2007

Integer Programming With Groebner Basis

Isabella Brooke Ginn
Virginia Commonwealth University

INTEGER PROGRAMMING WITH GROEBNER BASIS

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

Isabella Brooke Ginn
Mary Baldwin College, 1999

Director: Dr. James K. Deveney, Professor
Department of Mathematics

Virginia Commonwealth University
Richmond, Virginia
August 2007

Acknowledgements

I would like to extend my deepest gratitude to my committee chair Dr. James Deveney, his support throughout this process was far above any beyond the call of duty. The many hours he spent working with me on this thesis are greatly appreciated. I would also like to thank my committee members, Dr. Ghidewon Abay-Asmerom and Dr. Seth Roberts, for their assistance and support and sincerely appreciate their help.

I would like to thank my family and friends for their support and encouragement over the years.

Abstract

INTEGER PROGRAMMING WITH GROEBNER BASIS

By Isabella Brooke Ginn,
Bachelor of Arts, Mary Baldwin College, 1999

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University

Virginia Commonwealth University, 2007

Major Director: Dr. James Deveney, Professor
Department of Mathematics

Integer Programming problems are difficult problems to solve. The goal is to find an optimal solution that minimizes cost. With the help of Groebner based algorithms the optimal solution can be found if it exists. The application of this Groebner based algorithm and how it works is the topic of this research. The Algorithms are The Conti-Traverso Algorithm and the Original Conti-Traverso Algorithm. Examples are given as well as proofs that correspond to the algorithms. The latter algorithm is more efficient as well as more user friendly. The algorithms are not necessarily the best way to solve an integer programming problem, but they do find the optimal solution if it exists.

Contents

Introduction

Let A be an m x n matrix with integral entries, $b \in \mathbb{Z}^m$ and $c \in \mathbb{Z}^n$ (web.mit.edu.,2004). Given Ax = b, there is a cost vector that allows us to determine an optimal solution for Ax=b. The optimal solution(s) is a solution that minimizes the cost, based on our cost vector, c. An integer programming problem is similar to a linear programming problem except the solutions are positive integers. Based on applications we are only interested in whole units to minimize our cost. For example we would not be interested in making half of a bike to minimize our cost. Integer programming problems are very hard and are known as NP-complete problems, because we do not know how to solve them in polynomial time. The idea is to use Groebner Basis in order to find a solution that minimizes our cost.

The first section contains an introduction to Integer Programming. The second section discusses the idea behind finding a Groebner Basis. In the third section the Conti and Traverso Algorithm is introduced as well as examples that meet the requirements of the algorithm. The original Conti and Traverso Algorithm is discussed in the fourth section as well as examples. The following section concludes why finding the Groebner basis works to find the optimal solution to the Integer Programming problem and the last section concludes the paper.

Integer Programming

In 1827, the French mathematician J. B. J. Fourier published a method for solving systems of linear inequalities. (Sierksma, 1) This reference is known to be the first study of solving linear programming problems. Linear programming problems are used to solve various types of problems.

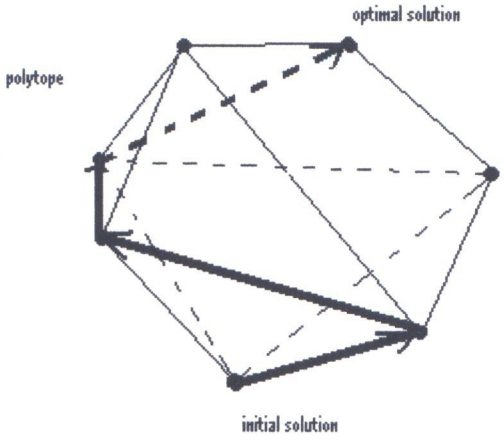
Given a linear programming problem, we want to solve for our given variables. If we are given a system of equations in n variables where Ax=b, we find our solution(s) to this system. If w is one of our solutions then Aw=b.

We want to find solutions to Ax=0, where the solutions are going to be a subspace of $\mathbb{R}^n$. Once the solution is found, say r, then w+r is a solution to Ax=b. However we are only interested in the solutions that are positive integers. Since integer programming involves solutions that are only integers the calculations tend to be more difficult. Linear programming problems can be done in various ways that can be solved in polynomial time, where integer solutions are a little harder and take longer to find. If we solve a linear programming problem and the solution is of integral value and it is optimal then we are done. If that is not the case then there is a more difficult problem that needs to be solved. Since we are looking at integer programming problems we are only interested in finding the solution(s) to Ax = b where the solutions are positive integers and A is an m x n matrix. Given Ax = b, and there exist more than one solution to this problem, we want to figure out which of the solutions minimizes our cost. Let *c* be our cost vector with n elements. The cost vector is used to figure out which solution is the optimal solution, the

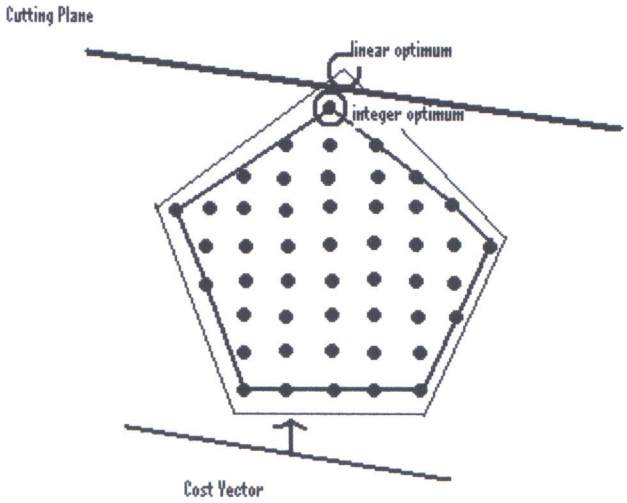
one that minimizes our cost. While a system of equations can have an infinite number of solutions in a contained area there are only a finite number of integer solutions. Since there are only a finite number of integer solutions then we do not have as many to work with to find the optimal one.

The region where all of the solutions are contained is known as the feasible region. The feasible region is a convex polytope that contains the solutions to our matrix equation. Only a finite number of the solutions are integer solutions, and those are the ones that we are interested in. This feasible region will lie in the quadrant where all values are positive. Within this region we are looking for the solution that minimizes our cost. The feasible region is a confined set of points and is defined as, $P_b^I = convex\ hull\{x \in \mathbb{N}^n : Ax = b\}$ (Thomas,122). The convex hull is enclosed by a polytope, and the integer solutions lie within that polytope. The area is feasible as long as it is not empty and is bounded. Within the bounded region the solution will not have an infinitely large solution, however if the area was not bounded then it would. The feasible region contains all points that satisfy all of the constraints. There may be several integer solutions that minimize the cost and therefore any of them would be optimal. The solutions to our integer programming problem are the points that are contained within a polytope that is formed by the intersection of the constraints. The optimal solution is one or more of the vertices of this polytope. The cost vector determines which of the vertices is optimal. From the cost vector a hyperplane can be formed and is shifted throughout the polytope. The hyperplane intersects the vertex and that point is the optimal solution.

To find this solution we will find the Groebner basis of the toric ideal of our matrix system.



This is a polytope that represents the route in which the cost vector would go in order to minimize the cost. The initial solution has an integral value but the next solution cost less than the previous solution. The next vertex is then tested to see if it cost less and it does, so this process is continued until we get to the optimal solution that costs the least. The following picture represents the cost vector cutting through a plane



(The previous pictures were recreated based on work done by Thomas, pages 120 & 121)

The cost vector intersects the cutting plane and therefore gives the optimal solution.

Groebner Basis

In this section we will discuss what a Groebner basis is and the different terminology associated with a Groebner basis.

Monomial Ordering.

A monomial ordering is done on polynomials where the monomials are put in order from largest to smallest. Given a polynomial in one variable, the ordering would be done where the monomial with the highest degree would be first and the next highest, would be next and so on.

There are two properties that must be considered in order to define a monomial order.

1. Well – Ordering. Any nonempty subset of monomials has a least element under $>$.

2. Compatibility. If $x^{\alpha} > x^{\beta}$, then $x^{\alpha}x^{\gamma} > x^{\beta}x^{\gamma}$ for any monomial x^{γ}.

The monomial ordering that we will use is lexicographic and it is defined as, $x_1^{a_1} \cdots x_n^{a_n} > x_1^{b_1} \cdots x_n^{b_n} \Leftrightarrow a_1 > b_1$, or $a_1 = b_1$ and $a_2 > b_2$, etc. (Cox, 3)

For example, given the following polynomial, with ordering x>y>z, $x^3y^2z^2 + x^2y^3z^4 + x^2y^3z^3 + x^5yz + x^4yz^2 + x^6y^2z$, with the monomials in order the polynomial would be $x^6y^2z + x^5yz + x^4yz^2 + x^3y^2z^2 + x^2y^3z^4 + x^2y^3z^3$.

Leading Terms of a polynomial.

Given a polynomial of the form $\sum_{i=0}^{n} a_i x^i$, where $a_n x^n$ is the leading term. Our a_n is non-zero and n is greater than or equal to zero. If two terms of a polynomial have the same variable with the same degree then the degree of the next variable in the order is used to determine the higher degree term. This process continues until there is a monomial that has the highest degree. The leading term is also known as the initial term.

For example, given $x^3y^4z^3$ and $x^2y^3z^4$, with lexicographic ordering x>y>z then the first monomial would be the leading term. By changing the lexicographic ordering to z>y>x, the second monomial would be the leading term. The total degree of the first monomial is 10 while the total degree of the second is nine. This is important if we are looking at monomial such as x^2y and y^{10}, with the lexicographic ordering as x>y, then the first monomial would be the leading term of $x^2y + y^{10}$ even though y^{10} has a higher total degree than x^2y.

Division Algorithm for Polynomials.

Given *f* and *g* that are polynomials, then *f* can be written as such:

$$f=qg+r,$$

where no term of *r* is divisible by LT(*g*). (Cox, 4)

Ideals.

A collection of polynomials $f_1,...,f_s \in k[x_1,...,x_n]$ generates the ideal

$\langle f_1,...,f_s\rangle = \{\sum_{i=1}^{s} h_i f_i \mid h_i \in k[x_1,...,x_n]\}$. (Cox, 5)

Hilbert Basis Theorem.

If $I \subset k[x_1,...,x_n]$ is an ideal, then we can find $f_1,...,f_s \in k[x_1,...,x_n]$ such that

$I = \langle f_1,...,f_s\rangle$

A Groebner basis for an ideal I in the polynomial ring $F[x_1, x_2,....,x_n]$ is a finite set of generators $\{g_1, g_2,....,g_m\}$ for I whose leading terms generate the ideal of all leading terms in I, i.e.,

$I = (g_1,.....,g_m)$ and $\langle \mathrm{LT}(I)\rangle = \langle \mathrm{LT}(g_1),...,\mathrm{LT}(g_m)\rangle$. (Dummit, 319)

In other words, a set of generators $\{g_1, g_2,....,g_m\}$ of I is a Groebner basis if the leading term of every nonzero element of I is divisible by some LT(g_i). Any Groebner basis of I is a basis of I. [Cox] Every element in I is a polynomial combination of generators. The process of finding the Groebner basis can take a while; the length depends on the number of variables.

Polynomial division must be done in order to apply what is known as the S-polynomial for the first step of finding the Groebner basis. An example of polynomial division is as follows:

We want to divide $x^2 + 1$ into $x^5 - 2x^4 + 3x + 7$:

$x^2+1\overline{)x^5-2x^4+3x+7}$, written with each degree term with zeros as coefficients would give us the following:

$x^2+1\overline{)x^5-2x^4+0x^3+0x^2+3x+7}$ and performing the operation gives us

$$
\begin{array}{r}
x^3-2x^2-x+2 \\
x^2+1\overline{)x^5-2x^4+0x^3+0x^2+3x+7} \\
\underline{-(x^5+0x^4+x^3)} \\
-2x^4-x^3 \\
\underline{-(-2x^4+0x^3-2x^2)} \\
-x^3+2x^2 \\
\underline{-(-x^3+0x^2-x)} \\
2x^2+4x \\
\underline{-(2x^2+2)} \\
4x+5
\end{array}
$$

To put it in the form of $f=qg+r$, we have:

$x^5 - 2x^4 + 3x + 7 = (x^2 + 1)(x^3-2x^2-x+2)+ 4x+5$, where $r=4x+5$, and no term of r is divisible by $x^2 + 1$.

The S-polynomial is used to get rid of the leading terms of the polynomials that we are using. The S – polynomial is defined as $S(f_1, f_2) = \frac{M}{LT(f_1)} f_1 - \frac{M}{LT(f_2)} f_2$, where $f_1, f_2 \in k[x_1,...,x_n]$ and M is the least common multiple of the leading monomial of f_1 and f_2. The S-polynomial combines f_1 and f_2 to cancel out the leading terms for each polynomial. We then have polynomials that are easier to solve.

Example of applying the S-Polynomial:

Let $f_1 = x^3 - 2xy$ and $f_2 = x^2y - 2y^2 - x$, with lexicographical ordering as x>y. Given F=$\{ f_1, f_2 \}$, then S(f_1, f_2)=$\frac{x^3y}{x^3} f_1 - \frac{x^3y}{x^2y} f_2 = y \cdot f_1 - xf_2 = x^2$.

Now to compute the Groebner basis for the above (f_1, f_2), we want to use Buchberger's Algorithm to do so. In order to do so we must perform general polynomial division. General polynomial division is used once the S-polynomial of two polynomials is found. Once the S-polynomial is calculated and the remainder is not zero, we use general polynomial division in order to find the Groebner basis.

General polynomial division is used once the S-polynomial is found. The result is then divided by each polynomial in the set. The remainder is then put in the set and the process is continued until each division gives a remainder of zero.

General Polynomial Division

Given a polynomial *P,* it can be expressed as a factored form using the set of polynomials that generate the ideal.

$$P = \sum_{i=0}^{n} p_i f_i$$

Where f_i represents the polynomials in *I*, and p_i represents the polynomial factors of f_i that when combined together form the polynomial *P*.

Example of General Polynomial Division

Given $g = xy^2 - x$, we want to divide *g* by $g_1 = xy + 1$ and $g_2 = y^2 - 1$, dividing by g_1 first and then by g_2.

$$xy^2 - x = y \cdot (xy + 1) + 0 \cdot (y^2 - 1) - x - y$$, where $-x - y$ is the remainder.

Dividing g in the other order $g_2 = y^2 - 1$ and then $g_1 = xy + 1$, then the remainder is zero.

$$xy^2 - x = x \cdot (y^2 - 1) + 0 \cdot (xy + 1)$$

When dividing by g_1 and g_2, look to see if the lead monomial of g is divisible by the lead monomial of g_1 or g_2. If it is not then add it to the remainder if it is then add what you get to the quotient.

When performing general polynomial division it is important to factor the polynomial as much as possible, therefore that latter factorization is the ideal general polynomial division because we get a remainder of zero.

Buchberger's Algorithm

Given $\{f_1, ..., f_s\} \subset k[x_1, ..., x_n]$, *consider the algorithm which starts with* $F = \{f_1, ..., f_s\}$ *and then repeat the two steps*

1. *Compute* $\overline{S(f_i, f_j)}^F$ *for all* $f_i, f_j \in F$ *with* $i < j$.

   ($\overline{S(f_i, f_j)}^F$ *is defined as the remainder of the S-polynomial with division by F)*

2. *Augment F by adding the nonzero* $\overline{S(g_i, g_j)}^F$.

   (For each time F has a polynomial added that is nonzero), until the Compute Step gives only zero remainders. This algorithm always terminates and the final value of F is a Groebner basis of $\langle f_1, ..., f_s \rangle$. (Cox, 8)

Bruno Buchberger developed the idea of a Groebner basis in 1965 and named it after his advisor Wolfgang Groebner. (Becker, vi)

The S-Polynomial is found for each different pair of polynomials in F. Do generalized polynomial division and then if the S-polynomial of the two polynomials is not equal to zero then the remainder is put in F as another polynomial. This process is continued until finding the S-Polynomial of each pair of polynomials gives you all remainders of zero.

Example of finding a Groebner basis for an ideal

To find the Groebner basis for (f_1, f_2) where $f_1 = x^3 - 2xy$, $f_2 = x^2y - 2y^2 - x$ and $f_3 = x^2$ because we know $\overline{S(f_1, f_2)} = x^2$. Now we have, $F_1 = \{f_1, f_2, f_3\}$, and we have to compute S-polynomials of possible pairs for the three polynomials in F_1, and see if we get a remainder of zero when doing generalized polynomial division.

$\overline{S(f_1, f_2)}^{F_1} = 0$

$\overline{S(f_1, f_3)}^{F_1} = -2xy = f_4$

$\overline{S(f_2, f_3)}^{F_1} = -x - 2y^2 = f_5$

This does not give us a Groebner basis so we have to continue with f_4 and f_5. We are now going to look at $F_2 = \{f_1, f_2, f_3, f_4, f_5\}$.

$\overline{S(f_1, f_5)}^{F_2} = -4y^3$

$\overline{S(f_4, f_5)}^{F_2} = -2y^3$

$\overline{S(f_i, f_j)}^{F_2} = 0$, for every other $i<j$.

Now $f_6 = y^3$, $F_3 = \{f_1, f_2, f_3, f_4, f_5, f_6\}$ and $\overline{S(f_i, f_j)}^{F_3} = 0$ for $1 \le i, j \le 6$.

The Groebner basis for the ideal generated by f_1 and f_2 with lexicographic ordering of x>y is

$F_3 = \{x^3 - 2xy, x^2y - x - 2y^2, x^2, -2xy, -x - 2y^2, y^3\}$.

Example to check if $\{f_1, f_2\}$ is a Groebner basis for *I*.

Given the ideal I generated by the following polynomials:

$$f_1 = x^3y - xy^2 + 1$$

$$f_2 = x^2y^2 - y^3 - 1,$$

We want to find out if F=$\{f_1, f_2\}$ is a Groebner basis. $S(f_1, f_2) = yf_1 - xf_2 = x + y$, this is it's own remainder when divided by $\{f_1, f_2\}$, so F is not a Groebner basis for I. In order for it to be a Groebner basis it would have to have a remainder of zero. So set $f_3 = x + y$.

Let $F_1 = \{f_1, f_2, f_3\}$ Now $\overline{S(f_1, f_2)}^{F_1} \equiv 0 \bmod F_1$, and

$\overline{S(f_1, f_3)}^{F_1} = f_1 - x^2yf_3 = -x^2y^2 - xy^2 + 1 \equiv 0 \bmod F_1$ and

$\overline{S(f_2, f_3)}^{F_1} = f_2 - xy^2f_3 = -xy^3 - y^3 - 1 \equiv y^4 - y^3 - 1 \bmod F_1$.

Now let $f_4 = y^4 - y^3 - 1$ and we get $F_2 = \{f_1, f_2, f_3, f_4\}$ and $\overline{S(f_2, f_3)}^{F_2} \equiv 0 \bmod F_2$, as well as every other pair of polynomials in F_2. So we have that F_2 is a Groebner basis for I and the LT(I) is generated by the leading terms of the four polynomials in F_2 and LT(I)= $(x^3y, x^2y^2, x, y^4) = (x, y^4)$. So x+y and $y^4 - y^3 - 1$ contain the leading terms in I and therefore $\{x+y, y^4 - y^3 - 1\}$ gives a minimal Groebner basis for I, where I=$(x+y, y^4 - y^3 - 1)$ □

With the help of technology, such as Maple, problems as the above can have the Groebner basis computed much faster than doing it by hand and problems with much larger degree can be done in a relatively timely fashion. The following is the above example executed in Maple.

```
f[1]:=x^3*y-x*y^2+1;
```

$$f_1 := x^3 y - x y^2 + 1$$

```
f[2]:=x^2*y^2-y^3-1;
```

$$f_2 := x^2 y^2 - y^3 - 1$$

```
b1:=gbasis([f[1],f[2]],plex(x,y));
```

$$b1 := [-y^3 - 1 + y^4, y + x]$$

```
b2:=gbasis([f[1],f[2]],plex(y,x));
```

$$b2 := [-1 + x^4 + x^3, y + x]$$

In b2, the Groebner basis with lexicographic ordering y>x is found. It is important to note that the basis depends on the monomial ordering.

First Conti-Traverso Algorithm and Example

A Groebner based algorithm was found by P. Conti and C. Traverso that solves an integer program, Ax=b, but only if all elements of b are positive. To do this we must find the toric ideal. The toric ideal of A is the kernel of the following homomorphism, $k\left[x_1 \cdot x_2 \cdots x_d\right] \to k\left[t_1^{\pm 1} \cdot t_2^{\pm 1} \cdots t_d^{\pm 1}\right]$. Where $x_j \to t^{a_j}$ and a_j is the jth column of A. (Thomas, 122)

Fact: *The toric ideal* $I_A = \langle x^{u_i^+} - x^{u_i^-} : u_i \in \ker_{\mathbb{Z}}(A), i = 1,...,t \rangle$.

The toric ideal is not easy to find because we have to know the solutions to Ax=0, but to give an idea of what the toric ideal of a matrix looks like we will look at the following matrix A.

$$A = \begin{bmatrix} 1 & 1 & -1 \\ -1 & 3 & 2 \end{bmatrix}$$

This is the toric ideal for the matrix A,

$$I_A = \left\langle -x_2 + x_3^4 x_1^5 \right\rangle.$$

Since the toric ideal is very hard to find it was found by work that we will look at in the next section on the Conti-Traverso Algorithm. This is just to give an idea of what the toric ideal of any matrix may look like.

The first step of the algorithm is to find the Groebner basis of the toric ideal, the lexicographic ordering is based on the value of the elements in the cost vector. To get a Groebner basis you need a monomial ordering. Our cost vector gives us the ordering that

one needs. For example, given c = (2, 1, 4, 3, 1), the monomial ordering would be $x_3 \succ x_4 \succ x_1 \succ x_2 \succ x_5$. We assume that all of the elements in the cost vector are positive, because it is impossible for something to cost negative dollars in order to be produced.

The following corollaries are important in the process of this algorithm.

Corollary 1:

The reduced Groebner basis $G_{\succ c}$ of the ideal that represents the toric ideal associated with the matrix A consists of a finite set of binomials of the form $x^{u_i^+} - x^{u_i^-}$, where $u \in \ker_{\mathbb{Z}}(A)$ and $\succ c$ is the ordering based on the cost vector. (Thomas, 123)

A reduced Groebner basis is when no monomial of any element of a Groebner Basis is divisible by the lead monomial of any other polynomial in the Groebner basis. The reduced Groebner basis is unique.

Corollary 2:

The normal form of a monomial in k[x] with respect to the reduced Groebner basis $G_{\succ c}$ of I_A is again a monomial.

The normal form of a polynomial *f* with respect to the lexicographic ordering is the unique remainder of dividing *f* with respect to $G_{\succ c}$. (Thomas, 123)

The Conti-Traverso Algorithm for $IP_{A,\succ c}$.

1. *Compute the reduced Groebner basis $G_{\succ c}$ of I_A*

2. *For* $b \in pos_{\mathbb{Z}}(A)$ *and any solution u of* $IP_{A,\succ c}(b)$*, compute* $x^{u'} = nf_{\succ c}(x^u)$. *The vector* u' *is an optimal solution of* $IP_{A,\succ c}(b)$*, (where* $nf_{\succ c}(f)$ *is the normal form of the polynomial f).* (Thomas,123)

Idea of the proof.

Given Ax=b, we are given the Groebner basis for the toric ideal of Matrix A.

Let $G_{\prec_c} = \{x^{\alpha_i} - x^{\beta_i}, i = 1,..., p\}$, where x^{α_i} is the leading term. The polynomial $x_1^{a_1} \cdots x_n^{a_n} - x_1^{b_1} \cdots x_n^{b_n}$ gives us the solution $a_1 \cdots a_n - b_1 \cdots b_n$ to Ax=0. This gives us $A\begin{bmatrix} a_1 \\ . \\ . \\ . \\ a_n \end{bmatrix} = A\begin{bmatrix} b_1 \\ . \\ . \\ . \\ b_n \end{bmatrix}$. Given that $A\begin{bmatrix} w_1 \\ . \\ . \\ . \\ w_n \end{bmatrix}$=b, then $A\left(\begin{bmatrix} w_1 \\ . \\ . \\ . \\ w_n \end{bmatrix} + \begin{bmatrix} a_1 \\ . \\ . \\ . \\ a_n \end{bmatrix} - \begin{bmatrix} b_1 \\ . \\ . \\ . \\ b_n \end{bmatrix}\right) = b$ and the b_i's can be traded for the a_i's, because the a_i's cost less than the b_i's, or vice versa. This is based on the cost vector. The cost vector allows us to determine the value of each variable. In this case since the a_i's cost less than the b_i's, we would want to have more of the a_i's □

Example:

Given Ax=b, where A=$\begin{bmatrix} 1 & 2 & 1 & 3 & 1 \\ 2 & 0 & 1 & 0 & 1 \end{bmatrix}$, b=$\begin{bmatrix} 250 \\ 100 \end{bmatrix}$ and the cost vector c = (5, 4, 3, 2, 1).

Given the solution $\begin{bmatrix} 10 \\ 20 \\ 30 \\ 40 \\ 50 \end{bmatrix}$, we want to use this solution to find the optimal solution.

The first step is to compute the reduced Groebner basis of the toric ideal and get $G_c = [-x_5 + x_3, -x_4^2 + x_2^3, -x_2x_5^2 + x_1x_4, x_2^2x_1 - x_5^2x_4, -x_5^4 + x_1^2x_2]$. We find the normal form with respect to G_c is $x_5^{100}x_4^{50}$, therefore (0, 0, 0, 50, 100) is the optimal solution.
For the first binomial in the Groebner basis to reduce cost all of the x_3's, that have value three, can be replaced with x_5's that have a value of one. For the second binomial three x_2's can take the place of two x_4's and have a value of six instead of eight. The third binomial two x_5's and one x_2 can be replaced with one x_1 and one x_4. For the fourth binomial, two x_5's and one x_4 can be switched with two x_2's and one x_1. For the final binomial four x_5's can be traded with two x_1's and one x_2 in order to minimize cost.

This method is only good when our b is positive and we are given a solution to the system to work with. However there is an algorithm to find the optimal solution with out knowing a solution and our b can have negative values, it is known as the original Conti - Traverso Algorithm.

Original Conti-Traverso Algorithm

Since the previous algorithm does not allow b to have negative elements and we must know at least one solution, it is not always practical. We are interested in finding the optimal solution for an integer program, if b has positive or negative integer values and not having any solution to our system. The following algorithm gives us the steps in order to do so. This algorithm is also helpful because it gives a method for finding the toric ideal of the matrix A.

Conti-Traverso Algorithm:

Consider the ideal J = $\left\langle x_j t^{a_j^-} - t^{a_j^+}, j = 1,...,n, t_0 t_1 \cdots t_d - 1 \right\rangle$ in the polynomial ring

k[$x_1,...,x_n,t_0,t_1,...,t_d$]. Let $t_{\bar{0}} = \{t_1,...,t_d\}$

1. *Compute the reduced Groebner basis $G_{\succ'}$ of J with respect to any elimination term order $\succ'$ such that $\{t_0,t_1,...,t_d\} \succ' \{x_1,...,x_n\}$ and $\succ'$ restricted to k[x] induces the same total order as $\succ_c$.*
2. *In order to solve $IP_{A,\prec_c}(b)$, form the monomial $t^b = t_0^{\beta} t_{\bar{0}}^{b+\beta(e_1+...+e_d)}$ where $\beta = \max\{|b_j| : b_j < 0\}$ and e_i is the i-th unit vector $\mathbb{R}^d$. Compute the normal form $t^{\gamma} x^{u'}$ of the monomial t^b with respect to $G_{\succ'}$.*

If $\gamma = 0$ then $IP_{A,\prec_c}(b)$ is feasible with optimal solution u'. Else $IP_{A,\prec_c}(b)$ is infeasible.

(Thomas,124)

The ideal $J = \left\langle x_j t^{a_j^-} - t^{a_j^+}, j = 1,...,n, t_0 t_1 \cdots t_d - 1 \right\rangle$ *has the property that* $J \cap k[x] = I_A$, *which in turn implies that* $G_{\succ} \cap k[x] = G_{\succ_c}$. (Thomas,124)

What this is telling us is that when we take our ideal J and intersect it with $k[x]$, we get the toric ideal which is made up of just our x's.

Since $J = \left\langle x_j t^{a_j^-} - t^{a_j^+}, j = 1,...,n, t_0 t_1 \cdots t_d - 1 \right\rangle$, we can say that in J we have

$x_j - \frac{t^{a_j^+}}{t^{a_j^-}} = x_j - t^{a_j}$, where j represents the columns in A. If we are given $\begin{bmatrix} 1 \\ -1 \\ 2 \\ -3 \end{bmatrix}$ as one of

the columns in matrix A, then we get $\left\langle x_j t_2^{-1} t_4^{-3} - t_1 t_3^2 \right\rangle = x_j - \frac{t_1 t_3^2}{t_2^1 t_4^3} = x_j - t_1 t_3^2 t_2^{-1} t_4^{-3} =$

$x_j - t_1 t_2^{-1} t_3^2 t_4^{-3} = x_j - t^{a_j}$. Therefore when reducing $x^{u^+} - x^{u^-}$, we can replace x_j with t^{a_j}. This is used to show that $x^{u^+} - x^{u^-}$ is in the ideal J.

The following proof has been directly taken from the work of Rekha R. Thomas.

Given $k[x_1 \cdot x_2 \cdots x_d][t_0 \cdot t_1 \cdots t_d] \to k\left[t_1^{\pm 1} \cdot t_2^{\pm 1} \cdots t_d^{\pm 1}\right]$ and $t_1 \to t_1$, $t_2 \to t_2, \ldots,$

$t_0 \to t_1^{-1} t_2^{-1} \cdots t_d^{-1}$.

Therefore, Step 1 of the algorithm indirectly achieves step 1 of the previous algorithm and the exponent vectors of all monomials of the form x^u *encountered during the reduction of* t^b *with respect to* $G_{\succ}$ *lie in* P_b^I. *Since* $\succ'$ *is elimination order, the*

algorithm with reduce t^b to a monomial of the form x^u as long as $IP_{A,\succ_c}(b)$ is feasible and then will proceed to reduce this solution to the optimal solution of the program. Suppose now that the normal form $t^\gamma x^{u'}$ of t^b has $\gamma \neq 0$ and $IP_{A,\succ_c}(b)$ has a solution of v. Then since $\succ'$ was an elimination order with $\{t_0,t_1,...,t_d\} \succ' \{x_1,...,x_n\}$, the binomial $t^\gamma x^{u'} - x^v \in J$ has $t^\gamma x^{u'}$ as leading term. This contradicts that $t^\gamma x^{u'}$ is the normal form of t^b with respect to $G_{\succ}$. Therefore, if $\gamma = 0$, by the same argument as in the proof of the previous algorithm, u' is the optimal solution to $IP_{A,\succ_c}(b)$. □ (Thomas, 124)

In order for the algorithm to be effective then the feasible region must be bounded.

Example 1

Given Ax=b, with A= $\begin{bmatrix} 1 & 2 & -3 & 4 \\ 2 & -2 & 4 & 3 \\ 1 & 3 & -4 & 2 \end{bmatrix}$ and b= $\begin{bmatrix} 15 \\ 11 \\ 11 \end{bmatrix}$, we want to find the optimal solution where c=(3, 2, 1). The first thing to do is compute the reduced Groebner basis of the ideal J=$[x_1 - t_1t_2^2t_3, x_2t_2^2 - t_1^2t_3^3, x_3t_1^3t_3^4 - t_2^4, x_4 - t_1^4t_2^3t_3^2]$ with monomial ordering $x_1 \succ x_2 \succ x_3$. Once that is done we get G=$[-x_2^{25}x_3^{17}x_4^4 + x_1^{15}, ..., -x_4 + t_1^3t_2t_3x_1]$ (see Maple work below). We then find the normal form of $t_1^{15}t_2^{11}t_3^{11}$ with respect to G and get $x_4^3x_2^4x_3^2x_1$, therefore the optimal solution is (1, 4, 2, 3).

Given $-x_2^{25}x_3^{17}x_4^4 + x_1^{15}$ is in the toric ideal. When we replace x_j with t^{a_j}, we get

$$-(t^{a_2})^{25}(t^{a_3})^{17}(t^{a_4})^4 + (t^{a_1})^{15} = -(t_1^2t_2^{-2}t_3^3)^{25}(t_1^{-3}t_2^4t_3^{-4})^{17}(t_1^4t_2^3t_3^2)^4 + (t_1t_2^2t_3)^{15} = 0.$$

Maple work for Example 1:

```
> J:=[x1-t1*t2^2*t3,x2*t2^2-t1^2*t3^3,x3*t1^3*t3^4-t2^4,x4-
t1^4*t2^3*t3^2];
```

$$J := [x1 - t1\,t2^2\,t3,\ x2\,t2^2 - t1^2\,t3^3,\ x3\,t1^3\,t3^4 - t2^4,\ x4 - t1^4\,t2^3\,t3^2]$$

```
> M1:=gbasis(J,plex(t1,t2,t3,x1,x2,x3,x4));
```

$$M1 := [-x2^{25}\,x3^{17}\,x4^4 + x1^{15},\ x2^{10}\,x3^7\,x4^2\,t3 - x1^7,\ -x2^{15}\,x3^{10}\,x4^2 + x1^8\,t3,\ -x2^5\,x3^3\,x4 + t3^2\,x1\,x4,$$
$$t3^2\,x1^2 - x1\,x2^5\,x3^3,\ -x1^6 + x2^5\,x3^4\,x4^2\,t3^3,\ -x1^5 + t3^5\,x3\,x4^2,\ t3^7\,x4^2 - x2^5\,x3^2\,x1^4,$$
$$t2\,x2^6\,x3^4\,x4 - x1^4,\ x2^6\,t2\,x1\,x3^3 - t3^5\,x4,\ -x4\,t3^3 + x2\,t2\,x1^2,\ -t3\,x2^4\,x3^3\,x4 + t2\,x1^3,$$
$$t3^2\,t2\,x2\,x3\,x4 - x1^3,\ -x2^4\,x1^2\,x3^2 + t3^4\,t2\,x4,\ x2^2\,x3\,x4\,t2^2 - t3\,x1\,x4,\ -t3\,x1^2 + x2^2\,t2^2\,x3\,x1,$$
$$t2^2\,x4\,t3 - x2^3\,x3^2\,x4,\ t2^2\,t3\,x1 - x2^3\,x3^2\,x1,\ t2^3\,x1^2 - t3^2\,x2^2\,x3^2\,x4,\ t2^4 - x2\,x3\,x1,$$
$$-x1\,x4 + x2^3\,x3^2\,x4\,t1,\ -x1^2 + x2^3\,x3^2\,x1\,t1,\ t1\,x2^2\,x1^2\,x3 - t2\,t3^2\,x4,\ -t2\,x2^3\,x3^2\,x4 + t1\,x1^3,$$
$$t1\,x2\,x3\,x4\,t3 - t2^2\,x4,\ -x1\,t2^2 + t1\,t3\,x2\,x3\,x1,\ -x2^2\,x3\,x4 + t3^2\,x4\,t1,\ t1\,t3^2\,x1 - x2^2\,x3\,x1,$$
$$-x2\,x3\,x4\,t3 + t1\,t2\,x1^2,\ -x1 + t1\,t2^2\,t3,\ t1\,t2^3\,x2\,x1 - t3^2\,x4,\ -t2\,x4 + t1^2\,x1^2,\ -x2\,t2^2 + t1^2\,t3^3,$$
$$-x4\,t3 + t1^2\,t2\,x2^2\,x3\,x1,\ t1^2\,t2^3\,x1 - x2\,x3\,x4,\ -x4 + t1^3\,t2\,t3\,x1]$$

```
> b:=Matrix([[15],[11],[11]]);
```

$$b := \begin{bmatrix} 15 \\ 11 \\ 11 \end{bmatrix}$$

```
> normalf(t1^15*t2^11*t3^11,M1,plex(t1,t2,t3,x1,x2,x3,x4));
```

$$x4^3\,x2^4\,x3^2\,x1$$

The result of the normal form tells us that because there are three x_4, four x_2, two x_3 and one x_1, the optimal solution is (1, 4, 2, 3).

Examples of the Original Conti-Traverso Algorithm

Example 2

Given Ax=b, with A=$\begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & 1 \end{bmatrix}$ and b=$\begin{bmatrix} -4 \\ 16 \end{bmatrix}$, we want to find the optimal solution.

Based on the algorithm the first thing to do is to compute the reduced Groebner basis of the ideal J=$[x_1 - t_1t_2^3, x_2t_1 - t_2^2, x_3t_2 - t_1^2, t_0t_1t_2 - 1]$ with monomial ordering $x_1 \succ x_2 \succ x_3$. When that is done we get $[x_1^3 - x_2^7x_3^5, ..., x_3t_0t_2^2 - t_1]$ (See attached Maple Work). We then form the monomialc $t_0^4t_1^0t_2^{20}$, since there was a negative element in b, four was added so that it would no longer be negative, because the normal form can not be found with rational expressions. We now compute the normal form of $t_0^4t_1^0t_2^{20}$ with respect to the Groebner basis. When this is done we get $x_2^7x_1x_3$. So the optimal solution to our problem is (1,7,1).

Maple work for Example 2:

```
> M:=[x1-t1*t2^3,x2*t1-t2^2,x3*t2-t1^2,t0*t1*t2-1];
```

$$M := [x1 - t1\,t2^3, x2\,t1 - t2^2, x3\,t2 - t1^2, t0\,t1\,t2 - 1]$$

```
> MM:=gbasis(M,plex(t0,t1,t2,x1,x2,x3));
```

$$MM := [x1^3 - x2^7\,x3^5, x2^4\,x3^3\,t2 - x1^2, x1\,t2 - x2^3\,x3^2, -x1 + x2\,t2^2\,x3, t2^3 - x2^2\,x3, x2\,t1 - t2^2,$$
$$-x2^2\,x3^2\,t2 + x1\,t1, t1\,t2 - x2\,x3, -x3\,t2 + t1^2, -1 + t0\,x3\,x2, x1\,t0 - t2^2, t0\,x3\,t2^2 - t1]$$

```
> normalf(t0^4*t2^20,MM,plex(t0,t1,t2,x1,x2,x3));
```

$$x2^7\,x1\,x3$$

Why the Groebner Basis Finds the Optimal Solution

If v is any solution to Ax=b, then $t^b - x^v \in J$, the binomial must be in the toric ideal. If that is the case then t^b must reduce to a polynomial in terms of x's since v is a solution. The reason this is so is, if we are given that $m_1 - m_2$ is in the kernel and given h and g, if m_1 divides h, then h= m_1 w and g= m_2 w, so h-g is in the kernel. From this we can say that m_1 can be replaced with m_2 when reducing h.

Given $t^b = t_0^{\beta} t_{\bar{0}}^{b+\beta(e_1+\ldots+e_d)}$ and the feasible solution exists, say v, so Av=b, we want to show that $t^b - v \in J$, so that when we reduce t^b, we can reduce it to v. Whatever the reduction is, it is going to be less than or equal to the solution v. Therefore that is why the normal form of t^b is a polynomial in terms of x.

Given

$$A\begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ a_d \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & \cdot & \cdot & c_d \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ a_d \end{bmatrix}$$

, where A can be represented as d columns of n by one matrices.

We then have, $\begin{bmatrix} c_1 & c_2 & \cdot & \cdot & c_d \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ a_d \end{bmatrix} = a_1c_1 + a_2c_2 + \ldots + a_dc_d$.

We know from previous work that,

$$x_1 \rightarrow t_1^{c_1}$$

$$x_1^{a_1} \rightarrow t_1^{a_1c_1}$$

$$x_1^{a_1} x_2^{a_2} \rightarrow t^{a_1c_1} t^{a_2c_2},$$

And $t_1^{a_1c_1} t_2^{a_2c_2} = t_0^{a_1c_1+a_2c_2}$, therefore $x_1^{a_1} x_2^{a_2} \cdots x_d^{a_d} \rightarrow t_1^{a_1c_1} t_2^{a_2c_2} \cdots t_d^{a_dc_d} = t_0^{a_1c_1+a_2c_2+\ldots+a_dc_d}$.

If v is a solution to Ax=b, so Av=b. So, $v = x_1^{w_1} x_2^{w_2} \cdots x_d^{w_d}$, and based on our homomorphism this becomes $t_1^{a_1w_1} t_2^{a_2w_2} \cdots t_d^{a_dw_d}$

$$= t_0^{a_1w_1+a_2w_2+\ldots+a_dw_d}$$

$$= t_0^{A\begin{bmatrix} w_1 \\ w_2 \\ \cdot \\ \cdot \\ w_d \end{bmatrix}}$$

$$= t_0^{\,b}.$$

If the homomorphism is applied to $t^b = t_0^{\beta} t_{\bar{0}}^{b+\beta(e_1+\ldots+e_d)}$, we get

$$= \frac{1}{t_1^{\beta} t_2^{\beta} \cdots t_d^{\beta}} t_{\bar{0}}^{b+\beta(e_1+\ldots+e_d)}$$

$$= \frac{1}{t_1^{\beta} t_2^{\beta} \cdots t_d^{\beta}} t_{\bar{0}}^{b} t_{\bar{0}}^{\beta(e_1+\ldots+e_d)}$$

$$= \frac{1}{t_1^{\beta} t_2^{\beta} \cdots t_d^{\beta}} t_{\bar{0}}^{b} t_1^{\beta} t_2^{\beta} \cdots t_d^{\beta}$$

$$= t_{\bar{0}}^{b}$$

So $t^b - v \in J$.

Therefore, v is any feasible reduced solution that is found from taking the normal form.

Conclusion

Integer Programming problems can have several solutions. When solving these types of problems only certain solutions work for the problem at hand. One thing that one may look for is to maximize profit. Something else that may be important when solving these types of problems is to minimize the cost. An integer programming problem may have several solutions but when the problem is to minimize the cost we are only looking for the solutions that can do that. This solution is known as the optimal solution.

Since integer programming problems can take a very long time to solve by hand we have found ways to solve this type of problem by using a method that involves finding the Groebner basis. This too can be very lengthy, but with the help of technology can be done in a timely fashion.

There were two methods that were looked at to deal with solving integer programming problems. The first Conti-Traverso Algorithm was good for solving these types of problems as long as our b had all positive values. This is not always the case so there was the second Conti-Traverso algorithm that found the optimal solution where b could have any type of integer value. Both of these methods involved finding the toric ideal of our Matrix A, and then computing the Groebner basis of this ideal. The cost vector gave us the order of our variables. The variables that cost the least were ordered first, since we were looking to minimize cost. Since the second method allowed for there

to be negative values in b, there were a few more steps in order to find our optimal solution. Once the Groebner basis was found we had to find the normal form of our b vector with respect to the Groebner basis. Once this was done, we had the optimal solution. With the use of technology and computing the Groebner basis, finding the optimal solution for an integer programming problem has been made easier.

Bibliography

Becker, T., Weispgenning V., and Kredel, H.: Grobner Basis, A Computational Approach to Commutative Algebra, Springer-Verlag, New York, NY, 1993.

Cox, David A.: Applications of Computational Algebraic Geometry, Proceedings of Symposia in Applied Mathematics, Volume 53, pp 1-24; American Mathematical Society, Providence, RI, 1998.

Dummit, David S. and Fook, Richards: Abstract Algebra, John Wiley and Sons, Inc., 2004, p 319.

Noble, Ben, Applied Linear Algebra, Prentice Hall, Inc., Englewood Cliffs, NJ, 1969.

Sierksma, Gerd, Linear and Integer Programming, Theory and Practice, second edition, Marcel Dekker, Inc., New York, NY, 2002.

Thomas, Rekha R.: Applications of Computational Algebraic Geometry, Proceedings of Symposia in Applied Mathematics, Volume 53, pp 119-142; American Mathematical Society, Providence, RI, 1998.

Websites:
http://en.wikipedia.org/wiki/Gr%c3%B6bner_basis page 2
http://web.mit.edu/singular.v2.0.5/distrib/Singular/2-0-5/html/sing_.htm
http://mathworld.wolfram.com/GroebnerBasis.html