, maintaining this simplification seems appropriate.

Choice of y. In line 14 we set y to $\max\{\mathrm{LV}(f) \mid f \in F_{\leq k+1}\}$ instead of $\max\{\mathrm{LV}(f) \mid f \in Mu_{=k}\}$ since this allows reductions among all elements of degree $k+1$ instead of only those in $Mu_{=k+1}$. Restricting reduction to the elements of $Mu_{=k+1}$ could lead to incomplete reductions and thus results. The actual implementation of MXL$_3$ uses "partial enlargement" in this step and thus increases y iteratively [29].

Incomplete reductions. In line 28 we removed the optimisation that only variables $\leq x$ are used for multiplication in the extension step. This optimisation can lead to an incorrect result as some reductions are never performed. As an example, consider $f = ab + 1$, $g = bc + a + b$ and $h = c$. The reduced Gröbner basis of the ideal $\langle f, g, h\rangle$ over $\mathbb{F}_2[a, b, c]$ with respect to a degree lexicographical term ordering is $\{a + 1, b + 1, c\}$. However, the pseudocode of MXL$_3$ as described in [30] will not perform the necessary reductions. The leading variable of h is c, thus $h \in \mathrm{LV}(F, c)$ and h is never extended using any variables except c since $a > c$ and $b > c$.
Furthermore, the S-polynomial $S(f, g) = cf - ag = (abc + c) - (abc + ab + a) = ab + a + c$ is not constructed since ag requires multiplication of g in $\mathrm{LV}(F, b)$ by a but $a > b$. Thus, on termination the output of MXL$_3$ is not a Gröbner basis.
Our change matches Proposition 3 from [30], which requires that for $H \longleftarrow \{t \cdot g \mid g \in G, t \text{ a term and } \deg(t \cdot g) \leq D + 1\}$ the reduced row echelon form of H is G. However, this property is not enforced by MXL$_3$ as presented in pseudocode in [30], since some $t \cdot g$ are prohibited from being constructed if $\deg(t) = 1$ and $t > \mathrm{LV}(g)$. We confirmed with the authors of [30] that their implementation catches up on those missing multiplications when $newExtend = True$ [29].

We also present a simplified version of the F_4 algorithm in Algorithm 3. For this, we need however to introduce the required notation.

Definition 4. *Let $F \subset \mathbb{F}[x_0, \ldots, x_{n-1}]$, and $(f, g) \in F \times F$ with $f \neq g$. We denote:*

$$\textsc{Pair}(f, g) = \big(\mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g)), m_f, f, m_g, g\big),$$

where $\mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g)) = \mathrm{LM}(m_g \cdot g) = \mathrm{LM}(m_f \cdot f)$.

Now, let $P = \{\textsc{Pair}(f, g) \mid \forall (f, g) \in P \times P \text{ with } g > f\}$, $\mathrm{p} = \textsc{Pair}(f, g) \in P$. *We define* Left *and* Right *as:*

$$\begin{aligned} \textsc{Left}(\mathrm{p}) &= (m_f, f) & \textsc{Right}(\mathrm{p}) &= (m_g, g), \\ \textsc{Left}(P) &= \bigcup_{\mathrm{p} \in P} \textsc{Left}(\mathrm{p}) & \textsc{Right}(P) &= \textstyle\bigcup_{\mathrm{p} \in P} \textsc{Right}(\mathrm{p}). \end{aligned}$$

Input: F – a list of polynomials $f_0, \ldots, f_{m-1} \in \mathbb{F}[x_0, \ldots, x_{n-1}]$ spanning a zero-dimensional ideal.
Result: A Gröbner basis for $\langle f_0, \ldots, f_{m-1} \rangle$.

begin
  $D \longleftarrow \max\{\deg(f) \mid f \in F\}$;
  $d \longleftarrow \min\{\deg(f) \mid f \in F\}$;
  $Mu \longleftarrow \varnothing$;
  $newExtend \longleftarrow True$;
  $x \longleftarrow x_0$;
  $CL \longleftarrow d$;
  while *True* **do**
    $\tilde{F}_{\leq d} \longleftarrow$ the row echelon form (or matrix form) of $F_{\leq d}$;
    $Mu \longleftarrow Mu \cup \{f \in \tilde{F}_{\leq d} \mid \deg(f) < d$ **and** $\mathrm{LM}(f) \notin \mathrm{LM}(F_{\leq d})\}$;
    $F_{\leq d} \longleftarrow \tilde{F}_{\leq d}$;
    `// did we find mutants?`
    if $Mu \neq \varnothing$ **then**
      $k \longleftarrow \min\{\deg(f) \mid f \in Mu\}$;
      $y \longleftarrow \max\{\mathrm{LV}(f) \mid f \in F_{\leq k+1}\}$;
      $Mu^{+}_{=k} \longleftarrow$ Multiply all elements of $Mu_{=k}$ by all variables $\leq y$;
      $Mu \longleftarrow Mu \setminus Mu_{=k}$;
      $F \longleftarrow F \cup Mu^{+}_{=k}$;
      $d \longleftarrow k + 1$;
    else
      `// does the basis contain all monomials of some degree` d_t`?`
      if $d < CL$ **and** $M_{=d_t} \subseteq \mathrm{LM}(F)$ *for some* $1 \leq d_t \leq d$ **then**
        `// We found a Gröbner basis`
        return F;
      `// did we do all enlargements at this degree already?`
      if $newExtend = True$ **then**
        $D \longleftarrow D + 1$;
        $x \longleftarrow \min\{\mathrm{LV}(f) \mid f \in F_{=D-1}\}$;
        $newExtend \longleftarrow False$;
      else
        `// do partial enlargement and eliminate`
        $x \longleftarrow \min\{\mathrm{LV}(f) \mid f \in F_{=D-1}$ **and** $\mathrm{LV}(f) > x\}$;
        $F^{+} \longleftarrow$ Multiply all elements of $\mathrm{LV}(F, x)$ by all variables $\not\leq x$ without redundancies;
        $F \longleftarrow F \cup F^{+}$;
        if $x = x_0$ **then**
          $newExtend \longleftarrow True$;
          $CL = D$;
      $d \longleftarrow D$;
end

Algorithm 2: MXL_3 (simplified)

Input: F – a tuple of polynomials $f_0, \ldots, f_{m-1}$
Input: SEL – a selection strategy
Result: a Gröbner basis for F

begin
  $G, i \longleftarrow F, 0$;
  $\tilde{F}_i^+ \longleftarrow F$;
  $P \longleftarrow \{\text{PAIR}(f, g) \mid \forall f, g \in G \textbf{ with } g > f\}$;
  while $P \neq \varnothing$ **do**
    $i \longleftarrow i + 1$;
    $P_i \longleftarrow \text{SEL}(P)$;
    $P \longleftarrow P \setminus P_i$;
    $\mathcal{L}_i \longleftarrow \text{Left}(P_i) \bigcup \text{Right}(P_i)$;
    `// Symbolic Preprocessing`
    $F_i \longleftarrow \{t \cdot f \mid \forall (t, f) \in \mathcal{L}_i\}$;
    $Done \longleftarrow \text{LM}(F_i)$;
    while $M(F) \neq Done$ **do**
      $m \longleftarrow$ an element in $M(F) \setminus Done$;
      add m to $Done$;
      if $\exists\ g \in G$ **such that** $\text{LM}(g) \mid m$ **then**
        $u = m/\text{LM}(g)$;
        add $u \cdot g$ to F_i;
    `// Gaussian Elimination`
    $\tilde{F}_i \longleftarrow$ the row echelon form of F_i;
    $\tilde{F}_i^+ \longleftarrow \{f \in \tilde{F} \mid \text{LM}(f) \notin \text{LM}(F)\}$;
    for $h \in \tilde{F}_i^+$ **do**
      $P \longleftarrow P \bigcup \{\text{PAIR}(f, h) : \forall f \in G\}$;
      add h to G;
  return G;
end

Algorithm 3: F_4 (simplified)

5 Relationship between the MXL Algorithms and F_4

In this section we discuss the relation between MXL_3 and F_4. It was shown in [3] that XL can be understood as a redundant variant of F_4 (cf. Lemma 1). Thus, we know that the "framework" of MXL_3 is compatible with F_4. Thus in order to study the connection between the two algorithms, we only have to consider the modifications made in MXL_3 compared to XL.

5.1 Mutants

The most visible change to XL in MXL_3 is the special treatment given to mutants. That is, instead of increasing the degree d in each iteration, if there is a fall of degree, then these new elements are treated at the current or perhaps a smaller degree before the algorithm proceeds to increase the degree as normally. Thus, compared to XL, the MXL family of algorithms may terminate at a lower degree.

On the other hand, the F_4 algorithm does not specify how to choose polynomials in each iteration of the main loop. Instead, the user passes a function Sel which specifies how to select pairs of polynomials. However, in [20] it is suggested to choose the normal selection strategy for most inputs. We recall here how the normal strategy has been adapted in F_4.

Definition 5 (Normal Strategy). *Let $F = \{f_0, \ldots, f_{m-1}\}$. We shall say that a pair $(f, g) \in F \times F$ with $f \neq g$ is a critical pair. Let then $\mathcal{P} \subset F \times F$ be the set of critical pairs. We denote by $\text{LCM}(p_{ij})$ the least common multiple of the leading monomials of the critical pair $p_{ij} = (f_i, f_j) \in \mathcal{P}$. We also call $\deg(\text{LCM}(p_{ij}))$ the degree of the critical pair p_{ij}. Further, let*

$$d = \min\{\deg(\text{LCM}(p)) \mid p \in \mathcal{P}\}$$

be the minimal degree of those least common multiples of p in $\mathcal{P}$. Then the normal selection strategy selects the subset

$$\mathcal{P}' = \{p \in \mathcal{P} \mid \deg(\text{LCM}(p)) = d\}.$$

We can now state our main result.

Theorem 1. *Let both MXL_3 and F_4 compute a Gröbner basis with respect to the same degree compatible ordering on the same input. Assume that until iteration i (inclusive) of the main loop both F_4 and MXL_3 computed the same list of polynomials. Furthermore, assume that $Mu \neq \varnothing$ in Algorithm 2 at line 12 and define k to be the minimal degree of a polynomial in Mu. The set of polynomials $F_{\leq k+1}$ considered by MXL_3 in the next iteration of the main loop is a superset of the polynomials considered by F_4 when using the* Normal Selection Strategy *in the next iteration $i+1$. Furthermore, every polynomial in $F_{\leq k+1}$ not in the set considered by F_4 is redundant in this iteration.*

Proof. First consider the F_4 algorithm, and assume that Sel is the Normal Selection Strategy. Then, the set $\mathcal{P}_{i+1}$ will contain the S-polynomials of lowest degree in $\mathcal{P}$. Every S-polynomial in $\mathcal{P}_{i+1}$ will have at least degree $k+1$, since the set $Mu_{=k}$ is in row echelon form and k is the minimal degree in Mu. If there exists a S-polynomial of degree $k+1$ then it is of the form $t_i f_i - t_j f_j$ with $\deg(t_i f_i) = k+1$ and $\deg(t_j f_j) = k+1$, where at least one of t_i, t_j has degree 1. Since MXL_3 constructs all multiples $t_{ij} f_i$ with $\deg(t_{ij}) = 1$ if $\deg(f_i) = k$ and all elements of degree $k+1$ in the next iteration, both components of the S-polynomial are included in $F_{\leq k+1}$.

In the *Symbolic Preprocessing* phase F_4 also constructs all components of *potential* S-polynomials that could arise during the elimination. These are always of the form $f_i - t_j f_j$ where $\deg(f_i) = \deg(t_j f_j)$. Since MXL_3 considers all monomial multiplies of all f_j up to degree $k+1$ in the next iteration, these components are also included in the set F_{k+1}.

Recall from Corollary 1 that all $f = \Sigma_{i=0}^{t-1} c_i x^{\alpha(i)} f_i$ can be rewritten as

$$f = \sum_{j=0}^{t-2} b_j x^{\delta - \tau_j} S(f_j, f_{j+1})$$

if $f < \max\{x^{\alpha(i)} f_i\}$. Note that $\deg(x^\delta) \leq k+1$ for $F_{\leq k+1}$ and that $\deg(x^{\tau_j}) = k+1$ for all S-polynomials contained in $F_{\leq k+1}$. It follows that $\deg(x^{\delta - \tau_j}) = 0$ if $b_j \neq 0$. That is, any f with a smaller leading term than its representation $\Sigma_{i=0}^{t-1} c_i x^{\alpha(i)} f_i$ can be computed by an $\mathbb{F}$-linear combination of S-polynomials: $f = \sum_{j=0}^{t-2} b_j S(f_j, f_{j+1})$.

It follows immediately from Corollary 1 that any multiple of f_i which does not correspond to a S-polynomial is redundant in this iteration since it cannot lead to a drop of a leading monomial. □

5.2 Partial Enlargement

The "partial enlargement" technique was introduced in MXL_2 and is also applied in MXL_3. Instead of multiplying every polynomial $f_i \in F$ by all variables $x_0, \ldots, x_{n-1}$ only a subset $\text{LV}(F, x)$ is considered. This subset is increased in each iteration by increasing x. In the language of linear algebra, the algorithm first computes the row echelon form of a submatrix in the lower right corner. If that does not suffice to produce elements of smaller degree, a bigger submatrix is considered.

This corresponds to selecting a subset of S-polynomials with small least common multiple in Sel instead of selecting all polynomials of minimal degree. We note that both the PolyBoRi package [7] and Magma computer algebra system [5] accept an option to restrict the number of S-polynomials considered in each iteration. However, the strategy how the number of S-polynomials is chosen in Magma and PolyBoRi is different from MXL_3. In the former ones, a *constant* number of S-polynomials is chosen as specified by the user; in the latter (MXL_3) a *changeable* number of S-polynomials is chosen based on the partition by leading variable. The strategy employed in MXL_3 will consider S-polynomials $S(f, g)$ where both f and g have leading variable at most x (inclusive). That is, if there is an S-polynomial $S(f, g) = t_f f - t_g g$ with $\text{LV}(f) < \text{LV}(g)$, MXL_3 will construct $t_f \cdot f$ when considering $\text{LV}(F, \text{LV}(f))$ and $t_g \cdot g$ when considering $\text{LV}(F, \text{LV}(g))$. Since $F_{\leq d}$ contains all elements of degree at most d, both components are included in the matrix when $\text{LV}(F, \text{LV}(g))$ are considered.

It is currently not clear which strategy for selecting subsets of S-polynomials is beneficial under which conditions. It should be noted however that if the size of the matrix is the main concern then selecting exactly the smallest S-polynomial in each iteration would be optimal; just as Buchberger's algorithm does. On the other hand, the contribution of algorithms such as F_4 is to improve performance by considering more than one S-polynomial in each iteration. Thus, it is not certain that using matrix sizes as a main measure of comparison gives an adequate picture of the performance of these algorithms.

5.3 Termination Criterion

The key contribution of the MXL_3 algorithm is the introduction of a new criterion to detect when a Gröbner basis is found. Since the MXL family does not use the concept of critical pairs, standard

termination criteria such as an empty list of pairs are not immediately applicable. In Lemma 3 we give an equivalent variant of this criterion, rephrased to be more suitable for our discussion.

Lemma 3 (Proposition 3 in [30]). *Let $G = \{g_0, \ldots, g_{s-1}\}$ be a finite subset of $\mathbb{F}[x_0, \ldots, x_{n-1}]$ with D being the highest degree of its elements. Suppose that the following hold:*

1. *all monomials of degree D in $\mathbb{F}[x_0, \ldots, x_{n-1}]$ are divisible by a leading monomial of some $g_i \in G$; and*
2. *if $H = G \cup \{t \cdot g_i \mid g_i \in G, t \text{ a monomial and } \deg(t \cdot g_i) \leq D + 1\}$, there exists $\tilde{H}$ – a row echelon form of H – such that $\mathrm{LM}(\tilde{H}_{\leq D}) \subset \langle \mathrm{LM}(G) \rangle$.*

Then G is a Gröbner basis.

Note that condition 1 implies that the ideal generated by G is 0-dimensional.

The MXL_3 algorithm uses a termination criterion based on Lemma 3 and thus will consider matrices up to degree $D + 1$ (where D is defined as in Lemma 3). The F_4 algorithm, on the other hand, will terminate once the list of critical pairs is empty. It is obvious that no new pairs will be created after the Gröbner basis is found, since all reductions will lead to zero in this situation. However, if we consider F_4 as given in Algorithm 3, one can see that the algorithm may consider pairs of degree $> D + 1$ after a Gröbner basis is discovered, if those pairs were constructed before the Gröbner basis is found. Put differently, the simplified F_4 variant considered in this work does not prune the list of critical pairs based on the current basis G. However, the full F_4 algorithm as specified in [20, p. 9] does indeed prune the list P by calling a subroutine called Update. In [20] a reference to [4, p. 230] is made – which applies Buchberger's first and second criteria using the Gebauer-Möller installation – as an example of such a routine.

The question thus becomes whether Buchberger's first and second criteria will remove all pairs of degree $> D + 1$ if the conditions (1) and (2) of Lemma 3 hold. An algorithmic variant of Buchberger's second criterion is given in the Lemma below.

Lemma 4 (Buchberger's second criterion). *Let G be a finite subset of the $\mathbb{F}[x_0, \ldots, x_{n-1}]$ and $p, g_1, g_2 \in \mathbb{F}[x_0, \ldots, x_{n-1}]$ be such that*

$$\mathrm{LM}(p) \mid \mathrm{LCM}(\mathrm{LM}(g_1), \mathrm{LM}(g_2)).$$

and $S(g_1, p)$, $S(g_2, p)$ have already been considered. Then $S(g_1, g_2)$ does not need to be considered and can be discarded.

We can now prove that the full F_4 algorithm will not consider pairs of a higher degree than the MXL_3 when applying Buchberger's second criterion.

Proposition 1. *We assume a degree compatible ordering on $\mathbb{F}[x_0, \ldots, x_{n-1}]$. If during a Gröbner basis computation using the full F_4 algorithm conditions (1) and (2) of Lemma 3 hold, then Buchberger's second criterion will remove any pair of degree $> D + 1$ from the list of critical pairs and thus F_4 will consider critical pairs of degree at most $D + 1$.*

Our proof follows very closely the original proof of Lemma 3 in [30].

Proof. Let $G = \{g_0, \ldots, g_{s-1}\}$ be a finite subset of $\mathbb{F}[x_0, \ldots, x_{n-1}]$ with D being the highest degree of its elements such that:

1. all monomials of degree D in $\mathbb{F}[x_0, \ldots, x_{n-1}]$ are divisible by a leading monomial of some $g_i \in G$; and
2. if $H = G \cup \{t \cdot g_i \mid g_i \in G, t \text{ a monomial and } \deg(t \cdot g_i) \leq D+1\}$, there exists $\tilde{H}$ – a row echelon form of H – such that $\mathrm{LM}(\tilde{H}_{\leq D}) \subset \langle \mathrm{LM}(G) \rangle$.

We denote the S-polynomial $S(g_i, g_j)$ by f, and let $d = \deg(f)$. We only have to consider pairs of degree $d > D+1$.

To do so, let $m = \mathrm{LCM}(\mathrm{LM}(g_i), \mathrm{LM}(g_j))$. There exist monomials m_i, m_j such that $m = m_i \cdot \mathrm{LM}(g_i) = m_j \cdot \mathrm{LM}(g_j)$. It is clear that $\mathrm{GCD}(m_i, m_j) = 1$.

By assumption $\deg(g_i)$ and $\deg(g_j)$ are at most equal to D. This implies that $\deg(m_j) \geq 2$ (resp. $\deg(m_j) \geq 2$) since $d > D+1$. It is then possible to write $m_i = m_{i,1} \cdot m_{i,2}$ such that $\deg(g_i) + \deg(m_{i,2}) = D+1$ and $\deg(m_{i,1}) \geq 1$. A similar decomposition can be found for $m_j = m_{j,1} \cdot m_{j,2}$. Thus, we have that all monomials $m_{i,1}, m_{i,2}, m_{j,1} \cdot m_{j,2}$ are of degree ≥ 1.

Now, let $m^* = \frac{m}{m_{i,1} \cdot m_{j,1}}$. By construction, we have

$$\mathrm{LCM}(m^*, \mathrm{LM}(g_i)) = m/m_{i,1} \text{ (resp. } \mathrm{LCM}(m^*, \mathrm{LM}(g_j)) = m/m_{j,1}),$$

which divides m properly. We also have $\deg(m^*) \leq D$. Since m_1 and m_2 must be distinct, we have that m^* cannot be equal to either $\mathrm{LM}(g_i)$ or $\mathrm{LM}(g_j)$. By condition 1, there exists $g \in G \setminus \{g_1, g_2\}$ such that with $\mathrm{LM}(g) = m^*$. In addition

$$\deg(\mathrm{LCM}(\mathrm{LM}(g), \mathrm{LM}(g_i)) < \deg(m)$$

and $\deg(\mathrm{LCM}(\mathrm{LM}(g), \mathrm{LM}(g_j)) < \deg(m)$. Thus, $S(g, g_i)$ and $S(g, g_j)$ are being considered at a lower degree than $D+1$.

Finally, m^* divides $m = \mathrm{LCM}(\mathrm{LM}(g_i), \mathrm{LM}(g_j))$ by construction. It then follows from Buchberger's second criterion that $f = S(g_i, g_j)$ does not need to be considered and is discarded. □

6 Conclusion

In this work we have studied the MXL family algorithms, and their connections to Gröbner bases theory. We demonstrated that the mutant strategy as used in the MXL algorithms is in fact a redundant variant of the Normal Selection Strategy. Furthermore, we showed that the partial enlargement strategy proposed in [31] corresponds to selecting a subset of S-polynomials of minimal degree in each iteration of algorithms such as F_4. As a result, we conclude that both the MXL and MXL_2 algorithms can be seen as redundant variants of the F_4 algorithm, although the latter may select critical pairs differently from usual F_4 implementations. Finally, we studied the novel termination criterion proposed in [30] and concluded that it does not allow the algorithm to terminate at a lower degree than F_4. Consequently, we conclude that MXL_3 too can be understood as a redundant variant of the F_4 algorithm.

We conclude with a brief discussion on what we view as the limitations of using running times as the basis for *comparison* between Gröbner basis *algorithms*. As noted early in this paper, linear algebra-based Gröbner bases algorithms allow several degrees of freedom to the designer and implementer of the algorithm to generate the matrices, and selection of strategies can drastically affect the efficiency of the computations. Furthermore, the specific implementation details and sub-algorithms used in the implementation (e.g. the package used for performing the Gaussian reductions, the internal representation of sparse matrices ,...) will also have great effect on running times and memory requirements (cf. Appendix A for an example).

In fact, we claim that three almost-independent aspects will affect running times of such algorithms: the mathematical details of the algorithm itself, the strategies and heuristics used in the implementation, and the low-level implementation details. The first aspect was the main focus of interest in this paper, but it should be clear that our results do not preclude that particular *implementations* of MutantXL algorithms can outperform particular *implementations* of F_4/F_5 in some situations. On the other hand, we are aware that it is difficult to compare the complexity of Gröbner basis algorithms and strategies and that designers often have little choice but to resort to experimental data to demonstrate the viability of their approach.

7 Acknowledgements

The work described in this paper has been supported by the Royal Society grant JP090728. We would like to thank Stanislav Bulygin, Jintai Ding and Mohamed Saied Emam Mohamed for helpful comments and discusssions on earlier drafts of this work.

References

1. Martin Albrecht and Carlos Cid. Algebraic techniques in differential cryptanalysis. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 193–208. Springer, 2009.
2. Martin Albrecht, Carlos Cid, Thomas Dullien, Jean-Charles Faugère, and Ludovic Perret. Algebraic Precomputations in Differential Cryptanalysis. In M. Yung and X. Lai, editors, *Information Security and Cryptology: 6th International Conference, Inscrypt 2010, Revised Selected Papers*, pages 1–18. Springer-Verlag, October 2010.
3. Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison between XL and Gröbner basis algorithms. In *Advances in Cryptology — ASIACRYPT 2004*, pages 338–353, 2004.
4. Thomas Becker and Volker Weispfenning. *Gröbner Bases - A Computational Approach to Commutative Algebra*. Springer Verlag, Berlin, Heidelberg, New York, 1991.
5. Wieb Bosma, John Cannon, and Catherine Playoust. The MAGMA Algebra System I: The User Language. In *Journal of Symbolic Computation 24*, pages 235–265. Academic Press, 1997.
6. Charles Bouillaguet, Jean-Charles Faugère, Pierre-Alain Fouque, and Ludovic Perret. Practical cryptanalysis of the identification scheme based on the isomorphism of polynomial with one secret problem. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography 2011*, volume 6571 of *Lecture Notes in Computer Science*. Springer, 2011.
7. Michael Brickenstein and Alexander Dreyer. PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials. *Journal of Symbolic Computation*, 44(9):1326–1345, September 2009.
8. Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal*. PhD thesis, Universität Innsbruck, 1965.
9. Johannes Buchmann, Daniel Cabarcas, Jintai Ding, and Mohamed Saied Emam Mohamed. Flexible partial enlargement to accelerate Gröbner basis computation over F2. In *Africacrypt*, 2010.
10. Johannes A. Buchmann, Jintai Ding, Mohamed Saied Emam Mohamed, and Wael Said Abd Elmageed Mohamed. MutantXL: Solving multivariate polynomial equations for cryptanalysis. In Helena Handschuh, Stefan Lucks, Bart Preneel, and Phillip Rogaway, editors, *Symmetric Cryptography*, number 09031 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, Germany.
11. Carlos Cid and Gaëtan Leurent. An Analysis of the XSL Algorithm. In *Advances in Cryptology — ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352, Berlin, Heidelberg, New York, 2005. Springer Verlag.
12. Nicolas Courtois and Willi Meier. Algebraic Attacks on Stream Ciphers with Linear Feedback. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 345–359. Springer–Verlag, 2003.
13. Nicolas T. Courtois. Algebraic Attacks over GF(2**k), Application to HFE Challenge 2 and Sflash-v2. In *Public Key Cryptography – PKC 2004*, pages 201–217, Berlin, Heidelberg, New York, 2004. Springer Verlag.

14. Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407, Berlin, Heidelberg, New York, 2000. Springer Verlag.
15. Nicolas T. Courtois and Jacques Patarin. About the XL algorithm over GF(2). In Marc Joye, editor, *Topics in Cryptology - CT-RSA 2003: The Cryptographers' Track at the RSA Conference 2003; Proceedings*, volume 2612 of *Lecture Notes in Computer Science*, pages 141–157, Berlin, Heidelberg, New York, 2003. Springer Verlag.
16. Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287, Berlin, Heidelberg, New York, 2002. Springer Verlag.
17. David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms*. Springer Verlag, Berlin, Heidelberg, New York, 1 edition, 1992.
18. Claus Diem. The XL-algorithm and a conjecture from commutative algebra. In P.J. Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 323–337. Springer-Verlag, 2004.
19. J.-C. Faugère and S. Lachartre. Parallel Gaussian Elimination for Gröbner bases computations in finite fields. In M. Moreno-Maza and J.L. Roch, editors, *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation*, PASCO '10, pages 89–97, New York, NY, USA, July 2010. ACM.
20. Jean-Charles Faugère. A new efficient algorithm for computing Gröbner basis (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.
21. Jean-Charles Faugère. A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5). In T. Mora, editor, *International Symposium on Symbolic and Algebraic Computation – ISSAC 2002*, pages 75–83, 2002.
22. Jean-Charles Faugère, Françoise Levy dit Vehel, and Ludovic Perret. Cryptanalysis of minrank. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 280–296. Springer, 2008.
23. Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, and Teo Mora. Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. In *Journal of Symbolic Computation 16*, pages 329–344. Academic Press, 1993.
24. Jean-Charles Faugère and Antoine Joux. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems using Gröbner Bases. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *LNCS*, pages 44–60. Springer–Verlag, 2003.
25. Jean-Charles Faugère and Ludovic Perret. Cryptanalysis of $2r^-$ schemes. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 357–372. Springer, 2006.
26. Jean-Charles Faugère and Ludovic Perret. Algebraic cryptanalysis of curry and flurry using correlated messages. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Inscrypt*, volume 6151 of *Lecture Notes in Computer Science*, pages 266–277. Springer, 2009.
27. Daniel Lazard. Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations. In J.A. van Hulzen, editor, *Proceedings of the European Computer Algebra Conference on Computer Algebra*, volume 162 of *LNCS*, pages 146–156. Springer–Verlag, 1983.
28. Chu-Wee Lim and Khoongming Khoo. An analysis of XSL applied to BES. In Alex Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 242–253. Springer, 2007.
29. Mohamed Saied Emam Mohamed. private communication, January 2011.
30. Mohamed Saied Emam Mohamed, Daniel Cabarcas, Jintai Ding, Johannes Buchmann, and Stanislav Bulygin. MXL3: An efficient algorithm for computing Gröbner bases of zero-dimensional ideals. In *12th International Conference on Information Security and Cryptology (ICISC)*, 2009.
31. Mohamed Saied Emam Mohamed, Wael Said Abd Elmageed Mohamed, Jintai Ding, and Johannes Buchmann. MXL2: Solving polynomial equations over GF(2) using an improved mutant strategy. In *PQCrypto*, pages 203–215, 2008.
32. Mohamed Saied Emam Mohamed, Fabian Werner, Jintai Ding, and Johannes Buchmann. Algebraic attack on the MQQ public key cryptosystem. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *8th International Conference on Cryptology and Network Security (CANS)*, volume 5888 of *Lecture Notes in Computer Science*, pages 392–401. Springer Verlag, 2009. pre-print available at `http://eprint.iacr.org/2008/451`.
33. Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In U. Maurer, editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *LNCS*, pages 33–48. Spinger–Verlag, 1996.

34. Ludovic Perret. A fast cryptanalysis of the isomorphism of polynomials with one secret problem. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 354–370. Springer, 2005.
35. Enrico Thomae and Christopher Wolf. Unravel XL and its variants. Cryptology ePrint Archive, Report 2010/596, 2010. `http://eprint.iacr.org/`.

A Effect of Linear Algebra Implementations on Gröbner Basis Computations

To show the effect of the linear algebra implementation, we compare two implementations of the F_4 algorithm. The only difference is the linear algebra package use to perform the Gaussian elimination step. We compare the original FGb implementation with the new linear algebra package described in [19]. However, to make the comparison fair we only us a sequential version of the package described in [19]. To compare, we consider the reduction of the 7th matrix occurring in the computation of a Gröbner basis of the standard benchmark Katsura 12 over $\mathbb{F}_{65521}$, as well as the full Gröbner basis computation. Typically, it takes 326.1 sec and 250 Mbytes to reduce the 7th matrix with FGb and 83.7 seconds and 682 Mbytes using FGb with the library from [19].

Table 1. Algorithm: F4 Benchmark: Katsura 14 mod $p = 65521$.

	Matrix 7 $(21,915 \times 23,127)$	Full Gröbner basis
FGb/CPU	83 s.	326 s.
FGb/Memory	250 Mbytes	262 Mbytes
FGb/Pasco/CPU [19] (1 core)	32 s.	151 s.
FGb/Pasco/Memory [19]	682 Mbytes	682 Mbytes