Mutant Gröbner Basis Algorithm

Jintai Ding, Daniel Cabarcas, Dieter Schmidt, Johannes Buchmann and Stefan Tohaneanu

Abstract. This paper explores how to use the concept of mutants in the computation of a Gröbner basis for polynomial equations. We compare our approach to the algorithm F4 of Faugère, and we perform experiments on polynomials system coming from multivariate public key cryptosystems in comparison with the experimental results from Magma implementation of F4. The experimental results demonstrate that the mutant strategy could indeed work better in the case of polynomial systems with underlying structures that could be fully exploited by the mutant strategy.

Keywords. Gröbner bases, cryptosystems.

1. Introduction

Solving polynomial equations, single or multivariate equations, has always been a central topic in mathematics, since polynomial equations are used almost everywhere in both theory and in practical applications. The general strategy has always been to develop an algorithm for solving multivariate polynomial equations, which is applicable universally to all polynomial equations. The general Gröbner basis algorithm [2] is of such a type as well as its extensions F4 and F5, which have been developed more recently [7, 8].

Due to problems from cryptography [6, 3], there exists an increasing interest in solving multivariate polynomial equations over a finite field. Although the attack on HFE by the F4 algorithm has been very successful, there is a general consensus that one should establish a different strategy, namely one should develop algorithms that could make use of the hidden structures of these equations, as was done in [10]. Different attempts already exist, for example the SAT solver, which uses methods from logic.

In this paper we will present a new idea for developing polynomial solving algorithm, which is based on the concept of mutant. Mutant, a new concept to characterize the degeneration of a polynomial system, was discovered in 2006 by
Ding [4]. A new strategy is proposed in [4] to utilize such a structure to improve various polynomial solving algorithms, including the Gröbner basis algorithms when mutants occur. In this article we will present how the mutant concept can be used to modify the F4 algorithm and we will demonstrate by experimental examples, that it results in an improvement.

The paper is organized as follows. First we present the concept of mutant and the mutant Gröbner basis algorithm. We will explain the difference of this new algorithm with the usual F4 algorithm. We then present experimental results to show the difference between the algorithms and why mutants can provide an improvement.

2. Mutants

Currently there exist several different algorithms to solve polynomial equations. Most of them use the basic concept of an ideal from algebraic geometry and formulate the strategy and the algorithm along this concept. The Gröbner basis algorithm and the XL algorithm are examples of this approach.

Given a set of polynomial equations

\[ f_1(x_1, \ldots, x_n) = f_2(x_1, \ldots, x_n) = \cdots = f_n(x_1, \ldots, x_n) = 0 \]

from the point of view of algebraic geometry, we know that to solve this set of equations, all we need to do is to find “good elements” in the ideal generated by \( f_1, \ldots, f_n \), which we denote by \( I(f_1, \ldots, f_n) \). By definition any element \( g \) in this ideal can be written as:

\[ g = \sum_{i=1}^{n} g_i f_i, \]

where the \( g_i \)'s are elements in the polynomial ring \( k[x_1, \ldots, x_n] \), and \( k \) is a field.

The Gröbner basis algorithm in this case looks for a new set of generators of \( I(f_1, \ldots, f_n) \), which is in triangular form, that is, in the form:

\[ g_1(x_1, \ldots, x_n), g_2(x_2, \ldots, x_n), \ldots, g_{n-1}(x_{n-1}, x_n), g_n(x_n), \]

where the \( g_i \)'s are elements of the ideal. The general strategy to achieve this is as follows: starting from \( f_1, \ldots, f_n \), which span a subspace in \( I(f_1, \ldots, f_n) \), we enlarge this subspace, on the one hand as much as possible, so that we can get the polynomials we want, namely the Gröbner basis, but on the other hand, as little as possible so that we do not waste too much time or memory. The Buchberger algorithm enlarges the subspace by adding the so called s-pairs of the subspace, and the F4 algorithm adds further multiples of the polynomials that can be used to reduce the new polynomials. The general strategy of the XL is to enlarge the subspace by adding all the multiples of this subspace by monomials up to a certain degree.

In both algorithms the key parameter is the maximum degree \( D \) of the polynomials of the subspace, which allows us to find a Gröbner basis. The maximum
degree $D$ makes a tremendous difference in terms of computational complexity
and memory requirement even if the degree increases just by 1.

We noticed that in all these algorithms, newly generated polynomials are
treated as if they were all equal, which clearly should not be the case. Intuitively,
we conclude that in the process of enlarging the space, we should treat elements
different degrees differently, in particular, the elements of lower degree should
play a more important role than the ones of higher degree. In doing so, it seems
that it should be possible that we can find the Gröbner basis with a smaller $D$.

To convey this idea more precisely, we would like to define first a new concept,
which we call mutants.

**Definition 1.** Let $I$ be the ideal generated by a set of polynomials $f_1, \ldots, f_n$. An
element $g$ in $I$ can be written as

$$ g = \sum_{i=1}^{n} f_i g_i . $$

We say that this expression is a reduced expression, if among all possible such
expressions $\max \{\deg(f_i g_i) \mid i = 1, \ldots, n\}$ is the smallest, or with a degree
compatible ordering of the monomials, if among all possible such expressions the
highest terms of all leading terms of $f_i g_i$ is the lowest.

**Definition 2.** Let $I$ be the ideal generated by a set of polynomials $f_1, \ldots, f_n$. An
element $g$ in $I$ is called a mutant if

$$ g = \sum_{i=1}^{n} f_i g_i $$
is a reduced expression and $\deg(g)$ is less than $L = \max \{\deg(f_i g_i) \mid i = 1, \ldots, n\}$. For any such an expression of $g$, we call the number $L$ the level of the
expression of $g$ and we call the number $L - \deg(g)$ the degeneracy degree of $g$.

For a set of non-homogeneous equations, the concept of mutants should
closely be related to the concept of regular sequence and semi regular sequence,
namely the regular sequence or semi-regular sequence corresponds exactly to the
set of polynomial equations, whose polynomials have no mutants. This is a very
interesting direction one should follow. From a practical point of view, we believe
that among the equations with a relatively large number of variables only those
which can be solved are of interest, otherwise the complexity is just out of reach.
The concept of mutant should be studied even more so that we could understand
more accurately the complexity of solving these equations.

### 3. The mutant Gröbner basis algorithm and its difference to F4

From this point on, we will use the notations and definitions in [7], which are
commonly used in the literature. We also adopt the recommended strategy which
selects the critical pairs with the lowest degree. We assume that \(<\) represents a graded term order.

In this paper, we will present a basic version of the MutantF4 algorithm, and we will leave out the details about some of the standard routines of the algorithm, such as the details about the Buchberger criteria, about the reduction or update step and the details about the efficient reduction, reuse and storage of matrices. For the details on these topics we refer the readers to \([1, 7]\).

Intuitively the main difference between F4 and MutantF4 is that the mutant polynomials (or more precisely, the “new” polynomials of low degree) are given a preferential treatment compared to the rest. As in F4, MutantF4 selects the pairs of minimal degree (\(P^*\)), applies a symbolic preprocessing to them (to obtain \(\tilde{H}\)), row reduces the result to a row echelon form (\(\tilde{H}^+\)), and discriminates the new leading terms (\(\tilde{H}^+\)) from those adjoined in the symbolic preprocessing. However, in the MutantF4 algorithm this last set of polynomials is divided into two subsets, the mutants (\(M\)) and the rest (\(\tilde{H}^-\)), the mutants are adjoined to the basis (\(G\)) whereas the rest is kept in hold. A detailed installation of the algorithm is given below.

**Algorithm MutantF4**

**Input** : \(F\) a finite subset of \(R\)

**Output** : \(G\) a Gröbner basis for \(I(F)\) w.r.t \(<\)

\(P = \{\text{pair}(f, g) \mid f, g \in G \text{ with } f \neq g\}\)

\(\tilde{H}^- = \emptyset\)

**while** \(P \neq \emptyset\) or \(\tilde{H}^- \neq \emptyset\)

\(P^* = \{p \in P \mid \text{deg}(p) \text{ is minimal w.r.t } <\}\)

\(P = P \setminus P^*\)

\(L = \text{mult}(\text{left}(P^*) \cup \text{right}(P^*))\)

\((H, T) = \text{symbolic preprocessing}(L \cup \tilde{H}^-, G)\)

\(\tilde{H} = \text{Reduction to row echelon form of } H \text{ w.r.t. } <\)

\(\tilde{H}^+ = \{f \in \tilde{H} \mid \text{HT}(f) \notin T\}\)

\(M = \{f \in \tilde{H}^+ \mid \text{deg}(f) \text{ is minimal w.r.t } <\}\)

\(\tilde{H}^- = \tilde{H}^+ \setminus M\)

\(P = P \cup \{\text{pair}(f, g) \mid f \in M, g \in G\} \cup \{\text{pair}(f, g) \mid f, g \in M \text{ with } f \neq g\}\)

\(G = G \cup M\)

**return** \(G\)
The symbolic preprocessing algorithm is a generalization of the algorithm proposed in [7]. In the original description, it is assumed that the input set comes from critical pairs. We suggest a more general procedure where this is not assumed.

**Algorithm SymbolicPreprocessing**

**Input**: \( H, G \) finite subsets of \( R \)

**Output**: \( H' \) is \( H \) increased so that for all \( t \in T(H) \), if \( t \) is divisible by an element of \( HT(G) \), then there exist \( t'g \in H \) for some term \( t' \) and some \( g \in G \) with \( HT(t'g) = t \)

**Output**: \( T \) is the subset of \( T(H') \) of all the terms divisible by an element of \( HT(G) \)

\[
T = \text{Done} = \{ t \in HT(H) \mid t = HT(t'g) \text{ with } t' \text{ a term, } g \in G \text{ and } t'g \in H \}
\]

while \( T(H) \neq \text{Done} \)

let \( m \) be an element of \( T(H) \) \( \setminus \) \( \text{Done} \)

\( \text{Done} = \text{Done} \cup \{ m \} \)

if \( m \) is top reducible by \( G \) then

let \( m' \) be a term and \( f \in G \) s.t. \( m = m' \ast HT(f) \)

\[
H = H \cup \{ m' \ast f \}
\]

\[
T = T \cup \{ HT(m' \ast f) \}
\]

return \((H, T)\)

Formally, we will call this new algorithm the MutantF4 Gröbner basis algorithm: the MF4GB algorithm.

**Remark 1.** The MF4GB algorithm cannot be realized as a selection strategy of the F4 algorithm. One way to see this is the fact that an element in \( \tilde{H}^- \) may never be included in the basis or be used to form pairs, whereas in the F4 algorithm the same polynomial would inevitably follow that path.

**Remark 2.** On an actual implementation, it is recommended to process \( \tilde{H}^- \) one degree at a time according to the degree of the step which is given by the degree of the critical pairs in \( P^* \).

**Remark 3.** What we have presented here is just the general idea of how to use mutants (low degree polynomials) in order to improve the F4 algorithm. The idea of exploiting mutants could be further refined, and there are several variants and possible improvements that arise from the basic MF4GB algorithm presented above. In particular we point out one, which has shown very good results. Let \( d_0 \) be the degree of the input basis \( F \). If the set of mutants \( \mathfrak{M} \) has degree one, the elements of degree higher than \( d_0 \) are discarded from the current basis \( G \) and a full reduction of the basis is undertaken, which accounts to eliminating as many variables as there are polynomials in \( \mathfrak{M} \).
The termination and correctness of the MutantF4 algorithm follows easily from the termination and correctness of the F4 algorithm, so we state the following corollary to theorem 2.2 in [7].

**Corollary 3.** The MutantF4 algorithm computes a Gröbner basis for \( I(f_1, \ldots, f_n) \).

The main reason why the new algorithm can work more efficiently is that the mutants are given preference in the computation process. They may actually reduce other “new polynomials” of high degree before these “new polynomials” become part of the basis and critical pairs are formed with them. By doing so, the computation of the Gröbner basis avoids the reduction of high degree polynomials from these critical pairs and the algorithm can finish at a lower degree. We will illustrate this with a concrete example, which comes from the MFE multivariate public key cryptosystems.

4. MFE and its variations

We will perform an experiment with the polynomial system derived from a simplified version of the MFE multivariate public key cryptosystem [9]. First, we will give a brief introduction of the MFE cryptosystem. The reason why we chose this example is that the attack on MFE [5] very much inspired the idea of mutant [4].

We will use the same notation as in [9] in this section. The MFE cryptosystem first chooses a finite field \( K \) of size \( 2^q \). Then it selects \( L \) to be a degree \( r \) extension of this field \( K \). The key construction is an invertible map, the central map,

\[
\phi_2 : L^{12} \rightarrow L^{15}.
\]

Then it is identified as a map \( \mathbb{K}^{12r} \rightarrow \mathbb{K}^{15r} \) via the field extension. The public key is derived from composing two randomly chosen affine maps on either side of \( \phi_2 \).

Our polynomials are derived as the public key polynomial minus the value of a randomly chosen ciphertext to ensure that the equations indeed have a solution. To simplify the experiment and its expositions, we choose \( K \) to be \( F_{2^r} \) instead of \( F_{2^{16}} \), and \( r = 2 \) instead of \( r = 4 \) or 5 as in the original MFE cryptosystems. This means we will experiment on a set of 30 quadratic polynomials with 24 variables over \( F_{2^r} \). The details of how these polynomials are built can be found in [9].

5. Experimental results

Here we compare the performance of the F4 algorithm against the MutantF4 algorithm in computing a Gröbner basis for a MFE system. Rather than raw computing time of several experiments, we present some data from the internal process of a single run. With this approach we avoid implementation factors which are not of concern for this paper. The choice of the random part of the MFE system does not affect the computation significantly so the data shown below can be thought as the typical data produced in the computation of a Gröbner basis.
for such a set of polynomials. To achieve a fair comparison, we will compare our results with that of the current version of Magma.

The results are in the table below, where the degree of the step refers to the degree of the critical pairs in $P^*$. The size of the matrix refers to the matrix interpretation of $H$ after symbolic preprocessing and before the row reduction. This choice obeys the fact that the bottle neck of the F4 algorithm is the computation of the row echelon form of the biggest matrix, so that a significant reduction of its size accounts to a significant speed up. In addition to this information, for the MutantF4 algorithm we include the number of polynomials inserted at each step and their degree (column labeled #new/deg).

For an F4 implementation, we used the latest version of Magma V2.14-10 (released 2008-02-15), which produces results consistent with our own implementation of F4. For the MutantF4 algorithm we used a basic implementation of the algorithm presented above and the one with variant as in Remark 3. Our implementation is basic in the sense that it is not at all optimized. However, a check for the Buchberger criteria as stated in [2] is included. These results are a clear indication that the MF4GB could really work better.

<table>
<thead>
<tr>
<th>Step</th>
<th>Deg</th>
<th>F4 Matrix size</th>
<th>MutantF4 Matrix size</th>
<th>#new/deg</th>
<th>MutantF4 Variant Matrix size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>30 × 140</td>
<td>2 × 58 × 140</td>
<td>29 / 2</td>
<td>2 × 58 × 140</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>587 × 1861</td>
<td>3 × 608 × 1861</td>
<td>128 / 3</td>
<td>3 × 608 × 1861</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>7104 × 14966</td>
<td>4 × 7247 × 14118</td>
<td>68 / 2*</td>
<td>4 × 7247 × 14118</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2146 × 2284</td>
<td>3 × 2274 × 2233</td>
<td>16 / 1*</td>
<td>3 × 2274 × 2233</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>338 × 273</td>
<td>2 × 325 × 272</td>
<td>15 / 2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2276 × 1833</td>
<td>3 × 2087 × 1879</td>
<td>5 / 2*</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>81 × 46</td>
<td>3 × 698 × 649</td>
<td>12 / 2*</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>50 × 27</td>
<td>3 × 678 × 484</td>
<td>8 / 1*</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>9196 × 8842</td>
<td>2 × 277 × 189</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*These are mutants, because the degree is less that the degree step.

Table 1. Performance of F4, MutantF4 and MutantF4 variant with public key of MFE.

We also did tests with the polynomials coming from MFE, but with the two affine transformations set to be identity maps, and it confirms our claim. At the beginning of the project, we used Magma version 2.11-11, and our results are even more impressive compared to this case. These experimental results are in the appendix.

6. Conclusion and Discussion

From the experiments above, it is clear that the MF4GB algorithm can work more efficiently for certain families of polynomial systems, namely the polynomial
systems which have a lot of hidden mutants in the ideal according to certain generators of the ideal. We do not at all claim that the MF4GB algorithm is always better than the F4 algorithm. For example, in the case of homogeneous polynomial system over characteristic zero, there should be no mutants at all, and therefore the MF4BG algorithm will simply degenerate into the usual F4 algorithm. From the concept of mutants, we also conclude that the homogeneous equations are much harder to solve than the non-homogeneous equations. The overall conclusion is that the MF4GB can not do worse than F4, but could do much better.

As we explained before, the mutant strategy is a general method and it can be used to improve various algorithms. Surely we are only at the beginning of exploring such a strategy and we believe that there is great potential. We are currently in the process to combine the mutant idea into F5 algorithm, which we believe also has great potential in improving the efficiency of F5.

7. Acknowledgments

We would like to thank Bo-yin Yang, Ralf-Philipp Weinmann and Andrei Pychkin for useful discussions.

References


Appendix A. Additional experimental results

<table>
<thead>
<tr>
<th>Step</th>
<th>Deg</th>
<th>Matrix size</th>
<th>Deg</th>
<th>Degree</th>
<th>Matrix size</th>
<th>#new/deg</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5 x 42</td>
<td>2</td>
<td>8 x 42</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>471 x 1450</td>
<td>3</td>
<td>491 x 1449</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>6384 x 12659</td>
<td>4</td>
<td>6569 x 12701</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2104 x 2239</td>
<td>3</td>
<td>2258 x 2202</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>338 x 273</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2276 x 1833</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>81 x 46</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>50 x 27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>9196 x 8842</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>0 x 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>0 x 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Performance of F4 and MutantF4 with MFE central map.

<table>
<thead>
<tr>
<th>Step</th>
<th>Size initial set</th>
<th>S-Pairs of min degree</th>
<th>Matrix size</th>
<th>Polys to insert at next step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
<td>4 of degree 2</td>
<td>5 x 42</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>34</td>
<td>128 of degree 3</td>
<td>471 x 1450</td>
<td>128</td>
</tr>
<tr>
<td>3</td>
<td>162</td>
<td>1237 of degree 4</td>
<td>6438 x 12713</td>
<td>491</td>
</tr>
<tr>
<td>4</td>
<td>653</td>
<td>840 of degree 3</td>
<td>2128 x 2263</td>
<td>376</td>
</tr>
<tr>
<td>5</td>
<td>1029</td>
<td>109 of degree 2</td>
<td>338 x 273</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>1048</td>
<td>647 of degree 3</td>
<td>2276 x 1833</td>
<td>48</td>
</tr>
<tr>
<td>7</td>
<td>1096</td>
<td>36 of degree 2</td>
<td>81 x 46</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1096</td>
<td>128 of degree 3</td>
<td>117 x 65</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1096</td>
<td>7926 of degree 4</td>
<td>10504 x 9880</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1096</td>
<td>5539 of degree 5</td>
<td>45066 x 44878</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1096</td>
<td>198 of degree 6</td>
<td>0 x 0</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Performance of F4 Version 2.11-11 with MFE central map.

Jintai Ding
University of Cincinnati, Cincinnati, OH, 45220, USA
e-mail: ding@math.uc.edu
Daniel Cabarcas  
University of Cincinnati, Cincinnati, OH, 45220, USA  
e-mail: cabarcas@gmail.com

Dieter Schmidt  
University of Cincinnati, Cincinnati, OH, 45220, USA  
e-mail: dieter.schmidt@uc.edu

Johannes Buchmann  
Technische Universität Darmstadt, Darmstadt, Germany  
e-mail: buchmann@cdc.informatik.tu-darmstadt.de

Stefan Tohaneanu  
University of Cincinnati, Cincinnati, OH, 45220, USA  
e-mail: tohanesn@ucmail.uc.edu