Finite Fields and Their Applications 41 (2016) 174–192

Contents lists available at ScienceDirect

Finite Fields and Their Applications

www.elsevier.com/locate/ffa

An improvement over the GVW algorithm for inhomogeneous polynomial systems ☆

Yao Sun [a], Zhenyu Huang [a,*], Dingkang Wang [b], Dongdai Lin [a]

[a] *SKLOIS, Institute of Information Engineering, CAS, Beijing 100093, China*
[b] *KLMM, Academy of Mathematics and Systems Science, CAS, Beijing 100190, China*

ARTICLE INFO

Article history:
Received 18 August 2015
Received in revised form 2 March 2016
Accepted 14 June 2016
Communicated by S. Gao

MSC:
13P10
13P15
11T55
68W30

Keywords:
Gröbner basis
The GVW algorithm
Signature-based algorithm
Linear algebra
Boolean polynomial ring

ABSTRACT

The GVW algorithm provides a new framework for computing Gröbner bases efficiently. If the input system is not homogeneous, some J-pairs with larger signatures but lower degrees may be rejected by GVW's criteria, and instead, GVW has to compute some J-pairs with smaller signatures but higher degrees. Consequently, degrees of polynomials appearing during the computations may unnecessarily grow up higher, and hence, the total computations become more expensive. This phenomenon happens more frequently when the coefficient field is a finite field and the field polynomials are involved in the computations. In this paper, a variant of the GVW algorithm, called M-GVW, is proposed. The concept of *mutant pairs* is introduced to overcome the inconveniences brought by inhomogeneous inputs. In aspects of implementations, to obtain efficient implementations of GVW/M-GVW over boolean polynomial rings, we take advantages of the famous library M4RI. We propose a new *rotating swap* method of adapting efficient routines in M4RI to deal with the one-direction reductions in GVW/M-GVW. Our implementations are tested with many examples from Boolean polynomial rings, and the timings show M-GVW usually

☆ The authors are supported by National Key Basic Research Program of China (No. 2013CB834203), National Natural Science Foundation of China (No. 11301523 and No. 61502485), the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDA06010701), and IEE's Research Project on Cryptography (No. Y4Z0061A02).

* Corresponding author.
E-mail addresses: sunyao@iie.ac.cn (Y. Sun), huangzhenyu@iie.ac.cn (Z. Huang), dwang@mmrc.iss.ac.cn (D. Wang), ddlin@iie.ac.cn (D. Lin).

http://dx.doi.org/10.1016/j.ffa.2016.06.002

performs much better than the original GVW algorithm if mutant pairs are found.

1. Introduction

Gröbner bases, proposed by Buchberger in 1965 [5], have been proven to be very useful in many aspects of algebra. In the past forty years, many efficient algorithms have been proposed to compute Gröbner bases. One important improvement is that Lazard pointed out the strong relation between Gröbner bases and linear algebra [23]. This idea has been implemented in F4 by Faugère [15], and also as XL type algorithms by Courtois et al. [7] and Ding et al. [9].

Faugère introduced the concept of signatures for polynomials and presented the famous F5 algorithm [16]. Since then, signature-based algorithms have been widely investigated, and several variants of F5 have been presented, including F5C [11], extended F5 [22], F5 with revised criterion (the AP algorithm) [4], and RB [13]. Gao et al. proposed another signature-based algorithm G2V [19] in a different way from F5, and GVW [20,21] is an extended version of G2V. The authors studied generalized criteria and signature-based algorithms in solvable polynomial algebra in [26,27]. For an overview of all signature-based algorithms, readers are referred to the survey by Eder and Faugère [14].

For implementations of signature-based algorithms, Roune and Stillman efficiently implemented GVW and AP without using linear algebra [24]. Faugère gave the matrix-F5 algorithm in [18]. An F5 in F4-style was described in more details by Albrecht and Perry [1].

In GVW, criteria always reject J-pairs with larger signatures, and process J-pairs with smaller signatures instead. Unfortunately, when the input systems are inhomogeneous, J-pairs with larger signatures probably have relatively lower degrees, where by saying degrees of polynomials, we mean the total degrees of polynomials. This phenomenon happens more frequently when the coefficient field is a finite field and the field polynomials are involved in the computations. This is not good for efficient Gröbner basis computations, because rejecting polynomials of lower degrees and reducing those of higher degrees will take more computing time. The case may be even worse if linear algebra is used for reductions. As suggested by Faugère in [15,16], a good strategy of dealing with critical pairs (equivalent to J-pairs in GVW) in a batch is to select all critical pairs with the minimal degree. So if polynomials with larger signatures and lower degrees are rejected by criteria, matrices with higher degrees may be constructed instead, which definitely leads to more computations. In fact, this case really happens when we are computing Gröbner bases for the HFE systems by using GVW. Some other influences of inhomogeneous input systems were discussed by Eder [12].

According to our observations, GVW's criteria may reject J-pairs with larger signatures and lower degrees. We believe such phenomenons are caused by a kind of *mutant pairs*, which will be defined in Section 3. On seeing this, we propose a variant algorithm of GVW, called M-GVW, to deal with mutant pairs. In M-GVW, criteria are not applied to mutant pairs anymore, such that J-pairs with larger signatures and lower degrees will not be rejected. Please note that for homogeneous polynomial systems, M-GVW is exactly the GVW algorithm since no mutant pairs are generated.

We implemented both the original GVW and M-GVW over Boolean polynomial rings. On eliminations of matrices, we hope to take advantages of fast arithmetics for dense matrices over GF(2) provided by the library M4RI [3]. However, reductions in signature-based algorithms must be done in one direction, i.e. rows with higher signatures can only be eliminated by rows with lower signatures. So routines from M4RI cannot be used directly. We present a method of doing such one-direction eliminations for dense matrices by modifying routines from M4RI. The experimental results show M-GVW usually performs much better than the original GVW algorithm if mutant pairs are found.

This paper is organized as follows. In Section 2, we introduce some necessary notations and revisit the GVW algorithm. In Section 3, M-GVW is presented. In Section 4, we discuss some details on implementing M-GVW over Boolean polynomial rings and show how routines from M4RI are modified to do one-direction eliminations. Some experimental results are shown in Section 5. Concluding remarks follow in Section 6.

2. The GVW algorithm revisited

Most of notations and definitions are inherited from Gao et al.'s original paper. For more details, please see [20,21].

Let $R = \mathrm{K}[x_1, \ldots, x_n]$ be a polynomial ring over a field K with n variables, and $\{f_1, \cdots, f_m\}$ is a finite subset of R. We want to compute a Gröbner basis for the ideal

$$I = \langle f_1, \cdots, f_m \rangle = \{p_1 f_1 + \cdots + p_m f_m \mid p_1, \cdots, p_m \in R\}$$

with respect to some monomial ordering on R.

Let $\mathbf{F} = (f_1, \cdots, f_m) \in R^m$, and consider the following R-module of $R^m \times R$:

$$\mathbf{M} = \{(\mathbf{u}, f) \in R^m \times R \mid \mathbf{u} \cdot \mathbf{F} = f\}.$$

Let $\mathbf{e}_i$ be the i-th unit vector of R^m, i.e. $(\mathbf{e}_i)_j = \delta_{ij}$ where δ_{ij} is the Kronecker delta. Then the R-module $\mathbf{M}$ is generated by $\{(\mathbf{e}_1, f_1), \cdots, (\mathbf{e}_m, f_m)\}$.

A monomial in R has the form $x^\alpha = \Pi_{i=1}^n x_i^{a_i}$, where $\alpha = (a_1, \ldots, a_n) \in \mathbb{N}^n$ and $\mathbb{N}$ is the set of all non-negative integers. A monomial in R^m is of the form $x^\alpha \mathbf{e}_i$, where $1 \leq i \leq m$ and $\alpha \in \mathbb{N}^n$. For monomials in R^n, we say $x^\alpha \mathbf{e}_i$ divides $x^\beta \mathbf{e}_j$ (or $x^\alpha \mathbf{e}_i \mid x^\beta \mathbf{e}_j$ for short), if $i = j$ and x^α divides x^β, and the quotient is defined as $(x^\beta \mathbf{e}_i)/(x^\alpha \mathbf{e}_i) = x^{\beta-\alpha} \in R$.

Fix any monomial ordering $\prec_p$ on R and any monomial ordering $\prec_s$ on R^m (subscripts p and s stand for *polynomial* and *signature* respectively). Please note that $\prec_s$ may or may not be related to $\prec_p$ in theory, although we always assume $\prec_s$ is **compatible** with $\prec_p$ practically, i.e. $x^\alpha \prec_p x^\beta$ if and only if $x^\alpha \mathbf{e}_i \prec_s x^\beta \mathbf{e}_i$ for $1 \leq i \leq m$. To make descriptions simpler, we use the following notations for leading monomials:

$$\mathrm{lm}(f) = \mathrm{lm}_{\prec_p}(f) \text{ and } \mathrm{lm}(\mathbf{u}) = \mathrm{lm}_{\prec_s}(\mathbf{u}),$$

for any $f \in R$ and any $\mathbf{u} \in R^m$. Leading monomials of $f \in R$ and $\mathbf{u} \in R^m$ are monomials without coefficients in R and R^m respectively. We define $\mathrm{lm}(f) = 0$ if $f = 0$, and $0 \prec_p x^\alpha$ for any non-zero monomial x^α in R; similarly for monomials in R^m. In the rest of this paper, we use $\prec$ to represent $\prec_p$ and $\prec_s$ for short, if no confusion occurs.

For a pair $(\mathbf{u}, f) \in \mathbf{M}$, $\mathrm{lm}(\mathbf{u})$ is called the **signature** of $(\mathbf{u}, f)$. This definition is the same as that used in GVW, but different from those used in [16,4]. The differences are discussed in [20,21].

Let $(\mathbf{u}, f) \in \mathbf{M}$ and $B \subset \mathbf{M}$, we say $(\mathbf{u}, f)$ is **top-reducible** by B, if there exists $(\mathbf{v}, g) \in B$ with $g \neq 0$, such that $\mathrm{lm}(g)$ divides $\mathrm{lm}(f)$ and $\mathrm{lm}(\mathbf{u}) \succeq \mathrm{lm}(t\mathbf{v})$ where $t = \mathrm{lm}(f)/\mathrm{lm}(g)$. The corresponding **top-reduction** is then

$$(\mathbf{u}, f) - ct(\mathbf{v}, g) = (\mathbf{u} - ct\mathbf{v}, f - ctg),$$

where $c = \mathrm{lc}(f)/\mathrm{lc}(g)$ and $\mathrm{lc}(f)$ denotes the leading coefficient of f. Particularly, this top-reduction is called **regular**, if $\mathrm{lm}(\mathbf{u}) \succ \mathrm{lm}(t\mathbf{v})$; and **super** if $\mathrm{lm}(\mathbf{u}) = \mathrm{lm}(t\mathbf{v})$.[1] Clearly, $(\mathbf{u} - ct\mathbf{v}, f - ctg)$ is also an element in $\mathbf{M}$.

A subset G of $\mathbf{M}$ is called a **strong Gröbner basis** for $\mathbf{M}$ if every nonzero pair (pairs $\neq (\mathbf{0}, 0)$) in $\mathbf{M}$ is top-reducible by G. By Proposition 2.2 of [20,21], let $G = \{(\mathbf{v}_i, g_i) \mid 1 \leq i \leq s\}$ be a strong Gröbner basis for $\mathbf{M}$. Then $\{g_i : 1 \leq i \leq s\}$ is a Gröbner basis for $I = \langle f_1, \ldots, f_m \rangle$.

Next, we define *joint pairs/J-pairs*. Suppose $(\mathbf{u}, f), (\mathbf{v}, g) \in \mathbf{M}$ are two pairs with f and g both nonzero. Let $t = \mathrm{lcm}(\mathrm{lm}(f), \mathrm{lm}(g))$, $t_f = t/\mathrm{lm}(f)$ and $t_g = t/\mathrm{lm}(g)$. Then the **J-pair** of $(\mathbf{u}, f)$ and $(\mathbf{v}, g)$ is defined as: $t_f(\mathbf{u}, f)$ (or $t_g(\mathbf{v}, g)$), if $\mathrm{lm}(t_f\mathbf{u}) \succ \mathrm{lm}(t_g\mathbf{v})$ (or $\mathrm{lm}(t_f\mathbf{u}) \prec \mathrm{lm}(t_g\mathbf{v})$). For the case $\mathrm{lm}(t_f\mathbf{u}) = \mathrm{lm}(t_g\mathbf{v})$, the J-pair is not defined. Note that the J-pair of $(\mathbf{u}, f), (\mathbf{v}, g) \in \mathbf{M}$ is also a pair in $\mathbf{M}$. Assume $t_f(\mathbf{u}, f)$ is the J-pair of $(\mathbf{u}, f)$ and $(\mathbf{v}, g)$, the **degree** of $t_f(\mathbf{u}, f)$ is defined as $\deg(t_f f)$, i.e. the degree of the polynomial part. For convenience, we call a J-pair is of $G \subset \mathbf{M}$, if it is the J-pair of two pairs in G.

For a pair $(\mathbf{u}, f) \in \mathbf{M}$ and a set $G \subset \mathbf{M}$, we say $(\mathbf{u}, f)$ is **covered** by G, if there is a pair $(\mathbf{v}, g) \in G$, such that $\mathrm{lm}(\mathbf{v})$ divides $\mathrm{lm}(\mathbf{u})$ and $t\mathrm{lm}(g) \prec \mathrm{lm}(f)$ (strictly smaller) where $t = \mathrm{lm}(\mathbf{u})/\mathrm{lm}(\mathbf{v})$.

Gao, Volny, and Wang give a simple characterization of strong Gröbner bases.

[1] Regular top-reduction defined here is slightly different from its original version in [20,21], but this will not affect proofs of related propositions and theorems.

Algorithm 1: The GVW algorithm.

Input : $f_1, \ldots, f_m \in R = \mathrm{K}[x_1, \ldots, x_n]$, monomial orderings for R and R^m.
Output: A Gröbner basis of $I = \langle f_1, \ldots, f_m \rangle$.

```
1  begin
2      H ⟵ {lm(f_j e_i − f_i e_j) | 1 ≤ i, j ≤ m}
3      G ⟵ {(lm(e_i), f_i) | 1 ≤ i ≤ m}
4      JPairSet ⟵ all J-pairs of G
5      while JPairSet ≠ ∅ do
6          Let t(x^α e_i, f) ∈ JPairSet and remove t(x^α e_i, f) from JPairSet.
7          if t x^α e_i is divisible by some monomial in H (Syzygy Criterion) or t(x^α e_i, f) is covered by
           G (Rewriting Criterion) then
8              GotoLine 5
9          (x^γ e_i, h) ⟵ Regular top-reduce (t x^α e_i, tf) by G.
10         if h = 0 then
11             H ⟵ H ∪ {x^γ e_i}
12         else
13             for (x^β e_j, g) ∈ G s.t. lm(g) x^γ e_i ≠ lm(h) x^β e_j do
14                 H ⟵ H ∪ {max(lm(g) x^γ e_i, lm(h) x^β e_j)}
15                 JPairSet ⟵ JPairSet ∪ {J-pair of (x^γ e_i, h) and (x^β e_j, g)}
16             G ⟵ G ∪ {(x^γ e_i, h)}
17     return {g | (x^β e_j, g) ∈ G}
```

Theorem 2.1 *(Gao–Volny–Wang [21]). Suppose $G \subset \mathbf{M}$ contains pairs with signatures $\{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$. Then G is a strong Gröbner basis for $\mathbf{M}$ if and only if every J-pair of G is covered by G.*

Note that checking whether a pair is covered by G does not need any reduction of polynomials. This immediately leads to the various rewrite rules used in the literature. Indeed, the rewrite rules can be rephrased as the following two criteria.

[Syzygy Criterion] For a J-pair $t_f(\mathbf{u}, f)$ of a set $G \in \mathbf{M}$, if there exists $(\mathbf{v}, 0) \in G$ such that $\mathrm{lm}(\mathbf{v})$ divides $t_f \mathrm{lm}(\mathbf{u})$, then this J-pair can be discarded.

[Second Criterion] For a J-pair of a set $G \in \mathbf{M}$, if this J-pair is covered by G, then this J-pair can be discarded.

In this paper, we call the second criterion **Rewriting Criterion**. Arri and Perry proposed a quite similar criterion to Rewriting Criterion in [4]. Comments on Arri–Perry's criterion and Rewriting Criterion can be found in [20,21,24].

The following GVW algorithm is slightly modified from its original version (see Algorithm 1). We delete the output of a Gröbner basis for the syzygy module of input polynomials, because we only care about the Gröbner basis of input polynomials in current paper. We emphasize that for a pair $(\mathbf{u}, f) \in \mathbf{M}$, only $(\mathrm{lm}(\mathbf{u}), f)$ is stored in the latest version of the GVW algorithm. Related conceptions, such as *top-reduction, J-pairs* and *cover*, are defined similarly. Please see [20,21] for more details.

There are some remarks on the GVW algorithm.

1. At Step 6, a J-pair can be selected from the set `JPairSet` in any order. In Section 4, we prefer to choosing J-pairs with minimal degrees first.
2. Proposition 2.2 in [20,21] ensures the correctness of GVW when J-pairs are computed in any order. The finite termination of GVW is proved by Theorem 3.1 in [20,21] when monomial orderings of R and R^m are compatible.
3. The GVW algorithm in [20,21] retains only one J-pair (the one with the minimal polynomial part) when there are several J-pairs having the same signature. This process can be implied by the "cover check" at step 7.

3. The M-GVW algorithm

3.1. Motivations and main ideas

The motivation of varying GVW arises from our implementation of GVW by using linear algebra for reductions. To control the size of matrices as small as possible during the computations, we deal with the J-pairs with the *minimal degree* first. That is, at Step 6 of GVW, we find the minimal degree of all J-pairs in `JPairSet` first, and then choose the J-pair with the smallest signature among the J-pairs with the minimal degree.

However, we are quite surprised to find that when computing a Gröbner basis for the HFE_25_96 system (from [25]), the degrees of matrices always grow up to 5. This phenomenon makes the efficiency of our implementation very poor, because the sizes of 5-degree matrices are much larger than those of degree 4. Here the degree of a matrix is the maximal degree of the polynomials used to construct this matrix. But it has been shown in [17] that the Gröbner basis of this example can definitely be obtained from matrices whose degree is not bigger than 4. So we believe there are some peculiar cases not considered in GVW when the input systems are inhomogeneous. We also notice that, this phenomenon has no relations to the computing orders of J-pairs.

We can illustrate this phenomenon clearly by the following example.

Example 3.1. Let $\{f_1, f_2, \ldots, f_{11}\} \subset R = \mathbb{F}_2[x_1, x_2, \ldots, x_9]$, where $\mathbb{F}_2$ is the Galois Field $GF(2)$, and

$$f_1 = x_1x_2x_5x_6 + x_2x_3x_7x_9 + x_7, f_2 = x_1x_2x_6x_8 + x_3x_4x_7,$$
$$f_{i+2} = x_i^2 + x_i, \text{ for } 1 \leq i \leq 9.$$

Monomial ordering $\prec_p$ in R is the Graded Reverse Lexicographic ordering, and $\prec_s$ in R^{11} is a position over term extension of $\prec_p$:

$$x^\alpha \mathbf{e}_i \prec_s x^\beta \mathbf{e}_j \text{ iff } i > j, \text{ or } i = j \text{ and } x^\alpha \prec_p x^\beta.$$

Thus, we have $\mathbf{e}_1 \succ \mathbf{e}_2 \succ \cdots \succ \mathbf{e}_{11}$.

For this example, GVW needs to process J-pairs of degrees bigger than 5 before a Gröbner basis is obtained. But the maximal degree of matrices in the F4 algorithm using criteria of [6] is 5. This implies some "useful" J-pairs with degrees not bigger than 5 have been rejected by GVW's criteria.

Now, we discuss this example in details. We compute a Gröbner basis for $\langle f_1, \ldots, f_{11}\rangle$ by GVW with the following strategy for selecting J-pairs from the set `JPairSet`:

1. $d \longleftarrow$ the *minimal* degree of J-pairs in the set `JPairSet`.
2. $t(x^\alpha \mathbf{e}_i, f) \longleftarrow$ J-pair with the *smallest* signature in $\{t(x^\alpha \mathbf{e}_i, f) \in \texttt{JPairSet} \mid \deg(tf) = d\}$.

Since the f_i's are all inhomogeneous, the above strategy implies that J-pairs are not handled in an increasing order on signatures.[2]

Initially, we have $G = \{(\mathbf{e}_1, f_1), (\mathbf{e}_2, f_2), \ldots, (\mathbf{e}_{11}, f_{11})\}$.

Before processing J-pairs of degree 6, the following polynomials are generated one by one:

$$
\begin{aligned}
&(x_8\mathbf{e}_2, f_{12} = x_3x_4x_7x_8 + x_3x_4x_7),\\
&(x_6\mathbf{e}_2, f_{13} = x_3x_4x_6x_7 + x_1x_2x_6x_8),\\
&(x_2\mathbf{e}_2, f_{14} = x_2x_3x_4x_7 + x_1x_2x_6x_8),\\
&(x_1\mathbf{e}_2, f_{15} = x_1x_3x_4x_7 + x_1x_2x_6x_8),\\
&(x_8\mathbf{e}_1, f_{16} = x_2x_3x_7x_8x_9 + x_3x_4x_5x_7 + x_7x_8),\\
&(x_6\mathbf{e}_1, f_{17} = x_2x_3x_6x_7x_9 + x_1x_2x_5x_6 + x_6x_7),\\
&(x_5\mathbf{e}_1, f_{18} = x_2x_3x_5x_7x_9 + x_1x_2x_5x_6 + x_5x_7),\\
&(x_2\mathbf{e}_1, f_{19} = x_2x_7 + x_7),\\
&(x_1\mathbf{e}_1, f_{20} = x_1x_2x_3x_7x_9 + x_1x_2x_5x_6 + x_1x_7),\\
&(x_2x_3x_8x_9\mathbf{e}_1, f_{21} = x_3x_4x_5x_7 + x_3x_7x_8x_9 + x_7x_8),\\
&(x_2x_3x_6x_9\mathbf{e}_1, f_{22} = x_3x_6x_7x_9 + x_3x_7x_9 + x_6x_7 + x_7),\\
&(x_2x_3x_5x_9\mathbf{e}_1, f_{23} = x_3x_5x_7x_9 + x_3x_7x_9 + x_5x_7 + x_7),\\
&(x_1x_2x_3x_9\mathbf{e}_1, f_{24} = x_1x_3x_7x_9 + x_2x_3x_7x_9 + x_1x_7 + x_7).
\end{aligned}
$$

During the computations, many leading monomial of syzygies in $\mathbf{M}$ have been generated, among them the one $x_2x_3x_4\mathbf{e}_1$ (obtained before f_{20}) is important, since it has been used to reject many other J-pairs.

[2] Even if J-pairs are processed in an increasing order on signatures, GVW still has to reduce J-pairs with degrees bigger than 5 before a Gröbner basis is obtained.

So far, all J-pairs with degrees not bigger than 5 have been considered. It is easy to check $\{f_1, f_2, \ldots, f_{24}\}$ is *not* a Gröbner basis of $\langle f_1, f_2, \ldots, f_{11}\rangle$. However, for the same ideal, F4 using criteria from [6] can obtain a Gröbner basis without computing any critical pairs of degrees bigger than 5. Comparing GVW and F4 step by step, we finally find the following J-pairs:

$$\begin{aligned}
&x_4(x_2x_3x_5x_9\mathbf{e}_1, f_{23}), && x_4(x_1x_2x_3x_9\mathbf{e}_1, f_{24}),\\
&x_4(x_2x_3x_6x_9\mathbf{e}_1, f_{22}), && x_1x_5x_6(x_2\mathbf{e}_1, f_{19}),\\
&x_1x_6x_8(x_2\mathbf{e}_1, f_{19}), && x_3(x_2x_3x_8x_9\mathbf{e}_1, f_{21}),\\
&x_4(x_2x_3x_8x_9\mathbf{e}_1, f_{21}), && x_3(x_1x_2x_3x_9\mathbf{e}_1, f_{24}),\\
&x_9(x_1x_2x_3x_9\mathbf{e}_1, f_{24}), && x_3(x_2x_3x_5x_9\mathbf{e}_1, f_{23}),\\
&x_9(x_2x_3x_5x_9\mathbf{e}_1, f_{23}), && x_3(x_2x_3x_6x_9\mathbf{e}_1, f_{22}),\\
&x_9(x_2x_3x_6x_9\mathbf{e}_1, f_{22}).
\end{aligned}$$

These J-pairs are rejected by GVW's criteria, but those corresponding critical pairs in F4 are not rejected by Buchberger's criteria.

Reducing these J-pairs, we get the following pairs:

$$(x_2x_3x_4x_5x_9\mathbf{e}_1, x_3x_4x_7x_9 + x_3x_7x_8x_9 + x_4x_5x_7 + x_7x_8x_9 + x_4x_7),$$

$$(x_1x_2x_3x_4x_9\mathbf{e}_1, x_1x_4x_7 + x_4x_7),$$

$$(x_2x_3x_4x_6x_9\mathbf{e}_1, x_4x_6x_7 + x_4x_7),$$

$$(x_1x_2x_5x_6\mathbf{e}_1, x_1x_5x_6x_7 + x_2x_3x_7x_9 + x_7^2),$$

$$(x_1x_2x_6x_8\mathbf{e}_1, x_1x_6x_7x_8 + x_3x_4x_7),$$

$$(x_2x_3^2x_8x_9\mathbf{e}_1, x_3x_7x_8 + x_7x_8),$$

$$(x_2x_3x_4x_8x_9\mathbf{e}_1, x_3x_4x_7x_9 + x_3x_7x_8x_9 + x_4x_7x_8 + x_7x_8),$$

$$(x_1x_2x_3^2x_9\mathbf{e}_1, x_1x_3x_7 + x_1x_7 + x_3x_7 + x_7),$$

$$(x_1x_2x_3x_9^2\mathbf{e}_1, x_1x_7x_9 + x_1x_7 + x_7x_9 + x_7),$$

$$(x_2x_3^2x_5x_9\mathbf{e}_1, x_3x_5x_7 + x_3x_7 + x_5x_7 + x_7),$$

$$(x_2x_3x_5x_9^2\mathbf{e}_1, x_5x_7x_9 + x_5x_7 + x_7x_9 + x_7),$$

$$(x_2x_3^2x_6x_9\mathbf{e}_1, x_3x_6x_7 + x_3x_7 + x_6x_7 + x_7),$$

$$(x_2x_3x_6x_9^2\mathbf{e}_1, x_6x_7x_9 + x_6x_7 + x_7x_9 + x_7).$$

There are 9 polynomials of degree 3 and 4 polynomials of degree 4. These polynomials are computed in F4, and prevent F4 to deal with critical pairs of degree bigger than 5.

However, these J-pairs are rejected by GVW's criteria, so GVW has to compute J-pairs of degree 6.

Next, we analyze why GVW is possible to reject J-pairs of lower degrees and prefer to reducing higher degree J-pairs. Take the J-pair $x_3(x_1x_2x_3x_9\mathbf{e}_1, f_{24})$ for example. Reducing this J-pair, we get

$$(x_1x_2x_3^2x_9\mathbf{e}_1, x_1x_3x_7 + x_1x_7 + x_3x_7 + x_7).$$

But this J-pair is rejected by $((x_3^2+x_3)\mathbf{e}_1 - f_1\mathbf{e}_5, 0) \in \mathbf{M}$ in GVW, which is the principal syzygy of f_1 and f_5. After GVW finishing all computations, we find the polynomial $x_1x_3x_7 + x_1x_7 + x_3x_7 + x_7$ is obtained from reducing the J-pair $x_3(x_1\mathbf{e}_1, f_{20})$, whose degree is 6 and whose signature is smaller. Combined with our experiences of proving F5 in [28], we have the following observation.

Observation 3.2. GVW's criteria alway reject J-pairs with higher signatures, and process some J-pairs with smaller signatures instead.

When the input systems are inhomogeneous, J-pairs with bigger signatures may have lower degrees than J-pairs with smaller signatures.

Consider the 5-degree J-pair $x_3(x_1x_2x_3x_9\mathbf{e}_1, f_{24})$ again. The syzygy $((x_3^2 + x_3)\mathbf{e}_1 - f_1\mathbf{e}_5, 0) \in \mathbf{M}$, which rejects this J-pair, corresponds to the equation

$$f_5f_1 - f_1f_5 = 0,$$

in which monomials of degree 6 appear. Thus, we believe that if we use the syzygy $((x_3^2 + x_3)\mathbf{e}_1 - f_1\mathbf{e}_5, 0)$ to reject this 5-degree J-pair, it is possible to deal with some J-pairs involving polynomials of degree 6 instead later, since the degrees of f_5f_1 and f_1f_5 are 6. On seeing this, our idea is to prevent GVW from rejecting J-pairs like $x_3(x_1x_2x_3x_9\mathbf{e}_1, f_{24})$.

Analyzing all J-pairs we have listed earlier, we find they have one common property: for any J-pair $t(x^\alpha\mathbf{e}_1, f_j)$ that is listed, we have

$$\deg(x^\alpha) + \deg(f_1) > \deg(f_j).$$

This property makes the degree of $x_3(x_1x_2x_3x_9\mathbf{e}_1, f_{24})$ lower than the degree of f_5f_1. In order to prevent this J-pair to be rejected, we should treat the pair $(x_1x_2x_3x_9\mathbf{e}_1, f_{24})$ specially. We find such pairs are similar to mutant polynomials defined in [9], so we give the following definition.

Definition 3.3. Let $\mathbf{M}$ be an R-module generated by $\{(\mathbf{e}_1, f_1), \ldots, (\mathbf{e}_m, f_m)\}$. A pair $(\mathbf{u}, f) \in \mathbf{M}$ with $\mathrm{lm}(\mathbf{u}) = x^\alpha\mathbf{e}_i$ and $f \neq 0$, is called **mutant**, if $\deg(x^\alpha)+\deg(f_i) > \deg(f)$. Particularly, a J-pair $t(\mathbf{u}, f)$ is called mutant where t is a monomial, if $(\mathbf{u}, f)$ is mutant.

Algorithm 2: The M-GVW algorithm.

Input : $f_1, \ldots, f_m \in R = \mathrm{K}[x_1, \ldots, x_n]$, monomial orderings for R and R^m.
Output: A Gröbner basis of $I = \langle f_1, \ldots, f_m \rangle$.

```
1  begin
2      H ⟵ {lm(f_j e_i − f_i e_j) | 1 ≤ i, j ≤ m}
3      G ⟵ {(lm(e_i), f_i) | 1 ≤ i ≤ m}
4      JPairSet ⟵ all J-pairs of G
5      while JPairSet ≠ ∅ do
6          Let t(x^α e_i, f) ∈ JPairSet and remove t(x^α e_i, f) from JPairSet.
7          if (1) deg(x^α) + deg(f_i) = deg(f), and (2) either t x^α e_i is divisible by some monomial in
           H or t(x^α e_i, f) is covered by G then
8              GotoLine 5
9          (x^γ e_i, h) ⟵ Regular top-reduce (t x^α e_i, tf) by G.
10         if h = 0 then
11             H ⟵ H ∪ {x^γ e_i}
12         else
13             for (x^β e_j, g) ∈ G s.t. lm(g) x^γ e_i ≠ lm(h) x^β e_j do
14                 H ⟵ H ∪ {max(lm(g) x^γ e_i, lm(h) x^β e_j)}
15                 JPairSet ⟵ JPairSet ∪ {J-pair of (x^γ e_i, h) and (x^β e_j, g)}
16             G ⟵ G ∪ {(x^γ e_i, h)}
17     return {g | (x^β e_j, g) ∈ G}
```

Due to the existence of syzygy pairs, there are lots of mutant pairs in $\mathbf{M}$. But only a few of mutant pairs will appear in the practical computations.

Remark 3.4. We only consider mutant pairs when the ordering $\prec_p$ is a graded ordering.

3.2. The M-GVW algorithm

From Example 3.1, we see that applying Syzygy Criterion to mutant J-pairs can cause an unexpected rise of the computing degree of J-pairs. In more complicated examples, we find that applying Rewriting Criterion to mutant J-pairs can also cause such phenomenon. So the main idea of M-GVW is to avoid applying both criteria to mutant J-pairs.[3] This idea is simple, but it works very well. We give M-GVW in Algorithm 2.

Please note that the only difference between M-GVW and GVW is that, in M-GVW criteria are not applied to mutant J-pairs. Particularly, mutant pairs cannot be found in M-GVW when input systems are *homogeneous*. In this case, M-GVW is exactly the GVW algorithm.

Lemma 3.5 *(ⓐ and ⓒ of Thm. 2.4 in [20]). Suppose G is a subset of M such that, for any monomial $x^\alpha \mathbf{e}_i \in R^m$, there exists a pair $(\mathbf{v}, g) \in G$ and a monomial t such that $x^\alpha \mathbf{e}_i = t\,\mathrm{lm}(\mathbf{v})$. Then G is a strong Gröbner basis for M if and only if every J-pair of G is covered by G.*

[3] Another method of dealing with mutant J-pairs is to append them to the input polynomials. This method is discussed in the previous version of current paper, and it can be found in [30].

Algorithm 3: M-GVW in matrix style.

Input : $f_1, \ldots, f_m \in R = \mathrm{K}[x_1, \ldots, x_n]$, monomial orderings for R and R^m.
Output: A Gröbner basis of $I = \langle f_1, \ldots, f_m \rangle$.

begin
  $H \longleftarrow \{\mathrm{lm}(f_j \mathbf{e}_i - f_i \mathbf{e}_j) \mid 1 \le i, j \le m\}$
  $G \longleftarrow \{(\mathrm{lm}(\mathbf{e}_i), f_i) \mid 1 \le i \le m\}$
  JPairSet $\longleftarrow$ all J-pairs of G
  while JPairSet $\neq \emptyset$ **do**
    $d \longleftarrow$ the minimal degree of J-pairs in JPairSet
    Todo $\longleftarrow$ all J-pairs of degree d in JPairSet and remove Todo from JPairSet
    $P \longleftarrow$ `SymbolicProcess`(Todo, G, H) *(element in P has the form of $(x^\alpha \mathbf{e}_i, f)$)*
    $F \longleftarrow$ `Elimination`(P) *(element in F has the form of $(x^\gamma \mathbf{e}_i, h)$)*
    $F^+ \longleftarrow F \setminus \{$pairs that are super top-reducible by $G\}$
    for *each* $(x^\gamma \mathbf{e}_i, h) \in F^+$ *s.t.* $h = 0$ **do**
      $H \longleftarrow H \cup \{x^\gamma \mathbf{e}_i\}$
    for *each* $(x^\gamma \mathbf{e}_i, h) \in F^+$ *s.t.* $h \neq 0$ **do**
      for $(x^\beta \mathbf{e}_j, g) \in G$, $\mathrm{lm}(g) x^\gamma \mathbf{e}_i \neq \mathrm{lm}(h) x^\beta \mathbf{e}_j$ **do**
        $H \longleftarrow H \cup \{\max(\mathrm{lm}(g) x^\gamma \mathbf{e}_i, \mathrm{lm}(h) x^\beta \mathbf{e}_j)\}$
        JPairSet $\longleftarrow$ JPairSet $\cup \{$J-pair of $(x^\gamma \mathbf{e}_i, h)$, $(x^\beta \mathbf{e}_j, g)\}$
      $G \longleftarrow G \cup \{(x^\gamma \mathbf{e}_i, h)\}$
  return $\{g \mid (x^\beta \mathbf{e}_j, g) \in G\}$

Since if $t(x^\alpha \mathbf{e}_i, f)$ is a J-pair and is regular top-reduced to $(x^\gamma \mathbf{e}_i, h)$ by G, then $t(x^\alpha \mathbf{e}_i, f)$ is covered by $(x^\gamma \mathbf{e}_i, h)$. Using this lemma, we directly have the following theorem.

Theorem 3.6. *The M-GVW algorithm is correct.*

4. On implementing M-GVW over Boolean polynomial rings

In this section, we discuss some details on implementing M-GVW over Boolean polynomial rings. First, to speed up M-GVW by using linear algebra, we rewrite M-GVW in a matrix style in Subsection 4.1. Next, we briefly mention how Gröbner bases are computed over Boolean polynomial rings in Subsection 4.2. At last in Subsection 4.3, we show how to do one-direction reductions in M-GVW by using routines modified from M4RI.

4.1. M-GVW in a matrix style

The matrix version of M-GVW is similar to F4. This matrix-style algorithm is also similar to Albrecht–Perry's version [1], but has small differences. The main function is given in Algorithm 3.

The function `SymbolicProcess`(Todo, G, H) does three things (see Algorithm 4). First, check whether J-pairs are rejected by criteria. Second, for each monomial that is not a leading monomial, search polynomials from G to reduce it. Third, sort all pairs according their signatures. Please note that, here we do not remove the pairs whose sig-

Algorithm 4: SymbolicProcess.

Input : Todo, a set of J-pairs; G, a set of pairs; H, a set of monomials in R^m.
Output: P, each element in P has the form of $(x^\alpha \mathbf{e}_i, f)$.

```
1  begin
2      P ⟵ ∅
3      for each t(x^α e_i, f) in Todo do
4          if (1) deg(x^α) + deg(f_i) = deg(f), and (2) either tx^α e_i is divisible by some monomial in
            H or t(x^α e_i, f) is covered by G then
5              GotoLine 3
6          P ⟵ P ∪ {(tx^α e_i, tf)}
7      Done ⟵ ∅
8      while M(P) ≠ Done do
9          m ⟵ an element of M(P) \ Done
10         Done ⟵ Done ∪ {m}
11         if ∃(x^β e_j, g) ∈ G s.t. lm(g) | m then
12             P ⟵ P ∪ {(m/lm(g))(x^β e_j, g)}
13     Sort P by an increasing order on signatures.
14     return P
```

natures equal to those of some other pairs, because we cannot apply Rewriting Criterion to mutant J-pairs. Besides, the function `elimination()` is not affected if there are some pairs having the same signatures. Denote $M(P)$ be the union of the sets of all monomials in h for all $(x^\gamma \mathbf{e}_i, h) \in P$.

This function is a bit different from Albrecht–Perry's version [1]. First, any monomial m can be selected from $M(P) \setminus Done$, while in [1] the maximal one is selected each time. Second, for any selected monomial m, we do not need to know its signature information.

The function `Elimination`(P) does three things. First, it writes pairs in P as rows of a matrix. Second, it computes the echelon form of this matrix. Third, it reads polynomials from rows of this matrix. In the first step, please note that building matrices from Boolean polynomials is a bit different from building matrices from polynomials in general polynomial rings, because the product of a monomial and a Boolean polynomial should be reduced by field polynomials automatically. We report our method in [29] and omit details here. The second step is critical for efficiency. Since naive Gaussian eliminations seem less efficient, we hope to take advantages of the efficient routines from M4RI to improve the efficiencies. However, in signature-based algorithms, since rows with higher signatures can only be eliminated by rows with lower signatures, routines from M4RI can not do this one-direction elimination directly. So we use a special kind of row swaps to replace the original row swaps in the routines of M4RI. This new swapping method will be discussed in Subsection 4.3.

4.2. On computing Gröbner basis over Boolean polynomial rings

In the rest of this section, we discuss the implementations of M-GVW over Boolean polynomial rings.

The polynomial ring is specialized as $R = \mathbb{F}_2[x_1, x_2, \ldots, x_n]$ with n variables over the Galois Field $GF(2)$. Polynomials $E = \{x_1^2+x_1, \ldots, x_n^2+x_n\}$ are called field polynomials. Let $F = \{f_1, \ldots, f_m\}$ be a subset of R. Then computing a Gröbner basis for the ideal generated by F over the Boolean polynomial ring $R/\langle E\rangle$, is equivalent to computing a Gröbner basis for the ideal generated by $F \cup E$ over R. In our implementation, we aim to compute Gröbner bases for $F \cup E$ over R, so all the computations are done in R. Since field polynomials have quite simple forms, we do not need to store them in practical implementations. Besides, normal forms of polynomials in R w.r.t. E are also done automatically in the implementation.

We specialize the monomial ordering $\prec_p$ on R to be the *Graded Reverse Lexicographic ordering*. And the monomial ordering $\prec_s$ on modules is a *position over term extension* of $\prec_p$, such that $\mathbf{e}_1 \succ_s \mathbf{e}_2 \succ_s \cdots \succ_s \mathbf{e}_m \succ_s \mathbf{e}_{m+1} \succ_s \cdots \succ_s \mathbf{e}_{m+n}$, where $\mathbf{e}_i$ corresponds to f_i for $i = 1, 2, \ldots, m$, and $\mathbf{e}_{m+j}$ corresponds to $x_j^2 + x_j$ for $j = 1, 2, \ldots, n$. With this monomial ordering $\prec_s$, field polynomials can be used to reduce other polynomials all the time.

4.3. One-direction elimination

Unlike F4, eliminations in signature-based algorithm can only be done from one direction. Because each row of matrix in signature-based algorithms corresponds to a signature, and rows with higher signatures can only be reduced by rows with lower signatures. Naive Gaussian eliminations can control eliminating directions easily. But to take advantages of efficient elimination routines from the library M4RI [3], we should revise the swapping methods in these routines. We call our method *rotating swap*. This method is a bit similar to that in [10].

We illustrate our rotating swap method through the following example. Let A be a matrix with entries in $\mathbb{F}_2$. Assume A has the following form:

$$\begin{array}{c} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \end{array} \begin{pmatrix} 0 & 1 & * & * & * & * \\ 0 & 0 & 0 & 1 & * & * \\ 0 & 0 & 1 & * & * & * \\ 0 & 1 & * & * & * & * \\ 1 & * & * & * & * & * \\ 1 & * & * & * & * & * \end{pmatrix},$$

where "$*$" may be 1 or 0, S_i is the signature of each row, and we assume $S_1 \prec_s S_2 \prec_s \cdots \prec_s S_6$.

To reduce A to row-echelon form, we first find the pivot entry in the first column. *We must search the pivot entry from top to bottom (i.e. from lower signatures to higher signatures)*. Then we find the entry at row 5 and col 1 is a pivot. If we use general methods of elimination, we need to swap row 1 and row 5 directly, and clear entries at column 1 by the row with signature S_5. Then matrix becomes:

$$\begin{matrix} S_5 \\ S_2 \\ S_3 \\ S_4 \\ S_1 \\ S_6 \end{matrix} \begin{pmatrix} 1 & * & * & * & * & * \\ 0 & 0 & 0 & 1 & * & * \\ 0 & 0 & 1 & * & * & * \\ 0 & 1 & * & * & * & * \\ 0 & 1 & * & * & * & * \\ 0 & * & * & * & * & * \end{pmatrix}.$$

Next, when doing elimination in the second column, the row with signature S_4 is selected as pivot row, and needs to eliminate other rows. However, this will leads to errors in signature-based algorithms, because the row with signature S_1 has a smaller signature than S_4 and cannot be eliminated by the row with signature S_4. Thus, in signature-based algorithms, we cannot swap row 1 and row 5 directly.

To make further eliminations correct, we swap row 1 and row 5 in a special manner. First, we pick up the row 5 with signature S_5. Second, we move rows 4, 3, 2, and 1 to rows 5, 4, 3, and 2 respectively. At last, we put the row with signature S_5 at row 1. After this swap, matrix A becomes the following form:

$$\begin{matrix} S_5 \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_6 \end{matrix} \begin{pmatrix} 1 & * & * & * & * & * \\ 0 & 1 & * & * & * & * \\ 0 & 0 & 0 & 1 & * & * \\ 0 & 0 & 1 & * & * & * \\ 0 & 1 & * & * & * & * \\ 1 & * & * & * & * & * \end{pmatrix}.$$

Next, we use the row with S_5 to clear all entries at column 1 below this row, and then column 1 is done. For column 2, we find pivots from rows with $S_1, ..., S_4$ and S_6, and repeat the above processes. Elimination terminates when the matrix becomes an upper triangular form.

This rotating swap method can ensure the rows with larger signatures are only reduced by rows with smaller signatures. Firstly, when we eliminating entries in some column (e.g. column 1), the pivot row (e.g. row of S_5) is always found from low signatures to high signatures (e.g. the row with the smallest signature such that its entry at column 1 is nonzero), so the rows with smaller signatures (e.g. rows of $S_1, \ldots, S_4$) will not be reduced by the pivot row (e.g. row of S_5). Thus, elimination in current column (e.g. column 1) is correct. Secondly, after this rotating swap, the rows (e.g. rows of $S_1, \ldots, S_4$ and S_6) below the pivot row (e.g. row of S_5) are still in an increasing order on signatures. So the eliminations afterwards (e.g. column 1) are correct as well.

Using this rotating swap, the echelon form of A is in an upper triangular form, such that divide-and-conquer methods of PLE decomposition [2] can be used, and hence, the eliminations can be speeded up significantly.

In our implementation, we modify many subroutines of `mzd_ple()` in the M4RI library to use this rotating swap. Our new function is called `gvw_ple()`. We compare

Table 1
`mzd_ple()` vs. `gvw_ple()`.

Examples (1k = 1000)	Density ≈ 50%		Density ≈ 3%	
	`mzd_ple()`	`gvw_ple()`	`mzd_ple()`	`gvw_ple()`
10k × 10k	0.378	0.382	0.345	0.354
10k × 30k	1.342	1.301	1.268	1.262
30k × 10k	1.432	1.443	1.403	1.418
30k × 30k	7.661	7.655	7.604	7.577
30k × 60k	18.684	18.671	18.651	18.634
60k × 30k	19.396	19.296	19.282	19.298
60k × 60k	58.373	58.636	54.509	54.263
60k × 100k	123.321	123.298	119.479	122.523
100k × 60k	119.991	118.388	108.565	108.501
100k × 100k	266.817	267.191	237.401	237.560
150k × 150k	817.682	817.750	700.032	700.781

the efficiency of `mzd_ple()` and `gvw_ple()` in the next section. The results show both functions almost have the same efficiency.

5. Experimental results

We implemented both the GVW and M-GVW algorithm over Boolean polynomial rings in C++. The library M4RI (ver. 20130416) [3] is used.[4]

Since our experiments in this paper only aim to test the affects of mutant pairs, our implementations here simply use routines for dense matrices. To obtain the best performance of the implementation, structured Gaussian eliminations and many other techniques should be used. We will report these techniques in another paper.

5.1. Tests for `gvw_ple()`

We tested the efficiency of the function `gvw_ple()`,[5] which is modified from `mzd_ple()` by using the rotating swap method. Examples with density ≈ 50% are generated directly by routines from M4RI, and we also generate some randomized matrices with density ≈ 3%. In Table 1, the first column is the size of matrices, and the timings in the other columns are given in seconds.

From Table 1, we can see that `mzd_ple()` and `gvw_ple()` almost have the same efficiency. So the new proposed rotating swapping method does not slow down the efficiency of eliminations.

5.2. Tests for M-GVW

In this subsection, we compare the performance of GVW and M-GVW when the input systems are inhomogeneous. We implemented both GVW and M-GVW in matrix

[4] Up to now, the latest version of M4RI is ver. 20140914. By our experiments, the latest routines are about 5% faster than those from ver. 20130416.

[5] The codes of `gvw_ple()` based on M4RI ver. 20130416 can be found at http://www.mmrc.iss.ac.cn/~dwang/software.html.

Table 2
GVW vs. M-GVW.

Example	GVW			M-GVW			
	max mat. (deg)	time for max mat.	total time	max mat. (deg)	time for max mat.	total time	mutant pairs
MQ16	9023 × 6863(5)	0.412	0.608	9023 × 6863(5)	0.404	0.611	0
MQ20	22875 × 21656(5)	4.445	5.918	22875 × 21656(5)	4.446	5.918	0
MQ24	226386 × 189944(6)	1546	1686	226386 × 189944(6)	1543	1684	0
HFE_25_96	60279 × 68334(5)	72.52	104.3	12903 × 14104(4)	1.479	4.527	1050
HFE_30_96	127991 × 174308(5)	697.0	994.0	20843 × 30259(4)	6.256	17.18	1320
HFE_35_96	240727 × 383969(5)	5495	8373	31430 × 57261(4)	23.01	57.77	1680

style over Boolean polynomial rings. Almost all of the implementations are the same, except that we do not apply both criteria to mutant J-pairs in M-GVW. On eliminating matrices over $GF(2)$ in both GVW and M-GVW, we use the routine `gvw_ple()` which is for dense matrices, although the practical matrices are really very sparse.

We tested many inhomogeneous systems over Boolean polynomial rings. Examples include "MQ n" systems (n quadratic equations with n variables) given by Courtois [8], and a few smaller HFE systems (HFE25/ HFE30/HFE35, 25/30/35 quadratic equations with 25/30/35 variables and $D = 96$) downloaded from [25].[6] The ordering $\prec_p$ is the Graded Reverse Lexicographic ordering, and $\prec_s$ is a position over term extension of $\prec_p$. The experimental platform is MacBook Pro with 2.6 GHz Intel Core i7, 16 GB memory.

Table 2 gives the global comparisons of GVW and M-GVW for all examples. For both GVW and M-GVW, we list the largest matrices generated during the computations, degrees of the largest matrices, the timings for eliminating these matrices by `gvw_ple()`, and the total computing time. For M-GVW, we also give the number of new mutant pairs. All the timings in this table are given in seconds. Please note that the number of mutant pairs is counted *before* linear polynomials are found, while the number of mutant pairs that are generated in the same matrix with linear polynomials, is not counted.

From Table 2, we can find that the maximal size of the matrix generated during the computations is exactly the same for Courtois' examples, and the corresponding computing time are also almost the same. This is because we can not find mutant polynomials in these examples. For the HFE systems, M-GVW performs much better than GVW because many mutant polynomials have been found in M-GVW and their J-pairs are not rejected in the following computations. So the maximal sizes of the matrix in M-GVW become much smaller than those in GVW, which results in that M-GVW cost much less eliminating time.

We also give some detailed comparisons for some HFE systems. In the following two tables, "no. of mat." means the i-th matrix computed in the corresponding algorithms. We also list: the numbers of J-pairs to be reduced (these J-pairs are not rejected by criteria), the size and degree of the matrices, the numbers of new generated pairs after eliminations, the number of new mutant pairs, the number of rows that are eliminated

[6] All tested systems can be found at http://www.mmrc.iss.ac.cn/~dwang/software.html.

Table 3
HFE_30_96 for GVW vs. M-GVW.

No. of mat.	Alg.	J-pairs	Mat. (deg)	New pairs	New mutant pairs	Zeros	Timing
1	GVW	186	930 × 4526(3)	278	0	0	0.020
	M-GVW	186	930 × 4526(3)	278	0	0	0.020
2	GVW	1433	13558 × 31861(4)	2864	60	113	4.408
	M-GVW	1433	13558 × 31861(4)	2864	60	113	4.394
3	GVW	379	12705 × 30202(4)	379	379	0	4.058
	M-GVW	1260	14007 × 30623(4)	1260	1260	0	4.782
4	GVW	10760	127991 × 174308(5)	27816	6083(3)	1046	746.0
	M-GVW	5450	20843 × 30259(4)	5095	5095(26)	425	7.949

Table 4
HFE_35_96 for GVW vs. M-GVW.

No. of mat.	Alg.	J-pairs	Mat. (deg)	New pairs	New mutant pairs	Zeros	Timing
1	GVW	232	1260 × 7176(3)	345	0	0	0.047
	M-GVW	232	1260 × 7176(3)	345	0	0	0.047
2	GVW	1886	21470 × 59416(4)	3818	70	135	14.60
	M-GVW	1886	21470 × 59416(4)	3818	70	135	14.47
3	GVW	502	20490 × 57227(4)	504	504	0	13.75
	M-GVW	1608	22098 × 57729(4)	1610	1610	0	15.90
4	GVW	15120	240727 × 383969(5)	39970	7844(2)	1335	5676
	M-GVW	6720	31430 × 57261(4)	6224	6224(30)	496	27.26

to 0, and the timing for finishing this loop (given in seconds). We do not give the data after linear polynomials are obtained.

In last matrices of Tables 3 and 4, the number of new mutant pairs "7844(2)" means there are 7844 new mutant pairs in current matrices and 2 of them correspond to linear polynomials. Besides, since linear polynomials are found, the number 7844 are not counted in Table 2.

It is not quite strange to find that we get more new pairs than the J-pairs to be reduced, e.g. in the second matrix of HFE_35_96, we get 3818 new pairs by reducing 1886 J-pairs. Because in the matrix version of M-GVW, pairs that used to reduce J-pairs can also be reduced by other pairs, which is the same as F5 does in [16].

From Tables 3 and 4, we can see that the computing procedures of these two examples are quite similar. Taking the system HFE_35_96 for example, GVW and M-GVW perform exactly the same for the first two matrices. Note that there are 70 mutant pairs generated in the second matrix. To build the third matrix, GVW applies both criteria to the J-pairs generated by these 70 mutant pairs, and only 502 pairs are not rejected. After the elimination, 504 new pairs are found, but unfortunately, non of them are of degree 3. So GVW has to build the forth matrix with 5-degree polynomials, and the matrix is extremely large. Reducing this large 5-degree matrix cost 5495 seconds. To make it worse, only two linear polynomials are found, which results in that GVW still have to take about 2600 seconds to find all other linear polynomials. However, the case

of M-GVW is completely different. 1608 J-pairs are generated by the 70 mutant pairs. Although it takes a bit more time for M-GVW to reducing the third matrix, we get more new pairs and many of these new pairs are of degree 3. Finally, after eliminating a 4-degree matrix, M-GVW get 30 linear polynomials, and the algorithm terminates immediately.

6. Conclusions

In this paper, we present a variant of the GVW algorithm, called M-GVW, to avoid rejecting J-pairs with larger signatures and lower degrees. M-GVW is exactly the same as GVW when input systems are homogeneous, but M-GVW usually has a better performance when the input systems are inhomogeneous, particularly for the HFE systems.

On implementing GVW/M-GVW, to take advantages of the efficient routines from M4RI, we proposed a rotating swap method to do one-direction eliminations of matrices. We modified the routine `mzd_ple()` of M4RI, and obtain `gvw_ple()`. The experimental results show that our rotating swap method does not slow down `gvw_ple()`. We also believe this rotating swap method can be easily applied to other linear algebraic package.

To obtain the best performance of the implementations of GVW/M-GVW, techniques of sparse linear algebra must be used, since the matrices generated by Gröbner basis algorithms are quite sparse. By using more sparse techniques, our implementations of GVW/M-GVW have been greatly improved and we will report more implementing details in another paper.

References

[1] M. Albrecht, J. Perry, F4/5, preprint, arXiv:1006.4933v2 [math.AC], 2010.
[2] M. Albrecht, C. Pernet, Efficient decomposition of dense matrices over GF(2), arXiv:1006.1744, 2011.
[3] M. Albrecht, G. Bard, The M4RI library – version 20130416, http://m4ri.sagemath.org, 2013.
[4] A. Arri, J. Perry, The F5 criterion revised, J. Symb. Comput. 46 (2011) 1017–1029.
[5] B. Buchberger, Ein Algorithmus zum auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal, PhD thesis, 1965.
[6] B. Buchberger, A criterion for detecting unnecessary reductions in the construction of Gröbner basis, in: Proceedings of EUROSAM'79, in: Lect. Notes in Comp. Sci., vol. 72, Springer, Berlin, 1979, pp. 3–21.
[7] N. Courtois, A. Klimov, J. Patarin, A. Shamir, Efficient algorithms for solving overdefined systems of multivariate polynomial equations, in: Proc. EUROCRYPT'00, in: Lect. Notes in Comp. Sci., vol. 1807, Springer, Berlin, 2000, pp. 392–407.
[8] N. Courtois, Benchmarking algebraic, logical and constraint solvers and study of selected hard problems, http://www.cryptosystem.net/aes/hardproblems.html, 2013.
[9] J. Ding, J. Buchmann, M.S.E. Mohamed, W.S.A.E. Mohamed, R.-P. Weinmann, MutantXL, in: Proceedings of the 1st International Conference on Symbolic Computation and Cryptography, SCC08, Beijing, China, 2008, pp. 16–22.
[10] J-G. Dumas, C. Pernet, Z. Sultan, Simultaneous computation of the row and column rank profiles, in: Proc. ISSAC'13, ACM Press, New York, USA, 2013.
[11] C. Eder, J. Perry, F5C: a variant of Faugère's F5 algorithm with reduced Gröbner bases, J. Symb. Comput. 45 (12) (2010) 1442–1458.
[12] C. Eder, An analysis of inhomogeneous signature-based Gröbner basis computations, J. Symb. Comput. 59 (2013) 21–35.

[13] C. Eder, B.H. Roune, Signature rewriting in Gröbner basis computation, in: Proc. ISSAC'13, ACM Press, New York, USA, 2013, pp. 331–338.
[14] C. Eder, J.-C. Faugère, A survey on signature-based Gröbner basis computations, arXiv:1404.1774, 2014.
[15] J.-C. Faugère, A new efficient algorithm for computing Gröbner bases (F_4), J. Pure Appl. Algebra 139 (1–3) (1999) 61–88.
[16] J.-C. Faugère, A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5), in: Proc. ISSAC'02, ACM Press, New York, USA, 2002, pp. 75–82.
[17] J.-C. Faugère, A. Joux, Algebraic cryptanalysis of Hidden Field Equation (HFE) cryptosystems using Gröbner bases, in: Proc. CRYPTO'03, in: LNCS, vol. 2729, Springer, Berlin/Heidelberg, 2003, pp. 44–60.
[18] J.-C. Faugère, S. Rahmany, Solving systems of polynomial equations with symmetries using SAGBI-Gröbner bases, in: Proc. ISSAC '09, ACM Press, New York, USA, 2009, pp. 151–158.
[19] S.H. Gao, Y.H. Guan, F. Volny, A new incremental algorithm for computing Gröbner bases, in: Proc. ISSAC'10, ACM Press, New York, USA, 2010, pp. 13–19.
[20] S.H. Gao, F. Volny, M.S. Wang, A new algorithm for computing Gröbner bases, Cryptology ePrint archive, report 2010/641, 2010.
[21] S.H. Gao, F. Volny, M.S. Wang, A new framework for computing Gröbner bases, Math. Comput. 85 (297) (2016) 449–465.
[22] A. Hashemi, G. Ars, Extended F5 criteria, J. Symb. Comput. 45 (12) (2010) 1330–1340.
[23] D. Lazard, Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations, in: Proc. EUROCAL'83, in: Lect. Notes in Comp. Sci., vol. 162, Springer, Berlin, 1983, pp. 146–156.
[24] B.H. Roune, M. Stillman, Practical Gröbner basis computation, in: Proc. ISSAC'12, ACM Press, 2012.
[25] A. Steel, Allan Steel's Gröbner basis timings page, http://magma.maths.usyd.edu.au/~allan/gb/, 2004.
[26] Y. Sun, D.K. Wang, A generalized criterion for signature related Gröbner basis algorithms, in: Proc. ISSAC'11, ACM Press, 2011, pp. 337–344.
[27] Y. Sun, D.K. Wang, D.X. Ma, Y. Zhang, A signature-based algorithm for computing Gröbner bases in solvable polynomial algebras, in: Proc. ISSAC'12, ACM Press, 2012, pp. 351–358.
[28] Y. Sun, D.K. Wang, A new proof for the correctness of the F5 algorithm, Sci. China Math. 56 (4) (2013) 745–756.
[29] Y. Sun, D.D. Lin, D.K. Wang, On implementing the symbolic preprocessing function over Boolean polynomial rings in Gröbner basis algorithms using linear algebra, J. Syst. Sci. Complex. 29 (3) (2016) 789–804.
[30] Y. Sun, D.D. Lin, D.K. Wang, An improvement over the GVW algorithm for inhomogeneous polynomial systems, arXiv:1404.1428 [cs.SC], 2014.