AJAMC, Vol 1, Issue 4, 2021

Grobner Basis and its application in Integer Programming

Anjan Samanta

Department of Basic Science and Humanities,
University of Engineering and Management, Kolkata, India

We are going to introduce Grobner basis, a particular kind of generating set of an ideal in a polynomial ring $k[x_1, \ldots, x_n]$ over field k, which is one of the main practical tools for solving system of polynomial equations. The upcoming discussion will focus on an application of Grobner basis, Integer Programming, in order to demonstrate how the techniques of Grobner basis work.

Keywords: Grobner basis, Buchberger's Algorithm, Integer Programming, Macaulay2, Laurent polynomial Ring.

1 Introduction

In this paper we are going to discuss about theory of Grobner basis and its application to Integer Programming. Grobner basis is sort of generalization of g.c.d of polynomials (Euclidean Algorithm) in one variable.

Integer programming belongs to the category of optimization problems, where we have to maximize or minimize an objective subject to some constraints. We will see how Grobner basis can be implemented for solving these kind of problems. Integer programming (IPP) is different from Linear programming (LPP) for the fact that IPP concerns about finding integer solutions, and finding integer-only solutions are much more difficult thing to deal with. We are going to use algorithms motivated by Grobner basis to solve IPP. Though algorithms are complicated compared to other available methods (like branch and bound method), they are worth investigating so that we can find a more efficient solution. For all algebraic computations, **Macaulay2** has been implemented.

2 Grobner Basis

2.1 Lexicographic Order:

Let $a = (a_1, ..., a_n)$ and $b = (b_1, ..., b_n)$ be in $\mathbb{Z}^n_{\geq 0}$. We define $a >_{lex} b$ if the leftmost nonzero entry of the vector difference $a - b \in \mathbb{Z}^n$ is positive. We will write $x^a >_{lex} x^b, if a >_{lex} b$.

Let $f = \Sigma a_m x^m$ be a nonzero polynomial in $k[x_1, ..., x_n]$ and let $>$ be a monomial order.

- The **multidegree** of f is, multideg(f)=max($m \in \mathbb{Z}^n_{\geq 0} | a_m \neq 0$)
- The **Leading coefficient** of f, LC(f)=$a_{multideg(f)} \in k$
- The **Leading monomial** of f is, LM(f)=$x^{multideg(f)}$
- The **Leading term** of f is, LT(f) = LC(f) . LM(f)

Let $f = 4x^3yz + 4z^2 - 13x^4 + 15xz^3$ and $>$ being Lex order. Then

$Multideg(f) = (4, 0, 0), LC(f) = -13, LM(f) = x^4, LT(f) = -13x^4$

2.2 Hilbert Basis Theorem and Grobner Basis

Theorem 2.1. (**Hilbert Basis**)[4] *Every ideal $I \subseteq k[x_1, ..., x_n]$($k$ being a field) has a finite generating set. Mathematically, $I =< g_1, ..., g_t >$ for some $g_1, ..., g_t \in I$.*

We call this generating sets of ideal as basis. So every ideal of $k[x_1, ..., x_n]$ has a basis. Grobner basis $G = \{g_1, ..., g_t\}$ is a basis of ideal I which satisfies the following property:

$$< LT(g_1), ..., LT(g_t)) >=< LT(I) > .$$

2.3 Properties of Grobner Basis

We define **S-polynomial** of f and g as follows :

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

where $x^a = lcm(LM(f), LM(g))$.

Buchberger's Criterion: [4] *Let I be a polynomial ideal. Then a basis $G = \{g_1, ..., g_t\}$ of I is a Grobner basis of I if and only if for all pairs $i \neq j$, the*

Email:anjan.samanta@uem.edu.in

remainder on division of $S(g_i, g_j)$ by G (listed in some order) is zero.

This criteria gives rise to an algorithm by which a Grobner basis for I can be constructed from any basis of that ideal. This algorithm is known as **Buchberger's Algorithm**. Hence we can say, in $k[x_1, ..., x_n]$, every ideal has a Grobner Basis.

2.4 Application of Grobner Basis: Solving Polynomial Equations[2]

Grobner basis is widely used to solve system of non-linear polynomial equations. Let us understand this by an example:
Consider the equations:

$$x^3 + y^3 + z^3 = 1$$

$$x + z = y$$

$$x = z$$

These equations can be solved by computing Grobner basis. Consider the ideal $I = (x^3 + y^3 + z^3 - 1, x + z - y, x - z) \subseteq \mathbb{C}[x, y, z]$. Now by hand it is difficult to perform Buchberger's algorithm to find Grobner basis. For computation we will use a software called **Macaulay2**. Here are the codes to find Grobner basis of above ideal:

```
i1 : CC[x,y,z,MonomialOrder=>Lex]
o1 : PolynomialRing
i2 : I=ideal(x^3+y^3+z^3-1,x+z-y,x-z)
o2 : ideal (x3+y3+z3-1,x-y+z,x-z)
i3 : gens gb I
o3 : ( z^3-.1  y-2z  x-z)
```

Hence Grobner basis is $\{z^3 - 0.1, y - 2z, x - z\}$. Since the first polynomial depends on z alone, we can find the roots. Then by back substitution we can find values of other variables.

3 Solving Integer Programming by Grobner Basis

The integer programming problem (IPP) can be defined as follows: Let $a_{ij} \in \mathbb{Z}, b_i \in \mathbb{Z}$ and $c_j \in \mathbb{R}, i = 1, ..., n, j = 1, ..., m$; we wish to find a solution $(l_1, l_2, .., l_m)$ in $\mathbb{N}^m$ of the system

$$a_{11}l_1 + a_{12}l_2 + ... + a_{1m}l_m = b_1$$

$$a_{21}l_1 + a_{22}l_2 + ... + a_{2m}l_m = b_2$$

$$\vdots$$

$$a_{n1}l_1 + a_{n2}l_2 + ... + a_{nm}l_m = b_n,$$

which minimizes the "cost function" $c(l_1, l_2 \dots, l_m) = \sum_{j=1}^{m} c_j l_j$

Grobner basis deals with polynomials, hence we first transfer these system of equations into a problem of polynomials. First, we assign a variable to each linear equation. We will let these variables be $x_1, x_2, \dots, x_n$ for the n equations. Then, we can represent the above equation as the following:

$$x_i^{a_{i1}l_1 + ... + a_{im}l_m} = x_i^{b_i}$$

But we are going to find a solution to the entire system, and so we must combine each of these equations to construct a single equation which represents the system. So we multiply the above n many equations to form a single equation. Thus we have

$$x_1^{a_{11}l_1 + a_{12}l_2 + \dots + a_{1m}l_m} \dots x_n^{a_{n1}l_1 + \dots + a_{nm}l_m} = x_1^{b_1} \dots x_n^{b_n}$$

This can be rewritten as:

$$(x_1^{a_{11}} x_2^{a_{21}} \dots x_n^{a_{n1}})^{l_1} \dots (x_1^{a_{1m}} x_2^{a_{2m}} \dots x_n^{a_{nm}})^{l_m} = x_1^{b_1} \dots x_n^{b_n}$$

Now we define a polynomial map $\phi : k[y_1, \dots, y_n] \longrightarrow k[x_1, \dots, x_n]$ such that $\phi(y_j) = f_j$,where $f_j = x_1^{a_{1j}} x_2^{a_{2j}} \dots x_n^{a_{nj}}, j = 1, 2, \dots, m.$
This can be showed that ϕ is a ring homomorphism.Then using this map, above equation transforms into

$$(\phi(y_1))^{l_1} \dots (\phi(y_m))^{l_m} = x_1^{b_1} x_2^{b_2} \dots x_n^{b_n}$$

Now using homomorphism of ϕ we can say

$$\phi(y_1^{l_1} \dots y_m^{l_m}) = x_1^{b_1} x_2^{b_2} \dots x_n^{b_n}$$

Therefore the monomial $x_1^{b_1} x_2^{b_2} ... x_n^{b_n}$ is in the image under ϕ of a monomial in $k[y_1, \dots, y_m]$, provided $(l_1, \dots, l_m)$ is a solution of the given Integer programming(IPP). Going backwards we can see that the converse is also true. Hence we have the following:

Theorem 3.1. [1] *Assume all a_{ij}'s and b_i's are non-negative. Then there exists a solution $(l_1, l_2, \dots, l_m) \in \mathbb{Z}^m_{\geq 0}$ of given IPP if and only if the monomial $x_1^{b_1} x_2^{b_2} \dots x_n^{b_n}$ is in the image under ϕ of a monomial in $k[y_1, \dots, y_m]$.Moreover, if*

$$\phi(y_1^{l_1} \dots y_m^{l_m}) = x_1^{b_1} x_2^{b_2} \dots x_n^{b_n}$$

then $(l_1, l_2, \dots, l_m) \in \mathbb{Z}^m_{\geq 0}$ is a solution to IPP.

So all we have to do is to find a preimage of the monomial $x_1^{b_1} x_2^{b_2} \dots x_n^{b_n}$ under ϕ. How to search for it? here comes the role of Grobner basis. The following theorem will tell us how reduced grobner basis will help us to find the preimage:

Theorem 3.2. [3] *Let $K = < y_1 - f_1, \dots, y_m - f_m > \in k[y_1, \dots, y_m, x_1, \dots, x_n]$ be an ideal where $f_j = x_1^{a_{1j}} x_2^{a_{2j}} \dots x_n^{a_{nj}}$, and let G be a Grobner basis for K with respect to an elimination order with the x variables larger than the y variables. Then $f \in k[x_1, \dots, x_n]$ is in the image of ϕ if and only if there exists $h \in k[y_1, \dots, y_m]$ such that h is the remainder when dividing f by Grobner basis G. In this case, $f = \phi(h) = h(f_1, \dots, f_m)$.*

The above theorem gives us the required preimage, which leads to the solution of the given IPP. So combining the two above mentioned theorems we have our following algorithm[1] of finding solution of IPP described above:

$$a_{11}l_1 + a_{12}l_2 + \ldots + a_{1m}l_m = b_1$$

$$\vdots$$

$$a_{n1}l_1 + a_{n2}l_2 + \ldots + a_{nm}l_m = b_n$$

1. Compute a Grobner basis G for $K =< y_j - x_1^{a_{1j}} x_2^{a_{2j}} \ldots x_n^{a_{nj}} | 1 \leq j \leq m >$ with respect to an elimination order with the x variables larger than the y variables;

2. Find the remainder h of the division of the monomial $x_1^{b_1} \ldots x_n^{b_n}$ by Grobner basis G.

3. If $h \notin k[y_1, ..., y_m]$, then the IPP does not have non-negative integer solutions. If $h = y_1^{l_1} ... y_m^{l_m}$,then $(l_1, l_2, ..., l_m)$ is a solution of given IPP.

Let us solve a IPP using this method.
Problem :

$$Maximize\ \ 3x + 4y + 2z$$

Subject to

$$3x + 2y + z \leq 45$$

$$x + 2y + 3z \leq 21$$

$$2x + y + z \leq 18$$

$$x, y, z \geq 0, x, y, z \in \mathbb{N}$$

This can be rephrased as:

$$min \quad -3x - 4y - 2z + 0.w_1 + 0.w_2 + 0.w_3$$

Subject to

$$3x + 2y + z + w_1 = 45$$

$$x + 2y + 3z + w_2 = 21$$

$$2x + y + z + w_3 = 18$$

$$x, y, z, w_1, w_2, w_3 \geq 0, w_1, w_2, w_3 \in \mathbb{N}$$

First we convert the IPP into a polynomial equation by above mentioned procedure to get

$$(x_1^3 x_2 x_3^2)^x (x_1^2 x_2^2 x_3)^y (x_1 x_2^3 x_3)^z x_1^{w_1} x_2^{w_2} x_3^{w_3} = x_1^{45} x_2^{21} x_3^{18}$$

Hence our required ideal in $k[x_1, x_2, x_3, y_1, y_2, ..., y_6]$ is $K = (y_1 - x_1^3 x_2 x_3^2, y_2 - x_1^2 x_2^2 x_3, y3 - x_1 x_2^3 x_3, y_4 - x_1, y_5 - x_2, y_6 - x_3)$.

Now we will use **Macaulay2** to compute reduced grobner basis G of K. Then we will divide the monomial $x_1^{45} x_2^{21} x_3^{18}$ by G to get our solution of IPP. Macaulay2 codes are written below:

```
i6 : R=QQ[x1,x2,x3,y1,y2,y3,y4,y5,y6]
o6 = R
o6 : PolynomialRing
i7 : k=ideal(y1-x1^3*x2*x3^3,y2
     -x1^2*x2^2*x3,y3-x1*x2^3*x3,
     y4-x1,y5-x2,y6-x3)

o7 = ideal (- x1^3 x2*x3^3  + y1,
     - x1^2 x2^2 x3 + y2,
     - x1*x2^3 x3 + y3,
     - x1 + y4,- x2+ y5, - x3 + y6)
o7 : Ideal of R
i8 : x1^45*x2^21*x3^18% gb k
       3  9  18
o8 = y1 y2 y4
o8 : R
```

Hence $(3, 9, 0)$ is a solution of given IPP. But this is not our aim. Our goal is to find the optimal solution (let's focus only minimization). So we have to minimize the cost function $c(l_1, ..., l_m)$ and to do so,we will define a term order on y variables as follows:

Definition 3.1. term order $<_c$ on the y variables is said to be compatible with the cost function c and the map ϕ if

$$\phi(y_1^{l_1} \ldots y_2^{l_m}) = \phi(y_1^{l_1'} \ldots y_m^{l_m'})$$

and

$$c(l_1, \ldots, l_m) < c(l_1' \ldots l_m')$$

implies

$$y_1^{l_1} \ldots y_2^{l_m} <_c y_1^{l_1'} \ldots y_m^{l_m'}$$

We will use this term order to the ring under consideration. The following theorem gives us the optimal solution:

Theorem 3.3. [3] Let G be a Grobner basis for K with respect to an elimination order with the x variables larger than the y variables, and an order $<_c$ on the y variables which is compatible with the cost function c and the map ϕ. If $\overline{x_1^{b_1} \ldots x_2^{b_n}}^G = y_1^{l_1} \ldots y_m^{l_m}$ (means, $y_1^{l_1} \ldots y_m^{l_m}$ is the remainder when dividing $x_1^{b_1} \ldots x_2^{b_n}$ by G), then $(l_1, \ldots l_m)$ is the optimal (minimal) solution of given Integer programming problem.

Now we will use this theorem to our previous algorithm to obtain the minimal solution. The only difference is that we have to define the term order initially. The **Macaulay2** codes are given below:

```
i1 : R=QQ[y1,y2,y3,y4,y5,y6,
     Weights=>{-3,-4,-2,0,0,0},
     MonomialOrder=>Lex,Global=>false]
o1 = R
o1 : PolynomialRing
i2 : R2=R[x1,x2,x3]
```

```
o2 = R2
o2 : PolynomialRing
i3 : K=ideal(y1-x1^3*x2*x3^2,y2-x1^2*x2^2*x3,
     y3-x1*x2^3*x3,y4-x1,y5-x2,y6-x3)
o3 = ideal (- x1^3 x2*x3^2  + y1, - x1^2 x2^2 x3 + y2,
     - x1*x2^3 x3 + y3, - x1 + y4, - x2+ y5, - x3 + y6)
o3 : Ideal of R2
i4 : x1^45*x2^21*x3^18 % gb K
       5  8  14
o4 = y1 y2 y4
o4 : R2
```

Hence $(5, 8, 0)$ is the **optimal solution** of given IPP.

Now let's try to solve another Integer programming problem where the constraints have negative integer coefficients as well:

$$minimize\ \ p = 3x + 4y + 5z + 8w$$

Subject to

$$3x - 2y + z + 2w = 45$$

$$x + 2y - 3z - 5w = 21$$

$$2x - y + z = 18$$

$$x, y, z, w \geq 0$$

Applying previous method to convert it to a polynomial equation:

$$(t_1^3 t_2 t_3^2)^x (t_1^{-2} t_2^2 t_3^{-1})^y (t_1 t_2^{-3} t_3)^z (t_1^2 t_2^{-5})^w = t_1^{45} t_2^{21} t_3^{18}$$

As negative powers of $t_i's$ are arising, we can't work with usual polynomial ring $k[t_1, t_2, t_3]$, somehow we have to allow the negative powers of $t_i's$ in the ring. There is a ring called Laurent polynomial ring which will serve our purpose.

Definition 3.2. (**Laurent polynomial ring**) Let $m = (a_1, \ldots, a_n) \in \mathbb{Z}^n$ be an integer vector. The corresponding Laurent monomial in variables $t_1, \ldots, t_n$ is

$$t^m = t_1^{a_1} \ldots t_n^{a_n}$$

Finite linear combinations

$$f = \sum_{m \in \mathbb{Z}^n} c_m t^m$$

of Laurent monomials are called Laurent polynomials. We represent the ring of Laurent polynomials with coefficients from field k by $k[t_1^{\pm 1} \ldots t_n^{\pm 1}]$.

One important observation is the following:

$$k[t_1^{\pm 1} \ldots t_n^{\pm 1}] = k[t_1, \ldots t_n, w]/(t_1 \ldots t_n w - 1)$$

Intuitively this isomorphism works by introducing a new variable w satisfying the relation, $wt_1 \ldots t_n - 1 = 0$, so that w can be written as product of the inverses of the $t_i's$: $w = t_1^{-1} \ldots t_n^{-1}$. So in this ring, negative exponents of variables are allowed, hence we can construct the ideal $K = (y_1 - f_1, \ldots y_m - f_m)$ as before, now in $k[t_1^{\pm 1} \ldots t_n^{\pm 1}, y_1, \ldots, y_m]$.
Then by proceeding by previous algorithm we get our optimal solution of given problem. So, let's try again to solve the above in Macaulay2:

```
i3 : R=QQ[y1,y2,y3,y4,Weights=>{3,4,5,8},
    MonomialOrder=>Lex,Global=>false]
o3 = R
o3 : PolynomialRing
i4 : R2=R[t1,t2,t3,w]/(t1*t2*t3*w-1)
o4 = R2
o4 : QuotientRing
i5 : K=ideal(y1-t1^3*t2*t3^2,y2-t1^-2*t2^2*t3^-1,
     y3-t1*t2^-3*t3,y4-t1^2*t2^-5)
                3     2            4    2
o5 = ideal (- t1 t2*t3  + y1, - t2 t3*w  + y2,
          4  4 3            7  5 5
     - t1 t3 w  + y3, - t1 t3 w  + y4)
o5 : Ideal of R2
i6 : t1^45*t2^21*t3^18 % gb K
       36  57  3  24
o6 = y1  y2  y3 y4
o6 : R2
```

Hence $(36, 57, 3, 24)$ is the optimal solution and we have tackled all the cases including negative integer coefficients.

4 Conclusion and Future work

We have seen a brief discussion about theory of Grobner basis and its numerous possibilities towards solving problems those are not much related with Commutative Algebra. My one friend is working on application of Grobner basis to Ordinary and partial differential equations.
We have solved so far small problems using Macaulay2. But for a large scale of variables, it is computationally difficult to find the reduced Grobner basis of an Ideal. So maybe if we try to convert the generators of a given ideal into an integer programming problem, and then by solving it numerically (by any method like branch and bound) and translating back, we may end up with the reduced Grobner basis. That's I am going to try in future and maybe we can develop an one-one correspondence between **Theory of Grobner basis** and **Integer Programming Problem**.

References

[1] William Wells Adams et al. *An introduction to Grobner bases.* 3. American Mathematical Soc., 1994.

[2] Brad A Lutes. "A Survey of Gröbner Bases and Their Applications Departmental Report". In: *Texas Tech University* (2004).

[3] Carolyn Wendler. "Groebner Bases with an Application to Integer Programming". In: *Senior Exercise, May* 17 (2004).

[4] David Cox, John Little, and Donal OShea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, 2013.