Divide and Conquer Algorithm for Gröbner Basis over Binary Field[*]

Teduh Wulandari, Sugi Guritman[†]
Nur Aliatiningtyas, Siswandi[‡]
Division of Pure Mathematics
Department of Mathematics
Faculty of Mathematics and Natural Sciences
IPB University

September 27, 2021

Abstract

In this paper we present an algorithm for finding Gröbner basis in binary multivariate polynomial ring $\mathbb{F}_2[x_1, x_2, ..., x_n]$ and for which the main goal is to solve binary multivariate nonlinear system; instead of the existing methods mostly based on linearization. The main idea of our method is well known divide and conquer recurrence; that in this case the Buchberger algorithm is applied only at the end of the combining phase. For data representation, every single polynomial in the ring can be considered to be a boolean object, so in symbolic computation perspective that the polynomial can be represented as a member of power set of integers and these integers represent variable symbol of the polynomial. Thus, with this point of view, details of the algorithm can be stated in algebraic computation codes. At the end of the paper, we give a general complexity of the algorithm and also present some facts from the implementation aspect.

Keywords: Gröbner basis, Binary multivariate polynomial ring, Divide and conquer algorithm, Set representation of polynomial.

1 Introduction

The concept of Gröbner bases was introduced by Bruno Buchberger (1965) in the context on performing algebraic computations in residue classes of multivariate polynomial rings. Gröbner bases techniques also have various applications

[*]Hasil pemikiran tidak dipublikasikan, terdokumentasi di Perpustakaan Departemen Matematika IPB untuk pembimbingan tugas akhir mahasiswa.

[†]teduhma@apps.ipb.ac.id, sugigu@apps.ipb.ac.id.

[‡]nural@apps.ipb.ac.id, siswandi@apps.ipb.ac.id

such as: in algebraic geometry, optimization, coding theory, control theory, cryptography, and many other fields.

In this paper, we study Gröbner bases only focus on binary case and for which the main goal is to solve binary multivariate nonlinear system. The special application of this study is state of the art algebraic attack on cryptosystems.

Problem of computing Gröbner bases associated with the solution of multivariate nonlinear system over general finite fields, with or without connected to algebraic attacks on cryptosystem, has been discussed in many articles over last decades. We refer to some of them in [1], [2], [3], [4], [6], [7], and for the basic theory we refer to [5].

It is different from the above our citations, the main idea of our method is well known divide and conquer recurrence. But still, they are all inspiring us especially for the binary case that we are focusing more on [3], [4], and [1] In our method, the Buchberger algorithm is applied only at the end of the combining phase. For data representation, every polynomial in the ring we consider as a boolean object, so in symbolic computation perspective that the polynomial can be represented as a member of power set of integers and these integers represent variable symbol of the polynomial. Based on this representation, we optimize the performance of splitting and combining part of the divide and conquer recurrence.

At the end of the paper, we give a general complexity of the algorithm and also present some facts from the implementation aspect.

2 Set Representation

For the subsequent discussion in this paper, we follow [5] as a basic theory and we customize for the binary case.

Let $\mathbb{F}_2$ be the binary field, then we will denote $\mathbb{F}_2[x_1, x_2, ..., x_n]$ be the binary ring of polynomials with coefficients in $\mathbb{F}_2$ and the indetermiates are $x_1, x_2, ..., x_n$. Since the main purpose of this paper is to solve binary multivariate nonlinear system, we may consider that every polynomial $f \in \mathbb{F}_2[x_1, x_2, ..., x_n]$ as a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$. So by this notion, the indetermiates $x_1, x_2, ..., x_n$ can be considered as variables. Subsequently, it is clear that $x_i^2 = x_i$ for every $i = 1, 2, ..., n$, and so that we have the following definition.

Definition 1 ***Binary multivariate polynomial ring*** $\mathbb{F}_2[x_1, x_2, ..., x_n]$ *is the set*

$$\left\{ \sum_{(i_1, i_2, ..., i_n) \in \mathbb{F}_2^n} a_{(i_1, i_2, ..., i_n)} . \left(x_1^{i_1} . x_2^{i_2} . \cdots . x_n^{i_x}\right) \mid a_{(i_1, i_2, ..., i_n)} \in \mathbb{F}_2 \right\}$$

with appropriate addition and multiplication of the binary polynomials follow from the general case. Then, $x_1^{i_1} . x_2^{i_2} . \cdots . x_n^{i_x}$ *will be called* ***monomials*** *of the polynomial.*

Simple illustration from the definition, we have

$$\mathbb{F}_2[x, y] = \{a_{00} + a_{10}x + a_{01}y + a_{11}xy \mid a_{00}, a_{10}, a_{01}, a_{11} \in \mathbb{F}_2\}$$

So, it is also clear that $\#\mathbb{F}_2[x,y] = 2^4 = 16$, and for n variables we have

$$\#\mathbb{F}_2[x_1, x_2, ..., x_n] = 2^{2^n}$$

For the shake of easier symbolic compution, variables $x_1, x_2, ..., x_n$ will be represented as integers $1, 2, ..., n$, repectively. Then, it is clear that the monomials will be presented as the subsets of $N = \{1, 2, ..., n\}$. In other words, the set of all monomials be the power set of N, call it as $\mathcal{P}(N)$. Then again, it is clear for the notion of monomials ordering is the ordering in the $\mathcal{P}(N)$, that is the typical *graded lexicographic* order in increasing order. Thus, based on these notions and the representation, it is reasonable to give alternative definitions about $\mathbb{F}_2[x_1, x_2, ..., x_n]$ as follows.

Definition 2 *We denote $\mathcal{M}_n = \mathcal{P}(N)$ be the set of all monomials of n variables. Then, for any two monomials $A, B \in \mathcal{M}_n$, we have the following definitions.*

1. ***Multiplication** of A and B, notation AB, is the set $AB := A \cup B$.*
2. *We mean A **divide** by B, notation $\frac{A}{B}$, is the set $\frac{A}{B} := A \setminus B$. In this case, B is said to be **devisible** by A, notation $A \mid B$, if only if $A \subseteq B$.*
3. *We mean $A \leq B$ follows the ordering set in $\mathcal{M}_n$.*

Definition 3 ***Binary multivariate polynomial ring of** n **variables**, denoted as $\mathcal{R}_n$, is the power set $\mathcal{R}_n = \mathcal{P}(\mathcal{M}_n)$ with the two operations on $\mathcal{R}_n$ are defined as follows..*

1. *For any two polynomials $P, Q \in \mathcal{R}_n$, **addision** of P and Q, notation $P \oplus Q$, is the set*
$$P \oplus Q := (P \cup Q) \setminus (P \cap Q)$$
2. *For any $P, Q \in \mathcal{R}_n$, let $P = \{P_1, P_2, ..., P_k\}$ and $Q = \{Q_1, Q_2, ..., Q_l\}$, **multiplication** of P and Q, notation $P \odot Q$, is the set*
$$P \odot Q := \bigoplus_{i,j=1}^{k,l} \{P_i Q_j\}$$

   Moreover,

3. *Empty set $\{\} \in \mathcal{R}_n$ be the addition identity and call it as the **zero polynomial** in $\mathcal{R}_n$.*
4. *The set $\{\{\}\} \in \mathcal{R}_n$ be the multiplication identity and call it as the **unity polynomial** in $\mathcal{R}_n$.*
5. *For any $P \in \mathcal{R}_n$, the **last monomial** of P, notation $lt(P)$, is defined as the set*
$$lt(P) = \max_{\leq}(P)\,,\ "\leq" \text{ be the set ordering in } P.$$

Definition 4 *Let $A \in \mathcal{M}_n$, $P \in \mathcal{R}_n$, and that $P = \{P_1, P_2, ..., P_k\}$. **Multiplication of** A **and** P, notation AP, is defined as the polynomial*

$$AP := \bigoplus_{i=1}^{k} \{AP_i\}.$$

*Furthermore, if $A \mid P_i$ for $i = 1, 2, ..., k$, we define **division of** P **by** A, notation $\frac{P}{A}$, as the polynomial*

$$\frac{P}{A} := \left\{ \frac{P_1}{A}, \frac{P_2}{A}, ..., \frac{P_k}{A} \right\}$$

which also can be said as "deleting variables of A in every monomial of P".

3 Buchberger's Algorithm

In this section we continue to use set representation to define Gröbner bases, and of course the most important thing is to describe Buchberger algorithm for which can be used to find bases in $\mathcal{R}_n$.

3.1 Gröbner Basis

Since the definition of Gröbner basis needs the notion of the division algorithm, so we begin with the following proposition.

Proposition 1 *Let $P \in \mathcal{R}_n$ and suppose that $P_1, P_2, ..., P_k \in \mathcal{R}_n$ be a sequence of non-zero polynomials. Then, there exist $Q_1, Q_2, ..., Q_k, R \in \mathcal{R}_n$ such that*

$$P = (Q_1 \odot P_1) \oplus (Q_2 \odot P_2) \oplus ... \oplus (Q_k \odot P_k) \oplus R$$

and either $R = \{\}$ or none of the monomials in R is divisible by

$$lt(P_1), lt(P_2), ..., lt(P_k).$$

Furthermore, for every $i = 1, 2, ..., k$, $lt(Q_i \odot P_i) \leq lt(P)$ if $(Q_i \odot P_i) \neq \{\}$.

We call the above proposition as *division algorithm* in $\mathcal{R}_n$, its proof follows from the general case (it can be found in [5]). For the symbolic computaion purpose, this algorithm can be represented as follows.

Algorithm 1 ***(Division Algorithm)***
INPUT: $P \in \mathcal{R}_n$ dan $P_1, ..., P_k \in (\mathcal{R}_n \setminus \{\{\}\})$
OUTPUT: $Q_1, ..., Q_k, R \in \mathcal{R}_n$

```
1. $Q_1 \leftarrow \{\}; ...; Q_k \leftarrow \{\}$.
2. $R \leftarrow \{\}; X \leftarrow P$.
```

```
3. while $X \neq \{\}$ do
     $Y \leftarrow X;$
     for $i = 1$ to $k$ do
       if $lt(P_i) \subseteq lt(X)$ then
         $X \leftarrow \left(X \oplus \frac{lt(X)}{lt(P_i)} P_i\right);$
         $Q_i \leftarrow \left(Q_i \oplus \left\{\frac{lt(X)}{lt(P_i)}\right\}\right);$
         break;
         end if;
       end if;
     end do;
     if $X = Y$ then
       $R \leftarrow (R \oplus \{lt(X)\});$
       $X \leftarrow (X \smallsetminus \{lt(X)\});$
     end if;
   end do;
4. return$(Q_1, ..., Q_k, R)$.
```

Definition 5 *From the Algorithm 1, we define the output R as the result of polynomial P **modulo** $(P_1, ..., P_k)$, that is*

$$R := P \bmod (P_1, ..., P_k)$$

*In other words, R is the **reminder** part af the division algorithm.*

Now, we are ready to define Gröbner basis as follows.

Definition 6 *A set of non-zero polynomials $G = \{P_1, P_2, ..., P_k\} \subseteq \mathcal{R}_n$ is called to be a **Gröbner basis** for an ideal I in R_n if $G \subseteq I$ and, for every polynomials P in $I \smallsetminus \{\{\}\}$,*

$$lt(P_i) \mid lt(P)$$

for some $i = 1, 2, ..., k$. Moreover, the set G is called a Gröbner basis if it is a Gröbner basis for the ideal $I = \langle P_1, P_2, ..., P_k \rangle$.

3.2 Algorithm Description

The most primitive part of Buchberger's algorithm is a function that we call it as *S-polynomial.* Below is the definition of the function.

Definition 7 *The **S-polynomial** of two non-zero polynomials P and Q in $\mathcal{R}_n$, notation $\mathbf{S}(P,Q)$, is defined as the function*

$$\mathbf{S}(P,Q) := (lt(Q) \smallsetminus X)\, P \oplus (lt(P) \smallsetminus X)\, Q$$

where $X = lt(P) \cap lt(Q)$.

Buchberger's algorithm is the process of continuously adding non-zero remainders of S-polynomials. The driven machine of the algorithm is the division algorithm and usually we want to reduce the divisions as few as possible. The algorithm can be quite horrible that is extremely time consuming to compute and also very dependent on the term ordering. Therefore, we need to speed up the algorithm by reducing input of polynomials. Below, we give an algorithm that can be included to the Buchberger's algorithm to speed up the performance.

Algorithm 2 ***(Reducing a sequence of polynomials)***
INPUT: Polynomials $(P_1, ..., P_k)$ in $(\mathcal{R}_n \smallsetminus \{\{\}\})$.
OUTPUT: Reduced polynomials $(Q_1, ..., Q_m)$ such that $\langle Q_1, ..., Q_m \rangle = \langle P_1, ..., P_k \rangle$.

```
1. $I \leftarrow (P_1, ..., P_k)$;
2. for $i = 1$ to $k$ do
       $Q_i \leftarrow P_i \bmod (P_1, ..., P_{i-1}, P_i, ..., P_k)$;
       $J \leftarrow (P_1, ..., P_{i-1}, Q_i, P_i, ..., P_k)$;
   end do;
3. if $J = I$ then return$(J)$; else
       $(P_1, ..., P_m) \leftarrow J$; go to Step 1;
   end if;
```

We define Algorithm 2 as the function

$$\mathbf{Redu}(P_1, ..., P_k)$$

Involving Algorithm 2 to the Buchberger's algorithm would not only speed up the performance, but also it will produce a *reduced Gröbner basis* as the output.

Definition 8 *A **minimal Gröbner basis** $(P_1, ..., P_k)$ is a Gröbner basis such that $lt(P_i)$ is not divisible by $lt(P_j)$ for $i \neq j$. A **reduced Gröbner basis** $(P_1, ..., P_k)$ is a minimal Gröbner basis such that no monomials (not just the last monomial) in P_i is divisible by $lt(P_j)$ for $i \neq j$.*

In fact, a reduced Gröbner basis is unique. Finally, here we present Buchberger's algorithm using set presentation for symbolic computation purpose.

Algorithm 3 ***(Buchberger's algorithm)***
INPUT: Polynomials $(P_1, ..., P_k)$ in $(\mathcal{R}_n \smallsetminus \{\{\}\})$.
OUTPUT: The reduced Gröbner basis $(Q_1, ..., Q_m)$ for the ideal $\langle P_1, ..., P_k \rangle$

```
1. $(Q_1, ..., Q_m) \leftarrow \mathbf{Redu}(P_1, ..., P_k)$ ; $U \leftarrow \{\}$ ;
2. for $i = 1$ to $m - 1$ do
     for $j = i + 1$ to $m$ do
       $S \leftarrow \mathbf{S}(Q_i, Q_j)$ ;
       $R \leftarrow S \bmod (Q_1, ..., Q_m)$ ;
       if $R = \{\{\}\}$ then return$((R))$ end if;
       if $R \neq \{\}$ then $U \leftarrow U \cup \{R\}$ end if;
     end do;
   end do;
3. while $U \neq \{\}$ do
     $P \leftarrow \{Q_1, ..., Q_m\} \cup U$;
     $(Q_1, ..., Q_m) \leftarrow \mathbf{Redu}(P_1, ..., P_k)$ ; $U \leftarrow \{\}$ ;
     for $i = 1$ to $m - 1$ do
       for $j = i + 1$ to $m$ do
         $S \leftarrow \mathbf{S}(Q_i, Q_j)$ ;
         $R \leftarrow S \bmod (Q_1, ..., Q_m)$ ;
         if $R \neq \{\}$ then $U \leftarrow U \cup \{R\}$ end if;
       end do;
     end do;
   end do;
4. return$(Q_1, ..., Q_m)$.
```

We define Algorithm 3 as the function

$$\mathbf{BbAlg}(P_1, ..., P_k)$$

4 Applying Divide and Conquer Recurrence

In this section, we consider that the Gröbner basis associated to the solution of binary multivariate nonlinear system. In this case, for any non-empty subset $F = \{f_1, f_2, ..., f_m\} \subseteq \mathbb{F}_2[x_1, x_2, ..., x_n]$, we interpretate as *binary multivariate nonlinear system* with m equations and n variables, that is

$$f_1(x_1, ..., x_n) = 0,\ f_2(x_1, ..., x_n) = 0, ..., f_m(x_1, ..., x_n) = 0$$

If $G = \{g_1, g_2, ..., g_k\}$ is the reduced Gröbner basis for the ideal $\langle f_1, f_2, ..., f_m \rangle$, then the solution of F if only if the solution of G. In fact, finding the solution of G much simpler than F.

4.1 Basic Idea

From the previous section, we are already given the basic method for finding Gröbner bases. Now, we enhance the performance of Buchberger's algorithm by applying divide and conquer recurrence. *Basic idea of the divide and conquer recurrence is to take large problem and divide it into smaller problems until problem is trivial, then combine the parts to make the solution.* In the following, we present the basic rationalization of the relationship between Buchberger's algorithm and the idea of divide and conquer recurrence.

Given binary multivariate nonlinear system $F = \{f_1, f_2, ..., f_m\}$, and the members of F are written as

$$f_1(x_1, ..., x_{n-1}, x_n),\ f_2(x_1, ..., x_{n-1}, x_n), ..., f_m(x_1, ..., x_{n-1}, x_n).$$

Then, for every $j = 1, ..., m$, there exist unique pair (f_{j0}, f_{j1}) of the polynomials such that f_j can be expressed on the form

$$f_j(x_1, ..., x_{n-1}, x_n) = f_{j,0}(x_1, ..., x_{n-1}) + f_{j,1}(x_1, ..., x_{n-1}).x_n$$

This means that F can be uniquely divided into two systems,

$$F_0 = \{f_{1,0}, f_{2,0}, ..., f_{m,0}\} \text{ and } F_1 = \{f_{1,1}, f_{2,1}, ..., f_{m1}\},$$

in which each system having $(n-1)$ variables: $x_1, ..., x_{n-1}$. It is just typical *division phase* of the divide and conquer recurrence which then the parts can be done recursively until we get the smallest parts of the system. The smallest parts should be trivial for the Buchberger's algorithm.

Suppose that F_0 and F_1 be the smallest parts of the system, the *combining phase* is just combining the solutions of F_0 and F_1 to compute the solution of F. Detail process of this phase will be given directly in the algorithm description, in the next subsection.

4.2 The Proposed Algorithm

From the basic idea in the previous subsection, in this subsection we will conctruct an algorithm that can be used to find a Gröbner bases from given binary multivariate nonlinear system. For the shake symbolic computation and the algorithm speed, we will take an advantage of using set representation described in Section 2 of this paper. Below is the description and rationalization of the algorithm.

- The input is any subset $B = \{P_1, ..., P_m\} \subseteq (\mathcal{R}_n \setminus \{\{\}\})$.
- If $\{\{\}\} \in B$, the algorithm will return $\{\{\{\}\}\}$. This means that the system is *inconsistent.*
- The algorithm take *the smallest sub-problem be the system of one variable.* This implies that the smallest system contains only one polynomial or two polynomials. Let A be the smallest system, we have: if $\#A = 1$ then algorithm return A; else return $\{\{\{\}\}\}$. The performance of this step is just like a bit operation.

- For the *division phase*, let $P \in B$ such that $P = P_0 \oplus P_1 . \{n\}$. In this case, strategy to compute P_0 and P_1, we describe in the following steps:
    - in pre-computation we define K be the power set of $\{1, 2, ..., n-1\}$;
    - compute $P_0 := P \cap K$ and then $P_1 := \frac{P \smallsetminus P_0}{\{n\}}$.

  Then, we have B has been divided into two systems

  $$B_0 = \{P_{1,0}, P_{2,0}, ..., P_{m,0}\} \text{ and } B_1 = \{P_{1,1}, P_{2,1}, ..., P_{m,1}\} .$$

- For the *combining phase*. In the recursive process, suppose that the algorithm has already calculated S_0 as the solution of B_0 and S_1 as the solution of $(B_0 \oplus B_1)$. Then, the algorithm will compute S as the solution of B by the following descriptions.
    - If $S_0 = \{\{\{\}\}\}$ and $S_1 = \{\{\{\}\}\}$, the algorithm returns $S = \{\{\{\}\}\}$. That means, for any cases the value of variabel x_n, the system B must be inconsistent.
    - If $S_0 = \{\{\{\}\}\}$ and $S_1 \neq \{\{\{\}\}\}$, the algorithm return

      $$S = S_1 \cup \{\{\{\}, \{n\}\}\} .$$

      which means that the solution of B is the set S_1 when $x_n = 1$.
    - If $S_0 \neq \{\{\{\}\}\}$ and $S_1 = \{\{\{\}\}\}$, the algorithm return

      $$S = S_0 \cup \{\{n\}\} .$$

      which means that the solution of B is the set S_0 when $x_n = 0$.
    - If $S_0 \neq \{\{\{\}\}\}$ and $S_1 \neq \{\{\{\}\}\}$, we compute the set:
        1. $U := S_0 \cap S_1$;
        2. $A_0 := S_0 \smallsetminus U$; and $A_1 := S_1 \smallsetminus U$;
        3. $Q := \{X_0 \oplus (X_0 \oplus X_1) \{n\} \mid X_0 \in A_0, X_1 \in A_1\}$; $R := U \cup Q$;
        4. then S is the output of Buchberger's algorithm of input R.

Eventually, here we present the proposed algorithm using set presentation for the symbolic computation purpose. We begin with a routine function that will be used to split polynomials; and a look-up table that will be used to speed up the splitting process. The look-up table is a list of the power set of $\{1, 2..., n-1\}$:

$$K := [\mathcal{P}(\{1\}), \mathcal{P}(\{1, 2\}), ..., \mathcal{P}(\{1, 2, .., n-1\})] .$$

Below is the algorithm of splitting set of polynomials.

Algorithm 4 ***(Splitting set of polynomials)***
INPUT: Subsets $\{P_1, ..., P_m\} \subseteq (\mathcal{R}_n \smallsetminus \{\{\}\})$, *interjer* j*, and the set* M *be the power set of* $\{1, 2, ..., j-1\}$ *refer to the look-up table* K*.*
OUTPUT: The two parts of set of polynomials: $\{P_{1,0}, ..., P_{m,0}\}$ *and* $\{P_{1,1}, ..., P_{m,1}\}$.

```
1. for $i = 1$ to $m$ do
      $P_{i,0} \leftarrow P_i \cap M; \quad Q_i \leftarrow \left(\frac{P_i \setminus P_{i,0}}{\{j\}}\right);$
      $P_{i,1} \leftarrow P_{i,0} \oplus Q_i;$
   end do;

2. return$([\{P_{1,0}, ..., P_{m,0}\}, \{P_{1,1}, ..., P_{m,1}\}]);$
```

We define the algorithm of splitting set of polynomials as the function

$$\mathbf{SplitP}\left(\{P_1, ..., P_m\}, j, M\right)$$

that will be called as a routine of the following *proposed algorithm.* This algorithm will be called as the function

$$\mathbf{GroBaDnC}\left(B, N, K\right)$$

where B stands for the set of polynomials that we will compute its Gröbner basis; $N = \{1, 2, ..., n\}$ be the set of variable symbols; K is a list that represent the look-up table K.

Algorithm 5 ***(The proposed algorithm))***
INPUT: Subsets $B = \{P_1, ..., P_k\} \subseteq (\mathcal{R}_n \setminus \{\{\}\})$, $N = \{1, 2, ..., n\}$, and the look-up table K.
OUTPUT: The reduced Gröbner basis $\{Q_1, ..., Q_k\}$ for the ideal $\langle P_1, ..., P_k\rangle$

```
1. If $\{\{\}\} \in B$ then return $(\{\{\{\}\}\})$ end if;

2. If $n = 1$ then
      If $\#B = 1$ then return $\{B\}$; else return $(\{\{\{\}\}\})$; end if;
   else
      $N \leftarrow N \setminus \{n\}$; $H \leftarrow$ SplitP $(B, n, K[n-1])$;
      $B0 \leftarrow H[1] \setminus \{\{\}\}$; $B1 \leftarrow H[2] \setminus \{\{\}\}$;
      $S0 \leftarrow$ GroBaDnC $(B0, N, K)$; $S1 \leftarrow$ GroBaDnC $(B1, N, K)$;
      if $S0 = \{\{\{\}\}\}$ then
         if $S1 = \{\{\{\}\}\}$ then return $(S1)$; end if;
         else $S \leftarrow S1 \cup \{\{\{\}, \{n\}\}\}$; return $(S)$; end if;
      else
         if $S1 = \{\{\{\}\}\}$ then $S \leftarrow S0 \cup \{\{\{n\}\}\}$; return $(S)$; end if;
         else
            $S \leftarrow S0 \cap S1$; $P \leftarrow S0 \setminus S$; $Q \leftarrow S1 \setminus S$;
            for $Y$ in $Q$ do
               for $X$ in $P$ do
            $R \leftarrow X \oplus Y$; $R \leftarrow \{n\}.R$; $R \leftarrow R \cup X$;
            $S \leftarrow S \cup \{R\}$;
          end do;
        end do;
        $S \leftarrow$ BbAlg$(S)$; return$(S)$;
      end if;
    end if;
  end if;
```

5 The Algorithms Analysis

In this last section, we present a general analysis for the proposed algorithm and implementation aspects. Description of the implementation aspects can also be considered as a general conclusion.

5.1 Speed Analysis

It is difficult for us to give a scrutiny analysis for the proposed algorithm because of the set representation of the algorithm. Even though we believe that the algorithm based on divide and conquer, however the usual analysis of divide and conquer algorithm is hard to describe. Therefore, the analysis of the algorithm will only be given on the form of speed description, instead of rigid time complexity.

Let $T(N)$ be the time compexity of the algorithm to solve the problem, where N be the input size. Then, master theorem of divide and conquer tells us that $T(N)$ satisfies an equation of the form

$$T(N) = aT\left(\frac{N}{b}\right) + f(N)$$

where $a \geq 1$ is the number of recursively calls, (N/b) with $b > 1$ is the size of a sub-problem, and $f(N)$ is the cost of the combine part. Below we present some items that could be used to assess the time complexity.

- Since the algorithm having input m polynomials in $\mathcal{R}_n$ and we assume that the input at random and $m \approx n$, then we may assume in general that the input size is $N = m.2^n$. In fact in avarage, each polynomial is a set containimg 2^{n-1} monomials.

- Since the algorithm take the smallest sub-problems of $n = 1$, and every polynomial is splitted into two parts, then we have $a = 2^n$ and $b = 2$, implies that $T(N/b) \in \mathcal{O}(1)$.

- *In the splitting polynomials process*, the algorithm is speeded up by the look-up table. The whole number of operations are involved in this process are mn times ($\cap$ and $\smallsetminus$ and *delete 1 symbol* and $\oplus$) set operations of polynomials of size 2^{n-1} monomials in avarage, see Algorithm 4. This process is hypotetically to be quite fast and it depends on the number of monomials in the polynomials operand.

- *Speed analysis in combining process can be used to assess the function* $f(N)$. If the Gröbner basis solution of the problem connected to the solution of binary multivariate nonlinear system, then we have the following hypothesis. Assuming that the input taken at random, and for $m > n$, then the sub-problems having output $\{\{\{\}\}\}$ (inconsistent solution), on every level in the recursive steps, should be having high probability. Especially for the smallest sub-problems, we have m polynomials of 1 variable, then the probability that the solution is inconsistent should be very high, tends to 1.

- In the case that two sub-problems will be combined, and both having a consistent output, applying Buchberger's algorithm to get the combined solution should be very fast. Our hypothetic reason of this assertion is each sub-problem having output reduced Gröbner basis, then combination of the two solution (see in the last steps in Algorithm 5) should be still near to the reduced Gröbner basis, so Algorithm 3 do not need too much effort to change it.

5.2 Implementation Facts

We have already implemented the Algorithm 5 running in MAPLE codes. We set all the inputs of the algorithm as a random data. All the experiments were done in the same computation environment and computer spesification. Below we describe some facts as a result of our obsevations from the implementation aspect.

- When we set $m = n$, then we may expect that the algorithm would output either inconsistent solution or having unique solution. This fact can be explained easily using elementary probability notion.

- We set that the system has already been consitent. If $m > n$, then we may expect that the algorithm would output unique solution. To get the unique solution we need to increase the value of m. This fact can be explained easily using elementary probability notion.

- When we convert all polynomials from set of set of integers representation to set of integers representation, Algorithm 5 will be getting faster significantly. As an illustrion, for $m = 20$ and $n = 16$, the new version takes only 7 seconds in avarage to solve the problem, compared to 16 seconds for the old one. In this case, it is clear that the performance of set operations will become much faster for the new one.

- Algorithm 5 is very much faster than Algorithm 3. As an illustrion, when we set $m = 10$ and $n = 8$, Algorithm 3 takes about 12 seconds to solve the problem in avarage; but Algorithm 5 in conversion version takes only 7 seconds even for very much larger input size $m = 20$ and $n = 16$. The explaination of this fact needs rigid time complexity analysis.

6 Future Works

From the results of this paper, at least we have three questions that would become our future works.

1. How to analyse a scrutiny time complexity for the algorithms?
2. Using set of integers representation, how to construct an efficient algorithm for solving binary multivariate nonlinear system without involving Buchberger's algorithm?
3. How to make an association the algorithms with algebraic attack to a cryptosytem, focus on special cases?

References

[1] M. Brickenstein and A. Dreyer. "PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials". In *Journal of Symbolic Computation*, Elsevier, vol. 44, issue 9, pp. 1326-1345. September 2009.

[2] N. T. Courtois and J. Patarin. "About the XL Algorithm over $GF(2)$". In *Springer-Verlag Berlin Heidelberg*, M. Joye (Ed.): CT-RSA 2003, LNCS 2612, pp. 141–157, 2003.

[3] S. Gao. "Counting Zeros over Finite Fields Using Gröbner Bases". *Master Thesis*, www.cs.cmu.edu, January 2009.

[4] F. Hinkelmann and E. Arnold. "Fast Gröbner Basis Computation for Boolean Polynomials". In *Published in ArXiv 2010*, http://semanticscholar.org>paper>. October 2018.

[5] N. Lauritzen, "Concrete Abstract Algebra: from Numbers to Gröbner Bases". *Cambridge University Press,* 2003. ISBN 978-0-521-53410-9.

[6] T. H. Nguyen. "Combinations of Boolean Gröbner Bases and SAT Solvers". *Doctoral Thesis*, Technische Universität Kaiserslautern, https://nbn-resolving.org/urn:nbn:de:hbz:386-kluedo-39582, December 2014.

[7] M. Sugita, M. Kawazoe, and H. Imai, "Relation between XL algorithm and Gröbner Bases Algorithms". In *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences,* E89A(1), pp. 11-18, January 2006.

[8] Y. Sun, Z. Huang, D. Lin, and D. Wang, “On Implementing the Symbolic Preprocessing Function over Boolean Polynomial Rings in Gröbner Basis Algorithms Using Linear Algebra”. In *Journal of Systems Science and Complexity*, Elsevier, vol. 29, issue 3, pp. 789–804. June 2016.