1. Introduction

Linear programs come in many different forms. Traditionally, one develops the theory for a few special formats. These formats are equivalent to one another and to any other linear programs in the sense that simple transformations can be used to transform from one format to the other.

We will take another approach. It turns out that the ideas we used to develop the simplex method for dictionaries can be easily extended to the most general cases.

We first make some general remarks on auxiliary problems that are used to get the starting feasible canonical form or dictionary. Then we extend the dictionary concept so that it can deal with the most general LPs.

2. Special Formats for Linear Programs

Dantzig [1963] defines three special formats, the canonical form, the standard form, and the inequality form. I will generally call the inequality form the symmetric form for reasons that will become clear when we discuss duality. The three formats in matrix notation are:

The Symmetric Form:

\[
\begin{align*}
\text{Minimize} \quad z &= cx \\
\text{subject to} \quad Ax &\geq b \\
&\quad x \geq 0
\end{align*}
\]

The Canonical Form:
Minimize $z = cx$
subject to $Ax + Iy = b$
$x \geq 0, y \geq 0$

The Standard Form:

Minimize $z = cx$
subject to $Ax = b$
$x \geq 0$

Without loss of generality, we can, and do, assume $b \geq 0$ in the standard form; we cannot make such an assumption for the symmetric and canonical forms.

3. Transforming Devices
Several devices or tricks allow one to move from one format to another. In addition sometimes an auxiliary problem has to be solved. We discuss this in the next section. Here are some “devices“:

\[
\begin{align*}
ax &= b & \iff & & ax \geq b, \text{ and } -ax \geq -b & \text{Converts equalities to inequalities} \\
ax &\leq b & \iff & & ax + s = b, s \geq 0 & \text{Converts inequalities to equalities} \\
ax &\geq b & \iff & & ax - s = b, s \geq 0 & \text{Converts inequalities to equalities} \\
x_j &\text{ (free, not sign)} & \iff & & x_j^+ - x_j^- = 0, x_j^+ \geq 0, x_j^- \geq 0 & \text{Converts unsigned variables to signed}
\end{align*}
\]
4. **Phase I**

Using these transformation, we can represent any linear program in the standard format. So, in our first pass at finding an initial feasible canonical form or dictionary, we assume we start with our linear program in standard form. Later, we will give a more general technique that starts with a linear program in a very general form.

As we have already observed, we can assume \( b \geq 0 \) in the standard form:

\[
\begin{align*}
\text{Minimize} & \quad z = cx \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0 
\end{align*}
\]

We now introduce an auxiliary problem, called the **Phase I Problem**. The sole purpose of solving this problem is to obtain an initial canonical/dictionary form.

\[
\begin{align*}
\text{Minimize}_{x,y} & \quad v = ey \\
\text{subject to} & \quad -z + cx = 0 \\
& \quad Ax + Iy = b \\
& \quad x \geq 0, \quad y \geq 0 
\end{align*}
\]

This is almost in canonical form. The only difficulty is that the basic vectors \( y \) appear in the objective. This is easily taken care of. Simply add all the rows of \( Ax + Iy \) together and subtract from the objective. We obtain:
Minimize \( \begin{align*} v &= -[eA]x + eb \\ \text{subject to} \quad &-z + cx = 0 \\ &Ax + Ly = b \\ &x \geq 0, \quad y \geq 0 \end{align*} \)

where \( e \) is a row vector of \( m \) ones. This is in canonical form except for the constant term, which, as we have already observed, matters little. The Phase I problem cannot have an unbounded solution because \( v \geq 0 \). Thus, the simplex method applied to the Phase I problem must end up with an optimal solution. If there is a feasible solution \( x^f \) to the original problem, then \( x = x^f \), and \( y = 0 \) is a solution with \( v^* = 0 \). So if Phase I ends with \( v^* > 0 \), there is not feasible solution, and we’re done.

Conversely if \( v^* = 0 \), then \( y_i = 0, \quad i = 1, \ldots, m \). This almost implies that we can just throw out all these auxiliary variables \( y_1, \ldots, y_m \), since they are all \( 0 \), and be left with a canonical form/dictionary that is equivalent to our original problem. This is almost true, but some of the artificial variables might end up in the last basis. Let us look at a couple of examples:

\[
\begin{align*}
\text{Min.} \quad z &= x_1 + x_2 \\
\text{s.t.} \quad &x_1 + x_2 + x_3 = 6 \\
&2x_1 - x_2 - x_3 = 3 \\
&x_1 \geq 0 \quad x_2 \geq 0 \quad x_3 \geq 0
\end{align*}
\]

Consider:

\[
\begin{align*}
\text{Min.} \quad v &= y_1 + y_2 \\
\text{s.t.} \quad &x_1 + x_2 = z \\
&x_1 + x_2 + x_3 + y_1 = 6 \\
&2x_1 - x_2 - x_3 + y_2 = 3 \\
&x_1 \geq 0 \quad x_2 \geq 0 \quad x_3 \geq 0 \quad y_1 \geq 0 \quad y_2 \geq 0
\end{align*}
\]

Subtract the constraint equations, yielding:
which is our Phase I Problem. Let’s solve it. We increase $x_1$ until $3/2$ when $y_2$ starts to go negative. We make $y_2$ non-basic and $x_1$ basic using $x_1 = \frac{3}{2} + \frac{1}{2} x_2 + \frac{1}{2} x_3 - \frac{1}{2} y_2$ resulting in:

\[ \begin{align*}
\text{Min.} & \quad v = \frac{3}{2} y_2 - \frac{3}{2} x_2 - \frac{3}{2} x_3 + \frac{9}{2} \\
\text{s.t.} & \quad \frac{3}{2} x_2 + \frac{1}{2} x_3 + y_1 = \frac{3}{2} \\
& \quad \frac{3}{2} x_2 + \frac{3}{2} x_3 + y_1 = \frac{9}{2} \\
& \quad \frac{1}{2} x_2 - \frac{1}{2} x_3 + \frac{1}{2} x_1 = \frac{3}{2} \\
& \quad x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad y_1 \geq 0, \quad y_2 \geq 0
\end{align*} \]

(Note, as soon as $y_2$ becomes non-basic we could just get rid of it, but we will keep it for now.)

At this point we could pick either $x_2$ or $x_3$ to come into the basis. We will take $x_2$. We can increase $x_2$ up to 3. At this point $x_2$ goes into the basis and $y_1$ leaves. Using $x_2 = 3 \frac{1}{3} y_2 - \frac{2}{3} y_1 - x_3$, we obtain:

\[ \begin{align*}
\text{Min.} & \quad v = y_2 + y_1 \\
\text{s.t.} & \quad -y_1 - x_3 = z - 6 \\
& \quad \frac{1}{3} y_2 + \frac{2}{3} y_1 + x_3 + x_2 = 3 \\
& \quad \frac{1}{3} y_2 + \frac{1}{3} y_1 + x_1 = 3 \\
& \quad x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad y_1 \geq 0, \quad y_2 \geq 0
\end{align*} \]

We have now achieved a canonical form for our problem. At this point we can eliminate the Phase I Objective ($v$), and all the
auxiliary variables $y_1$ and $y_2$. We then apply the simplex method to solve the problem. This is called Phase II.

Exercise 1: Apply the simplex method (Phase II) to the canonical form that we arrived at.

Exercise 2: Instead of bringing $x_2$ into the basis in the second iteration of Phase I, bring in $x_3$. Continue to the end of Phase I, and then perform Phase II.

Exercise 3: Perform Phase I on:

Minimize $x_1 + x_2$

s.t. $-x_1 - x_2 \leq 1$

$x_1 + x_2 \leq -1$

$x_1 \geq 0 \quad x_2 \geq 0$

Hint: Put in “slack variables” to convert the inequalities into equations, and put in standard form. Remember that the right hand side must be non-negative.

Exercise 4: Perform Phase I and Phase II on:

Min $x_1$

$x_1 - 2x_2 + x_3 = 4$

$x_1 + 3x_3 = 8$

$x_1 - x_2 + 2x_3 = 6$

$x_1 \geq 0 \quad x_2 \geq 0 \quad x_3 \geq 0$

Hint: At the end of Phase I, you might find that you need to make some adjustments to proceed to Phase II; discuss how you approach any issues that arise.

Now we are in a position to state the fundamental theorem for the simplex method:

**Fundamental Theorem of Linear Programming**: Every linear program has the following properties:
(i) If it has no optimal solution, then it is either infeasible or unbounded.

(ii) If it has a feasible solution, then it has a basic feasible solution.

(iii) If it has an optimal solution, then it has a basic optimal solution.

5. Extended Dictionaries

Linear programming problems can be expressed in many equivalent forms. A form that is convenient for our application, and is among the most general is:

\[
\begin{align*}
\text{Minimize } z &= cx \\
b^l &\leq Ax \leq b^u \\
l_j \leq x_j \leq u_j \text{ for } j = 1, \ldots, n
\end{align*}
\]

Here \( A \) is a given \( m \times n \) matrix, \( x \) is an \( n \)-vector of decision variables \( x_j \), each with given lower bounds \( l_j \) and upper bounds \( u_j \). The \( m \)-vectors \( b^l \) and \( b^u \) are given data that define constraints. With an appropriate implementation, the simplex method takes little more work to solve the extended format than the simpler forms given in textbooks. Obviously, minimizing \( z \) is equivalent to maximizing \( -z \). Thus, maximization problems can be handled using the same methods as minimization problems. The lower bound, \( l_j \), may take on the value \( -\infty \) and the upper bound, \( u_j \), may take on the value \( +\infty \). Similarly, some or all of the components of \( b^l \) may be \( -\infty \), and some or all of \( b^u \) may be \( +\infty \), this gives rise to the following terminology for variables:

<table>
<thead>
<tr>
<th>( l_j )</th>
<th>( u_j )</th>
<th>Variable Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>finite, less than ( u_j )</td>
<td>finite, greater than ( l_j )</td>
<td>bounded</td>
</tr>
</tbody>
</table>
Similar, but somewhat different terminology is used for the constraints:

<table>
<thead>
<tr>
<th>$b_i^l$</th>
<th>$b_i^u$</th>
<th>Constraint Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>finite, less than $b_i^u$</td>
<td>finite, greater than $b_i^l$</td>
<td>range</td>
</tr>
<tr>
<td>finite</td>
<td>$+\infty$</td>
<td>lower bounded</td>
</tr>
<tr>
<td>$-\infty$</td>
<td>finite</td>
<td>upper bounded</td>
</tr>
<tr>
<td>$-\infty$</td>
<td>$+\infty$</td>
<td>relaxed</td>
</tr>
<tr>
<td>equals $b_i^u$</td>
<td>equals $b_i^l$</td>
<td>equality</td>
</tr>
</tbody>
</table>

The simplex algorithm resolves this linear programming problem in a finite number of iterations resulting in one of the three following outcomes:

(i) An optimal solution to the problem

(ii) A linear class of solutions which satisfy the constraints and which give values to $z$ that are unbounded below, or

(iii) A demonstration that the constraints have no solution (independent of $z$ values)

We now replace the constraints, $b^l \leq Ax \leq b^u$ by:
\[ y = Ax \]
\[ b_i^l \leq y_i \leq b_i^u \quad \text{for } i = 1, \ldots, m \]

This leads us to:

\[
\begin{align*}
\text{Minimize} \quad & z = \sum_{j=1}^{n} c_j x_j \\
\text{Subject to} \quad & y_i = \sum_{j=1}^{n} a_{ij} x_j \quad (i = 1, 2, \ldots, m) \\
& l_j \leq x_j \leq u_j \quad \text{for } j = 1, \ldots, n \\
& b_i^l \leq y_i \leq b_i^u \quad \text{for } i = 1, \ldots, m
\end{align*}
\]

This equation together with an assignment of values to the non-basic variables \( x \) is a variant of the dictionary representation of Strum and Chvátal [Chvátal, 1983]. The dictionary is said to be feasible for given values of the independent (non-basic) variables \( x_1, \ldots, x_n \) if the given values satisfy their bounds and if the resulting values for the dependent (basic) variables \( y_1, \ldots, y_m \) satisfy theirs. If a dictionary, feasible or not, has the property that each non-basic variables is either at its upper bound or its lower, then the dictionary is said to be basic. Suppose our dictionary besides being feasible has the following optimality properties; (i) for every non-basic variable \( x_j \) that is strictly above its lower bound we have \( c_j \leq 0 \), and (ii) for every non-basic \( x_j \) that is strictly below its upper bound we have \( c_j \geq 0 \). Such a dictionary is said to be optimal. It is easy to see that no change in the non-basic variables will increase \( z \) and hence the current solution is optimal.

Starting with a feasible dictionary, the standard simplex method involves a sequence of feasible dictionaries. Each iteration consists of three steps:

1. **Select Column:**
Choose a non-basic variable, $x_s$, which violates one of the two optimality properties. Such a non-basic variable is said to be eligible. There may be many such; there are also several criteria for choosing the non-basic variable. We will discuss some shortly. If there is no such variable the current solution is optimal. In this latter case we stop with an optimal solution.

2. **Select Row:**

Increase the non-basic variable if the first optimality condition was violated (decrease the non-basic variable if the second optimality condition was violated) until the non-basic variable or one of the basic variables reaches its bound. If there is no limit to the change you can make in the non-basic variable, $x_s$, the value of $z$ is unbounded below and continuing to change $x_s$ will result in ever decreasing values of $z$. We then terminate with a class of feasible solutions with the objective unbounded below.

3. **Pivot:**

If the non-basic variable reaches its bound, then the next dictionary is determined by all the non-basics remaining the same except for $x_s$ which is set at the bound it reached. The basic variables are adjusted accordingly. If a basic variable $y_r$ starts to exceed its bound before $x_s$ reaches its bound then $x_s$ and $y_r$ exchange their roles; that is, $y_r$ becomes non-basic and $x_s$ becomes basic. This is accomplished by a pivot step. The result is the next dictionary.

It can be demonstrated that this process terminates in a finite number of dictionaries; however, there is a technical difficulty. The basic idea in showing finiteness is that each iteration will result in an increase in $z$ so that dictionaries cannot be repeated. It is fairly easy to see that only a finite number of different dictionaries can result from a given basic, feasible, initial dictionary and thus there must be a finite number of iterations in
the simplex method. The difficulty occurs in Step 2 in the case where some basic variables are already at one of their bounds so that we cannot change $x_s$ at all. This is called degeneracy. This can be handled (see Chvátal[1983, Chapter 3] for a general discussion, and [Gill et al, 1989] for the specific techniques we use); we will return to this point later. Another important observation is that there may be many eligible choices for $x_s$ but as far as the finiteness goes any one can be chosen. In practical implementations however the method (column choice rule) used to choose among the eligible non-basic variable can be critical in determining performance. We will come back to this point later. For now we will assume we use the original method used by Dantzig, which is to take the eligible column with the largest absolute value of $c_s$.

References:
Chvátal, Vasek, Linear Programming, Freeman, 1983,