CSCI 5454 CSL: Linear Programming and the Simplex Algorithm

Bryan Abbe

March 31, 2011

1 Origin

In 1947, George Dantzig recognized a common pattern in a wide variety of military programming and planning problems: one must optimize a linear form given some linear constraints. Linear forms are finite summations

$$\sum_{j=1}^{n} c_j x_j,$$

for constants c_j and variables x_j, which are usually both real-valued and in special cases one or the other could be integers. Linear constraints are linear forms which are less than, greater than, or equal to a given constant; these are of the form

$$\sum_{j=1}^{n} d_j x_j \leq b_j$$

some constants b_j, and where $\leq$ could be replaced by $\geq$ or $=$. For example, $\frac{9}{4}x_1 + 0 \cdot x_2 - 5x_3 \leq 10$ is a linear constraint. The linearity emerged because of summing quantities which were directly proportional to products of rates and quantities; the linear forms were

$$\sum_{\text{all items}} (\text{value per item}) \cdot (\text{quantity of item}),$$

similar to expected value. The constants on the right hand side of the linear constraints represented greatest or least amount needed, the latter occuring when $\leq$ is replaced by $\geq$. The recognition of these patterns led him (with his associates) to the study of linear programming, the study of such problems.

Because summations of products of rates with quantities occurs often in many fields, this relatively new idea had was eagerly applied to problems in agriculture, manufacturing, transportation, economic theory, engineering, general finance, time management, and many other areas.

Linear programming was also developed independently and without Dantzig's knowledge around 1939 by Russian mathematician and economist Leonid Kantorovich. However, Dantzig's development included what is now known as the simplex algorithm, which allowed problems in the areas above to be solved quickly with computers. It is so named because a region which results from the linear constraints mentioned above is a simplex, the convex hull of n vertices – which is the smallest convex set in some Euclidean space $\mathbb{R}^m$ containing the n vertices.

2 Motivating Example [2]

This is the long type of motivating example, because I wish to bring up several different issues at once.

2.1 Posing the Problem

Broad Motor Company (BMC) decides to sponsor a half hour TV show featuring a comedian and a band to promote its new product, the Bulkswagon.

- BMC insists on at least 3 minutes of commercials during the show.
- The TV network requires time allotted to commercials not to exceed 12 minutes and under no circumstances should this time exceed the time allotted for the comedian
- Comedian does not want to work more than 20 minutes of the half hour show, so the band is used for any remaining time.

BMC will be charged as follows:

- \$750 per minute that the comedian is on the air,
- \$600 per minute that the band is on the air, and
- \$450 per minute of commercials.

Finally, the TV network's experience on the air indicates that

- 12000 additional viewers tune in for every minute the comedian is on the air,
- 6000 additional viewers tune in for every minute the band is on the air, and
- 3000 viewers tune out for every minute of commercials.

The goal is to maximize the number of viewers at minimum cost to BMC.

2.2 The Solution

Let x be the number of minutes allotted to the comedian and y be the number of minutes allotted to the band. Then $30 - x - y$ is the number of minutes allotted to the band. Note the following linear constraints from the first set of demands above by the sponsor, network, and comedian:

$$\begin{gathered} x \geq 0 \; y \geq 3 \\ y \geq 0 \; y \leq 12 \\ 30 - x - y \geq 0 \; y \leq x \\ x \leq 20 \end{gathered}$$

By graphing the following inequalities and taking their intersection, one obtains the convex region, the simplex for the linear program. It is displayed as the first figure on the next page, with the interior in gray. Let n be the number of viewers and c be the cost to the sponsor. By the second and third sets of data, n and c are the linear forms given by

$$n = 12000x + 6000(30 - x - y) - 3000y = 3000(\underbrace{2x - 3y}_{M} + 60)$$

and

$$c = 750x + 600(30 - x - y) + 450y = 150(\underbrace{x - y}_{m} + 120).$$

The intersections of the convex region above with the lines given by $x - y = m$ and $2x - 3y = M$ using various values of m and M are in the second figure on the next page.

It is apparent that $m = 0$ is the least value possible for m; anything less would be outside the region. It is also apparent that $M = 31 = 2 \cdot 20 - 3 \cdot 3$ is the greatest value possible for M; anything greater would give a line which does not intersect the region above.

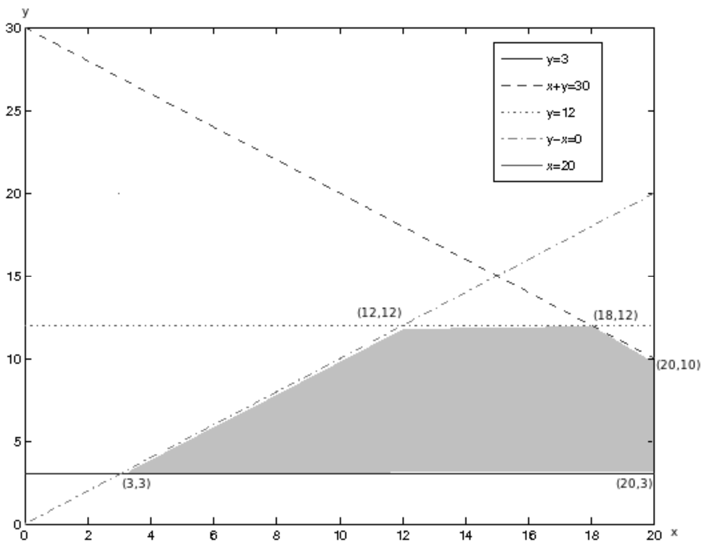

Figure 1: Intersections of half-planes given by linear constraints for example

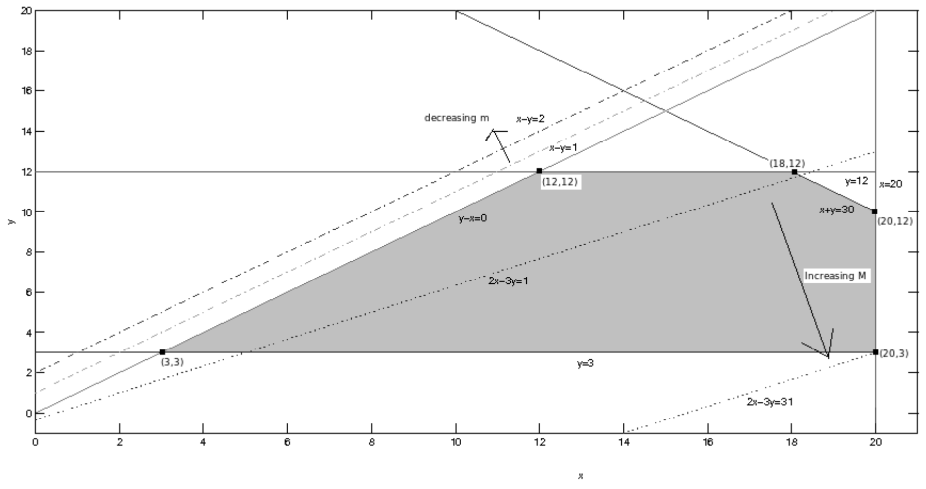

Figure 2: Intersections of convex region for example with lines giving values of m and M

One gets the maximum number of viewers when $x = 20$, $y = 3$, and this value is unique since it intersects at exactly one point;

$$n = 3000(31 + 60) = 3000(91) = 273,000 \text{ viewers}$$

and

$$c = \$150(20 - 3 + 120) = \$150(137) = \$20,550.$$

The solution if one wishes to prioritize greatest number of viewers is then to allot 20 minutes to the comedian, 3 minutes to commercials, and 7 minutes to the band, which seems more or less a common sense approach. Moreover, the extra cost might be offset by the expected number of Bulkswagon purchases resulting from the extra viewers.

On the other hand, one obtains the last cost of viewers whenever $x = y$ for $x \in [3, 12]$. The cost whenever $x = y$ on this closed interval is

$$c = \$150(0 + 120) = \$150(120) = \$18,000.$$

Since n is a linear function, its minima and maxima along the given line can be found by plugging in the endpoints of the closed interval: when $x = 3$,

$$n = 3000(2 \cdot 3 - 3 \cdot 3 + 60) = 3000(57) = 171,000 \text{ viewers}$$

and when $x = 12$,

$$n = 3000(2 \cdot 12 - 3 \cdot 12 + 60) = 3000(48) = 144,000 \text{ viewers}$$

The superior solution, if one wishes to prioritize least cost, is surprisingly *not* to fill the show up on commercials, but to allot 3 minutes to the comedian, 3 minutes to commercials, and 24 minutes to the band.

3 General Descriptions of Linear Programs

3.1 Linear Programs in Standard Form

In order to present the simplex algorithm, there must be agreement on a standard input representing a linear program (or linear programming problem; the two terms I will use synonymously), which will be found by giving the linear program in **standard form**, as follows:

$$\begin{aligned} \text{Maximize} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \le b_i, i \in \{1, \dots, m\}, \\ & x_j \ge 0, j \in \{1, \dots, n\}. \end{aligned}$$

The linear constraints $x_j \ge 0$ are also called the **nonnegativity constraints**.

This linear program can be written in a more compact form by letting $c = (c_1, \dots, c_n)$, $(x_1, \dots, x_n)$, $A_{m \times n} = (a_{ij})$, $b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$ so that the linear program in standard form is now

$$\begin{aligned} \text{Maximize} \quad & c^T x \\ \text{subject to} \quad & Ax \le b, \\ & x \ge 0, \end{aligned}$$

where $\le$ stands for the the m linear constraints that would result from multiplication of Ax and $x \ge 0$ means $x_j \ge 0$ for all j.

3.2 Converting a linear program to standard form

Before proceeding further with the general method, it should be verified that any linear program can be converted to standard form. A proof by example will be sufficient, provided that it illuminates the method for doing so. Consider the following linear program

$$\begin{aligned} \text{Minimize} \quad & -2x_1 + 3x_2 \\ \text{subject to} \quad & x_1 + x_2 = 7, \\ & x_1 - 2x_2 \le 4, \\ & x_1 \ge 0. \end{aligned}$$

To turn the linear program from a minimization problem into a maximization problem, first note that the feasible region does *not* change, for an objective function will need to be maximized subject to the same conditions as the original objective function which was to be minimized. That said, by maximizing the negation of the original objective function, one obtains the least objective value subject to the original conditions, so that the original objective value is minimized. That is, the original linear program is equivalent to (i.e., has the same feasible solutions and optimal solutions as)

$$\begin{aligned} \text{Maximize} \quad & 2x_1 - 3x_2 \\ \text{subject to} \quad & x_1 + x_2 = 7, \\ & x_1 - 2x_2 \leq 4, \\ & x_1 \geq 0. \end{aligned}$$

Note that if $a = b$, $a \leq b$ and $a \geq b$ for all real numbers a and b, so the above linear program is equivalent to

$$\begin{aligned} \text{Maximize} \quad & 2x_1 - 3x_2 \\ \text{subject to} \quad & x_1 + x_2 \leq 7, \\ & x_1 + x_2 \geq 7, \\ & x_1 - 2x_2 \leq 4, \\ & x_1 \geq 0. \end{aligned}$$

Moreover, if $a \geq b$, then $-a \leq -b$ for all real numbers a and b, so this third linear program is equivalent to

$$\begin{aligned} \text{Maximize} \quad & 2x_1 - 3x_2 \\ \text{subject to} \quad & x_1 + x_2 \leq 7, \\ & -x_1 - x_2 \leq -7, \\ & x_1 - 2x_2 \leq 4, \\ & x_1 \geq 0. \end{aligned}$$

Note that there is no condition for $x_2 \geq 0$. This can be remedied by setting $x_2 = x_2' - x_2''$, where without loss of generality, $x_2', x_2'' \geq 0$. This gives the equivalent linear program

$$\begin{aligned} \text{Maximize} \quad & 2x_1 - 3x_2' + 3x_2'' \\ \text{subject to} \quad & x_1 + x_2' - x_2'' \leq 7, \\ & -x_1 - x_2' + x_2'' \leq -7, \\ & x_1 - 2x_2' + 2x_2'' \leq 4, \\ & x_1 \geq 0, \\ & x_2' \geq 0, \\ & x_2'' \geq 0, \end{aligned}$$

which is in standard form. For convenience later, reassign $x_2 := x_2'$ and $x_3 := x_2''$, distribute and add like terms to get linear program (1),

$$\begin{aligned} \text{Maximize} \quad & 2x_1 - 3x_2 + 3x_3 \\ \text{subject to} \quad & x_1 + x_2 - x_3 \leq 7, \\ & -x_1 - x_2 + x_3 \leq -7, \\ & x_1 - 2x_2 + 2x_3 \leq 4, \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0. \end{aligned}$$

3.3 Feasible Regions

In the example from section 2, a convex region is obtained from the intersection of 7 half-planes. (A region is **convex** if for any two points P and Q in the region, the line segment connecting P and Q

is also in the region.) In a general linear program, a convex region is obtained from the intersection of m half-hyperplanes. This convex region is called the **feasible region** for the linear program, and a solution x which is is in this feasible region; i.e., a solution which satisfies the linear constraints to the linear program, is called a **feasible solution**. The linear form $c^T x$, sometimes denoted by z, is called the **objective function**, and for a particular assignment $(x_1, \ldots, x_n) := (d_1, \ldots, d_n)$, $c^T d$, sometimes denoted by v, is called the **objective value** when $x = d$.

From the example above, it is possible that there are infinitely many optimal feasible solutions to a linear programming problem and that the origin is not part of the feasible region. The optimum feasible values were attained on edges or vertices of the feasible region in the example above, and this will hold in general. For otherwise an optimum objective value would found at a point in the interior of the feasible region. When the hyperplane given by $z = c^T x$ for various values of z is shifted in the direction of increasing z (assuming the linear program is in standard form), a larger objective value would be found in the interior of the feasible region, since the region is convex. This gives a contradiction; therefore an optimum objective value will not be obtained in the interior of the feasible region.

Accordingly, a general simplex algorithm must search for optimum values along the boundary of the feasible region. An exhaustive search could potentially never terminate; instead, the algorithm will convert its standard form input into a **slack form**, which gives a system of equations in x, $Ax = b$, $z = c^T x$, and apply a method very similar to Gaussian elimination on them, based on the decision of a pivot. Recall that Gaussian elimination reduces a system of equations, in an augmented matrix representing a matrix equation, to **row-reduced echelon form**; that is, **basic variables** can be solved in terms of the **free variables** in the system of equations. In some cases there will be no free variables, and in other cases, there will be no solution, so that the system of equations is inconsistent. This same idea applies to a linear program in slack form. If the system of equations is inconsistent, the linear program is said to be **infeasible**.

3.4 Converting a Linear Program in Standard Form to Slack Form

Given a linear program $L = (A, b, c)$ (where A is $m \times n$) with unknowns $x = (x_1, \ldots, x_n)$, introduce new **slack variables** $x_{n+1}, \ldots, x_{n+m}$, defined by

$$x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j, i \in \{1, \ldots, m\}.$$

Since $Ax \leq b$, $x_{n+1}, \ldots, x_{n+m} \geq 0$. The slack variable x_{n+i} keep a record of how much of an item represented by $\sum_{j=1}^{n} a_{ij} x_j$ is being wasted. The simplex algorithm will use this idea, usually reducing at least one of the slack variables to 0 to find an optimal objective value for L. For example, given $m = 3$ equations with $n = 3$ variables, the linear program (1) above in slack form (2) is

$$\begin{aligned}
\text{Maximize} \quad & 2x_1 - 3x_2 + 3x_3 \\
\text{subject to} \quad & x_4 = 7 - x_1 - x_2 + x_3, \\
& x_5 = -7 + x_1 + x_2 - x_3, \\
& x_6 = 4 - x_1 + 2x_2 - 2x_3, \\
x_1 \geq 0, & x_4 \geq 0, \\
x_2 \geq 0, & x_5 \geq 0, \\
x_3 \geq 0, & x_6 \geq 0.
\end{aligned}$$

4 The Simplex Method

In a linear program we refer to basic variables and nonbasic variables (corresponding to the free variables). The basic variables are those on the left hand side of the equations, and the nonbasic variables are those on the right hand side of the equations; they are also those variables present in the objective function.

4.1 Basic Description of the Simplex Method

Given a linear program (A, b, c) is standard form, the simplex method first determines whether the program is feasible. If it is, the program is then converted to an equivalent slack form. If it is not, the method execution is complete.

The simplex method operates iteratively on equivalent linear programs in slack form. At each iteration, the simplex method picks a nonbasic variable x_e (called the **pivot** or **entering variable**) which has a positive coefficient in the current objective function. Since all of the basic variables must be nonnegative, the method then determines the tightest upper bound on x_e from the m equations, picks the equation which gives the tightest upper bound, and solves this equation for x_e. If the tightest upper bound is finite, it then substitutes the right hand side of this new equation everywhere x_e appears in the linear program, which gives an equivalent linear program in slack form. By setting each of the nonbasic variables equal to 0, one obtains a **basic solution**, where the basic variables values and the current intermediate objective value are uniquely determined. The intermediate basic solutions may not always be feasible, but the final one will be. (This is not obvious.) If there is no finite upper bound, the linear program will have an unbounded solution, so no explicit answer can be found, and at that point the method execution is complete.

Once the linear program is converted to one in which the all of the coefficients of the nonbasic variables in the objective function is negative, the method execution is complete.

4.2 Running the Simplex Method on Linear Program (2)

The simplex algorithmInitially, linear program (2) has basic solution $(0, 0, 0, 7, -7, 4)$ with objective value 0. (Note that it is not immediately clear that the program is feasible.) Since x_1 has positive coefficient 2 in the objective function, find the tightest upper bound for x_1 in this linear program: $x_1 \leq \frac{7}{1} = 7$ from the first equation, $x_1 \geq 7$ can be unbounded in the second equation, and $x_1 \leq \frac{4}{1} = 4$ from the third equation. Solve for x_1 in the third equation:

$$x_1 = 4 + 2x_2x_3 - x_6,$$

and substitute the right hand side in for x_1 everywhere it appears, then simplify. The entering variable x_1 is *entering* the basic variable set, and the variable x_6 is *leaving* the basic variable set, so x_6 is the **leaving variable**. One obtains the new linear program

$$\begin{aligned}
\text{Maximize} \quad & 8 + x_2 - x_3 - 2x_6 \\
\text{subject to} \quad & x_1 = 4 + 2x_2 - 2x_3 - x_6, \\
& x_4 = 3 - 3x_2 + 3x_3 + x_6, \\
& x_5 = -3 + 3x_2 - 3x_3 - x_6, \\
x_1 \geq 0,& x_4 \geq 0, \\
x_2 \geq 0,& x_5 \geq 0, \\
x_3 \geq 0,& x_6 \geq 0.
\end{aligned}$$

The basic solution is now $(4, 0, 0, 3, -3, 0)$ with objective value 8. Since x_2 has positive coefficient 1 in the objective function, and is the only such variable with a positive coefficient, it is the entering or pivot variable. Find the tightest upper bound for x_2 in this linear program: $x_2 \geq -2$ can be unbounded unbounded in the first equation, equation, $x_2 \leq \frac{3}{3} = 1$ from the second equation, and $x_2 \geq 1$ is unbounded in the third equation. Solve for x_2 in the second equation, making x_4 the leaving variable:

$$x_2 = 1 + x_3 - \frac{1}{3}x_4 + \frac{1}{3}x_6,$$

and substitute the right hand side in for x_2 everywhere it appears, then simplify. One obtains the new linear program

$$\begin{aligned}
\text{Maximize} \quad & 9 - \frac{1}{3}x_4 - \frac{5}{3}x_6 \\
\text{subject to} \quad & x_1 = 6 - \frac{2}{3}x_4 - \frac{1}{3}x_6, \\
& x_2 = 1 + x_3 - \frac{1}{3}x_4 + \frac{1}{3}x_6, \\
& x_5 = -x_4, \\
& x_1 \geq 0, x_4 \geq 0, \\
& x_2 \geq 0, x_5 \geq 0, \\
& x_3 \geq 0, x_6 \geq 0.
\end{aligned}$$

In this case, x_3 is a nonbasic variable even though it does not appear in the objective function or the equation for x_5. The basic solution is now $(6, 1, 0, 0, 0, 0)$ with objective value 9. Since there are no longer any positive coefficients in front of the nonbasic variables in the objective function, the iteration of converting to equivalent linear programs ends, and $\bar{x}$ is assigned to be $(6, 1, 0, 0, 0, 0)$. Note that if $\bar{x}$ is plugged into the original objective function, $9 = 2 \cdot 6 - 3 \cdot 1 + 3 \cdot 0$ is still obtained. The simplex method execution is now complete.

5 Formation of the Simplex Algorithm

The previous section outlined what needs to happen when implementing the simplex algorithm, which can now be presented. The comments link the simplex method described in the previous section to the simplex algorithm.

```
Simplex(A,b,c)
{
        // Determine whether the given linear program is feasible;
        // convert a feasible one into slack form
        (N,B,A,b,c,v):=Initialize-Simplex(A,b,c)

        // Loop while the objective function has (nonbasic) variables
        // with at least one coefficient c[1],...,c[n] which is
        // positive
        while some index j in N has c[j]>0 {

                // Find e to determine the variable x[e] which will
                // enter the set of basic variables in the next
                // equivalent linear program in slack form
                choose index e in N with c[e]>0

                for each index i in B {

                        // Get the tightest upper bound on the
                        // nonbasic variable x[e] and or set the
                        // upper bound to INFINITY if there isn't a
                        // finite upper bound
                        Delta[i]:=Upper-Bound(N,B,A,b,c,e)
                }

                // Find l to determine the variable x[l] which will
                // leave the set of basic variables in the next
                // equivalent linear program in slack form
                choose index l in B minimizing {Delta[i]:i=1,...,n}

31                  if (Delta[l]=INFINITY)
32                          return "unbounded"
33                  else {
34                          // With x[e] as the new pivot, convert the
35                          // current linear program L in slack to an
36                          // equivalent one by solving for x[e] in
37                          // equation l and replacing x[e] in all
38                          // other instances by the right hand side
39                          // of the obtained expression everywhere it
40                          // appears in L
41                          (N,B,A,b,c,v):=Pivot(N,B,A,b,c,v,l,e)
42                  }
43          }
44
45          // Assign basic solution to xbar, setting all nonbasic variables
46          // original to the linear programming problem equal to zero.  The
47          // new values b[1,...m] obtained from the function Pivot are the
48          // values for the basic variables original to the linear
49          // programming problem when the nonbasic variables are all set to
50          // zero.
51          for i=1 to n {
52                  if i is in B
53                          xbar[i]:=b[i]
54                  else
55                          xbar[i]:=0
56          }
57
58          return xbar[1,...,n]
59  }
```

Assuming that `Initialize-Simplex`, `Upper-Bound`, and `Pivot` do as directed, it is clear that `Simplex` returns a basic solution to the input linear program (A, b, c). It is not clear at the moment whether or not `Simplex` terminates, returns a feasible basic solution, and that this feasible solution is optimal. Verification, or at the very least a convincing argument giving references, of these claims will constitute the remainder of these notes.

6 Termination of `Simplex`

`Simplex` can fail to terminate if it is never the case that $c_j < 0$ for all j on a particular slack form, or the choice for `e` is chosen arbitrarily. The former can only arise if `Simplex` fails to strictly increase the objective value after an iteration and will be addressed later. The latter will be addressed presently.

6.1 Possibility of Cycling in `Simplex`

It is possible to give a linear program which is feasible and yet does not terminate under `Simplex` if the choice for `e` in line 15 is arbitrary. Admittedly, I gave an incorrect example in class; the example I gave illustrated that the objective value sometimes does not increase when the linear program is changed. In fact, this situation is rare: only three of a large number of linear programming problems considered by researchers have been known to cycle [3], and were artificially constructed. For examples, see [4] and [5]. The smallest counterexample known has 11 variables, which is not unrealistic since practical linear programming problems can have millions of variables. These papers are presented using the tableau representation of a linear program, which was not covered in this lecture.

To overcome this type of situation, one must choose deliberately the index `e`. Among other methods, one chooses the variable with the smallest index each time. This is known as Bland's Rule, named after Robert Bland, who first developed it can be used. A proof that the simplex method terminates using Bland's Rule when there is a choice for `e` is given in [6].

6.2 Possibility that `Simplex` never increases the objective value

First it is shown that `Simplex` never *decreases* the objective value. Assuming that `Pivot` operates as described in the comments of the pseudocode, given entering variable x_e, one finds b_e from equation with x_l on the left hand side, where l represents the leaving variable:

$$x_l = b_l - \sum_{j \in N} a_{lj} x_j.$$

Note $e \in N$ currently, and e will no longer be in N once `Pivot` returns. Thus, the new value $\hat{b}_e$ in the new linear program that will result is given by $\hat{b}_e = \frac{b_l}{a_{le}}$. Once x_e is solved in terms of the other variables in the equation above, and the right hand side is substituted into the objective function for x_e, the objective value increases by $c_e \hat{b}_e$. Recall again that x_e is currently a nonbasic variable, so would be on the right hand side of the objective function for the current linear program. But it is already known that $c_e > 0$ since that is how e was chosen, and $b_l \geq 0$ since x_e has a finite upper bound. If `Upper-Bound` works as prescribed, then the upper bound is finite only if $a_{le} > 0$; otherwise x_e can be taken arbitrarily large. Thus $\hat{b}_e \geq 0$, so the objective value cannot decrease.

Note that since there are $|N| + |B| = n + m$ variables, and m variables must be chosen at a time to get a slack form, there are at most $\binom{n+m}{m}$ unique slack forms. Now, if at some point in the iterations `Pivot` always generates a linear program in slack form that never increases the objective value, then the program cannot terminate. Yet, as the next proposition will show, `Pivot`, if it works as prescribed, returns a unique slack form.

Proposition 1. *If (A, b, c) is a linear program in standard form, then given a set B of basic variables, a slack form (N, B, A, b, c) is uniquely determined.*

Proof. To prove this, the following lemma is needed, which will be useful later as well.

Lemma 1. *Let I be a set of indices. For each $i \in I$, let α_i, β_i be real and let x_i be a real-valued variable. Fix γ real. Suppose that for any settings of $(x_i)_{i \in I}$,*

$$\sum_{i \in I} \alpha_i x_i = \gamma + \sum_{i \in I} \beta_i x_i.$$

Then $\alpha_i = \beta_i$ for each $i \in I$ and $\gamma = 0$.

Proof. Set $x_i = 0$ for all $i \in I$. Then $\gamma = 0$. Now for arbitrarily chosen $j \in I$, consider the settings

$$x_i = \begin{cases} 1, & i = j \\ 0, & \text{otherwise.} \end{cases}$$

In this case, $\alpha_j = \beta_j$. Since $j \in I$ was arbitrary and $\gamma = 0$, $\alpha_i = \beta_i$ for all $i \in I$. □

The proposition can now be proved. Suppose towards a contradiction that there are two different slack forms. Since B is given, $N = \{1, \ldots, n+m\} - B$ is uniquely determined, so the different slack forms are (N, B, A, b, c) and (N, B, A', b', c'). Write $v = \sum_{j \in B} c_j x_j$ and $v' = \sum_{j \in B} c'_j x_j$. Then one obtains slack forms, omitting the terms "Maximize,", "subject to," and all the nonnegativity constraints,

$$z = v + \sum_{j \in N} c_j x_j$$
$$x_i = b_i - \sum_{j \in N} a_{ij} x_j, i \in B$$

and

$$z = v' + \sum_{j \in N} c_j x_j$$
$$x_i = b'_i - \sum_{j \in N} a'_{ij} x_j, i \in B.$$

(Here, z just denotes a variable for the objective function.) Subtracting the equations for x_i, $i \in B$, one obtains, for each $i \in B$,

$$0 = (b_i - b_i') + \sum_{j \in N} (a_{ij} - a_{ij}')x_j,$$

or

$$\sum_{j \in N} a_{ij}x_j = (b_i - b_i') + \sum_{j \in N} a_{ij}'x_j.$$

Since the sequence of variables $(x_j)_{j \in N}$ is not specified, one can use Lemma 1 to conclude that $b_i - b_i' = 0$ and $a_{ij} = a_{ij}'$ for all $i \in B$. Therefore $b = b'$ and $A = A'$. In a similar manner, one can subtract the equation for the second objective function from the first, and end up with $c = c'$ and $v = v'$. This shows that the two forms are the same, contrary to the assumption above. $\square$

It follows that if `Simplex` fails to terminate for never increasing the objective function, then it must iterate through more than $\binom{n+m}{m}$ unique iterations. This means that `Simplex` will cycle, which reduces to the case considered in the previous subsection.

6.3 Running Time for `Simplex`

Since it is now apparent that `Simplex` will terminate, provided Bland's Rule is used in line 15, it makes sense to give the asymptotic running time. As noted above, there are at most $\binom{n+m}{m}$ iterations in the outermost loop, and the most time consuming part of this outerloop is calling `Pivot`, since all other operations take either $O(n)$ or $O(m)$ time. Assuming `Pivot` works as prescribed with entering index e and leaving index l, it must specify new values for b, c, and A since it must solve for x_e and then substitute for x_e wherever it appears outside the previous equation for x_l. This takes order $O(mn)$ time, so that the total running time, not including `Initialize-Simplex`, is $O\left(\binom{n+m}{m}mn\right) = O\left(\frac{n(n+m)^m}{(m-1)!}\right)$ (slight mistake in lecture; I overlooked the update of A and just looked at updating B and N). `Initialize-Simplex` actually takes the same time as the outer loop, since it will end up using this loop to create an auxiliary linear program with objective function $-x_0$ and checking whether the objective value is 0 to determine the feasibility of the program. (The details are more spelled out in [1]. They are not hard, but it was too much material to cover for this lecture.) Thus the asymptotic running time for `Simplex` is $O\left(\frac{n(n+m)^m}{(m-1)!}\right)$, which is not as bad as exponential time, so it is faster than an exhaustive search. The algorithm is useful for solving a problem which can be posed as a linear programming algorithm for which there is not a known efficient algorithm.

7 `Simplex` and the Return of a Feasible Basic Solution

This section shows that `Simplex` returns a feasible basic solution to the linear program. Note that a linear program (N, B, A, b, c) in slack form is feasible (i.e., will give a basic feasible solution) if and only if $b_i \geq 0$ for all $i \in \{1, \dots, m\}$. Certainly this condition gives sufficiency. If $b_k < 0$ for some $k \in \{1, \dots, m\}$ then the basic solution for this slack form has a negative entry for $\bar{x}_k$. But $\bar{x}_k \geq 0$ is required by the problem which gives the contradiction.

As a consequence of this, it suffices to focus on how `Pivot` updates b to get $\hat{b}$. As shown in the previous section, $\hat{b} \geq 0$. Note that one does not worry about $\bar{b}_l \in B$ because l will be removed from be at the termination of `Pivot`. For $i \in B - \{l, e\}$, and consider the equation

$$x_i = b_i - \sum_{j \in N} a_{ij}x_j.$$

Since $e \in N$ currently, and x_e is replaced by its new value on the right hand side of the above equation, one obtains $\hat{b}_i$ as $b_i - a_{ie}\hat{b}_e$. In the case that $a_{ie} \leq 0$, then because $b_i \geq 0$ (since the solution is feasible from the return of `Initialize-Simplex`) and $\hat{b}_e \geq 0$, so is $\hat{b}_i$. In the case that $a_{ie} > 0$, note that l was chosen in `Simplex` so that the upper bound for x_e was minimal in the equation for x_l. This means that $\frac{b_l}{a_{ie}} \leq \frac{b_i}{a_{ie}}$ for all $i \in B$, and so

$$\hat{b}_i = b_i - a_{ie}\frac{b_l}{a_{ie}} \geq b_i - a_{ie}\frac{b_i}{a_{ie}} = 0.$$

This gives the result.

8 Correctness of `Simplex`

To prove that `Simplex` gives an optimal solution, duality of the linear program is used. Given a linear program $L = (A, b, c)$ in standard form (with A is $m \times n$), its **dual** linear program is

$$\begin{aligned} \text{Minimize} \quad & by \\ \text{subject to} \quad & yA \geq c, \\ & y \geq 0, \end{aligned}$$

where $y = (y_1, \ldots, y_m)$. (Note the mistake from lecture.) It can be shown that the dual of the dual of the original linear program (called the **primal**) is again the primal, giving its name. To complete correctness of `Simplex`, two results are needed.

Claim 1. *Let $\bar{x}$ be a feasible solution to L and let $\bar{y}$ be a feasible solution to the dual of L. Then*

$$\sum_{j=1}^{n} c_j \bar{x}_j \leq \sum_{i=1}^{m} b_i \bar{y}_i.$$

Proof. Note that for each $j \in \{1, \ldots, n\}$, $c_j \leq \sum_{i=1}^{m} a_{ij} \bar{y}_i$ since $\bar{y}$ is a feasible solution to the dual of L (and therefore satisfies the constraints of the dual of L). Also, for each $i \in \{1, \ldots, m\}$, $\sum_{j=1}^{n} a_{ij} \bar{x}_j \leq b_i$ since $\bar{x}$ is a feasible solution to L. This results in

$$\begin{aligned} \sum_{j=1}^{n} c_j \bar{x}_j &\leq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} \bar{y}_i \right) \bar{x}_j \\ &= \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} \bar{x}_j \right) \bar{y}_i \\ &\leq \sum_{i=1}^{m} b_i \bar{y}_i, \end{aligned}$$

which gives the result. □

This shows that since the dual of L is a minimization problem, a feasible solution $\bar{y}$ to the dual of L which gives exact equality in the claim above implies that $\bar{y}$ cannot be optimized any further, nor can $\bar{x}$, since L is a maximization problem which is subject to the inequality above. It is therefore desired to show that given an input primal L to `Simplex` which returns feasible solution $\bar{x}$, one can come up with a feasible solution $\bar{y}$ to the dual of L such that

$$\sum_{j=1}^{n} c_j \bar{x}_j = \sum_{i=1}^{m} b_i \bar{y}_i,$$

which will show that both solutions are optimal, and in particular, that `Simplex` returns an optimal feasible solution (provided that L has finite upper bounds on all of its variables). So, let the last slack form of the primal L as input to `Simplex` be

$$\begin{aligned} z &= v' + \sum_{j \in N} c'_j x_j \\ x_i &= b'_i - \sum_{j \in N} a'_{ij} x_j, i \in B. \end{aligned}$$

Let

$$\bar{y}_i = \begin{cases} -c'_{n+i}, & n + i \in N, \\ 0, & \text{otherwise.} \end{cases}$$

Correctness Result 1. *Suppose that* `Simplex` *returns* $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n)$ *on the primal linear program* $L = (A, b, c)$*. Let* N *and* B *denote the indices of the nonbasic and basic variables in teh final slack form, with* c' *the coefficients in the objective function for the final slack form, and let* $\bar{y} = (\bar{y}_1, \ldots, \bar{y}_m)$ *be given as above. Then* $\bar{y}$ *is a feasible solution to the dual of* L*, and*

$$\sum_{j=1}^{n} c_j \bar{x}_j = \sum_{i=1}^{m} b_i \bar{y}_i,$$

which shows that $\bar{x}$ *and* $\bar{y}$ *are optimal solutions to* L *and the dual of* L*, respectively.*

Proof. First, note that $c'_j < 0$ for all $j \in N$, since otherwise `Simplex` would not have terminated. Since basic variables are not included in the objective function, $c'_j = 0$ for all $j \in B$. Thus one can rewrite z above as

$$\begin{aligned} z &= v' + \sum_{j \in N} c'_j x_j & &= v' + \sum_{j \in N} c'_j x_j + \overbrace{\sum_{j \in B} c'_j x_j}^{0} \\ &= v' + \sum_{j=1}^{n+m} c'_j x_j. \end{aligned}$$

But all the slack forms are equivalent, so the original objective function

$$\sum_{j=1}^{n} c_j x_j = v' + \sum_{j=1}^{n+m} c'_j x_j,$$

and so the objective value is $\sum_{j=1}^{n} c_j \bar{x}_j$, which gives

$$\sum_{j=1}^{n} c_j \bar{x}_j = v' + \sum_{j \in N} c'_j \overbrace{\bar{x}_j}^{0} + \overbrace{\sum_{j \in B} c'_j x_j}^{0} = v'$$

(recall that all nonbasic variables are assigned 0 at the end `Simplex`). That is,

$$\sum_{j=1}^{n} c_j \bar{x}_j = v'.$$

Then

$$\begin{aligned}
\sum_{j=1}^{n} c_j x_j &= v' + \sum_{j=1}^{n+m} c'_j x_j \\
&= v' + \sum_{j=1}^{n} c'_j x_j + \sum_{j=n+1}^{n+m} c'_j x_j \\
&= v' + \sum_{j=1}^{n} c'_j x_j + \sum_{i=1}^{m} c'_{n+i} x_{n+i} \\
&= v' + \sum_{j=1}^{n} c'_j x_j + \sum_{i=1}^{m} (-\bar{y}_i) x_{n+i} \qquad \text{by definition of } \bar{y}_i \\
&= v' + \sum_{j=1}^{n} c'_j x_j + \sum_{i=1}^{m} (-\bar{y}_i) \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right) \qquad \text{from the original slack form for } L \\
&= v' + \sum_{j=1}^{n} c'_j x_j - \sum_{i=1}^{m} b_i \bar{y}_i + \sum_{i=1}^{m} \sum_{j=1}^{n} (a_{ij} x_j) \bar{y}_i \\
&= v' + \sum_{j=1}^{n} c'_j x_j - \sum_{i=1}^{m} b_i \bar{y}_i + \sum_{j=1}^{n} \sum_{i=1}^{m} (a_{ij} \bar{y}_i) x_j \\
&= \left(v' - \sum_{i=1}^{m} b_i \bar{y}_i \right) + \sum_{j=1}^{n} \left(c'_j + \sum_{i=1}^{m} (a_{ij} \bar{y} \right).
\end{aligned}$$

Therefore

$$\sum_{j=1}^{n} c_j x_j = \left(v' - \sum_{i=1}^{m} b_i \bar{y}_i \right) + \sum_{j=1}^{n} \left(c'_j + \sum_{i=1}^{m} (a_{ij} \bar{y} \right) x_j.$$

By Lemma 1 above, this means that, for all $j \in \{1, \ldots, n\}$,

$$c_j = c'_j + \sum_{i=1}^{m} a_{ij} \bar{y}_i$$

and

$$v' - \sum_{i=1}^{m} b_i \bar{y}_i = 0$$

But then, by Claim 1 and the results above,

$$v' = \sum_{j=1}^{n} c_j \bar{x}_j \leq \sum_{i=1}^{m} b_i \bar{y}_i = v',$$

so

$$\sum_{j=1}^{n} c_j \bar{x}_j = \sum_{i=1}^{m} b_i \bar{y}_i,$$

showing optimality. It remains to show that $\bar{y}$ is a feasible solution to the dual of L. For all $j \in \{1, \ldots, n\}$, one gets from the other equation derived from Lemma 1 that

$$\begin{aligned}
c_j &= c'_j + \sum_{i=1}^{m} a_{ij} \bar{y}_i \\
&\leq \sum_{i=1}^{m} a_{ij} \bar{y}_i.
\end{aligned}$$

But this means that $\bar{y}$ is feasible since then $yA \geq c$. This gives the result.

□

9 References

[1] Cormen, T.H, Leiserson, C.E, Rivest, R.L, and Stein, C. *Introduction to Algorithms, Third Edition,* MIT Press, 2009.

[2] Glicksman, A.M. *An Introduction to Linear Programming and the Theory of Games,* John Wiley and Sons, Inc, 1963.

[3] Gass, S. *Linear Programming, Methods and Applications,* McGraw-Hill Book Co., 1958.

[4] Hoffman, A. "Cycling in the Simplex Algorithm," National Bureau of Standards Report, 1953.

[5] Beale, E.M.L. "Cycling in the Dual Simplex Algorithm," *Naval Research Logistics Quarterly,* v.2, no.4, 1955.

[6] Rizzi, R. "ADDENDUM: Bland's pivoting rule," `http://users.dimi.uniud.it/~romeo.rizzi/classes/LP/homeworks/Bland/Bland.html`, 2001.