Revised Simplex Algorithm

The algorithm

We consider an LP in the following form (c and x are of dimension n, b of dimension $m, n > m$):

$$maximise\ c^T x$$

Subject to:

$$\begin{gathered} Ax = b \\ x \geq 0 \end{gathered}$$

First we note that this form can handle all possible linear variants:

- We can solve a minimisation problem by multiplying c by -1;
- We can handle "$\leq$" or "$\geq$" constraints by adding additional variables which only appear in the relevant constraint and have coefficient of 1 or -1.

Assume we have an initial set of variables of dimension m which we can denote as the **basic variables**. (We can, if necessary, construct such a solution by introducing artificial start-up variables which have a very large negative cost coefficients).

The basis matrix is those columns of the constraint matrix A relating to the basic variables:

$$B = \begin{bmatrix} B_{11} & \cdots & B_{1m} \\ \vdots & . & \vdots \\ B_{m1} & \cdots & B_{mm} \end{bmatrix}$$

The basic variables take on values: $x^B = \begin{bmatrix} x_1^B \\ \vdots \\ x_m^B \end{bmatrix}$

At any iteration all the non-basic variables are 0.

Therefore:

$$\begin{gathered} Bx^B = b \\ x^B = B^{-1}b \end{gathered}$$

At any iteration, given the original b vector and the inverse basis matrix B^{-1}, x^B (the current solution) can be calculated. We also note the current objective function can be calculated as:

$$z^B = {c^B}^T x^B$$

where c^B holds the objective function coefficients of the basic variables.

Revised Simplex Algorithm

1. Determine the entering variable x_{j^*} with associated vector of the constraint matrix p^{j^*}:
    a. Compute the dual variables $y = {B^{-1}}^T c^B$ (or alternatively $y^T = {c^B}^T B^{-1}$).
    b. Compute the "reduced cost": $c_j' = c_j - y^T p^j$ for all non-basic variables.
    c. Choose j^* such that $c_{j^*}' > 0$ We require a positive reduced cost for a maximisation problem (most positive or some other rule).
    d. If there are no positive reduced costs, then stop with the optimal solution.
2. Determine the leaving variable x_r with associated vector p^r:
    a. Compute the direction in which we are going to move $\alpha^{j^*} = B^{-1} p^{j^*}$. As we increase x_{j^*} then the other variables reduce by α^{j^*}.
    b. x_r is the basic variable which achieves the value $\theta = \min_k \left\{ {x_k^B}\big/{\alpha_k^{j^*}} \,\middle|\, \alpha_k^{j^*} > 0 \right\}$.
    c. If there is no such x_r then stop with an unbounded solution.
3. Remove x_r from the basis and add x_{j^*}. Calculate the new B^{-1} and $x^B = B^{-1}b$ and go to step 1.

If we move distance λ in this direction then the new point (dimension $m + 1$) is $\begin{bmatrix} x^B - \lambda\alpha^{j^*} \\ \lambda \end{bmatrix}$. Multiplying these variables by the relevant portion of the constraint matrix gives:

$$\begin{aligned} \begin{bmatrix} B & p^{j^*} \end{bmatrix} \begin{bmatrix} x^B - \lambda\alpha^{j^*} \\ \lambda \end{bmatrix} &= Bx^B - \lambda B\alpha^{j^*} + \lambda p^{j^*} \\ &= b - \lambda B B^{-1} p^{j^*} + \lambda p^{j^*} \\ &= b \end{aligned}$$

So moving in this direction ensures the constraints stay satisfied. We need only ensure that all the coefficients of $x^B - \lambda\alpha^{j^*}$ remain positive.

Also, the new objective function will be:

$$\begin{aligned} \begin{bmatrix} {c^B}^T & c_{j^*} \end{bmatrix} \begin{bmatrix} x^B - \lambda\alpha^{j^*} \\ \lambda \end{bmatrix} &= {c^B}^T x^B - \lambda {c^B}^T \alpha^{j^*} + \lambda c_{j^*} \\ &= z^B - \lambda {c^B}^T B^{-1} p^{j^*} + \lambda c_{j^*} \\ &= z^B - \lambda y^T p^{j^*} + \lambda c_{j^*} \\ &= z^B + \lambda (c_{j^*} - y^T p^{j^*}) \\ &\geq z^B \end{aligned}$$

Large scale simplex algorithms need to cope with several issues:

- Maintaining and updating a representation of B^{-1} without explicitly recalculating it at every iteration. B is typically very sparse. Specialist sparse LU factorisation techniques are used.
- Rank 1 updates.
- Determining the entering variable. There can be a huge number of variables and calculating the reduced cost for all of them can be very time consuming.
- Calculating $B^{-1}p^{j^*}$ and similar operations when the vector in question is very sparse. Doing this efficiently led to a dramatic speed increase in the late 90's.

Worked Example

Consider this problem. Giapetto's Woodcarving Inc manufactures toy trains and toy soldiers. A soldier uses \$10 worth of raw materials and \$14 worth of labour and sells for \$27. A train uses \$9 worth of raw materials and \$10 worth of labour and sells for \$21. A soldier needs 2 hours finishing and 1 hour of carpentry. A train needs 1 hour of finishing and 1 hour of carpentry. Giapetto can obtain all the raw materials it needs but only 100 finishing hours and 80 hours of carpentry per week. They can sell any number of trains each week, but at most 40 toy soldiers.

How many of each kind of toy should be produced per week to maximise revenue.

If we let x_1 be the number of soldiers produced each week and x_2 be the number of trains produced each week, the problem can be modelled as:

$$maximise\ z = 3x_1 + 2x_2$$

Subject to:

$$\begin{array}{rl} 2x_1 + x_2 \leq 100 & (1)\ (Finishing) \\ x_1 + x_2 \leq 80 & (2)\ (Carpentry) \\ x_1 \leq 40 & (3)\ (Demand\ for\ soldiers) \\ x_1, x_2 \geq 0 & (Sign\ constraints) \end{array}$$

We convert this to standard form by the addition of 3 slack variables:

$$maximise\ z = 3x_1 + 2x_2$$

Subject to:

$$\begin{array}{lllllllllll} 2x_1 & + & x_2 & + & x_3 & & & & & = & 100 \\ x_1 & + & x_2 & & & + & x_4 & & & = & 80 \\ x_1 & & & & & & & + & x_5 & = & 40 \\ \multicolumn{11}{l}{x_1, x_2, x_3, x_4, x_5 \geq 0} \end{array}$$

The initial basic variables are x_3, x_4, x_5 with $B = B^{-1} = I, x^B = \begin{pmatrix} 100 \\ 80 \\ 40 \end{pmatrix}$ and $z^B = 0$

Iteration 1

1. Entering variable:
    a. Compute $y = 0$
    b. Reduced costs: $c'_1 = 3, c'_2 = 2$
    c. Entering variable is x_1 with $p^1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$
2. Determine the leaving variable:
    a. $\alpha^1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$
    b. $\theta = 40$, exiting variable is x_5

3. Basis is x_1, x_3, x_4. $B = \begin{pmatrix} 2 & 1 & \\ 1 & & 1 \\ 1 & & \end{pmatrix}, B^{-1} = \begin{pmatrix} & & 1 \\ 1 & & -2 \\ & 1 & -1 \end{pmatrix}, x^B = \begin{pmatrix} 40 \\ 20 \\ 40 \end{pmatrix}, z^B = 120$

Iteration 2

1. Entering variable:
   a. Compute $y^T = (3,0,0)\begin{pmatrix} & & 1 \\ 1 & & -2 \\ & 1 & -1 \end{pmatrix} = (0,0,3)$
   b. Reduced costs: $c_2' = 2, c_5' = -3$
   c. Entering variable is x_2 with $p^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$
2. Determine the leaving variable:
   a. $\alpha^2 = \begin{pmatrix} & & 1 \\ 1 & & -2 \\ & 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$
   b. $\theta = 20$, exiting variable is x_3
3. Basis is x_1, x_2, x_4. $B = \begin{pmatrix} 2 & 1 & \\ 1 & 1 & 1 \\ 1 & & \end{pmatrix}, B^{-1} = \begin{pmatrix} & & 1 \\ 1 & & -2 \\ -1 & 1 & 1 \end{pmatrix}, x^B = \begin{pmatrix} 40 \\ 20 \\ 20 \end{pmatrix}, z^B = 160$

Iteration 3

1. Entering variable:
   a. Compute $y^T = (3,2,0)\begin{pmatrix} & & 1 \\ 1 & & -2 \\ -1 & 1 & 1 \end{pmatrix} = (2,0,-1)$
   b. Reduced costs: $c_3' = -2, c_5' = 1$
   c. Entering variable is x_5 with $p^5 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$
2. Determine the leaving variable:
   a. $\alpha^5 = \begin{pmatrix} & & 1 \\ 1 & & -2 \\ -1 & 1 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$
   b. $\theta = 20$, exiting variable is x_4
3. Basis is x_1, x_2, x_5. $B = \begin{pmatrix} 2 & 1 & \\ 1 & 1 & \\ 1 & & 1 \end{pmatrix}, B^{-1} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & 0 \\ -1 & 1 & 1 \end{pmatrix}, x^B = \begin{pmatrix} 20 \\ 60 \\ 20 \end{pmatrix}, z^B = 180$

Iteration 4

1. Entering variable:
   a. Compute $y^T = (3,2,0)\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & 0 \\ -1 & 1 & 1 \end{pmatrix} = (1,1,0)$
   b. Reduced costs: $c_3' = -1, c_4' = -1$
   c. No entering variable – stop.

Other Considerations

The description of the Revised Simplex Algorithm so far gives rise to a couple of technical questions. How do we ensure we can construct an initial basis? How can we ensure the basis is always invertible?

The Initial Basis

Any method of constructing an initial basis that works and is efficient can be used by solvers. For the purposes of proving the correctness of the Revised Simplex Algorithm we need merely show that it is possible to construct some initial basis. This can be done as follows:

Artificial Basis

For each constraint, add an artificial variable that only appears in that constraint, with a coefficient of 1 (or -1 if the right hand side of that constraint is negative). Let the cost of every artificial variable be $-M$, where M is chosen big enough to ensure none of the artificial variables will be non-zero in an optimal solution. In practice solvers choose a value for M based on the problem data and then increase it if needed to drive all artificial variables from the solution.

Clearly an all artificial basis is invertible, as it is I.

Does the Basis stay Invertible?

Recall from matrix theory that a matrix is invertible if all the columns are linearly independent. Assume the basis at the beginning of an iteration is invertible. Then the basis at the end of the iteration is invertible if the column of A corresponding to the entering variable (p^{j^*}) is not linearly dependent on the basis with the column corresponding to the exiting variable (r) removed.

Recall that we calculate the direction of movement as $\alpha^{j^*} = B^{-1}p^{j^*}$. This means that $p^{j^*} = B\alpha^{j^*}$. Moreover, $\alpha_r^{j^*}$ must be strictly greater than zero due to the way the exiting variable is selected. This gives the desired result as the coefficient corresponding to the exiting variable is non-zero.