ANALYSIS OF FAUGÈRE’S ALGORITHMS AND GRÖBNER WALK FOR TORIC IDEALS

トーリックイデアルに対するFaugèreのアルゴリズムおよびGröbner walkの解析

by

NAKAYAMA Hiroki

中山 裕貴

A Master Thesis

修士論文

Submitted to

the Graduate School of the University of Tokyo

on February 3, 2004

in Partial Fulfillment of the Requirements

for the Degree of Master of Information Science and Technology

in Computer Science

Thesis Supervisor: IMAI Hiroshi 今井 浩

Professor of Information Science
ABSTRACT

Gröbner basis is very useful tool in solving systems of algebraic equations and ideal membership problems. Gröbner bases of polynomial ideals had been computed by some existing algorithms of Buchberger, but recently, $F_4$ and $F_5$ algorithms were proposed by Faugère. The former aims to improve speed of computation of normal forms, which are critical steps of Buchberger algorithm, by transforming multiple polynomials into matrices and reducing these polynomials at a time. On the other hand, if the ideal is toric, we cannot compute its Gröbner bases efficiently with $F_4$ algorithm because such matrices become very sparse. The latter eliminates generations of unnecessary polynomials completely by holding past results of computation in generations of new polynomials, but the efficiencies on toric ideals have not been examined.

Additionally, Gröbner basis of toric ideals is an important concept in the field of geometry and combinatorics, and Hosten and Robbiano et al. proposed algorithms to compute Gröbner bases by focusing on geometric structures of ideals. $F_4$ and $F_5$ algorithms were inferior to those geometric methods on speed, but by analyzing computation time with various incidence matrices and term orders, it is expected not only to deepen knowledge of $F_4$ and $F_5$ but also to give some insight to the structure of ideals.

In our study, first we improve data structures and algorithms by focusing on the properties of generated matrices in $F_4$ algorithm. If the ideal is toric, matrices that are generated when polynomials are being reduced have a good property that it is unimodular. In the case we can replace elementary operations of matrices with those on digraphs, and we can regard it as a network flow problem. We show that this gives both time and memory great improvements by $F_4$ algorithm.

Next we compare computation time on $F_4$ and $F_5$ algorithms with existing Buchberger algorithm. When the term order is a graded reverse lexicographic ordering, $F_4$ and $F_5$ are superior to Buchberger algorithm. Contrary when one is pure lexicographic ordering, $F_4$ and $F_5$ are inferior. These reasons on $F_4$ are assumed that increase of reducions compared to reduced polynomials, and that on $F_5$ are assumed that inefficiency of selection strategy of polynomial pairs.

Last we analyze algorithms through computing Gröbner bases by another term order and changing the ordering, instead of computing Gröbner bases by lexicographic ordering directly. We implement an algorithm using Gröbner walk and compare the computation times. In consequence a lot of time was spent in computing normal forms, therefore the algorithm did not make speeding-up compared to a method computing a Gröbner basis with respect to a lexicographic order directly.
論文要旨

グレブナ基底は連立代数方程式の求解やイデアルの所属判定問題などにおいて非常に有用なものである。多項式イデアルのグレブナ基底を求める算法としては、長らく従来の Buchberger によるアルゴリズムが用いられてきたが、最近になって Faugère による $F_4$、$F_5$ アルゴリズムが提案された。$F_4$ アルゴリズムは多項式の定義を一度に列でとる形をとることで、計算時間の大幅を占める正規形の計算を高速化するという考え方であり、いくつかの例でその高速性が実証されている。一方で、イデアルがトーリックである場合、$F_4$ アルゴリズムにおいて生成される列式は非常に複雑なものとなり、効率よくグレブナ基底を求めることが難しい。一方の $F_5$ アルゴリズムは新しい多項式の生成時に前の計算の情報を記憶することで、無駄な多項式の生成を完全に排除するものであるが、トーリックイデアルに対する有効性はまだ検証されていない。

また、トーリックイデアルのグレブナ基底は、代数幾何学や組み合わせ論において重要な概念であり、イデアルの幾何的構造に着目してグレブナ基底を求めるアルゴリズムが Hosten や Robbiano により提案されている。$F_4$ や $F_5$ はこれらの手法には実行速度の面では及ばないが、イデアルを生成する列式の性質や順序による実行時間の変化を解析することは、$F_4$ や $F_5$ の特徴について深く知見を得る他に、イデアルの構造に新たな意味付けを与えるものと期待される。

そこで本研究ではまず、$F_4$ アルゴリズムにおいて生成される列式の性質に着目したデータ構造およびアルゴリズムの改善を行う。イデアルがトーリックである場合、多項式の順序に生成される列式は単純であるという利点を持っている。この場合、列式の基本形を有向グラフ上の操作に置き換えることができ、ネットワーク上の問題と見なすことができる。これにより、実行速度および使用メモリ量が従来の $F_4$ アルゴリズムより大きく改善された。

次に、$F_4$、$F_5$ アルゴリズムの実行速度を従来の Buchberger アルゴリズムと比較したところ、順序が全数逆辞書式順序である場合は従来の Buchberger アルゴリズムより良い結果が得られたものの、辞書式順序である場合は良い結果が得られなかった。この原因は、$F_4$ の場合は節約される多項式に比べて節約に使う多項式の数が多くなるため、$F_5$ の場合は多項式ペアの選択効率が悪いためと考えられる。

さらに、最初から辞書式順序でグレブナ基底を求めるのではなく、異なる順序によってグレブナ基底を求める (Change of Ordering) アルゴリズムを使用した手法に着目する。そのためのアルゴリズムとして Gröbner walk を用いた方法を実装し、実行時間の比較を行った。結果、Gröbner walk 中の正規形を求め部分に多くの時間がかかるが、最初から辞書式で求める方法と比較して効果を上げることはできなかった。
Acknowledgements

I would like to thank Prof. IMAI Hiroshi, who is my supervisor, for his helpful advice and suggestions about the theme of this thesis. I also would like to thank Prof. TAKAYAMA Nobuki and Prof. NORO Masayuki for their useful suggestions and discussion at a past workshop.

Finally I would like to thank all members of IMAI laboratory and Dr. ISHIZEKI Takayuki.
## Contents

1 **Introduction** ................................................. 1  
   1.1 Background .................................................. 1  
   1.2 Our objective ............................................... 2  
   1.3 Organization of this thesis ................................. 3  

2 **Preliminaries** ........................................... 4  
   2.1 Ideals ........................................................ 4  
      2.1.1 Basic definitions ....................................... 4  
      2.1.2 Toric ideals ........................................... 6  
   2.2 Gröbner bases .............................................. 7  
      2.2.1 Buchberger algorithm .................................. 10  
      2.2.2 Some criteria .......................................... 11  
      2.2.3 Strategies .............................................. 11  
   2.3 State Polytopes ........................................... 12  

3 **F₄ algorithm** ............................................. 16  
   3.1 Overview of F₄ algorithm .................................. 16  
   3.2 Improvements of F₄ algorithm ............................. 19  
      3.2.1 Improvements of Data Structure ....................... 19  
      3.2.2 Improvements of Reduction Algorithm ............... 20  

4 **F₅ algorithm** ............................................. 23  
   4.1 Overview of F₅ algorithm .................................. 23  
      4.1.1 Optimal Criterion ..................................... 24  
   4.2 Description of F₅ algorithm .............................. 25  
   4.3 Adjustment of F₅ algorithm .............................. 28  

5 **Change of Ordering algorithm** ......................... 30  
   5.1 Gröbner walk .............................................. 30  
   5.2 Improvements for toric ideals ........................... 33
6 Experiments

6.1 Conditions of the experiments .................................................. 35
6.2 Results of improved $F_4$ algorithm .............................................. 36
6.3 Results of $F_5$ algorithm ......................................................... 40
6.4 Results of Change of Ordering algorithm ..................................... 43

7 Concluding remarks ............................................................... 44

References ................................................................. 46
# List of Figures

2.1 $P$ in example 2.43 ................................................. 13  
2.2 The normal cone of $F_1$ at $P$ (left) and the normal fan of $P$ (right) in Example 2.45 14  
2.3 Gröbner fan of $I$ in Example 2.51 ................................................. 14  
3.1 A digraph associated to $v$ .................................................. 20  
3.2 A digraph associated to $v'$ .................................................. 21  
5.1 move of $\omega$ in Gröbner walk .................................................. 31  
6.1 Comparison of gr, old and new $F_4$ on $A_n$ wrt DRL (sec) .................. 37  
6.2 Comparison of gr, old and new $F_4$ on $A_n$ wrt LEX (sec) .................. 38  
6.3 Comparison of gr, $F_4$ and $F_5$ on $A_n$ wrt DRL (sec) .................. 40  
6.4 Comparison of gr, $F_4$ and $F_5$ on $A_n$ wrt LEX (sec) .................. 41
List of Tables

6.1 Matrices and the number of elements of generators . . . . . . . . . . . . . . . . . . . . . . 36
6.2 Computation time of gr, old and new $F_4$ on $A_n$ wrt DRL (sec) . . . . . . . . . . . . 36
6.3 Computation time of gr, old and new $F_4$ on $A_n$ wrt LEX (sec) . . . . . . . . . . . . 37
6.4 Computation time of gr, old and new $F_4$ on $A_n^h$ wrt DRL (sec) . . . . . . . . . . . . 38
6.5 Computation time of gr, old and new $F_4$ on $A_n^h$ wrt LEX (sec) . . . . . . . . . . . . 39
6.6 Computation time of gr, old and new $F_4$ on $P_{3 \times 3}$ wrt DRL (sec) . . . . . . . . . . 39
6.7 Computation time of gr, old and new $F_4$ on $P_{3 \times 3}$ wrt LEX (sec) . . . . . . . . . . 39
6.8 Computation time of gr, new $F_4$ and $F_5$ on $A_n$ wrt DRL (sec) . . . . . . . . . . . . 40
6.9 Computation time of gr, new $F_4$ and $F_5$ on $A_n$ wrt LEX (sec) . . . . . . . . . . . . 41
6.10 Computation time of gr, new $F_4$ and $F_5$ on $A_n^h$ wrt DRL (sec) . . . . . . . . . . . . 42
6.11 Computation time of gr, new $F_4$ and $F_5$ on $A_n^h$ wrt LEX (sec) . . . . . . . . . . . . 42
6.12 Comparison between LEX and DRL+Gröbner walk on $A_n$ (sec) . . . . . . . . . . . . . 43
6.13 Comparison between LEX and DRL+Gröbner walk on $P_{3 \times 3}$ (sec) . . . . . . . . . . 43
Chapter 1

Introduction

1.1 Background

Gröbner basis was first proposed by Buchberger [6] in order to compute of ideals on polynomial ring and to inspect structures of modules. This basis has several good properties and enabled us to describe various problems on computational algebra and algebraical geometry as definite algorithms, for example, the ideal membership problem and primary decomposition of ideals. In result algorithms using Gröbner basis are implemented on many computation algebra systems nowadays [1, 20, 27]. Gröbner basis is also widely used as one of the main tools for eliminating variables and solving systems of nonlinear algebraic equations. Additionally applications of Gröbner bases span to integer programming [12], invariant theory [28], coding theory [26] and so on. Therefore in recent years, the importance to improve efficiencies of computing Gröbner bases is increasing.

As an algorithm to compute Gröbner basis, Buchberger algorithm [7, 9] is the most general method. Although this algorithm is so simple, the explosions of time and memory space during the computation are tremendous. Then some improvements of the algorithm were proposed. Especially following approaches were commonly used:

- To decrease the number of pairs to compute by eliminating useless pairs [8, 18].
- If the ideal is toric, to focus on geometric properties of polytopes associated to certain matrices [24, 21].

Meanwhile Faugère recently proposed new algorithms $F_4$ [15] and $F_5$ [16]. The former focuses on the reduction step of polynomials, which is one of dominant steps of Buchberger algorithm. By regarding the reduction as transformation of matrices, computation times spent in reduction are pretty lowered and further improvements is expected by parallelizations. Whereas the latter improves Buchberger algorithm by generating no useless pair (not partial, but complete elimination) using past calculation results. By using these two algorithms, it is said that some previously intractable problems were solved.
Toric ideals are specific case of ideals, whose generators are binomial and it represents the algebraic relations of finite sets of power-products. The importance of toric ideals is derived from the fact that toric ideals show up in many problems arising from various field, for instance integer programming and combinatorics. Especially in network flow problems, Gröbner bases of toric ideals generated from certain incidence matrices have close relations with test sets, hence the structures of Gröbner bases were studied in order to inspect the complexity of problems for various network problems (i.e. minimum cost flow problems and multidimensional transportation problems) [23]. Therefore to improve the efficiency to compute Gröbner bases of toric ideals is greatly expected.

In multivariable systems, since the result of division algorithm varies according to term orderings, the form of Gröbner basis also varies according to them. To obtain a Gröbner basis with respect to desired ordering, instead of computing Gröbner basis wrt the ordering directly, it may be faster to compute Gröbner basis with respect to another ordering first, then change the ordering by specific algorithm. We call those algorithms Change of Ordering algorithms. For example, [2, 11, 17] are well known methods.

1.2 Our objective

Now we focus on $F_4$ and $F_5$ algorithms to compute Gröbner bases of toric ideals. Since little theoretical results [3] were found about the complexity of computing Gröbner basis so far, it is important to inspect the behavior of algorithms on specific instances through computational experiments. Toric ideals have very simple forms; which are generated by only binomials and their coefficients are 1 and $-1$. For original $F_4$ algorithm, toric ideals are thought the worst case because of the sparsity of constructed matrices when polynomials are being reduced. By improving the data structures, however, we can replace transformations on matrices with that on digraphs, in consequence the reduction step becomes considerably efficient. On the other hand, although $F_5$ algorithm is expected to make improvements on selections of pairs, enough experiments have not done yet. These algorithms are thought that they do not surpass to algorithms using triangulations of polytopes, but the analyses of $F_4$ and $F_5$ using toric ideals are expected to give us more insight about $F_4$ and $F_5$ algorithm.

Additionally, the forms of Gröbner bases vary according to the fixed term orders. Although it is convenient for solving equations to compute Gröbner bases with respect to a pure lexicographic order, computations with that order may be very time-consuming. Then we compare two algorithms for computing Gröbner bases with respect to a lexicographic order; one computes the desired basis directly, and the other first does compute a Gröbner basis with respect to a graded reverse lexicographic order and then obtains the desired basis by a specific algorithm, i.e. Change of Ordering algorithm. We use Gröbner walk algorithm in this thesis. This algorithm is said the effectiveness in cases that each polynomial has many terms, now we apply this algorithm
to toric cases, where each polynomial is binomial.

1.3 Organization of this thesis

This thesis is organized as follows. In Chapter 2 we give some important definitions of toric ideals, Gröbner basis and state polytopes. In Chapter 3 first we introduce \( F_k \) algorithm, then we propose improvements of this algorithm specialized for the case of toric ideals. In Chapter 4 we introduce a new criterion and \( F_5 \) algorithm. To work the criterion accurately, we adjust a part of this algorithm. In Chapter 5 we consider of Change or Ordering algorithm. As such algorithm, we use Gröbner walk algorithm and specialized it for the case of toric ideals. In Chapter 6 results of experiments of \( F_k, F_5 \) and Change of Ordering algorithms on various instances and term orderings are shown. Finally in Chapter 7 we conclude this thesis.
Chapter 2

Preliminaries

In this chapter we give basic definitions ideals, toric ideals and Gröbner bases. We refer to [13, 14] for the introduction of ideals and Gröbner bases, and [29, 30] for the introduction of toric ideals.

Definition 2.1 In this paper, $\mathbb{N}^+, \mathbb{N}, \mathbb{Z}, \mathbb{Q}$ and $\mathbb{R}$ represent sets of natural numbers, non-negative integers, integers, rational numbers and real numbers respectively.

2.1 Ideals

2.1.1 Basic definitions

Let $k$ be a field and $k[x_1, \ldots, x_n]$ the polynomial ring with coefficients in $k$, and $\mathbf{x} = \{x_1, \ldots, x_n\}$ the set of variables of $k[x_1, \ldots, x_n]$. For an exponent vector $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{N}^n$, we denote $\mathbf{x}^\mathbf{a} = x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}$.

Definition 2.2 We define a total degree of $f = x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n} \in k[x_1, \ldots, x_n]$ by $\sum_{i=1}^n a_i$. We write it $\text{tdeg}(f)$.

Furthermore, for a weight vector $\omega$, we define a weighted degree $\text{deg}_\omega(f)$ by $\sum_{i=1}^n \omega_i a_i$ where $\omega = (\omega_1, \ldots, \omega_n)$.

Definition 2.3 A non-empty subset $I \subseteq k[x_1, \ldots, x_n]$ is an ideal if it satisfies the following conditions:

1. For any $f, g \in I$, $f - g \in I$

2. For any $f \in I$ and $h \in k[x_1, \ldots, x_n]$, $fh \in I$

Proposition 2.4 If $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ be polynomials, then

$$\langle f_1, \ldots, f_s \rangle := \left\{ \sum_{i=1}^s h_i f_i \bigg| h_1, \ldots, h_s \in k[x_1, \ldots, x_n] \right\}$$

is an ideal of $k[x_1, \ldots, x_n]$. 

4
We call \( \langle f_1, \ldots, f_s \rangle \) the ideal generated by polynomials \( f_1, \ldots, f_s \), and \{ \langle f_1, \ldots, f_s \rangle \} \) a generator of \( \langle f_1, \ldots, f_s \rangle \). By the Hilbert Basis Theorem, every ideal \( I \) over \( k[x_1, \ldots, x_n] \) is finitely generated, i.e., there exist finite \( f_1, \ldots, f_s \in I \) such that \( I = \langle f_1, \ldots, f_s \rangle \).

For ideals \( I, J \subset k[x_1, \ldots, x_n] \), we can define some algebraic operations naturally. Following operations are binary operations: to each pair of ideals, they associate a new ideal.

**Definition 2.5** If ideals \( I, J \in k[x_1, \ldots, x_n] \), then the sum of \( I \) and \( J \), denoted \( I + J \), is defined as the set

\[
I + J = \{ f + g \mid f \in I \text{ and } g \in J \}
\]

and this is also an ideal in \( k[x_1, \ldots, x_n] \). If \( I = \langle f_1, \ldots, f_r \rangle \) and \( J = \langle g_1, \ldots, g_s \rangle \), then \( I + J = \langle f_1, \ldots, f_r, g_1, \ldots, g_s \rangle \).

**Corollary 2.6** If \( f_1, \ldots, f_r \in k[x_1, \ldots, x_n] \), then

\[
\langle f_1, \ldots, f_r \rangle = \langle f_1 \rangle + \cdots + \langle f_r \rangle.
\]

**Definition 2.7** If ideals \( I, J \in k[x_1, \ldots, x_n] \), the the product, denoted \( I \cdot J \), is defined as the set

\[
I \cdot J = \{ f_1g_1 + \cdots + f_rg_r \mid f_1, \ldots, f_r \in I, g_1, \ldots, g_r \in J, r \in \mathbb{N}^+ \}
\]

and this is also an ideal in \( k[x_1, \ldots, x_n] \). If \( I = \langle f_1, \ldots, f_r \rangle \) and \( J = \langle g_1, \ldots, g_s \rangle \), \( I \cdot J \) is generated by the set of all products of generators of \( I \) and \( J \):

\[
I \cdot J = \langle f_ig_j \mid 1 \leq i \leq r, 1 \leq j \leq s \rangle.
\]

**Definition 2.8** If ideals \( I, J \in k[x_1, \ldots, x_n] \), then the intersection of \( I \) and \( J \), denoted \( I \cap J \), is the set of polynomials which belong to both \( I \) and \( J \).

As same as the cases of sum and product, \( I \cap J \) is also an ideal in \( k[x_1, \ldots, x_n] \).

**Theorem 2.9** Let ideals \( I, J \in k[x_1, \ldots, x_n] \). Then

\[
I \cap J = (tI + (1-t)J) \cap k[x_1, \ldots, x_n].
\]

**Definition 2.10** If \( I, J \in k[x_1, \ldots, x_n] \) are ideals, then \( I : J \) is the set

\[
I : J = \{ f \in k[x_1, \ldots, x_n] \mid fg \in I \text{ for all } g \in J \}
\]

and is called the ideal quotient of \( I \) by \( J \).

**Proposition 2.11** Let ideals \( I, J, K \subset k[x_1, \ldots, x_n] \). Then

1. \( I : k[x_1, \ldots, x_n] = I \).
2. $IJ \subset K$ if and only if $I \subset K : J$.

3. $J \subset I$ if and only if $I : J = k[x_1, \ldots, x_n]$.

**Definition 2.12** Let $A$ be a ring, let $I$ be an ideal in $A$ and $F$ be a non-zero divisor in $A$. The saturation of $I$ with respect to $F$ is the ideal

$$IA_F \cap A = \{ G \in A \mid GF^r \in I \text{ for some } r \in \mathbb{N}^+ \},$$

that we denote by $I : F^\infty$. The ideal $I$ is said to be $F$-saturated if $I = I : F^\infty$.

For a polynomial $f \in k[x_1, \ldots, x_n]$ and an ideal $J \subset k[x_1, \ldots, x_n]$, the following two subsets of $k[x_1, \ldots, x_n]$ are also ideals:

$$(J : f) = \{ g \in k[x_1, \ldots, x_n] \mid fg \in J \},$$

$$(J : f^r) = \{ g \in k[x_1, \ldots, x_n] \mid f^r g \in J \text{ for some } r \in \mathbb{N}^+ \}.$$  

**Lemma 2.13** Let $I$ be an ideal in $A$. Then

1. $I : (fg)^\infty = (I : f^\infty) : g^\infty$.

2. If $I$ is $f$-saturated and $g$-saturated, then $I$ is $fg$-saturated.

**Definition 2.14** Given a commutative unit ring $R$, and an $R$-module $M$, a sequence $\{x_1, \ldots, x_n\}$ of elements of $R$ is called a regular sequence for $M$, if for all $i = 1, \ldots, n$,

1. The multiplication by $x_i$ is injective on $M/\langle x_1, \ldots, x_{i-1} \rangle M$, and

2. $M/\langle x_1, \ldots, x_n \rangle M \neq 0$.

That is, in terms of a polynomial ring, for all $i = 1, \ldots, n$, $f_i \notin \langle f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_n \rangle$ stands.

### 2.1.2 Toric ideals

In this part, we consider $A \in \mathbb{Z}^{d \times n}$ as a set of column vectors $\{a_1, \ldots, a_n\}$. Each vector $a_i$ is identified with a monomial $t^{a_i}$ in the Laurent polynomial ring $k[t^{\pm 1}] = k[t_1, \ldots, t_d, t_1^{-1}, \ldots, t_d^{-1}]$.

**Definition 2.15** For a vector $u \in \mathbb{Z}^n$, we define a support of $u$ as follows:

$$\text{supp}(u) = \{ i \mid u_i \neq 0, u = (u_1, \ldots, u_n) \}.$$  

**Definition 2.16** Consider the homomorphism

$$\pi : k[x_1, \ldots, x_n] \longrightarrow k[t^{\pm 1}], \quad x_i \longmapsto t^{a_i}.$$

The image of $\pi$ is the semigroup $\mathbb{N}A = \{ \lambda_1 a_1 + \cdots + \lambda_n a_n \mid \lambda_1, \ldots, \lambda_n \in \mathbb{N} \}$, and the kernel of $\pi$ is denoted $I_A$ and called the toric ideal of $A$.  

6
Every vector $\mathbf{u} \in \mathbb{Z}^n$ can be written uniquely as $\mathbf{u} = \mathbf{u}^+ - \mathbf{u}^-$ where $\mathbf{u}^+$ and $\mathbf{u}^-$ are non-negative and have disjoint support.

**Lemma 2.17**

$$I_A = \langle \mathbf{a}^{\mathbf{u}^+} - \mathbf{a}^{\mathbf{u}^-} \mid \mathbf{u} \in \ker(A) \cap \mathbb{Z}^n, i = 1, \ldots, s \rangle.$$  

Furthermore, a toric ideal is generated by finite binomials.

From a matrix $A \in \mathbb{Z}^{d \times n}$, its generator of toric ideal $I_A$ is found by computing $I_L : (x_1 \cdots x_n)^\infty$ where $I_L$ denotes the ideal generated by the binomials associated with the vectors in $L \subseteq \text{Ker}(A)$ and $I_L : (x_1 \cdots x_n)^\infty$ denotes the saturation of $I_L$ with respect to $x_1 \cdots x_n$ [5]. From lemma 2.13, $I_L : (x_1 \cdots x_n)^\infty = \left((I_L : x_1^\infty) : x_2^\infty \cdots : x_n^\infty\right)$ stands. Therefore we can obtain the generators of toric ideals $I_i$ by computing saturations wrt $x_1, \ldots, x_n$ sequentially. A saturation $(I_L : x_i^\infty)$ is calculated by taking an ideal $H = I_L + (tx_i - 1) \in k[x_1, \ldots, x_n,t]$ and computing an intersection $H$ and $k[x_1, \ldots, x_n]$.

Above-mentioned algorithm is implemented in CoCoA [10].

**Example 2.18** Let $A = \begin{pmatrix} 1 & 3 & 1 & 5 \\ 1 & 2 & 3 & 2 \end{pmatrix}$. The kernel of $A$ is $\begin{pmatrix} -7 & 2 & 1 & 0 \\ 4 & -3 & 0 & 1 \end{pmatrix}^T$, therefore

$I_L = \langle x_1^2 - x_2^2 x_3, x_4^2 x_4 - x_5^2 \rangle$.

By computing the saturations sequentially, we obtain

$\left((I_L : x_1^\infty) : x_2^\infty : x_4^\infty\right) = \langle x_1^4 x_4 - x_2^2, -x_1^2 + x_2^2 x_3, -x_3^2 x_2 - x_3 x_4, -x_2^4 + x_1 x_3 x_4^2 \rangle$

We minimize this ideal and finally obtain the toric ideal $I_A = \langle x_1^4 x_4 - x_2^2, x_1^2 - x_2^2 x_3, x_4^2 x_2 - x_3 x_4 \rangle$.

This ideal is homogeneous with respect to a weight vector $\langle 2 & 5 & 4 & 7 \rangle$, which is a summation of each row of $A$.

### 2.2 Gröbner bases

To define a Gröbner basis of an ideal, it is first necessary to define a term order, i.e., a good total order on a set of monomials.

**Definition 2.19** A total order $\succ$ on the set of monomials in $k[x_1, \ldots, x_n]$ is a term order if it satisfies the following conditions:

1. For any $\mathbf{u} \in \mathbb{N}^n \setminus \{0\}$, $\mathbf{x}^\mathbf{u} \succ 1$

2. If $\mathbf{x}^\mathbf{u} \succ \mathbf{x}^\mathbf{v}$ and $\mathbf{w} \in \mathbb{N}^n$, then $\mathbf{x}^\mathbf{u} \mathbf{x}^\mathbf{w} \succ \mathbf{x}^\mathbf{v} \mathbf{x}^\mathbf{w}$

The following lists some well-known examples of term orders.
Definition 2.20 Fix variable ordering \( x_1 > x_2 > \cdots > x_n \). A (pure) lexicographic order (LEX) induced by this variable ordering is defined as follows:

\[
\mathbf{x}^u \succ_{\text{LEX}} \mathbf{x}^v \iff \text{the leftmost non-zero element of } (u_1 - v_1, \ldots, u_n - v_n) \text{ is positive.}
\]

Definition 2.21 Fix variable ordering \( x_1 > x_2 > \cdots > x_n \). A graded reverse lexicographic order (DRL) induced by this variable ordering is defined as follows:

\[
\mathbf{x}^u \succ_{\text{DRL}} \mathbf{x}^v \iff \text{the leftmost non-zero element of } \left( \sum_{k=1}^n (u_k - v_k), u_n - v_n, \ldots, v_1 - u_1 \right) \text{ is positive.}
\]

Definition 2.22 Fix variable ordering \( x_1 > x_2 > \cdots > x_n \), a term order \( \succ \), and let a vector \( \mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{Q}^n_{\geq 0} \). A weight order \( \succ_{\mathbf{c}} \) induced by this variable ordering and vector is defined as follows:

\[
\mathbf{x}^u \succ_{\mathbf{c}} \mathbf{x}^v \iff \sum_{k=1}^n c_k (u_k - v_k) > 0 \text{ or } \sum_{k=1}^n c_k (u_k - v_k) = 0 \text{ and } \mathbf{x}^u \succ \mathbf{x}^v.
\]

For a term order \( \succ \) and a polynomial \( f \in k[x_1, \ldots, x_n] \), we denote \( \text{in}_{\succ} (f) \) the largest term of \( f \) with respect to \( \succ \) and call the initial term of \( f \). Then, for an ideal \( I \subseteq k[x_1, \ldots, x_n] \), we define the initial ideal of \( I \) as \( \text{in}_{\succ} (I) = \langle \text{in}_{\succ} (f) \mid f \in I \rangle \).

Example 2.23 Let \( f = x_1^3 x_3 + x_2^5 + x_3^4 \). For some term orders, we determine the initial term of \( f \).

- If \( \succ \) is a lexicographic order induced by the variable ordering \( x \succ y \), then \( x_1^3 x_3 \succ x_2^5 \succ x_3^4 \) and \( \text{in}_{\succ} (f) = x_1^3 x_3 \).

- If \( \succ \) is a degree reverse lexicographic order induced by the variable ordering \( x \succ y \), then \( x_2^5 \succ x_1^3 x_3 \succ x_3^4 \) and \( \text{in}_{\succ} (f) = x_2^5 \).

- If \( \succ_{\mathbf{c}} \) is a weight order where \( \mathbf{c} = (1, 2, 5) \) and \( \succ \) is a degree reverse lexicographic order induced by the variable ordering \( x \succ y \), then \( x_3^4 \succ x_2^5 \succ x_1^3 x_3 \) and \( \text{in}_{\succ_{\mathbf{c}}} (f) = x_3^4 \).

Given a term order, we can define a Gröbner basis for an ideal with respect to the order.

Definition 2.24 Let \( I \) be an ideal over \( k[x_1, \ldots, x_n] \) and \( \succ \) a term order. First we define the initial ideal of \( I \) as \( \text{in}_{\succ} (I) = \langle \text{in}_{\succ} (f) \mid f \in I \rangle \). A Gröbner basis of \( I \) with respect to \( \succ \) is a finite set \( G_{\succ} = \{ g_1, \ldots, g_s \} \subseteq I \) such that

\[
\text{in}_{\succ} (I) = \langle \text{in}_{\succ} (g_1), \ldots, \text{in}_{\succ} (g_s) \rangle.
\]
That is, if \( G_{\succ} = \{g_1, \ldots, g_s\} \) is a Gröbner basis of \( I \), for any \( f \in I \), there exists some \( g_i \in G \) such that \( \text{in}_{\succ}(g_i) \) divides \( \text{in}_{\succ}(f) \).

A Gröbner basis is called reduced if, for any \( i, j \) with \( i \neq j \), no term of \( g_i \) is divisible by \( \text{in}_{\succ}(g_j) \) and the coefficient of \( \text{in}_{\succ}(g_i) \) is 1 for any \( i \). When an ideal and a term order are fixed, its reduced Gröbner basis is determined uniquely.

**Theorem 2.25** Fix an order \( \succ \) and a Gröbner basis \( G = \{g_1, \ldots, g_t\} \) for \( I \) with respect to \( \succ \). Then every \( f \in k[x_1, \ldots, x_n] \) can be written as

\[
f = a_1g_1 + \cdots + a_tg_t + r, \quad a_1, \ldots, a_t, r \in k[x_1, \ldots, x_n]
\]

where either \( r = 0 \) or no term of \( r \) is divisible by any of \( \text{in}_{\succ}(g_1), \ldots, \text{in}_{\succ}(g_t) \). Then \( r \) is uniquely determined and called normal form of \( f \) by \( G \). We write \( r = f^G \). Each polynomial \( a_1, \ldots, a_t, r \) is calculated by following division algorithm.

**Algorithm 2.26 (The division algorithm)**

**Input:** polynomial \( f \), Gröbner basis \( G = \{g_1, \ldots, g_t\} \) and a term order \( \succ \)

**Output:** \( \{a_1, \ldots, a_t, r \mid f = a_1g_1 + \cdots + a_tg_t + r\} \)

\( a_1 = a_2 = \cdots = a_t = 0, r = 0; \)

\( p = f; \)

\textbf{while} \( (p \neq 0) \) \{
  \( i = 1; \)
  \( \text{divisionoccurred} = \text{false}; \)
  \textbf{while} \( (i \leq s \text{ and } \text{divisionoccurred} = \text{false}) \) \{
    \textbf{if} \( (\text{in}_{\succ}(f_i) | \text{in}_{\succ}(p)) \) \{
      \( a_i = a_i + \text{in}_{\succ}(p)/\text{in}_{\succ}(g_i); \)
      \( p = p - g_i \cdot \text{in}_{\succ}(p)/\text{in}_{\succ}(g_i); \)
      \( \text{divisionoccurred} = \text{true}; \)
    \} \textbf{else} \{
      \( i = i + 1; \)
    \}
  \}
  \textbf{if} \( (\text{divisionoccurred} = \text{false}) \) \{
    \( r = r + \text{in}_{\succ}(p); \)
    \( p = p - \text{in}_{\succ}(p); \)
  \}
\}

**Example 2.27** Let us divide \( f = x^2y + xy^2 + y^2 \) by \( g_1 = y^2 - 1 \) and \( g_2 = x - y \). \( \{g_1, g_2\} \) is a reduced Gröbner basis for \( \langle g_1, g_2 \rangle \) with respect to lexicographic order induced by the variable ordering \( x \succ y \).
In first step, since \( \text{in}_\succ (g_2) \) divides \( \text{in}_\succ (f) \), we obtain \( a_2 = xy \) and \( p = 2xy^2 + y^2 \). Next \( \text{in}_\succ (g_1) \) divides \( \text{in}_\succ (p) \), hence \( a_1 = 2x \) and \( p = 2x + y^2 \). Then \( \text{in}_\succ (g_2) \) divides \( \text{in}_\succ (p) \), thus \( a_2 = xy + 2 \) and \( p = y^2 + 2y \). Then \( \text{in}_\succ (g_1) \) divides \( \text{in}_\succ (p) \), finally \( a_1 = 2x + 1 \) and \( p = 2y + 1 \). Since \( \text{in}_\succ (p) \) is not divisible by neither \( \text{in}_\succ (g_1) \) nor \( \text{in}_\succ (g_2) \), we obtain \( r = 2y + 1 \). This shows that

\[
x^2y + xy^2 + y^2 = (2x + 1)\cdot (y^2 - 1) + (xy + 2)\cdot (x - y) + 2y + 1.
\]

### 2.2.1 Buchberger algorithm

As an algorithm for computing Gröbner basis, Buchberger algorithm is most basic and well-known. In this algorithm, \( S \)-polynomial plays an important role.

**Definition 2.28** Let \( f, g \in k[x_1, \ldots, x_n] \) be nonzero polynomials. We denote as the least common multiple of \( \text{in}_\succ (f) \) and \( \text{in}_\succ (g) \) by \( \gamma \), i.e. \( \gamma = \text{LCM} (\text{in}_\succ (f), \text{in}_\succ (g)) \). Then we define the \( S \)-polynomial of \( f \) and \( g \) as

\[
S(f, g) = \frac{\gamma}{\text{in}_\succ (f)} \cdot f - \frac{\gamma}{\text{in}_\succ (g)} \cdot g.
\]

**Proposition 2.29** Let \( I \) be an ideal. Then a basis \( \mathcal{G} = \{g_1, \ldots, g_l\} \) for \( I \) is Gröbner basis for \( I \) if and only if \( S(g_i, g_j) = 0 \) for all pairs \( i \neq j \).

With this proposition, Buchberger [9] constructed an algorithm for computing a Gröbner basis.

**Algorithm 2.30 (Buchberger Algorithm)**

**Input:** \( F = \{f_1, \ldots, f_s\} \subset k[x_1, \ldots, x_n] \) and a term order \( \succ \)

**Output:** A Gröbner basis \( \mathcal{G} \) for \( I = \langle f_1, \ldots, f_s \rangle \) with respect to \( \succ \)

\( \mathcal{G} = F; \)

\[ \text{do } \]

\[ \quad \mathcal{G}' = \mathcal{G}; \]

\[ \quad \text{for each pair } \{p, q\}, p \neq q \text{ in } \mathcal{G}' \{ \]

\[ \quad \quad S = S(p, q)'; \]

\[ \quad \quad \text{if } S \neq 0 \text{ then } \mathcal{G} = \mathcal{G} \cup \{S\}; \]

\[ \quad \} \]

\[ \text{while } (\mathcal{G} \neq \mathcal{G}') \]

**Proposition 2.31** For any term order and any ideal, a Gröbner basis can be computed in finite steps by the above algorithm.

**Proposition 2.32** Let ideals \( I, J \subset k[x_1, \ldots, x_n] \) and a term order \( \succ \). Then \( I = J \) if and only if a reduced Gröbner bases of \( I \) with respect to \( \succ \) equals that of \( J \) with respect to \( \succ \).

Therefore we have a facultativity on choosing generators of toric ideals.
2.2.2 Some criteria

By Buchberger algorithm, a Gröbner basis is guaranteed to be computed in finite steps. However, if that algorithm is applied without any modification, it costs enormous time and space, because of explosion of polynomial pairs. In fact, most of generated S-polynomial are useless, that is, their normal forms is 0. Then we need to eliminate these useless pairs with simple checks. As such checking methods, following criteria are used:

First we introduce an appropriate total order among all pairs. In following sentence, we write LCM\(in_\succ(g_i), in_\succ(g_j)\) as \(T_{ij}\) and \(in_\succ(g_k)\) as \(T_k\).

**Definition 2.33** Let \(\mathcal{G} = \{g_1, \ldots, g_t\}\). For \(\{S\text{pol}(g_i, g_j) \mid 1 \leq i < j \leq t\}\), a total order \(<\) is defined as follows:

\[
S\text{pol}(i, j) < S\text{pol}(k, l) \iff T_{ij} < T_{kl} \quad \text{or} \\
T_{ij} = T_{kl} \text{ and } (j < l \text{ or } (j = l \text{ and } i < k))
\]

**Definition 2.34** For each pair \((i, j)\), following three properties are defined:

1. \(M(i, j) \iff \) There exists some \(k < j\), then \(T_{kj} \mid T_{ij}\) and \(T_{kj} \neq T_{ij}\) are satisfied.
2. \(F(i, j) \iff \) There exists some \(k < i\), then \(T_{kj} = T_{ij}\) is satisfied.
3. \(B(i, j) \iff \) There exists some \(k > j\), then \(T_{k} \mid T_{ij}\), \(T_{jk} \neq T_{ij}\) and \(T_{ik} \neq T_{ij}\) are satisfied.

**Proposition 2.35** ([18]) Let \(g_i, g_j \in \mathcal{G}\). For a pair \((i, j)\), if at least one of \(M(i, j), F(i, j)\) and \(B(i, j)\) is satisfied, then \(\overline{S\text{pol}(g_i, g_j)}^\mathcal{G} = 0\).

In addition to above-mentioned criteria, following Buchberger criterion is also used.

**Proposition 2.36** Let \(g_i, g_j \in \mathcal{G}\). If \(\text{GCD}(in_\succ(g_i), in_\succ(g_j)) = 1\), then \(\overline{S\text{pol}(f, g)}^\mathcal{G} = 0\).

2.2.3 Strategies

In Buchberger algorithm, the way that which pair to select first for computing S-polynomial does not affect the finite termination of the algorithm. However, the selection strategy of pairs affect the efficiencies of computation severely. First Buchberger proposed below strategy.

**Definition 2.37** (Normal strategy) From \(F = \{f_1, \ldots, f_m\}\), Normal strategy chooses a pair such that \(T_{ij}\) is minimum with respect to term order \(\succ\) and compute \(S\text{pol}(f_i, f_j)\).

This strategy intends to generate elements of the bases whose head terms are low with respect to \(\succ\) first. But this strategy may give disastrous results in case of lexicographic order, which does not compare pairs by total degree.

Giovini et al. proposed a new strategy [19], which introduces sugar, a virtual degree of a homogenized polynomial.
Definition 2.38 Let $t$ be a variable for homogenization and we denote $R = k[x_1, \ldots, x_n]$ and $R_h = k[x_1, \ldots, x_n, t]$. We write the homogenization of $f \in R$ as $f^h$ and the dehomogenization of $g \in R_h$ as $g_h$ and define them as follows.

$$f^h = t^{\text{tdeg}(f)} f(x_1/t, \ldots, x_n/t)$$

$$g_h = g|_{t=1}$$

For sets of polynomials $F \subset R$ and $G \subset R_h$, we define their homogenization and dehomogenization as $F^h = \{f^h \mid f \in F\}$ and $G_h = \{g_h \mid g \in G\}$. When for an order $<$ on $R$, an order $<_h$ on $R_h$ satisfies

For any polynomials $g \in R_h$, $(\text{in}_{<_h}(g))_h = \text{in}_<(g_h)$,

then we call $<_h$ a homogenization of $<$.

Proposition 2.39 Let $F \subset k[x_1, \ldots, x_n]$ and $<_h$ be a homogenized order of $<$. Then if $G$ is a Gröbner basis of homogeneous polynomials in $\langle F^h \rangle$, $G_h$ is a Gröbner basis of $\langle F \rangle$.

Definition 2.40 (Sugar strategy) For each polynomial $f \in F$, we relate $f$ to $s_f \in \mathbb{N}$ according to following rules.

1. If $f$ is an element of input polynomials, $s_f = t^{\text{tdeg}(f)}$.

2. If $m$ is a monomial, $s_{mf} = t^{\text{tdeg}(m)} + s_f$.

3. $s_{f+g} = \max(s_f, s_g)$.

From $F = \{f_1, \ldots, f_m\}$, sugar strategy choose a pair $(f_i, f_j)$ such that the sugar of $\text{Spol}(f_i, f_j)$ is minimum and compute $\text{Spol}(f_i, f_j)$.

2.3 State Polytopes

In this section, we introduce the state polytope [4] of an ideal $I$. It has the property that the vertices are in a bijection with the distinct reduced Gröbner bases for $I$.

At first, we review some basic concepts from polyhedral geometry.

Definition 2.41 A polyhedron is an intersection of finite closed half-spaces in $\mathbb{R}^n$.

Definition 2.42 Let $P$ be a polyhedron in $\mathbb{R}^n$ and $c \in \mathbb{R}^n$. We define the face of $P$ with respect to $c$ by

$$\text{face}_c(P) = \{u \in P : c \cdot u \geq c \cdot v \text{ for all } v \in P\}.$$
Example 2.43 Let
\[ P = \{ x = (x_1, x_2) \in \mathbb{R}^2 : -2 \leq x_1, x_2 \leq 2 \} \]

When \( c_1 = (1, 1) \), then \( \text{face}_{c_1}(P) = \{(1, 1)\} \) and when \( c_2 = (1, 0) \), then \( \text{face}_{c_2}(P) = \{(1, x_2) : -1 \leq x_2 \leq 1\} \) (see Figure 2.1).

Definition 2.44 Let \( P \subseteq \mathbb{R}^n \) be a polyhedron and \( F \) be a face of \( P \). The normal cone of \( F \) at \( P \) is
\[ N_P(F) = \{ c \in \mathbb{R}^n : \text{face}_c(P) = F \}. \]

The collection of normal cones as \( F \) ranges over all the faces of \( P \) is called the normal fan of \( P \) and written by \( \mathcal{N}(P) \).

Example 2.45 Let \( P \) be the same polyhedron as Example 2.43. Then the normal cone of \( F_1 = \{(1, 1)\} \) at \( P \) is the shaded area in Figure 2.2(left). The normal fan of \( P \) is drawn in Figure 2.2(right). Let
\[ F_1 = \{(1, 1)\}, F_2 = \{(-1, 1)\}, F_3 = \{(-1, -1)\}, F_4 = \{(1, -1)\} \]
be faces of \( P \). Then in Figure 2.2(right), the cone \( C_i \) (\( i = 1, 2, 3, 4 \)) is the normal cone of \( F_i \) at \( P \).

Next we introduce the Gröbner fan [25] of an ideal \( I \) and state polytope of \( I \).

Definition 2.46 Fix \( \omega \in \mathbb{R}^n \). For any polynomial \( f = \sum_i c_i x^{a_i} \), we define the initial form \( \text{in}_\omega(f) \) to be the sum of all terms \( c_i x^{a_i} \) such that the inner product \( \omega \cdot a_i \) is maximal. We define the initial ideal of \( I \) with respect to \( \omega \) as
\[ \text{in}_\omega(I) = \langle \text{in}_\omega(f) : f \in I \rangle \]
Figure 2.2: The normal cone of $F_1$ at $P$ (left) and the normal fan of $P$ (right) in Example 2.45

**Remark 2.47 ([30], Proposition 1.11.)** For any term order $\succ$ and any ideal $I$, there exists a non-negative integer vector $\omega \in \mathbb{N}^n$ such that $\text{in}_\omega(I) = \text{in}_{\cdot \omega}(I)$.

**Definition 2.48** Let $I \subset k[x_1, \ldots, x_n]$ be an ideal. Two weight vectors $\omega_1, \omega_2 \in \mathbb{R}^n$ are called equivalent with respect to $I$ if and only if $\text{in}_{\omega_1}(I) = \text{in}_{\omega_2}(I)$.

**Proposition 2.49** The set of all weight vectors that are equivalent to $\omega \in \mathbb{R}^n$ form a relatively open polyhedral cone in $\mathbb{R}^n$, the closure of which is called the Gröbner cone of $\omega$.

**Definition 2.50** We define the Gröbner fan of $I$ to be the collection of all Gröbner cones of $I$.

**Example 2.51** Let an ideal $I = (xy + x + y) \subset k[x, y]$. Then $I$ has three Gröbner fans (Figure 2.3):

$$
C_1 = \{(\omega_1, \omega_2) : \text{in}_{(\omega_1, \omega_2)}(I) = (xy)\}
$$

$$
C_2 = \{(\omega_1, \omega_2) : \text{in}_{(\omega_1, \omega_2)}(I) = (y)\}
$$

$$
C_3 = \{(\omega_1, \omega_2) : \text{in}_{(\omega_1, \omega_2)}(I) = (x)\}
$$

Figure 2.3: Gröbner fan of $I$ in Example 2.51
Theorem 2.52 Let $I$ be a homogeneous ideal in $k[x_1, \ldots, x_n]$. Then there exists a polytope $St(I) \subset \mathbb{R}^n$ whose normal fan $N(St(I))$ coincides with the Gröbner fan of $I$. This is called state polytope of $I$.

Corollary 2.53 Let $I$ be an ideal in $k[x_1, \ldots, x_n]$. Then $I$ has finite distinct reduced Gröbner bases.
Chapter 3

$F_4$ algorithm

Faugère's $F_4$ algorithm [15] aims to improve reductions of $S$-polynomials. The key point of $F_4$ consists of two parts, which are Symbolic Preprocessing and reduction by matrices.

3.1 Overview of $F_4$ algorithm

In normal Buchberger algorithm, we choose only one pair whose degree (or sugar) is minimal and reduce it by many reducers. In general, only a small number of reducers are used for reduction of $S$-polynomial. To find such reducers may be time-consuming, hence we improve this step by generating multiple $S$-polynomial at once and selecting reducers. We call the selection of reducers Symbolic Preprocessing.

First we define a simple version of Symbolic Preprocessing, which selects candidates of polynomials to reduce one $S$-polynomial. To obtain a uniform designation, we introduce below notations in following sentence.

- $T(f) \cdots$ a set of all terms appearing in $f$.
- $\text{reductum}(f) \cdots$ a set of terms appearing in $f$ except the head term of $f$.

Algorithm 3.1 (Highlight of Symbolic Preprocessing)

Input: $S$-polynomial $f$, a set of polynomials $G$

Output: $\{ah | a$ is monomial, $h \in G\}$

$T = T(f)$;

$\text{Red} = 0$;

while ($\exists g \in G, \exists t \in T | \text{in}_\succ(g)$ divides $t$)

$\text{Red} = \text{Red} \cup \{t / \text{in}_\succ(g) \cdot g\}$;

$T = (T \setminus \{t\}) \cup T(\text{reductum}(t / \text{in}_\succ(g) \cdot g))$;

}

return $\text{Red}$;

This output $\text{Red}$ has following properties.
• If \( t \in T(\{f\} \cup \text{Red}) \) is divided by \( \text{in}_\succ (g) \mid g \in G \), there exists \( x \in \text{Red} \) where \( \text{in}_\succ (x) = t \).

• The normal form of \( \{f\}\) by \( \text{Red} \) becomes that of \( \{f\}\) by \( G \).

We rewrite these procedures in terms of matrices. First we denote all terms appearing in \( \{f\} \cup \text{Red} \) as \( T \), then we sort elements of \( T \) decreasingly, that is \( t_1 > t_2 > \cdots \). Each polynomial is represented as a row vector whose basis is \( (t_1, t_2, \ldots) \). We denote the polynomial associated to \( i \)-th row of \( A \) as \( \text{poly}(A_i) \). A matrix \( A \) consists of a set of those row vectors, i.e. for \( S \)-polynomial \( f \) and a set of reducers \( \text{Red} = \{r_1, \ldots, r_m\} \), the matrix becomes as follows:

\[
A = \begin{pmatrix}
  f_1 & f_2 & \cdots \\
  r_{11} & r_{12} & \cdots \\
  \vdots & \vdots & \ddots \\
  r_{m1} & r_{m2} & \cdots 
\end{pmatrix}
\]

In above matrix, \( f_i \) and \( r_{ki} \) are the coefficients of \( f \) and \( r_k \) associated to \( t_i \). Next by elementary row transformation, we transform this matrix to a matrix \( B \) which satisfies a following property.

• If \( B_i \neq 0 \), then \( B_k (i \neq k) \) does not contain \( \text{in}_\succ (\text{poly}(B_i)) \).

That means \( B \) is a diagonalized matrix. Finally, the normal form \( \mathcal{F}^G \) is \( \text{poly}(B_i) \) such that \( \text{in}_\succ (\text{poly}(B_i)) \) is not contained in \( \{\text{in}_\succ (r) \mid r \in \text{Red}\} \). Now we note that when \( S \)-polynomial is begin reduced, used operations are only addition (or subtraction) of polynomials and multiplication of constant. Unlike usual reductions, multiplication of monomial is unnecessary, because reducers are already multiplied in Symbolic Preprocessing.

We can easily extend this algorithm to select candidates of polynomials to reduce multiple \( S \)-polynomials.

**Algorithm 3.2 (Symbolic Preprocessing)**

**Input:** A set of \( S \)-polynomials \( F \), a set of polynomials \( G \)

**Output:** \( \{ah \mid a \text{ is monomial}, h \in G\} \)

\[
T = \bigcup_{f \in F} T(f); \\
\text{Red} = 0; \\
\text{while} (\exists g \in G, \exists t \in T \mid \text{in}_\succ (g) \text{ divides } t) \\
\text{Red} = \text{Red} \cup \{t/\text{in}_\succ (g) \cdot g\}; \\
T = (T \setminus \{t\}) \cup T(\text{reductum}(t/\text{in}_\succ (g) \cdot g)); \\
\text{return} \text{Red};
\]

Let \( F = \{f_1, \ldots, f_n\} \) be a set of \( S \)-polynomials and \( G = \{g_1, \ldots, g_m\} \) be reducers. Now a matrix is constructed as follows:
\[ A = \begin{pmatrix}
  f_{11} & f_{12} & \cdots \\
  f_{n1} & f_{n2} & \cdots \\
  r_{11} & r_{12} & \cdots \\
  \vdots & \vdots & \ddots \\
  r_{m1} & r_{m2} & \cdots 
\end{pmatrix} \]

Likewise the case \( S \)-polynomial is only one, \( f_{ki} \) and \( r_{ki} \) are the coefficients of \( f_k \) and \( r_k \). For a matrix \( B \), which is a diagonalized matrix of \( A \), we divide every row into two groups \( F' \) and \( \text{Red}' \).

\[
F' = \{ h = \text{poly}(B_i) \mid h \neq 0, \text{in}_\succ(h) \not\in \{\text{in}_\succ(r) \mid r \in \text{Red}\} \}
\]
\[
\text{Red}' = \{ h = \text{poly}(B_i) \mid h \neq 0, \text{in}_\succ(h) \in \{\text{in}_\succ(r) \mid r \in \text{Red}\} \}
\]

**Proposition 3.3**

1. \( h' \in F' \) is normal form for \( G \cup (F' \setminus \{ h \}) \).
2. If \( f \in F \), then \( f^{\text{Red}' \cup F} \equiv 0 \).

**Example 3.4** Let \( S \)-polynomials \( f_1 = 2x^2 + 3x + 6y^2 - 3y - 1, f_2 = 2xy + x + 3y^2 - y - 1 \) and reducers \( g_1 = 2x^2 - 3y + 2, g_2 = x + 2y^2 - 1 \). And we fix a variable order on \( x > y \) and a term order on lexicographic order.

First we construct a following matrix.

\[
A = \begin{pmatrix}
  2 & 0 & 3 & 6 & -3 & -1 \\
  0 & 2 & 1 & 3 & -1 & -1 \\
  2 & 0 & 0 & 0 & -3 & 2 \\
  0 & 0 & 1 & 2 & 0 & -1
\end{pmatrix}
\]

The row of \( A \) is associated to polynomials \( f_1, f_2, g_1, g_2 \) and the column of \( A \) is associated to terms \( (x^2, xy, x, y^2, y, 1) \) from left to right. By elementary row transformation, we obtain a following diagonalized matrix \( B \).

\[
B = \begin{pmatrix}
  2 & 0 & 0 & 0 & -3 & 2 \\
  0 & 2 & 0 & 1 & -1 & 0 \\
  0 & 0 & 1 & 2 & 0 & -1 \\
  0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

Then we obtain \( F' = \{\text{poly}(B_2)\} \) and \( \text{Red}' = \{\text{poly}(B_1), \text{poly}(B_3)\} \). As a result of reduction, one of \( S \)-polynomial is reduced to \( 2xy + y^2 - y \), and the other is reduced to 0.

As same as Buchberger algorithm, the selection strategy of pairs is important for \( F_1 \) algorithm. But it is known that the affection of selection strategy decreases because multiple pairs can be chosen at once. For example, following strategy is often used.

**Definition 3.5** Choose all pairs whose sugar of the \( S \)-polynomials become minimum.
3.2 Improvements of $F_4$ algorithm

$F_4$ algorithm can be regarded as a variant of Buchberger algorithm, and has following advantages.

- Comparisons of terms with respect to a term order $\succ$ are replaced with only those of indexes of columns of the matrices.
- The bases are kept to be nearly reduced during whole computation.

On the other hand, $F_4$ algorithm has several problems. Many $S$-polynomials and reducers are represented as a matrix at once, therefore huge memory space is required. Additionally, for the sparsity of the matrices, the structure of data is needed to be efficient. If the ideal is toric, generated $S$-polynomials have properties as below:

- Every polynomial is binomial.
- About the binomial, the coefficient of one term is 1, and that of the other is $-1$. Without loss of generality, we can determine the coefficient of a leading term is 1.

From the point of view of sparsity of matrices, toric ideals are assumed to be the worst instance for $F_4$ algorithm. Therefore we represent the matrices with more efficient structure of data and refine the reduction algorithm.

3.2.1 Improvements of Data Structure

Let $A \in \{0,\pm1\}^{d \times n}$. Instead of this matrix, we use a vector $\mathbf{v} \in \mathbb{Z}^d_{\leq 0}$. $\mathbf{v}$ has a same information with $A$ and is defined as follows:

**Definition 3.6** Let $A \in \{0,\pm1\}^{d \times n}$ and $a_i$ be the $i$-th row vector of $A$. Each $a_i$ contains 1 and $-1$ respectively only one time, and we can assume the column index associated to coefficient 1 is smaller than that to coefficient $-1$.

Then for each $a_i = \begin{pmatrix} 0 & 1 & 0 & -1 & 0 \end{pmatrix}$, where two indexes associated to nonzero coefficients are $i < j$, we define $\mathbf{v}[i] \succ j$.

**Example 3.7** Let $A = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & -1 & 0 & 0 \end{pmatrix}$. Then $\mathbf{v}[1] = \{2, 3\}, \mathbf{v}[2] = \{4\}, \mathbf{v}[3] = \mathbf{v}[4] = \emptyset$.

**Remark 3.8** As we say afterwards, rows of a coefficient matrix $A$ is read and processed sequentially, thus any $\mathbf{v}[i]$ never have more than two elements at the same time like above example 3.7.
3.2.2 Improvements of Reduction Algorithm

Now we remark that after a reduction, the vector (or matrix) becomes diagonalized. To consider the reduction algorithm intuitively, we represent \( \mathbf{v} \) as digraph.

**Definition 3.9** We define an incidence graph of a vector \( \mathbf{v} \) (or a matrix \( A \)) as follows:

- **Vertices** \( \cdots \) the set of all indexes
- **Edges** \( \cdots \) \( \{(i,j) \mid \mathbf{v}[i] \ni j\} \)

**Example 3.10 (Continue from Example 3.7)** The vertices of the incidence graph of \( \mathbf{v} \) are \( \{1,2,3,4\} \) and the edge of that are \( \{\{1,2\},\{1,3\},\{2,4\}\} \). See below figure 3.1.

![Figure 3.1: A digraph associated to \( \mathbf{v} \)](image)

By reduction algorithm, the vector \( \mathbf{v} \) is reduced to a new vector \( \mathbf{v}' \). In terms of digraph, the term that reduced is newly defined.

**Definition 3.11** If following two conditions

- Outdegrees of all vertices are at most 1, and
- If an outdegree of vertex \( v_i \) is 1 for some \( i \), then an indegree of vertex \( v_i \) vertex is 0 are satisfied, then the incidence graph is reduced.

In terms of a vector, “reduced” means (i) for all index \( i \), \( \mathbf{v}'[i] \) contains at most 1 elements and (ii) if \( \mathbf{v}'[i] = \{j\} \) for some \( i > j \), then \( \mathbf{v}'[j] = \emptyset \).

**Example 3.12 (Continue from Example 3.10)** For the matrix \( A \) in Example 3.7, the result of reduction is \( B = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \) and \( \mathbf{v}'[1] = \mathbf{v}'[2] = \mathbf{v}'[3] = \{4\}, \mathbf{v}'[4] = \emptyset \). See figure 3.2.
Now we propose the reduction algorithm specifically. The algorithm consists of two steps, the forward step and the backward step. In the forward step, let the result of step be $v''$, the numbers of all elements of $v''$ are assured at most 1. Additionally after the backward step, if $v'[i] = \{j\}$ for some $i > j$, then $v'[j] = 0$ is guaranteed. We remark this algorithm processes the input data sequentially, therefore a vector $v$ is not constructed actually.

Algorithm 3.13 (Reduction Algorithm on vector)

**Input:** $\{p_1, \ldots, p_d = (i, j) \mid 1 \leq i < j \leq n\}$

**Output:** $v' = v[1], \ldots, v'[n]$

**forward step:**

$v'[1] = \cdots = v'[n] = 0$

for ($i = 1$ to $d$) {
    Read $p_i$
    if ($v'[i] = 0$)
        $v'[i] = j$
    else {
        $k = i; \ l = v'[i]$
        while ($v'[k] \neq 0$) {
            if ($j = 1$) break;
            $j_1 = \max(j, l); \ j_2 = \min(j, l)$;
            $v'[k] = j_1$
            $k = j_2; \ l = j_1$
        }
        $v'[k] = l$
    }
}
backward step:
for (i = n downto 1) {
    if (v'[i] ≠ 0)
        v'[j] = v'[i] for all j such that v'[j] = i;
}
return v';

As the complexity of these procedures, the forward step may cost \(O(d^2)\) time complexity in the worst cases, e.g. \(p_i = (i, i + 2) \mid 1 \leq i \leq \lfloor \frac{d}{2} \rfloor \}, \{p_{\lfloor \frac{d}{2} \rfloor + 1} = (1, 2)\} \) and \(p_i = (1, i + 2 - \lfloor \frac{d}{2} \rfloor) \mid \lfloor \frac{d}{2} \rfloor + 2 \leq i \leq d\}. \) But in most cases it costs only \(O(n + d)\). Contrary the backward step only costs \(O(n)\) time apparently. Hence whole reduction algorithm takes \(O(n + d)\) time on average and \(O(n)\) memory space. Since it takes \(O(n^3)\) time and \(O(n^2)\) memory space for diagonalization of matrices in general, we can say the new reduction algorithm achieves a great improvement.

**Example 3.14 (Continue from Example 3.12)** Based on the matrix \(A\) in Example 3.7, let \(p_1 = (1, 3), p_2 = (2, 4)\) and \(p_3 = (1, 2)\). First we initialize an array \(v' = (0, 0, 0, 0)\). In a forward step, the algorithm reads \(p_1, p_2\) then \(v'[1] = 3\) and \(v'[2] = 4\). Next \(p_3\) is read and since \(v'[1] ≠ 0, v'[1] = \max(2, 3) = 3, v'[2] = \max(3, 4) = 4\) and \(v'[3] = 4\). Now a forward step ends and we obtain \(v' = (3, 4, 0)\).

Subsequently by a backward step, \(v'[1] = v'[3] = 4\) for \((i, j) = (1, 3)\). The the reduction algorithm terminates and we obtain \(v' = (4, 4, 4, 0)\). This vector corresponds to the graph represented in Figure 3.2.
Chapter 4

\(F_5\) algorithm

Faugère’s \(F_5\) algorithm [16] introduced a new optimal criteria instead of Buchberger criteria. The new criteria ensures that it generates no useless critical pairs if the input is a regular sequence.

4.1 Overview of \(F_5\) algorithm

Let \((f_1, \ldots, f_m)\) be a polynomial \(m\)-tuple and \(I\) the ideal generated by \((f_1, \ldots, f_m)\). First we associate a unique and canonical signature for all the elements of \(T(I)\), that is to say all the leading terms of all the polynomials in the ideal. In following sentence, we denote \(\mathcal{P} = k[x_1, \ldots, x_n]\) is a polynomial ring.

We consider the evaluation function \(v:\)

\[v \left( \begin{array}{c}
\mathcal{P}^m \\
g = (g_1, \ldots, g_m) \\
\sum_{i=1}^{m} f_i g_i
\end{array} \right) \rightarrow \mathcal{P} \]

For \(\mathbf{F}_i\), which is the canonical \(i\)-th unit vector in \(\mathcal{P}^m\), we have \(v(\mathbf{F}_i) = f_i\) and \(g = \sum_{i=1}^{m} g_i \mathbf{F}_i\). An \(m\)-tuple \(g = (g_1, \ldots, g_m)\) is called a syzygy if \(v(g) = 0\). The so-called principal syzygies \(s_{i,j} = f_j \mathbf{F}_i - f_i \mathbf{F}_j\) are syzygies. Let \(\text{PSyz}\) be the module generated by the principal syzygies.

We can extend the admissible ordering \(<\) to \(\mathcal{P}^m\) with the following definition:

**Definition 4.1** For any element in \(\mathcal{P}^m\), \(\sum_{k=i}^{m} g_k \mathbf{F}_k > \sum_{k=j}^{m} h_k \mathbf{F}_k\) stands if and only if

\[(i > j \text{ and } h_j \neq 0) \text{ or } (i = j \text{ and } \text{in}_{>}(g_i) < \text{in}_{>}(h_i))\]

**Proposition 4.2** Let \(\omega\) be

\[
\begin{pmatrix}
T & \rightarrow & \mathcal{P}^m \\
t & \mapsto & \min_{>} W(t)
\end{pmatrix}.
\]

If \((t_1, t_2) \in T(I)^2\), then \(\text{in}_{>}(\omega(t_1)) \neq \text{in}_{>}(\omega(t_2))\) if \(t_1 \neq t_2\).

**Corollary 4.3** For all the polynomials \(f \in I\), we define \(v_1(f)\) to be \(\text{in}_{>}(\omega(\text{in}_{>}(f)))\). If \(\text{in}_{>}(f_1) \neq \text{in}_{>}(f_2)\) for \(f_1\) and \(f_2\), then we have \(v_1(f_1) \neq v_2(f_2)\).
We call $v_1(f)$ the *signature* of $f$: it is unique and does not depend on the order of the computations. In algorithm $F_5$, we have a pair $r = (v_1(f), f) \in R$ instead of $f$ and call it an *expanded polynomial*. $r = (eF_k, f)$ means that there exists an representation $f = eF_k + \sum_{i=k+1}^n a_if_i$ where $f_1 \supset \cdots \supset f_k \supset \cdots \supset f_n$. We introduce below notations for $r = (tF_i, f)$; $\text{poly}(r) = f$, $\mathcal{S}(r) = tF_i$ and $\text{index}(r) = i$. For $0 \neq \lambda \in k, v$ be monomial, and $r \in R$ we define $\lambda r = (uF_i, \lambda p)$ and $vr = (uvF_i, vp)$. We also extend the definition of usual operators to $R$: $\text{in}_\prec(r) = \text{in}_\prec(\text{poly}(r))$ and $\text{NF}(r, G)$, which is the normal form of $r$ by $G \in \mathcal{P}$, equals to $(\mathcal{S}(r), \text{NF}(\text{poly}(r), G))$.

### 4.1.1 Optimal Criterion

We introduce two definitions from [16].

**Definition 4.4** Let $P$ be a finite subset of $R$, and $r, t \in R$. If

$$\text{poly}(r) = \sum_{p \in P} m_p p \quad m_p \in k[x_1, \ldots, x_n]$$

we say it is a $t$-representation of $r$ if $\text{in}_\prec(t) > \text{in}_\prec(m_pp)$ and $\mathcal{S}(t) \supset \mathcal{S}(m_pp)$ for all $p \in P$. We write this property as $f = o_P(t)$.

**Definition 4.5** We say that $r \in R$ is normalized if $\mathcal{S}(r) = eF_k$ and $e$ is not reduced by $\langle f_{k+1}, \ldots, f_m \rangle$. If $r$ such that $u \in (tF_i, r) \in R$ is normalized, we define that $(u, r)$ is normalized. Additionally we say a pair of expanded polynomials $(r_i, r_j)$ is normalized if $\mathcal{S}(r_j) \supset \mathcal{S}(r_i), (u_i, r_i)$ and $(u_j, r_j)$ are normalized, where $u_i = T_{ij}/\text{in}_\prec(r_i)$ and $u_j = T_{ij}/\text{in}_\prec(r_j)$.

We remark if obtained the Gröbner basis of ideal $\langle f_{k+1}, \ldots, f_m \rangle$ in advance, we can determine if $r \in R$ is reducible using divisibility check of head terms.

**Theorem 4.6** Let $F = \{f_1, \ldots, f_m\}$ be polynomials in $k[x_1, \ldots, x_n]$ and $I = \langle F \rangle$. Let $G = \{r_1, \ldots, r_{ng} \} \in R^{ng}$ such that

- $F \subset \text{poly}(G)$. Let $g_i = \text{poly}(r_i)$ and $G_1 = \{g_i \mid i = 1, \ldots, ng\}$.
- For all $(i, j) \in \{1, \ldots, ng\}$ such that the pair $(r_i, r_j)$ is normalized, $S\text{pol}(g_i, g_j) = o_{G_1}(u_i r_i)$ or 0 where $u_i = T_{ij}/\text{in}_\prec(r_i)$.

Then $G_1$ is a Gröbner basis of $I$.

For the proof, see [16].

In $F_5$ algorithm, we add *simplification rules* when a new expanded polynomial (and signature) is generated. The rules are stored as an array of lists. That is, we add a pair $(t, k)$ to a head of $\text{Rule}[i]$ when $r_k = (tF_i, f)$ is newly generated.
The following function checks whether a product \( u \times r_k \) is simplified with simplification rules:

**Rewritten** \( (u, r_k) \) if \( u \) is a term, \( r_k = (tF_i, f_k) \)

**Rule** \( [i] = [(t_1, k_1), \ldots, (t_r, k_r)] \);

for \( j = 1 \) to \( r \) {
    if \( (t_j | u) \) {
        if \( (k = k_j) \) then return false;
        else return true;
    }
}
return false;

**Example 4.7** Let expanded polynomials \( r_4 = (F_2, f_4) \) and \( r_6 = (xF_2, f_6) \) are generated in advance. Then \( \text{Rule}[2] = [(x, 6), (1, 4)] \). Now \( \text{Rewritten}(y^2, r_4) \) returns false and \( \text{Rewritten}(xy, r_4) \) returns true because \( xyf_4 \) is simplified to \( yf_6 \).

For a pair \( (r_i, r_j) \) \( \in R \), we denote \( u_i = T_i / \text{in}_{\succ}(\text{poly}(r_i)), u_j = T_j / \text{in}_{\succ}(\text{poly}(r_j)) \). If either \((u_i, r_i)\) or \((u_j, r_j)\) is simplified with an existing rule, from theorem 4.6, we can assure \( \text{Spol}(\text{poly}(r_i), \text{poly}(r_j)) \) is reduced to 0.

**4.2 Description of F_5 algorithm**

To use the fact of theorem 4.6, we sort input polynomials as a descending order \( f_1 \succ \cdots \succ f_n \) and compute a Gröbner basis of \( \langle f_i, \ldots, f_n \rangle \) for each addition of a polynomial \( f_i \in I \). We call it incremental algorithm. Whole \( F_5 \) algorithm is iterations of following algorithm from \( n \) to 1. Description of following algorithm is from a paper of Faugère [16].

**Algorithm 4.8** (incremental step of \( F_5 \))

**Input:** polynomial \( f_i, G_{i+1} \subset R \) where \( \text{poly}(G_i) \) is a Gröbner basis of \( \langle f_{i+1}, \ldots, f_n \rangle \),

**term order \( \succ \)**

**Output:** Gröbner basis \( \text{G} \) of \( \langle f_i, \ldots, f_m \rangle \)

\( r_i = (F_i, f_i) \);
\( G_i = G_i \cup \{r_i\} \);
\( P = \{\text{CritPair}(r_i, r) \mid r \in G_{i+1}\} \);

while \( (P \neq \emptyset) \) {
    \( P_d = \text{subset of } P \) where the degree of lcm is minimal;
    \( P = P \setminus P_d \);
    \( F = \text{Spol}(P_d) \);
    \( R_d = \text{Reduction}(F, G_i) \);
    for each \( (r \in R_d) \) {
        \( \)
\[ P = P \cup \{ \text{CritPair}(r, g) \mid g \in G_i \}; \]
\[ G_i = G_i \cup \{ r \}; \]
\}
\}
return \( G_i \);

Subfunctions in this algorithm are described as follows:

CritPair has two expanded polynomials \( r_1, r_2 \) as arguments. If a pair \((r_1, r_2)\) is normalized, this function returns a critical pair \((\tau, r_i, u_i, r_j, u_j)\) where \( \tau = \text{LCM}(\text{in}(r_i), \text{in}(r_j)) \) and \( u_i = \tau/r_i, u_j = \tau/r_j \).

CritPair
Input: \( r_1, r_2 \in R \)
Output: \([\text{lcm}, u_1, r_1, u_2, r_2] \) or \( \emptyset \)
if \( S(r_1) \prec S(r_2) \) Swap\((r_1, r_2)\);
\( t_1F_{k_1} = S(r_1), t_2F_{k_2} = S(r_2) \);
if \( (k_1 > k) \) then return \( \emptyset \);
\( \text{lcm} = \text{lcm}(\text{in}(r_1), \text{in}(r_2)) \);
\( u_1 = \text{lcm}/\text{in}(r_1), u_2 = \text{lcm}/\text{in}(r_2) \);
if \( (u_1t_1 \) is reducible by \( G_{i+1} \) then return \( \emptyset \);
if \( (k_2 = i \) and \( u_2t_2 \) is reducible by \( G_{i+1} \) then return \( \emptyset \);
return \([\text{lcm}, u_1, r_1, u_2, r_2] \);

For each surviving critical pair, optimal criterion is tested by \( \text{Spol} \). If the pair can be written with rules, the pair is removed by optimal criterion.

Spol
Input: \( P = [p_1, \ldots, p_h] \) a list of critical pairs
Output: \( F \) a list of \( R \)
for each \( (p_i = [\text{lcm}, u_1, r_1, u_2, r_2]) \) {
if (either \( (u_1, r_1) \) or \( (u_2, r_2) \) is simplified) continue;
\( N = N + 1 \);
\( r_N = (u_1S(r_1), u_1\text{poly}(r_1) - u_2\text{poly}(r_2)) \);
add Rule\((r_N)\);
\( F = F \cup \{ r_N \} \);
}
return \( F \);

As a major difference with Buchberger algorithm, the reduction of a polynomial in \( F_5 \) algorithm may return several polynomials. In function \( \text{Reduction} \), polynomials are reduced by \( \text{TopReduction} \) in ascending order for signatures. The result of \( \text{TopReduction} \) is a pair \((r, F')\),
where \( r \) is a reduced (expanded) polynomial and \( F' \) is a list of polynomials which are being reduced once again. \( F' = 0 \) means that \( r \) is irreducible (or zero). Contrary \( F' \neq 0 \) (then \( r = 0 \)) means all the elements of \( F' \) are being reduced by \( \text{TopReduction} \).

**Reduction**

**Input:** \( S \)-polynomials \( ToDo \) and reducer \( G \)

**Output:** reduced polynomials \( Done \)

while \( (ToDo \neq \emptyset) \) {
  choose the minimal element \( h \) for \( S \) and remove \( h \) from \( ToDo \);
  reduce \( h \) by \( G_{i+1} \);
  \( (h_1, ToDo_1) = \text{TopReduction}(h, G \cup Done) \);
  \( Done = Done \cup h_1 \);
  \( ToDo = ToDo \cup ToDo_1 \);
}

return \( Done \);

**TopReduction**

**Input:** \( h \in R \) and reducer \( G \)

**Output:** normal form of \( h \) or new \( ToDo \)

\( r' = IsReducible(h, G) \);

if \( (r' = 0) \) return \( (h, 0) \);
else {
  \( u = \text{in}_\succ(h)/\text{in}_\succ(r') \);
  if \( (uS(r') \prec S(h)) \) {
    \( \text{poly}(h) = \text{poly}(h) - \text{upoly}(r') \);
    return \( (\emptyset, \{h\}) \);
  }
  else {
    \( N = N + 1 \);
    \( r_N = (uS(r'), \text{upoly}(r') - \text{poly}(h)) \);
    add Rule\( (r_N) \);
    return \( (\emptyset, \{r_N, r'\}) \);
  }
}

Function \( IsReducible \) checks whether \( h \) is reducible by \( G \). For \( r \in G \), if all of four following conditions

1. \( \text{in}_\succ(h) \) is divided by \( \text{in}_\succ(r) \).

2. \( (u, r) \) is normalized where \( u = \text{in}_\succ(h)/\text{in}_\succ(r) \).
3. \((u, r)\) is not be simplified by function \(Rewritten\).

4. \(uS(r) \neq S(h)\).

are satisfied, then \(h\) is reduced by \(r\).

IsRedducible

**Input:** expanded polynomial \(h\) and reducer \(G\)

**Output:** reducer \(g_i \in G\) such that reduces \(h\), or \(\emptyset\)

\(t_0F_{k_0} = S(h)\);

for each \((g_i \in G)\) {
\[
  tF_k = S(g_i);
\]

if \(((u = in_\omega(poly(h))/in_\omega(poly(g_i))) \text{ is a term}) \text{ and}
\]

\((ut \text{ is not reduced by } G_{i+1}) \text{ and}
\]

\(((u, g_i) \text{ is not simplified}) \text{ and}
\]

\((utF_k \neq t_0F_{k_0})\)

then return \(g_i\);

}

return \(\emptyset\);

### 4.3 Adjustment of \(F_5\) algorithm

\(F_5\) algorithm is said to work well in many examples, but when there exists a polynomial whose terms are not disjoint each other.

**Example 4.9** Let \(F = \{f_1, f_2, f_3, f_4\} = \{x_3x_7 - x_2x_8, x_3x_7 - x_1x_9, x_6x_7 - x_4x_9, x_5x_8 - x_4x_9\}\). We compute the Gröbner basis of \(F\) with respect to graded reverse lexicographic order. By \(F_5\) algorithm, we obtain \(G_2 = \{r_2, r_3, r_4, r_5 = (x_6F_2, f_5 = x_3x_4x_9 - x_1x_6x_9)\}\), which is a Gröbner basis of \(\{f_2, f_3, f_4\}\). We remark \(r_2 = (F_2, f_2), r_3 = (F_3, f_3)\) and \(r_4 = (F_4, f_4)\).

Then we read \(r_1 = (F_1, f_1)\). A critical pair \((r_1, r_2)\) is normalized and \(r_6 = (F_1, x_2x_8 - x_1x_9)\) is newly generated. Next by \((r_6, r_4)\), \(r_7 = (x_5F_1, x_2x_4x_9 - x_1x_5x_9)\) is generated. By \((r_7, r_5)\), \(r_8 = (x_3x_5F_1, x_1x_3x_5x_9 - x_1x_2x_6x_9)\) is generated. Now \(G_1 = \{r_4, r_3, r_2, r_5, r_1, r_6, r_7, r_8\}\).

Next we apply Critpair to a critical pair \((r_8, r_5)\). \(lcm(in_\omega(r_8), in_\omega(r_5)) = x_1x_2x_4x_5x_9\), then \(u_1 = x_4\) and \(u_2 = x_1x_5\). Neither \(x_4 \cdot x_3x_5\) nor \(x_1x_5 \cdot x_6\) is not top reducible by \(\{r_2, r_3, r_4, r_5\}\), therefore the pair \((r_8, r_5)\) is normalized.

Since \((r_8, r_5)\) is normalized, we apply optimal criterion to \((r_8, r_5)\). Since both \(Rewritten(x_4, r_8)\) and \(Rewritten(x_1x_5, r_5)\) return false, \(F_5\) generates \(r_9 = (x_3x_4x_5F_1, f_9)\) where \(f_9 = Spol(f_8, f_5) = x_1x_2x_4x_6x_9 - x_2^2x_5x_6x_9\). But \(f_9\) is reduced to 0 by \(f_7\). This contradicts to the definition of \(F_5\) that there is no reduction to 0.
In above example, the cause of contradiction is thought to be a imperfect definition of normalized. In fact, from a following equation

\[ Spol(f_8, f_5) = -x_4 f_8 + x_1 x_5 f_5 = x_1 x_6 f_7 \]

we obtain a \( x_4 f_8 \)-representation of \( r_9 \) since \( \text{in}_{\prec} (x_4 f_8) > \text{in}_{\prec} (x_1 x_6 f_7) \) and \( S(x_4 f_8) = x_3 x_4 x_6 F_1 > S(x_1 x_6 f_7) = x_1 x_5 x_6 F_1 \). From theorem 4.6, this pair \((r_8, r_5)\) should be eliminated, but optimal criterion does not work well in this case.

Therefore we need to adjust the definition of normalized. A binomial \( Spol(f_8, f_5) = -x_1 x_6 (x_2 x_4 x_9 - x_1 x_5 x_9) \) has a common factor \( x_9 \) since both \( f_8 \) and \( f_5 \) have a common factor \( x_9 \). Consequently we rewrite algorithm \textit{CritPair}, to consider a common factor of two terms in each binomial.

\textbf{New CritPair}

\textbf{Input}: \( r_1, r_2 \in R \)

\textbf{Output}: \([\text{lcm}, u_1, r_1, u_2, r_2]\) or \( \emptyset \)

\begin{align*}
\text{if } S(r_1) &< S(r_2) \text{ Swap}(r_1, r_2); \\
t_1 F_{k_1} &= S(r_1), t_2 F_{k_2} = S(r_2); \\
\text{if } (k_1 > k) \text{ then return } \emptyset; \\
\text{lcm} &= \text{lcm}(\text{in}_{\prec}(r_1), \text{in}_{\prec}(r_2)); \\
u_1 &= \text{lcm} / \text{in}_{\prec}(r_1), u_2 = \text{lcm} / \text{in}_{\prec}(r_2); \\
p_1 &= \text{GCD of two terms in } \text{poly}(r_1), p_2 = \text{GCD of two terms in } \text{poly}(r_2); \\
\text{if } (u_1 t_1 p_1 \text{ is reducible by } G_{i+1}) \text{ then return } \emptyset; \\
\text{if } (k_2 = i \text{ and } u_2 t_2 p_2 \text{ is reducible by } G_{i+1}) \text{ then return } \emptyset; \\
\text{return } [\text{lcm}, u_1, r_1, u_2, r_2];
\end{align*}

Now we apply new \textit{Critpair}.

\textbf{Example 4.10 (Continue from Example 4.9)} Let \( F \) be a set of polynomials in example 4.9.

As same as a past example, we obtain \( G_1 = \{ r_4, r_3, r_2, r_1, r_6, r_7, r_8 \} \). Then we apply \textit{Critpair} to \((r_8, r_5)\). Since \( u_1 = x_4 \) and \( p_1 = x_9, x_4 \cdot x_3 x_5 \cdot x_9 \) is top reducible by \( f_5 \). Hence a pair \((r_8, r_5)\) is not normalized and its \( S \)-polynomial is not computed. Also a critical pair \((r_8, r_7)\) is identified not normalized by new \textit{Critpair}.

As a result, the \( F_5 \) algorithm terminates and it returns \( G_1 = \{ g_2, g_3, g_4, g_5, g_6, g_7, g_8 \mid g_i = \text{poly}(r_i) \} \) as a reduced Gröbner basis of \( F \).
Chapter 5

Change of Ordering algorithm

With a view to some applications, e.g. solving multivariable systems, it is generally convenient to compute Gröbner bases with respect to a lexicographic order. But it may be pretty time-consuming to compute desired Gröbner bases directly, therefore a method using Change of Ordering algorithm is frequently adopted. This means that first we compute Gröbner bases with less time-consuming order (e.g. a graded reverse lexicographic order) and then we change the ordering of bases using a specific algorithm. As such algorithms, FGLM algorithm [17] for zero-dimensional ideals and an algorithm with LLL [2] for two variables system are also known, but now we apply an algorithm using Gröbner walk [11] in this thesis.

5.1 Gröbner walk

The Gröbner walk algorithm takes as input two term orders \(<, \ll\) and the Gröbner basis \(G_<\) of ideal \(I\). To compute another Gröbner basis \(G_{\ll}\), the algorithm sequentially constructs a finite number of term orders \(< = \ll_0, \ldots, \ll_m = \ll\) and Gröbner bases \(G_1, \ldots, G_m\) such that \(G_k\) is a Gröbner basis with respect to \(\ll_k\). For any \(i\), Gröbner cones associated to \(G_i\) and \(G_{i+1}\) are adjacent each other, therefore we can compute \(G_{i+1}\) from \(G_i\) with less complexity.

A weight vector \(\omega\) is an element that satisfies

\[
\Omega^n = \{ (\psi_1, \ldots, \psi_n) \in \mathbb{Q}^n \mid \psi_i \geq 0 \text{ for all } 1 \leq i \leq n \}.
\]

\(\omega\) moves from a starting point associated to \(<\) toward a goal point associated to \(\ll\), across a line segment in Gröbner fan (see figure 5.1). An \(\omega\)-degree of a monomial \(t = \alpha x_1^{a_1} \cdots x_n^{a_n}\) is defined by

\[
\deg_\omega(t) = \sum_{i=1}^n a_i \omega_i.
\]

For a polynomial \(f = t_1 + t_2 + \cdots + t_m\), \(\deg_\omega(f)\) is defined by \(\max(\deg(t_1), \ldots, \deg(t_2))\).

**Definition 5.1** We define \(\text{in}_\omega(f)\) as an initial form of \(f\) with respect to \(\omega\), which consists of all monomials \(t_{i_1}, \ldots, t_{i_k}\) in \(f\) such that \(\deg(t_{i_1}) = \cdots = \deg(t_{i_k}) = \deg_\omega(f)\).
For a set of polynomials $G$, we denote a set \( \{ \text{in}_\omega(g) \mid g \in G \} \) by $G_\omega$ and an ideal \( \langle \{ \text{in}_\omega(g) \mid g \in G \} \rangle \) by $\langle I_\omega \rangle$. For a vector $\omega$ and a term order $\bar{\prec}$, we can define another term order $\bar{\prec}_\omega$ (see 2.22). In this sentence, we denote this order by $\omega^\prec$.

**Definition 5.2** For an ideal $I$ and term order $\prec$, we define a Gröbner cone $C_\prec(I)$ as a topological closure of \( \{ \omega \in \Omega^n \mid \langle I_\omega \rangle = \langle I_\omega \rangle \} \).

This is a convex cone in $\mathbb{Q}^n$ with nonempty interior. The Gröbner fan of $I$ is the set $C_\prec(I)$ for all term order $\prec$.

Let $\prec$ and $\prec'$ be term orders whose cones contain a common weight vector $\omega$ (i.e., $\omega \in C_\prec(I) \cap C_{\prec'}(I)$) and $G = \{ g_1, \ldots, g_r \}$ be a reduced Gröbner basis of $I$ with respect to $\prec$. Then a Gröbner basis of $I$ with respect to $\prec'$ is obtained by following lemma:

**Lemma 5.3** ([11]) Let $\{ m_1, \ldots, m_s \}$ be the reduced Gröbner basis of $\langle I_\omega \rangle$ with respect to $\prec'$. Then, for all $i = 1, \ldots, s$, there is a representation

$$m_i = \sum_{j=1}^{r} h_{ij} \text{in}_\omega(g_j)$$

with $\omega$-homogeneous polynomials $h_{i1}, \ldots, h_{ir}$. Furthermore, the set $\{ f_1, \ldots, f_s \}$ with $f_i = \sum_{j=1}^{r} h_{ij}g_j$ is a Gröbner basis of $I$ with respect to $\prec'$.

We call the operation that constructs $f_i$ from $m_i$ 'lift of $m_i$'.

Now we describe a full picture of Gröbner walk algorithm. First we determine two weight vectors $\sigma, \tau$ associated to $\prec, \ll$. To satisfy a demand, we choose $\sigma, \tau$ such that $\sigma \in C_\prec(I)$ and $\tau \in C_{\ll}(I)$. When $\prec$ is graded reverse lexicographic order and $\ll$ is lexicographic order, we can take $\sigma = (1, 1, \ldots, 1)$ and $\tau = (1, 0, \ldots, 0)$. When we walk on the Gröbner fan of $I$, $\omega$ moves along the line segment $[\sigma, \tau]$. Therefore every weight vectors $\omega_1, \ldots, \omega_m$ are represented as $\{(1 - t)\sigma + t\tau \mid 0 \leq t \leq 1\}$.

Now we show a process to obtain $G_{\prec k}$ from $G_{\prec k-1}$ and $\omega_k$, where $\prec_{k-1} = \omega_{k-1}^{\ll}$ and $\prec_k = \omega_k^{\ll}$.

1. From $G_{\prec k-1}$ and $\omega_k$, we construct $\langle G_{\prec k-1} \omega_k \rangle = \{ \text{in}_{\omega_k}(g) \mid g \in G_{\prec k-1} \}$, which is a Gröbner basis of $\langle I_{\omega_k} \rangle$ with respect to $\prec_{k-1}$. 

31
2. By Buchberger algorithm, we compute the reduced Gröbner basis of \((G_{<k-1})_{\omega_k}\) with respect to \(<_k\). We denote it by \(G\).

3. We reduce each polynomial \(m_i \in \{m_1, \ldots, m_s\} = G\) modulo the Gröbner basis \((G_{<k-1})_{\omega_k}\) and obtain the following representation

\[
m_i = \sum_{j=1}^{r} h_{ij} in_{\omega_k}(g_j)
\]

with \(g_j \in (G_{<k-1})_{\omega_k}\). Then by lifting \(m_i\), we obtain

\[
f_i = \sum_{j=1}^{r} h_{ij} g_j.
\]

Since \(\omega_k \in (C_{<k-1}(I) \cap C_{<k-1}(I))\), the set \(F = \{f_1, \ldots, f_s\}\) is a Gröbner basis of \(I\) with respect to \(<_k\), which we call \(G_{<k}\).

4. Determine the next weight vector \(\omega_{k+1}\) and goto 1.

For determining the next vector, we use a following lemma.

**Lemma 5.4** Let \(<, <'\) be two term orders and \(G = \{g_1, \ldots, g_r\}\) the reduced Gröbner basis of \(I\) with respect to \(<.\) Then, \(C_{<}(I) = C_{<'}(I)\) if and only if \(in_{<}(g_i) = in_{<'} g(i)\) for every \(i \in \{1, \ldots, r\}\).

That is, we can move the current weight vector \(\omega(t) = (1-t)\sigma + t\tau\) toward \(\tau\) until encountering a facet of the Gröbner fan. We determine the next vector \(\omega(t') = (1-t)\sigma + t'\tau\) where \(t'\) satisfies the following property.

\[
t' = \min\{s : \deg_{\omega(s)}(p_1) = \deg_{\omega(s)}(p_i) | g = p_1 + \cdots + p_n, g \in G\} \cap (t, 1]\)

If \(t' = 1\), that is the next vector is equal to \(\tau\), the next conversion is the last step of the walk. Because the Gröbner fan of a polynomial ideal has finite cardinality, Gröbner walk algorithm terminates in finite steps.

The full description of Gröbner walk is as follows:

**Algorithm 5.5 (Gröbner walk algorithm)**

Input: orders \(<, \ll,\) reduced Gröbner basis \(G_{<}\)

Output: reduced Gröbner basis \(G_{\ll}\)

Fix \(\sigma \in C_{<}(G_{<})\) and \(\tau \in C_{\ll}(G_{<})\);

\(G = G_{<}; \omega = \sigma; <_{\ll} = \omega_{<}; <_{+} = \omega_{\ll}\);

while (1) {

\(G_{\omega} = \{in_{\omega}(g) | g \in G\}\);

\(G_{\omega}^+ = \text{Gröbner basis of } G_{\omega} \text{ wrt } \omega_{+}\);
\[ G = \{ f_i (= \text{lift of } m_i) \mid m_i \in G_\omega^+ \}; \]
\[ \text{if } (\omega = \tau) \text{ return } G; \]
\[ \text{determine new } \omega; \]
\[ \omega^+ = \omega^+; \quad \omega^+ = \omega^+; \]
\}

In this algorithm most elements of \( G_\omega \) are monomials, therefore Gröbner basis of \( G_\omega \) is expected to compute faster than that of \( G \).

### 5.2 Improvements for toric ideals

In this section we specialize Gröbner walk algorithm for cases that the input ideal is toric.

**Proposition 5.6** In algorithm 5.5, \( G_\omega \) and \( G_\omega^+ \) contains only monomials and binomials if the input ideal is toric.

We denote \( G_\omega \) by \( \{m_1, \ldots, m_s, h_1 - j_1, \ldots, h_t - j_t\} \), where \( \{m_1, \ldots, m_s\} \) are monomials and \( \{h_1 - j_1, \ldots, h_t - j_t\} \) are binomials where \( h_i > j_i \) for all \( 1 \leq i \leq t \).

**Remark 5.7** Any element of \( G_\omega^+ \) can be represented as a syzygy of \( \{m_1, \ldots, m_s, h_1 - j_1, \ldots, h_t - j_t\} \) since \( \langle G_\omega \rangle = \langle G_\omega^+ \rangle \).

**Theorem 5.8** For a monomial \( m \in G_\omega^+ \), the result of lift of \( m \) is \( m - m' \), where \( m' \) is a normal form of \( m \) with respect to \( G_\omega \).

**Proof.** From the definition of Buchberger algorithm, \( m \) is obtained from \( G_\omega \), by repeating to make S-polynomial and to reduce it. \( m \) is a monomial, hence appearances of monomials in above two step must be only once. We denote the monomial by \( m_k \) and can write \( m \) as following form:

\[ m = \alpha_1(h_{i_1} - j_{i_1}) + \alpha_2(h_{i_2} - j_{i_2}) + \cdots + \beta m_k \]

In above equation, adjacent two terms (e.g. \(-\alpha_1 j_{i_1} + \alpha_2 h_{i_2}\)) is cancelled each other. Since this left-hand side is a monomial, terms of right-hand side are vanished except one term. If \( \beta = 1 \) and the rest term is \( m_k \), that means any \( \alpha_i = 0 \), we obtain \( m = m_k \). The normal form of \( m \) wrt \( G_\omega \) is \( m' \) where \( m - m' \in G \), and the result of lift of \( m \) is \( m - m' \). Otherwise \( (\beta \neq 1) \) the rest term is \( h_{i_1} (= m) \). By lifting above equation, we obtain

\[ m - m^* = \alpha_1(h_{i_1} - j_{i_1}) + \alpha_2(h_{i_2} - j_{i_2}) + \cdots + \beta(m_k - m') \]

where \( m - m' \in G \). Since \( m = h_{i_1} \), we obtain \( m^* = \beta m' \) where the normal form of \( m \) wrt \( G_\omega \).

Both cases satisfy the theorem. \( \Box \)

33
Theorem 5.9 For a binomial $h - j \in G_\omega^+$, the result of lift of $h - j$ is also $h - j$.

Proof. It is enough for the proof to show that $h_i$ is represented with a syzygy whose elements are only binomials, that is

$$h - j = \sum_{i=1}^{t} \alpha_i(h_i - j_i).$$

Any binomials in $G_\omega^+$ is obtain from $G_\omega$ by Buchberger algorithm. It is apparent that $\text{Spol}(g_i, g_j)$ becomes a binomial iff both $g_i$ and $g_j$ are binomials and that a result of reductions becomes a binomial iff all reducers are binomials. Hence any binomial in $G_\omega^+$ is written as a form of above equation. □

Example 5.10 Let $\prec$ be a graded reverse lexicographic order, $\ll$ be a lexicographic order and $G_\prec$ be \{ce - d^2, be - cd, ae - bd, ad - e, ae - c^2, bc - e, ac - d, b^2 - d, ab - c, a^2 - b\}.

First we fix $(\omega, \tau) = (1, 1, 1, 1)$ and $(\omega, 1, 0, 0, 0)$. Then the set of initial terms $G_\omega$ is \{ce - d^2, be - cd, ae - bd, ad - c^2, ae - c^2, bc, ac, b^2, ab, a^2\}. The underlined terms is leading terms with respect to a lexicographic order.

The Gröbner basis $G_\omega^+$ of $G_\omega$ with respect to $\omega^{LEX}$ is

$$G_\omega^+ = \{d^6, ce - d^2, cd^3, c^2d, c^3, be - cd, bd - c^2, bc, b^2, ae - c^2, ac, ab, a^2\}$$

By lifting $G_\omega^+$, we obtain

$$G = \{d^6 - e^4, ce - d^2, cd^3 - e^3, c^2d - e^2, c^3 - de, be - cd, bd - c^2, bc - e, b^2 - d, ae - c^2, ad - e, ac - d, ab - c, a^2 - b\}$$

By updating the weight vector, we newly obtain $\omega = (1, 0, 0, 0, 0)$. Then the algorithm backs to the top of while loop.

The set of initial terms $G_\omega$ is

$$\{d^6 - e^4, ce - d^2, cd^3 - e^3, c^2d - e^2, c^3 - de, be - cd, bd - c^2, bc - e, b^2 - d, ae, ad, ac, ab, a^2\}.$$ 

This is already a Gröbner basis wrt $\omega^{LEX}$. Finally the algorithm is terminated and the output is as follows:

$$G_\ll = \{d^6 - e^4, ce - d^2, cd^3 - e^3, c^2d - e^2, c^3 - de, be - cd, bd - c^2, bc - e, b^2 - d, ae - c^2, ad - e, ac - d, ab - c, a^2 - b\}.$$
Chapter 6

Experiments

6.1 Conditions of the experiments

The comparison between existing algorithms and our proposed algorithms is done as follows. We implemented algorithms $F_4,F_5$ with C as new functions of Asir [1], and Change of Ordering algorithm as a user script language on Asir, in order to compare the results with existing algorithms on Asir. The machine environment of experiments is

- CPU : UltraSPARC-II 360MHz,
- Memory : 2GB.

Existing Buchberger algorithm, which we call $gr$, adopts criteria in Proposition 2.35 and a sugar strategy. And existing $F_4$ algorithm, which we call old $F_4$, uses Gauss elimination and Chinese Remainder theorem for diagonalization of matrices. As input term orders, we use two orders, a graded reverse lexicographic order(DRL) and a pure lexicographic order(LEX).

In our experiments, we use following matrices. Some of these examples are from [22].

- $A_n = \begin{pmatrix} 1 & 2 & \cdots & n \end{pmatrix} \cdots$ the Gröbner basis elements of the toric ideal $I_{A_n}$ correspond to primitive partition identities with largest part $n$ (see [30]).
- $A_n^h = \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & 2 & \cdots & n & 0 \end{pmatrix} \cdots$ The homogenized matrix of $A_n$.
- $P_{3 \times s} = \begin{pmatrix} 1 & 2 & 3 & \cdots & s \\ 1 & 4 & 9 & \cdots & s^2 \\ 1 & 8 & 27 & \cdots & s^3 \end{pmatrix} \cdots 3 \times s$ matrix.

We compute the generators of toric ideals corresponding to the matrices with CoCoA [10] in advance and use them as input. The number of elements of each generator is as follows:
Table 6.1: Matrices and the number of elements of generators

<table>
<thead>
<tr>
<th>matrices</th>
<th>#(elements of generators)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_n, A^h_n$</td>
<td>equal to $n$</td>
</tr>
<tr>
<td>$P_{3\times 8}$</td>
<td>102</td>
</tr>
<tr>
<td>$P_{3\times 9}$</td>
<td>277</td>
</tr>
<tr>
<td>$P_{3\times 10}$</td>
<td>509</td>
</tr>
<tr>
<td>$P_{3\times 11}$</td>
<td>1359</td>
</tr>
</tbody>
</table>

6.2 Results of improved $F_4$ algorithm

First we observe the efficiency of improved $F_4$ algorithm by comparing it with gr and old $F_4$. The improvements are done only in the part of reduction, therefore we also show computation time for reduction of old and new $F_4$. Just for information of complexities, the number of elements of Gröbner bases are also shown.

Table 6.2: Computation time of gr, old and new $F_4$ on $A_n$ wrt DRL (sec)

<table>
<thead>
<tr>
<th>size of $n$</th>
<th>#elements</th>
<th>gr</th>
<th>old $F_4$ (reduction)</th>
<th>new $F_4$ (reduction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>45</td>
<td>0.04826</td>
<td>0.3582 (0.320447)</td>
<td>0.04848 (0.00184131)</td>
</tr>
<tr>
<td>15</td>
<td>105</td>
<td>0.4298</td>
<td>3.004 (2.64308)</td>
<td>0.255 (0.018609)</td>
</tr>
<tr>
<td>20</td>
<td>190</td>
<td>1.935</td>
<td>19.08 (17.3325)</td>
<td>1.044 (0.127311)</td>
</tr>
<tr>
<td>25</td>
<td>300</td>
<td>7.006</td>
<td>74.86 (69.731)</td>
<td>3.332 (0.486787)</td>
</tr>
<tr>
<td>30</td>
<td>435</td>
<td>20.97</td>
<td>199.8 (187.827)</td>
<td>9.276 (1.3382)</td>
</tr>
<tr>
<td>35</td>
<td>595</td>
<td>61.6</td>
<td>764.8 (671.696)</td>
<td>19.55 (3.18394)</td>
</tr>
<tr>
<td>40</td>
<td>780</td>
<td>155.6</td>
<td>1832 (1651.63)</td>
<td>42.14 (7.09988)</td>
</tr>
</tbody>
</table>

36
Figure 6.1: Comparison of gr, old and new $F_4$ on $A_n$ wrt DRL (sec)

<table>
<thead>
<tr>
<th>size of $n$</th>
<th>#elements</th>
<th>$gr$</th>
<th>old $F_4$ (reduction)</th>
<th>new $F_4$ (reduction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>59</td>
<td>0.1342</td>
<td>0.8391 (0.759665)</td>
<td>0.1304 (0.06541115)</td>
</tr>
<tr>
<td>15</td>
<td>132</td>
<td>0.709</td>
<td>8.749 (8.04571)</td>
<td>0.6792 (0.0803082)</td>
</tr>
<tr>
<td>20</td>
<td>231</td>
<td>2.771</td>
<td>68.82 (64.4011)</td>
<td>3.292 (0.627788)</td>
</tr>
<tr>
<td>25</td>
<td>360</td>
<td>9.387</td>
<td>322.2 (306.012)</td>
<td>13.38 (2.98794)</td>
</tr>
<tr>
<td>30</td>
<td>509</td>
<td>26.23</td>
<td>1366 (1180.59)</td>
<td>40.65 (9.81033)</td>
</tr>
<tr>
<td>35</td>
<td>688</td>
<td>70.28</td>
<td>4578 (3685.63)</td>
<td>113.4 (29.3145)</td>
</tr>
<tr>
<td>40</td>
<td>891</td>
<td>168.7</td>
<td>24320 (16626.7)</td>
<td>273 (73.9212)</td>
</tr>
</tbody>
</table>
Figure 6.2: Comparison of gr, old and new $F_4$ on $A_n$ wrt LEX (sec)

Table 6.4: Computation time of gr, old and new $F_4$ on $A_n^h$ wrt DRL (sec)

<table>
<thead>
<tr>
<th>size of n</th>
<th>#elements</th>
<th>gr</th>
<th>old $F_4$ (reduction)</th>
<th>new $F_4$ (reduction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>45</td>
<td>0.1125</td>
<td>0.5065 (0.459446)</td>
<td>0.08881 (0.00358605)</td>
</tr>
<tr>
<td>15</td>
<td>105</td>
<td>0.4752</td>
<td>3.042 (2.61124)</td>
<td>0.36 (0.0303888)</td>
</tr>
<tr>
<td>20</td>
<td>190</td>
<td>2.113</td>
<td>19.35 (17.4367)</td>
<td>1.156 (0.180416)</td>
</tr>
<tr>
<td>25</td>
<td>300</td>
<td>7.312</td>
<td>75.78 (70.3721)</td>
<td>3.416 (0.672355)</td>
</tr>
<tr>
<td>30</td>
<td>435</td>
<td>22.16</td>
<td>207.7 (194.424)</td>
<td>9.282 (2.03491)</td>
</tr>
<tr>
<td>35</td>
<td>595</td>
<td>65.68</td>
<td>765.8 (670.46)</td>
<td>21.89 (4.79189)</td>
</tr>
<tr>
<td>40</td>
<td>780</td>
<td>165.2</td>
<td>1793 (1632.54)</td>
<td>46.24 (10.3447)</td>
</tr>
</tbody>
</table>
Table 6.5: Computation time of gr, old and new $F_4$ on $A_n^4$ wrt LEX (sec)

<table>
<thead>
<tr>
<th>size of $n$</th>
<th>#elements</th>
<th>gr</th>
<th>old $F_4$ (reduction)</th>
<th>new $F_4$ (reduction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>59</td>
<td>0.1973</td>
<td>1.919 (1.68101)</td>
<td>0.16 (0.0176654)</td>
</tr>
<tr>
<td>15</td>
<td>132</td>
<td>0.721</td>
<td>8.976 (8.17039)</td>
<td>0.7118 (0.125416)</td>
</tr>
<tr>
<td>20</td>
<td>231</td>
<td>2.822</td>
<td>67.73 (62.929)</td>
<td>3.702 (0.991405)</td>
</tr>
<tr>
<td>25</td>
<td>360</td>
<td>9.561</td>
<td>334.5 (315.97)</td>
<td>15.35 (5.01556)</td>
</tr>
<tr>
<td>30</td>
<td>509</td>
<td>27.35</td>
<td>1439 (1233.2)</td>
<td>47.86 (16.9182)</td>
</tr>
<tr>
<td>35</td>
<td>688</td>
<td>72.02</td>
<td>4760 (3733.69)</td>
<td>132.2 (49.5848)</td>
</tr>
<tr>
<td>40</td>
<td>891</td>
<td>172.5</td>
<td>17350 (12544.6)</td>
<td>322.1 (126.204)</td>
</tr>
</tbody>
</table>

Table 6.6: Computation time of gr, old and new $F_4$ on $P_{3 \times s}$ wrt DRL (sec)

<table>
<thead>
<tr>
<th>size of $(r, s)$</th>
<th>#elements</th>
<th>gr</th>
<th>old $F_4$ (reduction)</th>
<th>new $F_4$ (reduction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3, 8)</td>
<td>96</td>
<td>0.3439</td>
<td>4.966 (4.31867)</td>
<td>0.4213 (0.0120394)</td>
</tr>
<tr>
<td>(3, 9)</td>
<td>255</td>
<td>2.001</td>
<td>150.5 (147.072)</td>
<td>2.496 (0.15273)</td>
</tr>
<tr>
<td>(3, 10)</td>
<td>573</td>
<td>9.714</td>
<td>(Memory Exhausted)</td>
<td>12.47 (1.517)</td>
</tr>
<tr>
<td>(3, 11)</td>
<td>1213</td>
<td>84.73</td>
<td>(Memory Exhausted)</td>
<td>108.7 (15.7334)</td>
</tr>
</tbody>
</table>

Table 6.7: Computation time of gr, old and new $F_4$ on $P_{3 \times s}$ wrt LEX (sec)

<table>
<thead>
<tr>
<th>size of $(r, s)$</th>
<th>#elements</th>
<th>gr</th>
<th>old $F_4$ (reduction)</th>
<th>new $F_4$ (reduction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3, 8)</td>
<td>390</td>
<td>2.009</td>
<td>22.58 (20.7604)</td>
<td>1.869 (0.0761878)</td>
</tr>
<tr>
<td>(3, 9)</td>
<td>1264</td>
<td>16.21</td>
<td>929.6 (907.666)</td>
<td>19.93 (1.22666)</td>
</tr>
<tr>
<td>(3, 10)</td>
<td>3503</td>
<td>136.3</td>
<td>(Memory Exhausted)</td>
<td>195.5 (20.0841)</td>
</tr>
<tr>
<td>(3, 11)</td>
<td>9492</td>
<td>1334</td>
<td>(Memory Exhausted)</td>
<td>2232 (295.597)</td>
</tr>
</tbody>
</table>

39
6.3 Results of $F_5$ algorithm

We compare the computation times of $F_5$ algorithm with gr and new $F_4$ algorithm. In following results, the computation times of gr and $F_4$ is as same as the corresponding results in section 6.2.

Table 6.8: Computation time of gr, new $F_4$ and $F_5$ on $A_n$ wrt DRL (sec)

<table>
<thead>
<tr>
<th>size of $n$</th>
<th>gr</th>
<th>new $F_4$</th>
<th>$F_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.04826</td>
<td>0.04848</td>
<td>0.1001</td>
</tr>
<tr>
<td>15</td>
<td>0.4298</td>
<td>0.255</td>
<td>0.3944</td>
</tr>
<tr>
<td>20</td>
<td>1.935</td>
<td>1.044</td>
<td>1.719</td>
</tr>
<tr>
<td>25</td>
<td>7.006</td>
<td>3.332</td>
<td>5.685</td>
</tr>
<tr>
<td>30</td>
<td>20.97</td>
<td>9.276</td>
<td>15.55</td>
</tr>
<tr>
<td>35</td>
<td>61.6</td>
<td>19.55</td>
<td>37.48</td>
</tr>
<tr>
<td>40</td>
<td>155.6</td>
<td>42.14</td>
<td>84.46</td>
</tr>
</tbody>
</table>

Figure 6.3: Comparison of gr, $F_4$ and $F_5$ on $A_n$ wrt DRL (sec)
Table 6.9: Computation time of $gr$, new $F_4$ and $F_5$ on $A_n$ wrt LEX (sec)

<table>
<thead>
<tr>
<th>size of $n$</th>
<th>$gr$</th>
<th>new $F_4$</th>
<th>$F_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1342</td>
<td>0.1304</td>
<td>0.5125</td>
</tr>
<tr>
<td>15</td>
<td>0.709</td>
<td>0.6792</td>
<td>12.92</td>
</tr>
<tr>
<td>20</td>
<td>2.771</td>
<td>3.292</td>
<td>133.9</td>
</tr>
<tr>
<td>25</td>
<td>9.387</td>
<td>13.38</td>
<td>803.8</td>
</tr>
<tr>
<td>30</td>
<td>26.23</td>
<td>40.65</td>
<td>3341</td>
</tr>
<tr>
<td>35</td>
<td>70.28</td>
<td>113.4</td>
<td>12350</td>
</tr>
<tr>
<td>40</td>
<td>168.7</td>
<td>273</td>
<td>(too large)</td>
</tr>
</tbody>
</table>

Figure 6.4: Comparison of $gr$, $F_4$ and $F_5$ on $A_n$ wrt LEX (sec)
Table 6.10: Computation time of gr, new $F_4$ and $F_5$ on $A_n^h$ wrt DRL (sec)

<table>
<thead>
<tr>
<th>size of n</th>
<th>gr</th>
<th>new $F_4$</th>
<th>$F_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1125</td>
<td>0.08881</td>
<td>0.1022</td>
</tr>
<tr>
<td>15</td>
<td>0.4752</td>
<td>0.36</td>
<td>0.4046</td>
</tr>
<tr>
<td>20</td>
<td>2.113</td>
<td>1.156</td>
<td>1.748</td>
</tr>
<tr>
<td>25</td>
<td>7.312</td>
<td>3.416</td>
<td>5.736</td>
</tr>
<tr>
<td>30</td>
<td>22.16</td>
<td>9.282</td>
<td>15.82</td>
</tr>
<tr>
<td>35</td>
<td>65.68</td>
<td>21.89</td>
<td>37.93</td>
</tr>
<tr>
<td>40</td>
<td>165.2</td>
<td>46.24</td>
<td>84.95</td>
</tr>
</tbody>
</table>

Table 6.11: Computation time of gr, new $F_4$ and $F_5$ on $A_n^h$ wrt LEX (sec)

<table>
<thead>
<tr>
<th>size of n</th>
<th>gr</th>
<th>new $F_4$</th>
<th>$F_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1973</td>
<td>0.16</td>
<td>0.9505</td>
</tr>
<tr>
<td>15</td>
<td>0.721</td>
<td>0.7118</td>
<td>14.16</td>
</tr>
<tr>
<td>20</td>
<td>2.822</td>
<td>3.702</td>
<td>136.3</td>
</tr>
<tr>
<td>25</td>
<td>9.561</td>
<td>15.35</td>
<td>835</td>
</tr>
<tr>
<td>30</td>
<td>27.35</td>
<td>47.86</td>
<td>3629</td>
</tr>
<tr>
<td>35</td>
<td>72.02</td>
<td>132.2</td>
<td>12630</td>
</tr>
<tr>
<td>40</td>
<td>172.5</td>
<td>322.1</td>
<td>(too large)</td>
</tr>
</tbody>
</table>
6.4 Results of Change of Ordering algorithm

We measure the computation time of changing Gröbner bases with respect to graded reverse lexicographic order into those with lexicographic order. We compare two computation results, one computes bases with a lexicographic order directly by gr, and the other does bases with a graded reverse lexicographic order by new $F_4$ first, then converts the bases to a lexicographic order by Gröbner walk.

We remark that toric ideals generated from $A^4_n$ are homogeneous with respect to a graded reverse lexicographic order. Hence Gröbner walk algorithm is of no use since in algorithm 5.5, all elements of $\{in_{\omega}(g) \mid g \in G\}$ are binomial.

Table 6.12: Comparison between LEX and DRL+Gröbner walk on $A_n$ (sec)

<table>
<thead>
<tr>
<th>size of n</th>
<th>LEX</th>
<th>DRL+Gröbner Walk</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1342</td>
<td>0.563</td>
</tr>
<tr>
<td>15</td>
<td>0.709</td>
<td>2.728</td>
</tr>
<tr>
<td>20</td>
<td>2.771</td>
<td>9.837</td>
</tr>
<tr>
<td>25</td>
<td>9.387</td>
<td>31.86</td>
</tr>
<tr>
<td>30</td>
<td>26.23</td>
<td>88.74</td>
</tr>
<tr>
<td>35</td>
<td>70.28</td>
<td>220.6</td>
</tr>
<tr>
<td>40</td>
<td>168.7</td>
<td>516.5</td>
</tr>
</tbody>
</table>

Table 6.13: Comparison between LEX and DRL+Gröbner walk on $P_{3 \times 8}$ (sec)

<table>
<thead>
<tr>
<th>size of n</th>
<th>LEX</th>
<th>DRL+Gröbner Walk</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3,8)</td>
<td>2.009</td>
<td>72.75</td>
</tr>
<tr>
<td>(3,9)</td>
<td>16.21</td>
<td>794.9</td>
</tr>
<tr>
<td>(3,10)</td>
<td>136.3</td>
<td>11120</td>
</tr>
<tr>
<td>(3,11)</td>
<td>1334</td>
<td>(too large)</td>
</tr>
</tbody>
</table>
Chapter 7

Concluding remarks

In this thesis, first we made improvements of $F_4$ algorithm for toric ideals, in terms of data structure by replacing incidence matrices by vectors. In our experiments, computation times for reductions, which is a critical step of original $F_4$ algorithm, were speeded up by several hundred times by our improvements. As a result, new $F_4$ algorithm achieved computation speed as a couple of times faster as Buchberger algorithm in cases of a graded reverse lexicographic order with some examples. Contrary new $F_4$ algorithm is inferior to Buchberger algorithm in a case of a lexicographic order. This reason is thought that the total degree of elements of Gröbner bases rises high when the term order is a lexicographic order, moreover as degrees of $S$-polynomials rise higher, less $S$-polynomial are being reduced by needlessly more reducers. Furthermore from the result on $A_n$ and $P_{3\times 3}$, it may be presumed that if there exists many elements in generators compared to the number of variables, $F_4$ algorithm becomes slower than gr.

Next we modified and implemented $F_5$ algorithm, which eliminates all reduction to 0 by holding past histories of computations. In our experiments, this $F_5$ algorithm is faster than normal Buchberger algorithm but slower than our improved $F_4$ in cases of a graded reverse lexicographic order. Contrary, the computation time of $F_5$ with respect to lexicographic order explodes tremendously, which exceeds one hundred times longer computation times than gr. On the whole, we can say that the merit of $F_5$, which generates no $S$-polynomial reduced to 0, is not worked very well.

Finally we focused on Change of Ordering algorithms. We adopt Gröbner walk algorithm and improved for toric ideals. As a result of experiments, it was found that the computation of lifting of monomials to binomials (i.e. computation of normal forms) was a critical step, which spent much more time than computation of Gröbner basis of initial terms, in whole of algorithm. Moreover, the elements of toric ideals are all binomial, thus it does not have advantages to compute Gröbner bases of initial terms, compared with general polynomial ideals. These facts cause the inferiority of Gröbner walk in toric cases.

As future works, following interesting problems and ideas remain:
• For $F_4$ algorithm, consider of a method mixing Buchberger algorithm and $F_4$ into one algorithm. That means, first we use Symbolic Preprocessing and a reduction by matrices as long as degrees of polynomials is not so large, afterwards we switch to Buchberger algorithm.

• For $F_5$ algorithm, apply implemented $F_5$ algorithm to general ideals. And about explosions of computation time when the ordering is a lexicographic order, do further analyses.

• For Change of Ordering algorithm, use other Change of Ordering algorithms. And in Gröbner walk algorithm, improve the process to compute normal forms of monomials using the idea of $F_4$ algorithm.
References


[27] SINGULAR research group. SINGULAR. http://www.singular.uni-kl.de/.

