Linear Programming: The Simplex Method and Beyond

Rohan Saraogi, Mihir Trivedi

December 16, 2016

1 Linear Programming

1.1 Introduction

Linear programming is a technique for the optimization of a linear objective function, subject to certain constraints. The constraints define a region in $\mathbb{R}^n$, called the 'feasible region'. A linear programming algorithm finds a point in the feasible region where the objective function has a maximum or minimum value, if such a point exists.

There are three preliminary terms in linear programming – linear objective function, decision variables, and constraints. The linear objective function is the n-dimensional linear function

$$f(x_1, ..., x_n) = a_1x_1 + ... + a_nx_n, a_1, ..., a_n \in \mathbb{R}$$

in $\mathbb{R}^n$ that needs to be optimized. The decision variables are the x_i variables that define the linear objective function. The constraints define the feasible region $\mathbb{R}^n$. They are of two kinds:

- Closed half-spaces in $\mathbb{R}^n$ given by

  $$b_1x_1 + ... + b_nx_n \leq \text{constant}, b_1, ..., b_n \in \mathbb{R}.$$

  Given m such closed half-spaces, the region in $\mathbb{R}^n$ that satisfies all m constraints is termed a **closed convex polytope.** Thus, the polytope is the solution to the system:

  $$\begin{gathered} a_{11}x_1 + ... + a_{1n}x_n \leq b_1 \\ ... \\ ... \\ ... \\ a_{m1}x_1 + ... + a_{mn}x_n \leq b_m \end{gathered}$$

- Non-negativity of the decision variables

  $$x_i \geq 0, 1 \leq i \leq n$$

The form of the constraints applies specifically to the problem of maximizing the objective function. If the goal is to minimize it, the negative of the objective function can be considered, since the point at which the negative of the objective function is maximized, is precisely the point at which the objective function itself is minimized.

The objective function and the constraints are collectively termed a 'linear program'. The solution of the linear program is the point in the feasible region where the objective function is maximized or minimized (as required).

1.2 Graphical Method

While solving linear programs in general requires an algorithmic approach, it is quite easy to solve them in $\mathbb{R}^2$ or $\mathbb{R}^3$ using the graphical method. Consider the following linear program:

$$z = 750x_1 + 1000x_2$$

Let the constraints be:

$$\begin{gathered} x_1 + x_2 \leq 10 \\ x_1 + 2x_2 \leq 15 \\ 4x_1 + 3x_2 \leq 25 \\ x_1, x_2 \geq 0 \end{gathered}$$

Figure 1 shows the region in $\mathbb{R}^2$ where all the constraints are satisfied. This is indicated in grey.

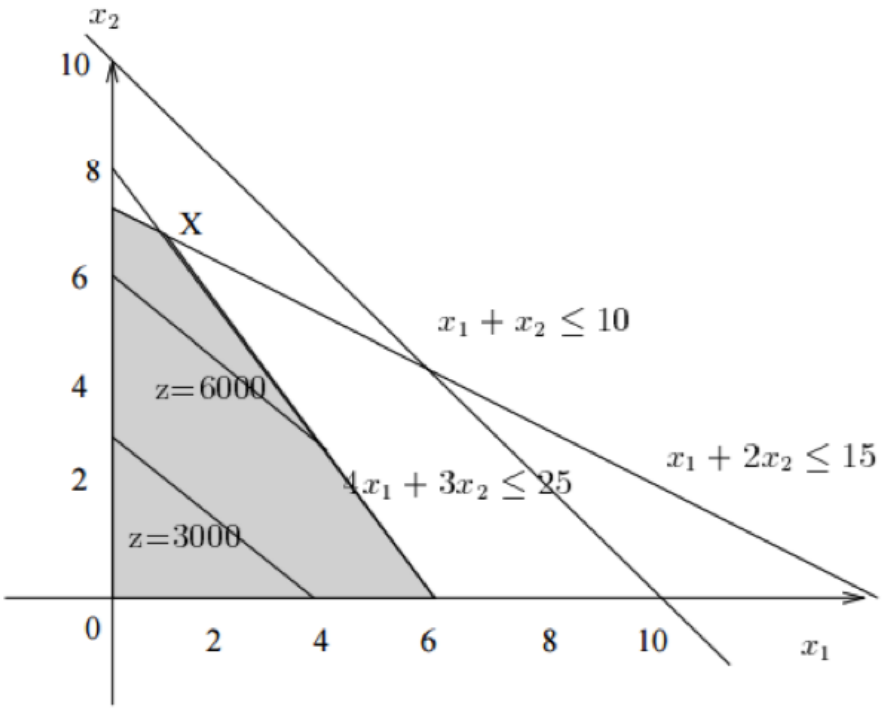
There are several approaches to determine the point in the grey region where the value of z is maximized. Consider the following approach:

- Set the value of z to a constant and plot the associated line in the grey region. Ensure that the plotted line has at least some segment lying within the grey region
- Increment z by a certain amount, and then repeat step 1
- Repeat Step 2 3-5 times.

The plotted lines are called **iso-lines** because z is a constant along these lines. In Figure 1, this is done for $z = 3000$ and $= 6000$. It is observed that as z is increased, the lines move 'rightwards.' Hence, z attains a maximum value at the line that is further right, while still remaining within the grey region. Such a line would pass through the point X in Figure 1. Thus, z is maximized at point X.

Note that X lies at the boundary of the grey region. This is relevant, since it can be shown that, in general, the solution to any linear program (if it exists) will always be found at the boundary of the region. The only exception to this is when the objective function is a constant, in which case the solution is found everywhere within the region. To further develop this idea, it is necessary to consider the notions of convexity and concavity in $\mathbb{R}^n$.

Figure 1: Graphical Method



1.3 Convexity and Concavity

There are three basic definitions to consider.

Definition 1.1. A subset C of $\mathbb{R}^n$ is convex if given any two points $u, v \in C$, the set of convex combinations of u and v are in C. The convex combinations w_t are given by:

$$\{w_t \in \mathbb{R}^n \text{ s.t } w_t = ut + (1-t)v, 0 \leq t \leq 1\}$$

Definition 1.2. Let $C \subset \mathbb{R}^n$ be a convex set, and let $f : C \to \mathbb{R}$. Then $f(x)$ is convex on C if for all x, $y \in C$ and $t \in [0, 1]$:

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

Definition 1.3. Let $C \subset \mathbb{R}^n$ be a convex set, and let $f : C \to \mathbb{R}$. Then $f(x)$ is concave on C if for all x, $y \in C$ and $t \in [0, 1]$:

$$f(tx + (1-t)y) \geq tf(x) + (1-t)f(y)$$

It follows trivially that any linear function in $\mathbb{R}^n$ is both concave and convex. There are two relevant theorems.

Theorem 1.1. Let $C \subset \mathbb{R}^n$ be a convex set, and let $f : C \to \mathbb{R}$ be a convex function. Let x^* be a local minimizer of f. This implies that there exists a $p > 0$ such that for every $x \in C$:

$$\|x - x^*\| \leq p \Rightarrow f(x) \geq f(x^*)$$

Then x^* is a global minimizer of f

Proof: Assume that there exists an x_0 such that $f(x_0) < f(x^*)$. Then, for any $t \in (0, 1]$:

$$f((1-t)x^* + tx_0) \leq (1-t)f(x^*) + tf(x_0) < (1-t)f(x^*) + tf(x^*) = f(x^*) \ (1)$$

Choose t small enough so that:

$$\|(1-t)x^* + tx_0 - x^*\| < p$$

By, from (1):

$$f((1-t)x^* + tx_0) < f(x^*)$$

Then $(1-t)x^* + tx_0$ is a new minimizer of f. This contradicts the fact that x^* is a local minimizer of f. Hence, there cannot exist such an x_0. This completes the proof.

Theorem 1.2. Let $C \subset \mathbb{R}^n$ be a convex set, and let $f : C \to \mathbb{R}$ be a concave function. Let x^* be a local maximizer of f. This implies that there exists a $p > 0$ such that for every $x \in C$:

$$\|x - x^*\| \leq p \Rightarrow f(x)leqf(x^*)$$

Then x^* is a global maximizer of f. The proof of this theorem is symmetric to theproof of theorem 1.1. Theorems 1.1 and 1.2 say that if a local maximum (or minimum) of the objective function is found within the feasible region, then the global maximum or minimum of the objective function has also been found. It can now be shown that for a non-constant linear objective function, the maximum (or minimum) will always be found at the boundary of the feasible region.

1.4 Maximum and Minimum Principle

Maximum Principle The linear objective function necessarily attains its maximum at the boundary of the feasible region, unless it is a constant, in which case it has a maximum value at all points.

Proof The definition of convexity of the linear objective function says that for $t \in (0, 1)$

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) \leq \max(f(x), f(y))$$

So when the objective function is evaluated at the convex combination of two points, the value is not greater than the max of the values of the function at the two points

Every interior point of the feasible region can be expressed as convex combinations of other interior points. Moreover, given an interior point, every other interior point can be used in the construction of convex combinations for the given interior point. Thus, no interior point can be a maximizer of the objective function since in that case every interior point would be a maximizer, and the objective function would be a constant. The only points exempt from this are boundary points of the feasible region that cannot be expressed as

convex combinations. In particular, these boundary points must be 'corners' or 'vertices' of the feasible region since if they lie along the 'flat' (half space) sides, they could be expressed as convex combinations of other boundary points. This proves the maximum principle. The minimum principle is defined similarly, and its proof is symmetric.

Minimum Principle The linear objective function necessarily attains its minimum value at the boundary of the feasible region, unless it is a constant, in which case it has a minimum value at all points.

Thus, the linear objective function will always have its optimum value at the boundary, and in particular at the 'vertices', of the feasible region. These vertices are called basic solutions.

As a side note, it is observed that it was crucial to have non – strict inequalities in the definition of the linear program, since using strict inequalities would have restricted access to the boundary of the feasible region. This would have been problematic since, as has been stated above, the optimum of the objective function is always found at the boundary.

1.5 Existence of Solutions

It is not necessary for solutions to the linear program to exist. Linear programs with no solutions are said to be 'infeasible'. For example, inconsistent constraints like $x \geq 2$ and $x \leq 1$ cannot jointly be satisfied.

In addition, while the feasible region is a closed convex polytope, it need not be bounded. If it is unbounded in the direction of the gradient of the objective function (the gradient is the vector or coefficients of the objective function), it may have no optimal value. This is because the gradient vector can be scaled by setting all decision variables equal to a constant. This can be done indefinitely since the region is unbounded, and the function would attain more extreme values.

As an example, consider the aforementioned problem of maximizing the linear objective function:

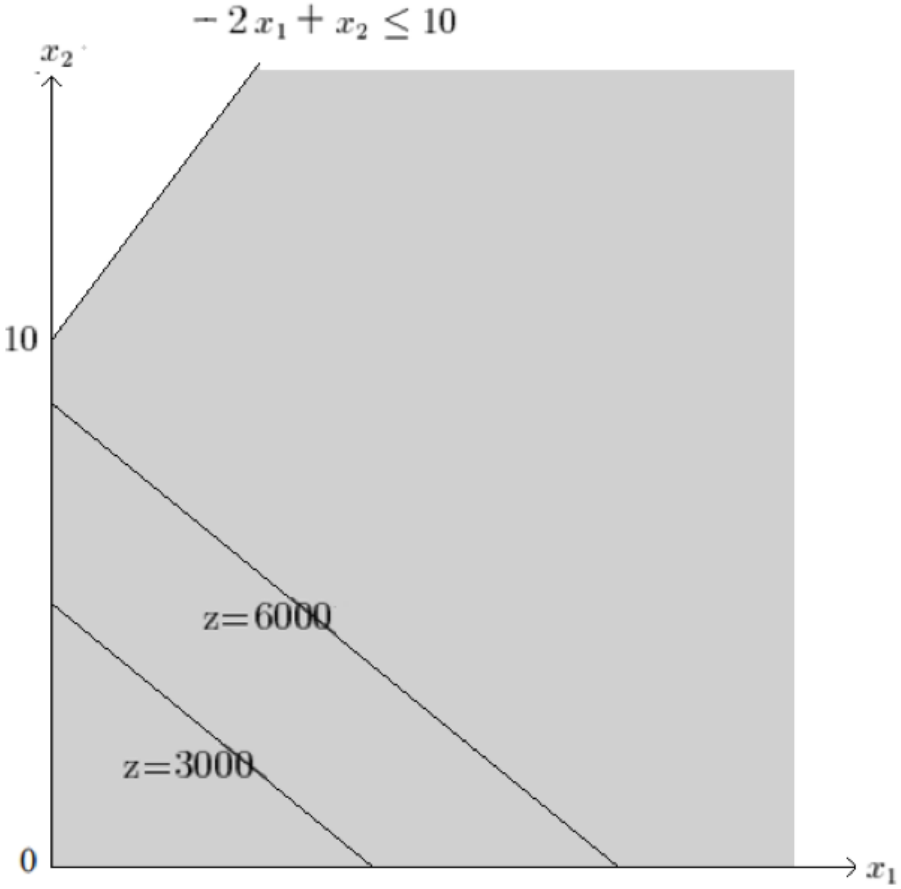
$$z = 750x_1 + 1000x_2$$

Suppose the constraints are now:

$$\begin{gathered} -2x_1 + x_2 \leq 10 \\ x_1, x_2 \geq 0 \end{gathered}$$

Figure 2 shows the feasible region in grey. It is observed that the region is unbounded and extends to infinity in the first quadrant of the Cartesian plane. If the process of drawing 'iso – lines' is once again used, it can be seen that there is no limit to the lines moving rightward, and hence there is no finite point at which the objective function z attains a maximum value.

Figure 2: Unbounded Feasible Region



2 Simplex Method

2.1 Simplex Tableau

In the simplex method, the standard problem is to maximize the linear objective function. If instead the minimization problem is considered, the negative of the objective function can be subjected to the maximization problem, since the point at which the negative of the objective function is maximized is precisely the point at which the objective function itself is minimized.

Prior to the application of the simplex method, it is necessary to transform the constraints of the linear program in the following manner:

- The closed half spaces in $\mathbb{R}^n$ are expressed as linear equations by introducing non-negative slack variables s_i

$$a_{i1}x_1 + ... + a_{in}x_n \leq b_i \rightarrow a_{i1}x_1 + ... + a_{in}x_n + s_i = b_i$$
$$a_{i1}x_1 + ... + a_{in}x_n \geq b_i \rightarrow a_{i1}x_1 + ... + a_{in}x_n - s_i = b_i$$

- If a decision variable x_i is unrestricted, it is replaced in all equations with $x_a - x_b$ with both x_a and x_b being non-negative

As an example, consider the problem of maximizing the linear objective function

$$z = x_1 + x_2$$

The constraints (with the slack variables) are:

$$\begin{gathered} 2x_1 + x_2 + s_1 = 4 \\ x_1 + 2x_2 + s_2 = 3 \\ x_1, x_2, s_1, s_2 \geq 0 \end{gathered}$$

The linear program is now expressed in the form of a simplex tableau, given by:

$$\begin{array}{lrrrrll} z & -x_1 & -x_2 & & & = 0 & \text{Row 0} \\ & +2x_1 & +x_2 & +s_1 & & = 4 & \text{Row 1} \\ & +x_1 & +2x_2 & & +s_2 & = 3 & \text{Row 2} \end{array}$$

In particular, the objective function is expressed as an equation, and each variable is given a column of its own.

2.2 Basic Solutions

There are two kinds of variables in the simplex tableau – basic and non-basic. Basic variables are the variables that feature only once in their respective columns. All other variables are non-basic. In the above example, z, s_1 and s_2 are basic, and x_1 and x_2 are non-basic.

A basic solution is obtained by setting all non-basic variables to zero, and basic variables to the value on the RHS of the equation. In the above example, the basic solution is $z = 0$, $s_1 = 4$, $s_2 = 3$, $x_1 = 0$, and $x_2 = 0$.

The basic solutions of the simplex tableau correspond to the vertices of the feasible region. Hence, intuitively, the simplex method applies elementary row operations to move from basic solution (vertex) to basic solution (vertex), until it arrives at the basic solution (vertex) at which the objective function is maximized.

2.3 Rules of Implementation

The basic solution of the example problem is $z = 0$, $s_1 = 4$, $s_2 = 3$, $x_1 = 0$, and $x_2 = 0$.

It is observed that since x_1 and x_2 have negative coefficients in Row 0, z can be increased by increasing the values of x_1 or x_2, provided that the constraints are not violated. Hence, the first rule of the simplex method is:

Rule 1: If all variables have a nonnegative coefficient in Row 0, the current basic solution is optimal. Otherwise, pick a variable x_i with a negative coefficient in Row 0

The variable x_i selected is called the 'entering variable', and the column to which it belongs is called the 'pivot column'. Suppose x_1 is selected. The next step is to apply a pivot operation to make the non-basic variable x_1 basic. To do this, a 'pivot element' must be selected. This is accomplished using Rule 2 of the simplex method. To provide intuition, suppose $2x_1$ in Row 1 is selected as the pivot element. Applying elementary row operations, the simplex tableau is found to be:

$$\begin{array}{rrrrrrl} z & & -(1/2)x_2 & +(1/2)s_1 & & = 2 & \text{Row 0} \\ & +x_1 & +(1/2)x_2 & +(1/2)s_1 & & = 2 & \text{Row 1} \\ & & +(3/2)x_2 & -(1/2)s_1 & +s_2 & = 1 & \text{Row 2} \end{array}$$

The basic solution is $z = 2$, $x_1 = 2$, $x_2 = 0$, $s_1 = 0$, $s_2 = 1$.

Suppose instead x_1 in Row 2 is selected as the pivot element. Applying elementary row operations, the simplex tableau is found to be:

$$\begin{array}{rrrrrrl} z & & +x_2 & & +s_2 & = 3 & \text{Row 0} \\ & & -3x_2 & +s_1 & -2s_2 & = -2 & \text{Row 1} \\ & x_1 & +2x_2 & & +s_2 & = 3 & \text{Row 2} \end{array}$$

The basic solution is $z = 3$, $x_1 = 3$, $x_2 = 0$, $s_1 = -2$, $s_2 = 0$. The pivot element that must be used is $2x_1$ since using x_1 leads to negative, and hence infeasible values for the other variables. The following theorem states which pivot element must be selected to obtain a feasible basic solution.

Theorem 2.1. The pivot element that guarantees a feasible solution is the one with the smallest positive ratio:

$$\frac{\text{RHS}}{\text{Entering Variable Coefficient}}$$

Proof:
When a pivot element is selected, it is made into a basic variable by applying elementary row operations. These operations are applied in two stages. Firstly, the row containing the pivot element is scaled so that the coefficient of the pivot element is 1. It is observed that scaling does not influence the aforementioned ratio. Secondly, to eliminate the elements in the other rows of the pivot column, a row is selected, the row containing the pivot element is scaled so that the coefficient of the pivot element is the same as the coefficient of the element in the pivot column of the selected row, and the row is then replaced with the difference between the row and the scaled row containing the pivot element. This is repeated for all rows other than the row containing the pivot element. As an example, when $2x_1$ was selected as the pivot element, the row operations applied were:

$$\begin{gathered} \text{Row 1} \to (1/2)\text{Row 1} \\ \text{Row 0} \to \text{Row 0} - (-)\text{Row 1} \\ \text{Row 2} \to \text{Row 2} - \text{Row 1} \end{gathered}$$

When x1 was selected as the pivot element, the row operations applied were:

$$\text{Row } 0 \rightarrow \text{Row } 0 - (-)\text{Row } 2$$
$$\text{Row } 1 \rightarrow \text{Row } 1 - (2)\text{Row } 2$$

Suppose Row i is the row with the smallest positive ratio. It is scaled so that the pivot element has a coefficient of 1, with no change to the ratio.

Consider Row j, with j distinct from i. Now:

$$\frac{\text{RHS}_{Rowi}}{\text{Coefficient}_{Rowi}} \leq \frac{\text{RHS}_{Rowj}}{\text{Coefficient}_{Rowj}}$$
$$\text{Coefficient}_{Rowj} * \frac{\text{RHS}_{Rowi}}{\text{Coefficient}_{Rowi}} \leq \text{RHS}_{Rowj}$$

When the element in the pivot column of Row j is eliminated, the RHS in Row j becomes:

$$\text{RHS}_{Rowj} - \frac{\text{RHS}_{Rowi}}{\text{Coefficient}_{Rowi}} * \text{Coefficient}_{Rowj} \geq 0$$

This ensures that all the RHS values remain non negative after applying the elementary row operations, thereby guaranteeing a feasible solution. This completes the proof. Thus, Rule 2 of the simplex method is:

Rule 2:
For each Row i, where there is a strictly positive 'entering variable coefficient', compute the ratio of the RHS to the 'entering variable coefficient'. Choose the pivot row as being the one with the minimum ratio.

These two rules of the simplex method are applied alternately until we have an optimal solution. The simplex tableau presently is:

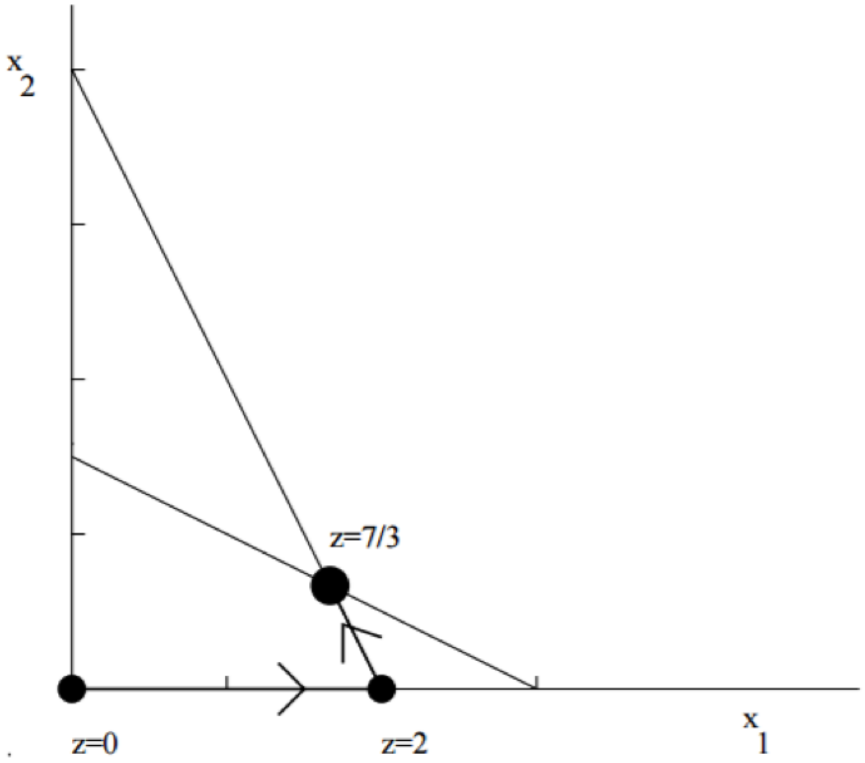
$$\begin{array}{lllll} z & -(1/2)x_2 & +(1/2)s_1 & & = 2 \quad \text{Row } 0 \\ x_1 & +(1/2)x_2 & +(1/2)s_1 & & = 2 \quad \text{Row } 1 \\ & +(3/2)x_2 & -(1/2)s_1 & +s_2 & = 1 \quad \text{Row } 2 \end{array}$$

Selecting the column of x_2 as the pivot column and applying elementary row operations, the simplex tableau is found to be:

$$\begin{array}{llll} z & +(1/3)s_1 & +(1/3)s_2 & = (7/3) \quad \text{Row } 0 \\ x_1 & +(2/3)s_1 & -(1/3)s_2 & = (5/3) \quad \text{Row } 1 \\ x_2 & -(1/3)s_1 & +(2/3)s_2 & = (2/3) \quad \text{Row } 2 \end{array}$$

The basic solution is $z = 7/3$, $x_1 = 5/3$, $x_2 = 2/3$, $s_1 = 0$, $s_2 = 0$. Since there are no negative coefficients in Row 0, this the optimal solution. The following figure shows the graphical interpretation of this method. In the above example, the basic solutions (x_1, x_2), in the order in which they were obtained, are $(0, 0)$, $(2, 0)$, and $(5/3, 2/3)$. This exactly corresponds to a movement along the vertices of the feasible region, as desired.

Figure 3: Graphical Interpretation



3 Excel's Implementation: Karmarkar's Method

3.1 Overview and Intuition

Though powerful, the Simplex Algorithm is not a polynomial-time algortihm. If we start from a particular vertex, then we may end up visiting all the vertices of the feasible region before reaching the solution. This worst-case scenario has exponential running time since the number of vertices could be exponential in number. In 1984, a 28-year-old mathematician named Narendra Karmarkar developed a polynomial-time algorithm to solve LP problems. The Solver tool in Excel implements the Karmarker method.

We begin by exploring the intuition behind Karmarkar's method. A standard LP problem can be geometrically interpreted as a geodesic dome. In two dimensions, the feasible region is a plane; the addition of each new variable adds another dimension to the feasible region. Beyond three dimensions, this is impossible to visualize graphically. Therefore, for an n-dimensional LP problem, a geodesic dome serves as a useful interpretation of the geometry of the problem. Each of the dome's corners serves as a possible solution. With the simplex method, the algorithm lands on one corrner and inspects it; it then visits adjacent corners to find a better answer. If it finds a better answer, then it proceeds in that direction and performs another iteration. The procedure is repeated until the program no longer finds a better solution.

Karmarkar's method employs the same intuition, but follows a radically different tactic. Rather than initialize to a conrner, the algorithm starts from a point within the feasible region. From this interior point, the feasible region is transformed and projected to reconfigure its shape. Suppose that the LP problem exists in the x-space. The transformation T then translates the problem into the y-space, and importantly, the the current point x_k is transformed in such a way that it is in the center of the feasible region in the y-space. This guarantees that the improvement in the optimal solution will be significant. Then, the method determines which direction the solution lies in and makes a step of length l in that direction. The region is then transformed into its original configuration, and the program uses the new point as the interior point for the next iteration.

3.2 Assumptions

Karmarkar's algorithm follows three specific assumptions. The first assumption is standard, namely that the lienar program has a strictly feasible point, and that the set of optimal points is bounded. The second assumption is that the linear program has a special, "canonical" form. Specifically:

$$\begin{aligned} \text{minimize} \quad & z = c^T x \\ \text{subject to} \quad & Ax = 0 \\ & a^T x = 1 \\ & x \geq 0 \end{aligned}$$

Note that this assumption isn't restrictive, since any LP problem can be written in such a form. If we consider a problem in standard, Simplex form:

$$\begin{aligned} \text{minimize} \quad & c'^T x \\ \text{subject to} \quad & A'x' = b \\ & x' \geq 0 \end{aligned}$$

as assume that x' has dimension $n-1$, we can convert this formulation into canonical form by introducing a new variable, $x_n = 1$. Then, we can write the problem as:

$$\begin{aligned} \text{minimize} \quad & z = c'^T x' \\ \text{subject to} \quad & A'x' - bx_n = 0 \\ & x_n = 1 \\ & x' \geq 0 \end{aligned}$$

Here, $A = [A', -b]$, $x = [x', x_n]^T$, and a is the n^{th} unit vector $\mathbf{1}$. The third assumption is that the value of the objective at the optimum is known and equal to 0. This assumption is unlikely to hold, and indeed it is possible to adapt the method to solve problem swhere the optimal objective value is unknown.

3.3 Mathematical Formulation

We now explore the mathematical formulation of Karmarkar's algorithm. During each iteration, the algorithm performs four steps. First, the problem is transformed via a projective transformation. Then, the projected steepest-descent direction is computed. Next, a step is taken in that direction, and finally, the resulting point is mapped back to the original space via the inverse transformation. The point serves as the beginning of the next iteration. The projective transformation T is defined as follows:

$$y = T(x) = \frac{X^{-1}x}{e^T X^{-1}x}$$

where $X = \text{diag}(x_k)$. The transformation maps the current point x_k to $e/n = (1/n, ..., 1/n)^T$. The corresponding inverse transformation T^{-1} is defined as follows:

$$x = T^{-1}(y) = \frac{Xy}{e^T Xy}$$

Recall that the original linear program was in homogenous form. Substituting the inverse transformation into the original program yields a program in the y-space. That is, the new problem P' is defined as follows:

$$\begin{aligned} \text{minimize} \quad & z = \frac{c^T Xy}{e^T Xy} \\ \text{subject to} \quad & AXy = 0 \\ & e^T y = 1 \\ & y \geq 0 \end{aligned}$$

Importantly, P' is not a linear program. However, since we assumed that the value of the objective at the optimum is 0 in the $x - space$, it is correspondiy true in the y-space. Thus, the objective can be rewritten as: minimize $c^T Xy$. The constraints remain the same.

We now compute the steepest-descent direction in the y-space. We want to move from a feasible point y_k to another feasible point y_{k+1} that, for some fixed vector v, will have a larger value of vy. If we choose to move in such a direction $d = (d_1, ..., d_n)$, that solves the optimization problem:

$$\begin{aligned} &\text{maximize } vd \\ &\text{subject to } Ad = 0 \\ &\qquad\qquad d_1 + ... + d_n = 0 \\ &\qquad\qquad \|d\| = 1 \end{aligned}$$

then we will be moving in a feasible direction that maximizes the increase in vy per unit length moved. In the context of the LP formulation in the y space, the direction d that solves this optimization problem is given by the projection of $c^T X$ onto the y-space. To compute this projection, we first denote the constraint matrix as:

$$\text{B} = \begin{bmatrix} AX \\ \mathbf{1}^T \end{bmatrix}$$

The corresponding orthogonal projection matrix is $P_B = I - B^T(BB^T)^{-1}B$. We further define:

$$c_p = -P_B c^T X$$

The projected steepest-descent direction is $\Delta y =$
$ddfracc_p\|c_p\|$. Starting from $\mathbf{1}/n$, we now take a step of length α along the projected steepest-descent direction. Our choice of α is importnat. Any step length less than α_{max}, the step to the boundary, fulfills non-negativity constraints, but this alone does not guarantee polynomial complexity. Note that we divide c_p by its Euclidean norm because this formulation always maintains feasibility. Karmarkar determined that a suitable measure for $\alpha = (1/3)(n(n-1))^{-1/2}$. Therefore:

$$y_{k+1} = y_k + \frac{(n(n-1))^{-1/2}}{3}\Delta y$$

The final step of the iteration is the map y_{k+1} back to the x-space, which yields the new estimate:

$$x_{k+1} = \frac{Xy_{k+1}}{e^T X y_{k+1}}$$

3.4 Example of Karmarkar's Algorithm

We now walk through an example of an iteration of Karmarkar's algorithm to further elucidate the relevant concepts. Therefore, consider the following LP problem:

$$\begin{gathered}\text{minimize } z = 2x_2 - x_3 \\ x_1 - 2x_2 + x_3 = 0 \\ x_1 + x_2 + x_3 = 1 \\ x_1, x_2, x_3 \geq 0\end{gathered}$$

Note that the constraint matrix $A = (1, 2, -1)$. The feasible region is the line segment between Importantly, the point $\mathbf{1}/n = (1/3, 1/3, 1/3)^T$ lies within the feasible region. Now, suppose that the current point Let $x_k = (0.2692, 0.3333, 0.3974)^T$.

The first step is to transform the problem from the x-space to the y-space using the projective transformation. The projective transformation can be conducted in two steps. First, we computed $X^{-1}x$:

$$X^{-1}x = \begin{bmatrix} 3.715 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2.643 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3.715x_1 \\ 3x_2 \\ 2.643x_3 \end{bmatrix}$$

Second, we scale the result by the sum of the variables. That is:

$$y = \frac{1}{3.715x_1 + 3x_2 + 2.643x_3} \begin{bmatrix} 3.715x_1 \\ 3x_2 \\ 2.643x_3 \end{bmatrix}$$

Recall that the projective transformation changes the LP formulation. The new program has the objective function minimize $c^T Xy$ and the constraint $AXy = 0$. Therefore, we now calculate AX. Simple matrix math yields: $AX = (0.2692, 0.6666, 0.3974)$

We now need to compute the steepest-descent direction in the y-space. In order to compute the steepest-descent direction, we first compute the orthogonal projection matrix P_B. Recall that B is denoted as:

$$\begin{bmatrix} AX \\ 1 \end{bmatrix} = \begin{bmatrix} 0.2692 & 0.6666 & 0.3974 \\ 1 & 1 & 1 \end{bmatrix}$$

Therefore, we calculate $P_B = I - B^T(BB^T)^{-1}B$:

$$P_B = \begin{bmatrix} 0.441 & 0.067 & 0.492 \\ 0.067 & 0.992 & -0.059 \\ 0.492 & -0.059 & 0.567 \end{bmatrix}$$

In order to calculate c_p, we now need to calculate $c^T X$. We do so as follows:

$$c^T X = \begin{bmatrix} 0 & 2 & -1 \end{bmatrix} \begin{bmatrix} 0.2692 & 0 & 0 \\ 0 & 0.3333 & 0 \\ 0 & 0 & 0.3974 \end{bmatrix} = \begin{bmatrix} 0 & 0.6666 & -0.3974 \end{bmatrix}$$

Therefore, c_p is computed as:

$$\Delta y = P_B c^T X = \begin{bmatrix} 0.151 \\ -0.018 \\ -0.132 \end{bmatrix}$$

Scaling this by the vector's euclidean norm yields the projected steepest descent direction: $(0.7497, -0.0894, -0.6554)^T$. If we use an α value as specified by Karmarkar:

$$y_{k+1} = y_k + \frac{(n(n-1))^{-1/2}}{3} \Delta y = (0.2653, 0.3414, 0.3928)^T$$

4 MATLAB's Optimization Toolbox: Interior Point Legacy Algorithm

4.1 Linprog

MATLAB's linear programming capabilities are programmed into the command **linprog**. The command allows the user to specify which of three algorithms it wishes to use to solve the LP problem: the interior point legacy algorithm, the interior point algorithm, and the dual simplex method. The default algorithm is the interior point legacy algorithm, which is a variant of primal-dual interior point method. We now explore the mathematical properties of this algorithm.

4.2 Interior Point Legacy Algorithm

4.2.1 Preprocessing

The default interior-point-legacy method is based on LIPSOL, which is a variant a primal-dual interior point method known as Mehrotra's predictor-corrector algorithm, .

The algorithm begins by applying a series of preprocessing steps. These preprocessing steps function to remove redundancies and simplify constraints. Various tasks are performed in this step. For instance, variables with equal upper and lower bounds are removed. Similarly, linear inequality constraints involving just variable are converted into strict bounds. As before, linear inequality constraints are converted into equalities through the addition of slack variables. The MATLAB documentation specifies a more thorough list of the exact pre-processing steps.[1].

If the algorithm detects an infeasible or unbounded problem, then it halts and returns an appropriate error message. Alternatively, if the algorithm arrives at a single feasible point, this point represents the solution.

4.2.2 Primal and Dual Problem

After preprocessing (also referred to as presolve), the problem has the form:

$$\begin{aligned} \text{minimize} \quad & f^T(x) && (1)\\ \text{subject to } & Ax = b && (2)\\ & 0 \leq x \leq u && (3) \end{aligned}$$

With the addition of slack variables s, the inequality constraits can be written as equalities. That is, the problem can be re-written as:

$$\begin{aligned} \text{minimize} \quad & f^T(x) && (4)\\ \text{subject to } & Ax = b && (5)\\ & x + s = u && (6)\\ & x \geq 0, s \geq 0 && (7) \end{aligned}$$

This is referred to as the primal problem, where x consists of the primal variables and s consists of the primal slack variables. The dual problem can also be specified:

$$\begin{aligned} \text{maximize} \quad & b^T y - u^T w && (8)\\ \text{subject to } & A^T y - w + z = f && (9)\\ & z \geq 0, w \geq 0 && (10) \end{aligned}$$

In the dual problem, y and w consist of the dual variables and z consists of the dual slacks. The linear program is therefore defined by both the primal problem and the dual problem. The optimality constraints for the linear program are:

[1]MATLAB Documentation. Linear Programming Algorithms.

$$F(x,y,z,s,w) = \begin{bmatrix} Ax - b \\ x + s - u \\ A^T y - w + z - f \\ x_i z_i \\ s_i w_i \end{bmatrix} = 0$$

where $x_i z_i$ and $s_i w_i$ represent component-wise multiplication.

4.2.3 Primal-Dual Algorithm

The Interior Point Legacy Algorithm is a primal-dual algorithm, which means that the primal and dual programs are solved simultaneously. Consider $v = [x, y, z, s, w]^T$, where $[x, z, s, w] > 0$. The algorithm first computes the prediction and correction direction:

$$\Delta v_p = -(F^T(v))^{-1} F(v)$$
$$\Delta v_c = -(F^T(v))^{-1} F(v + \Delta v_p) - \mu \hat{e}$$

where μ is the centering parameter and $\hat{e}$ is binary vector with the ones corresponding to the quadratic equations in $F(v)$. The algorithm then takes a step of length α in the combined direction:

$$v^+ = v + \alpha(\Delta v_p + \Delta v_c)$$

The step-length parameter α is chosen so that $v^+ = [x^+, y^+, z^+, s^+, w^+]^T$ satisfies $[x^+, z^+, s^+, w^+] > 0$. The algorithm then loops until the iterates converge. The algorithm employs a standard stopping criterion:

$$\frac{\|r_b\|}{\max(1, \|b\|)} + \frac{\|r_f\|}{\max(1, \|f\|)} + \frac{\|r_u\|}{\max(1, \|u\|)} + \frac{|f^T x = b^T y + u^T w|}{\max(1, |f^T x|, |b^T y - u^T w|)} \leq tol$$

where *tol* is some tolerance, $r_b = Ax - b$, $r_f = A^T y - w + z - f$ and $r_u = x + s - u$. These constitute the primal residual, dual residual, and upper-bound feasibility, respectively.

5 Example Implementation in MATLAB

The syntax for MATLAB's linprog function is as follows:

```
x = linprog(f, A, b)
x = linprog(f, A, b, Aeq, beq)
x = linprog(f, A, b, Aeq, beq, lb, ub)
x = linprog(f, A, b, Aeq, beq, lb, ub, options)
x = linprog(problem)
[x, fval] = linprog(____)
[x, fval, exitflag, output] = linprog(____)
[x, fval, exitflag, output, lambda] = linprog(____)
```

Note that linprog assumes that we are minimizing the objective function. In order to maximize the objective function, we need to take the negative. The following is the implementation of the first example presented in the paper:

```
A = [1, 1; 1, 2; 4, 3];
f = [-750; -1000];
b = [10; 15; 25];
x = linprog(f, A, b);
Optimization terminated

x =
1.0000
7.0000
z = 750*x(1) + 1000*x(2)
z =
7.7500e+03
```

If we try to solve the problem from section 1.5 where there was no solution, Matlab throws an error:

```
f = [-750; -1000];
A = [-2, 1];
b = [3];
linprog(f, A, b)
Exiting: One or more of the residuals, duality grap, or total relative error
has grown 100000 times greater than its minimum value so far: the dual appears
to be unfeasible (and the primal unbounded).
The primal residual < OptimalityTolerance=1.00e-08.

ans =
1.0e+16 *

2.0897
4.1795
```

If we were to solve the problem outlined in section 2.1, Matlab produces the following results:

```
f = [-1; -1];
A = [2, 1; 1, 2];
b = [4; 3];
x = linprog(f, A, b)
Optimization terminated

x =
1.6667
0.6667

z = x(1) + x(2)
z =
2.3333
```

6 Concluding Remarks

Linear programming represents a vast field of research, and several clever algorithms already exist to optimize objective functions of different forms. In addition to exploring the canonical Simplex Algorithm, we also explored two interior point methods, namely Karmarkar's algorithm and the Interior Point Legacy method implemented by MATLAB. There remain several open problems in the theory of linear programming. For instance, does LP admit a strongly polynomial-time algorithm? Or, are there pivot rules which lead to polynomial time Simplex variants? Though difficult, solutions to these questions would represent fundamental breakthroughs in mathematics. More importantly, such answers could potentially advance our ability to solve large-scale linear problems.

References

[1] Igor Griva, Stephen G. Nash, Ariela Sofer (2009). *Linear and Nonlinear Optimization.* Society for Industrial Mathematics

[2] AK Dhamija (2009). *Karmarkar's Algorithm: An Interior Point Method of Linear Programming Problem.* Defence Research and Development Organization.

[3] Gerard Sierksma (2001). *Linear and Integer Programming: Theory and Practice, Second Edition..* CRC Press.

[4] Alexander Schrijver (1998). *Theory of Linear and Integer Programming.* John Wiley Sons.

[5] *Reminiscences about the origins of linear programming.* Operations Research Letter.

[6] *Linear Programming.* Wikipedia.

[7] *Linear Programming Algorithms.* MathWorks MATLAB Documentation.