Chapter 4

The Simplex Method

4.1 Recap

In the previous chapters we introduced the linear programming problem. We saw how this could be solved (graphically) for two variables, and also constructed an impractical, brute-force method that would work (very slowly) for the general case.

Problems of current interest can have thousands of constraints and millions of variables. To solve these we want to introduce and analyse the *simplex method*. Introduced by Dantzig in 1947, it builds on the fundamental theorem by giving a method of moving, step by step, from one vertex to an adjacent one until the optimal solution is found.

There are a number of steps we need to complete.

1. How do we characterise a vertex (algebraically)?
2. How do we find a vertex to start from?
3. How do we verify if the current vertex is optimal?
4. How do we move from one vertex to the next?

4.2 Setup and assumptions

In this chapter we will use the standard matrix form,

$$\begin{array}{lrcl} \min & cx & & \\ \text{subject to} & Ax & = & b \\ & x & \geq & 0. \end{array} \tag{4.1}$$

Here $A \in \mathbb{R}^{m \times n}, c, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ are the given coefficients that define the problem. As usual m is the number of constraints and n the number of variables.

We make three assumptions:

1. $b \geq 0$. If this is not true for some row i so that $b_i < 0$ we can multiply that row (both b_i and A_{ij} for all columns j) by -1.
2. $m < n$. That is, the number of constraints is less than the number of variables. If this is not true then the linear system defined by $Ax = b$ is either square or overdetermined, implying one or no solutions (the feasible region is a single point or empty).
3. A has full row rank so its rows are linearly independent.

The second assumption may seem to contradict earlier examples where there were more constraints than variables. However, in all those cases the constraints were *inequalities.* To transform to standard form we need to increase the number of variables (using slack or surplus variables). This will change all those problems to ones with $m < n$.

The use of the third assumption can be checked by assuming A is not of full row rank. In that case we can use row operations to transform the system to $A'x = b'$, where the matrix A' contains two rows i and k such that

$$\begin{aligned} a'_i x &= b'_i, && (4.2)\\ a'_k x &= b'_k, && (4.3)\\ a'_i &= a'_k. && (4.4) \end{aligned}$$

There are then two cases.

1. $b'_i = b'_k$: the two constraints are identical and one can be dropped from the problem.
2. $b'_i \neq b'_k$: the constraints are inconsistent and so the linear problem is infeasible.

So the third assumption stops us from considering infeasible problems and ensures constraints are not repeating information.

4.3 Basic matrices

There is a general mathematical strategy of solving a complex problem by first solving a simple case, then transforming the general case to the simple one. This can seem indirect. The next few sections will build up a simple case for linear programs.

Let $B \subseteq \{1, \ldots, n\}$ be a subset of the column indexes of A, with $|B| = m$. That is, B will identify a number of columns of A, corresponding to variables in the linear program, and the number of columns identified will match the number of constraints.

A *basic matrix* A_B is the submatrix of A with *linearly independent columns* that is obtained by dropping every column with index not in B. If A_B is a basic matrix then

- A_B is square;
- A_B has full row and column rank (by the assumptions);
- A_B is invertible, as it has full row and column rank.

We can use this construction to split, of *partition*, the linear program. Let $N = \{1, \ldots, n\} \setminus B$ be the complement of B. Let c_B, x_B, A_B be the components (for the vectors c, x) or columns (for the matrix A) that are indexed by B, and c_N, x_N, A_N be the components (for the vectors c, x) or columns (for the matrix A) that are indexed by N. Then we can use column operations on the linear program to re-order the vectors and matrices, writing

$$\begin{aligned} c &= (c_B, c_N), && (4.5)\\ A &= (A_B, A_N), && (4.6)\\ x &= (x_B, x_N). && (4.7) \end{aligned}$$

This means that our standard matrix form description of the linear program becomes

$$\begin{array}{lrcl} \text{min} & \begin{pmatrix} c_B & c_N \end{pmatrix} \begin{pmatrix} x_B \\ x_N \end{pmatrix} & & \\ \text{subject to} & \begin{pmatrix} A_B & A_N \end{pmatrix} \begin{pmatrix} x_B \\ x_N \end{pmatrix} & = & b \\ & \begin{pmatrix} x_B \\ x_N \end{pmatrix} & \geq & \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \end{array} \tag{4.8}$$

Rewriting this by expanding out the matrix multiplications gives

$$\begin{array}{lrcl} \min & c_B x_B + c_N x_N & & \\ \text{subject to} & A_B x_B + A_N x_N & = & b \\ & x_B & \geq & 0 \\ & x_N & \geq & 0. \end{array} \tag{4.9}$$

Why have we made the problem more complicated? Because the basic matrix is invertible, we can *directly solve* for the optimal basic solution. We have

$$Ax = b \tag{4.10}$$

$$\iff \quad A_B x_B + A_N x_N = b \tag{4.11}$$

$$\iff \quad x_B = A_B^{-1} b - A_B^{-1} A_N x_N. \tag{4.12}$$

This allows us to write the objective function as

$$cx = c_B x_B + c_N x_N \tag{4.13}$$

$$= c_B A_B^{-1} b + \left(c_N - c_B A_B^{-1} A_N\right) x_N. \tag{4.14}$$

We can also add the term $(c_B - c_B A_B^{-1} A_B) x_B$ to the objective function as it vanishes. This is useful later.

This defines the *basic form* of a standard form linear program: if A_B is a basic matrix then the basic form is

$$\begin{array}{lrcl} \min & c_B A_B^{-1} b + \left(c_N - c_B A_B^{-1} A_N\right) x_N & + & \left(c_B - c_B A_B^{-1} A_B\right) x_B \\ \text{subject to} & x_B + A_B^{-1} A_N x_N & = & A_B^{-1} b \\ & x_B & \geq & 0 \\ & x_N & \geq & 0. \end{array} \tag{4.15}$$

4.4 Basic solutions

When the linear program is in basic form we call x_B the *basic variables* and x_N the *non-basic variables.* If there is a solution with $x_N = 0$ we call it a *basic solution.* A basic solution need not satisfy $x_B \geq 0$ and so it may not fall in the feasible region. If there is a solution $x = (x_B, x_N)$ with $x_N = 0$ and also $x_B \geq 0$ then it is called *basic feasible.*

Note that the constraints automatically imply that if the solution is basic and so $x_N = 0$, then

$$x_B = A_B^{-1} b. \tag{4.16}$$

Basic solutions can be constructed by solving the linear system defined by the basic matrix.

4.4.1 Basic solutions and vertices

There is a direct link between basic solutions and vertices.

Theorem 4.4.1. *x is a basic feasible solution if, and only if, it is a vertex of the feasible region.*

Proof. First prove the implication using contradiction.

Let x be a basic feasible solution with $x = (x_B, x_N)$. Assume, by contradiction, that x is not a vertex. As the feasible region is convex, if x is not a vertex then there are two distinct points y, z, both distinct from x, such that

$$x = \lambda y + (1 - \lambda) z, \quad \lambda \in (0, 1). \tag{4.17}$$

We can split both points in the same way as the basic feasible solution x, as $y = (y_B, y_N)$ and $z = (z_B, z_N)$. This gives

$$x_N = \lambda y_N + (1 - \lambda) z_N, \quad \lambda \in (0, 1). \tag{4.18}$$

As x is a basic feasible solution we have $x_N = 0$. As $\lambda \in (0,1)$ we have $\lambda > 0$ and $1-\lambda > 0$. As y and z are feasible we have $y_N, z_N \geq 0$. This requires that $y_N = 0 = z_N$.

Similarly, as y and z are feasible, we also have that $Ay = b$ and $Az = b$. As all the non-basic pieces must vanish, we have that $A_B x_B = b = A_B y_B = A_B z_B$. Since A_B is non-singular (as it is a basic matrix), any linear system defined by it has a unique solution. Therefore $x_B = y_B = z_B$. As all the non-basic pieces vanish, we have that $x = y = z$. This contradicts the assumption that they are all distinct. Therefore a basic feasible solution is a vertex.

Next prove that a vertex is a basic feasible solution. Define $\hat{B} \subseteq \{1, \dots, n\}$ as the set of indexes of the components of the vertex x which are positive, $\hat{B} = \{i \colon x_i > 0\}$. Let $\hat{N}$ be the complement, $\hat{N} = \{1, \dots, n\} \setminus \hat{B}$. Write $x = (x_{\hat{B}}, x_{\hat{N}})$ and re-write $Ax = b$ as

$$A_{\hat{B}} x_{\hat{B}} + A_{\hat{N}} x_{\hat{N}} = b. \tag{4.19}$$

By construction $x_{\hat{N}} = 0$ and so $A_{\hat{B}} x_{\hat{B}} = b$.

We need to show that $A_{\hat{B}}$ is a basic matrix. We do this by contradiction. Assume the columns of $A_{\hat{B}}$ are not linearly independent. Then there is some $y_{\hat{B}} \neq 0$ such that $A_{\hat{B}} y_{\hat{B}} = 0$. By construction $x_{\hat{B}} > 0$. Therefore there is some constant $\epsilon > 0$ such that

$$x_{\hat{B}} - \epsilon y_{\hat{B}} \geq 0 \text{ and } x_{\hat{B}} + \epsilon y_{\hat{B}} \geq 0. \tag{4.20}$$

Also by construction we have

$$A_{\hat{B}} \left(x_{\hat{B}} \pm \epsilon y_{\hat{B}} \right) = b. \tag{4.21}$$

Thus the two distinct points $(x_{\hat{B}} \pm \epsilon y_{\hat{B}}, 0)$ are feasible solutions to $Ax = b$. Finally, note that

$$x = (x_{\hat{B}}, 0) = \frac{1}{2} \left(x_{\hat{B}} - \epsilon y_{\hat{B}} \right) + \frac{1}{2} \left(x_{\hat{B}} + \epsilon y_{\hat{B}} \right) \tag{4.22}$$

This shows that x can be written as the convex combination of two distinct feasible points. This contradicts x being a vertex. Therefore the columns of $A_{\hat{B}}$ are linearly independent.

To show that $A_{\hat{B}}$ is basic we need to show that it is a square matrix. If $|\hat{B}| = m$ then the proof is complete. If not, we can select more indices from $\hat{N}$ so that the columns of the enlarged $A_{\hat{B}}$ are linearly independent. This is possible as A has full row rank. The solution will then be degenerate - it will have basic variables with zero value - but remain basic. □

4.5 Optimality

The previous sections have shown how to algebraically characterise a vertex (and we remember that the optimal solution must lie on some vertex): a vertex corresponds to a basic feasible solution. We now want to understand how to check if a vertex (or equivalently a basic feasible solution) is optimal, without enumerating all cases.

4.5.1 Reduced costs

Remember that the linear program can be written in basic form, with some basic matrix A_B, as

$$\begin{array}{llcl} \min & c_B A_B^{-1} b + \left(c_N - c_B A_B^{-1} A_N \right) x_N & + & \left(c_B - c_B A_B^{-1} A_B \right) x_B \\ \text{subject to} & x_B + A_B^{-1} A_N x_N & = & A_B^{-1} b \\ & x_B & \geq & 0 \\ & x_N & \geq & 0. \end{array} \tag{4.23}$$

Certain terms are sufficiently important to get names, which are all types of *reduced costs*:

$$\bar{c}_N = c_N - c_B A_B^{-1} A_N \qquad \text{reduced costs of non-basic variables,} \tag{4.24}$$

$$\bar{c}_B = c_B - c_B A_B^{-1} A_B \qquad \text{reduced costs of basic variables,} \tag{4.25}$$

$$\bar{c} = c - c_B A_B^{-1} A \qquad \text{reduced costs (of the variables).} \tag{4.26}$$

Note that by construction $\bar{c}_B = 0$ automatically. We also note that $\bar{c} = (\bar{c}_B, \bar{c}_N)$ in a similar fashion to the split of a solution into basic and non-basic pieces (this can be checked explicitly using the matrix form split).

In terms of the reduced costs the basic form linear program is

$$\begin{array}{lrcl} \text{min} & c_B A_B^{-1} b + \bar{c}_N x_N + \bar{c}_B x_B & & \\ \text{subject to} & x_B + A_B^{-1} A_N x_N & = & A_B^{-1} b \\ & x_B & \geq & 0 \\ & x_N & \geq & 0. \end{array} \tag{4.27}$$

This emphasises, as $\bar{c}_B = 0$, that x_B has no impact on the objective function. It also gives us an interpretation of the reduced costs of the non-basic variables: the component $(\bar{c}_N)_j$ is equal to the change in the objective function value when changing the non-basic variable $(x_N)_j$ by one unit.

Therefore, if we start from a basic feasible solution (where, by construction, $x_N = 0$), we can improve (make smaller) the objective function by increasing any component $(x_N)_j$ where the corresponding component of the reduced cost is negative, $(\bar{c}_N)_j < 0$.

4.5.2 Optimality test

Theorem 4.5.1. *If, given a basic feasible solution x with basis B, the reduced cost vector $\bar{c}$ is non-negative, then x is optimal.*

Proof. We show that, for any other feasible solution y, $cy \geq cx$. To do this define $d = y - x$. As

$$cy \geq cx \tag{4.28}$$
$$\iff \quad c(y - x) \geq 0 \tag{4.29}$$
$$\iff \quad cd \geq 0, \tag{4.30}$$

we are looking to prove $cd \geq 0$.

Since x and y are both feasable solutions we have $Ax = b$ and $Ay = b$, implying

$$Ad = A(x - y) \tag{4.31}$$
$$= Ax - Ay \tag{4.32}$$
$$= 0. \tag{4.33}$$

Writing $d = (d_B, d_N)$ as usual we have

$$Ad = 0 \tag{4.34}$$
$$\iff \quad A_B d_B + A_N d_N = 0. \tag{4.35}$$

The basic matrix A_B is non-singular by construction, so

$$d_B = -A_B^{-1} A_N d_N. \tag{4.36}$$

We can then evaluate

$$cd = c_B d_B + c_N d_N \tag{4.37}$$
$$= c_B \left(-A_B^{-1} A_N d_N\right) + c_N d_N \tag{4.38}$$
$$= \left(c_N - c_B A_B^{-1} A_N\right) d_N \tag{4.39}$$
$$= \bar{c}_N d_N. \tag{4.40}$$

By assumption we have $\bar{c}_N \geq 0$. To show that $cd \geq 0$ we need to show that $d_N = y_N - x_N \geq 0$.

We note that y is a feasible solution which implies that $y_N \geq 0$. Similarly x is a *basic* feasible solution which requires that $x_N = 0$. Hence $d_N \geq 0$, implying that $cd \geq 0$ and hence $cy \geq cx$. Thus any feasible solution that is not x increases the value of the objective function, meaning x is optimal. □

4.6 Moving between vertices

We now have both a characterisation of vertices (as basic feasible solutions) and a test of their optimality (the associated reduced cost vector is non-negative). If we can easily move from one vertex to another then we have (most of) the ingredients of an algorithm.

We know intuitively that a vertex is connected to a neighbouring vertex by an edge; a one dimensional boundary line of the feasible region. We know a vertex corresponds to a basic feasible solution. The basis B is the set of indices defining which variables are basic at a particular vertex, and has complement N. If we move along an edge we might expect to only be changing two variables (in an appropriate coordinate system the one dimensional line defining the edge lies in a two dimensional plane spanned by two basis vectors). Therefore, moving along an edge from one vertex to another corresponds to *swapping one non-basic variable with one basic variable*. This is equivalent to swapping one entry of N with one entry of B.

4.6.1 Minimum ratio test

We go back to our linear program

$$\begin{array}{lrcl} \min & c_B A_B^{-1} b + \bar{c}_N x_N + \bar{c}_B x_B & & \\ \text{subject to} & x_B + A_B^{-1} A_N x_N & = & A_B^{-1} b \\ & x_B & \geq & 0 \\ & x_N & \geq & 0. \end{array} \tag{4.41}$$

For compactness define $\bar{A} = A_B^{-1} A_N$ and $\bar{b} = A_B^{-1} b$, so the system of equations becomes

$$x_B + \bar{A} x_N = \bar{b}. \tag{4.42}$$

For each single row $i \in \{1, \dots, m\}$ we can write this explicitly as

$$(x_B)_i + \sum_{j \in N} \bar{a}_{ij} (x_N)_j = \bar{b}_i. \tag{4.43}$$

Now, if x corresponds to a vertex then it is a basic feasible solution so that $x_N = 0$. We assume at this point that some component has negative reduced cost. That is, $\exists s \in N$ such that $\bar{c}_s < 0$. It must be the case that $s \in N$ (if one exists) as $\bar{c}_B = 0$ automatically. Therefore we can improve the value of the objective function by increasing the value of x_s (and hence removing s from N). The constraints then require that

$$(x_B)_i = \bar{b}_i - \bar{a}_{is} x_s. \tag{4.44}$$

There is no sum as all terms in N are still basic, so $(x_N)_j = 0$ for all $j \in N \setminus \{s\}$.

We also need to retain all the positivity constraints $(x_B)_i \geq 0$. This requires that

$$\bar{b}_i - \bar{a}_{is} x_s \geq 0. \tag{4.45}$$

If $\bar{a}_{is} \leq 0$ this cannot cause a problem: increasing x_s can never violate this constraint. However, if $\bar{a}_{is} > 0$ then this implies that we must impose

$$x_s \leq \frac{\bar{b}_i}{\bar{a}_{is}}. \tag{4.46}$$

As we need this condition to hold for all constraints, this means we must impose

$$x_s \leq \min_{i \in B \colon \bar{a}_{is} > 0} \frac{\bar{b}_i}{\bar{a}_{is}}. \tag{4.47}$$

We use this to define the intermediate variable

$$\theta^* = \min_{i \in B \colon \bar{a}_{is} > 0} \frac{\bar{b}_i}{\bar{a}_{is}}. \tag{4.48}$$

If $\theta^* > 0$ then setting $x_s = \theta^*$ ensures that

- x_s becomes basic (so that s moves from N to B);
- The value of every variable $(x_B)_r$ with
$$r \in \operatorname*{argmin}_{i \in B \colon \bar{a}_{is} > 0} \frac{\bar{b}_i}{\bar{a}_{is}} \tag{4.49}$$
is decreased to zero. We can select any one of these variables to leave the basis (so that r moves from B to N), whilst the others remain (with value zero).

This gives us a recipe for moving from vertex to vertex: find the value of θ^* for given index s and an index r that gives the minimum. Swap s and r. This is the *minimum ratio test.*

We restricted to the case where $\bar{a}_{is} > 0$ when computing the minimum ratio as the other case does not violate the positivity constraint. However, we also need to consider the objective function.

Theorem 4.6.1. *If there is an index s such that $\bar{c}{+}s < 0$ with $\bar{a}_{is} \leq 0$ for every row $i \in \{1, \dots, m\}$, the linear program is unbounded.*

Proof. Since $\bar{a}_{is} \leq 0$ for all i, x_s can be increased indefinitely without violating any constraint. As $\bar{c}_s < 0$ the objective function can be improved indefinitely. □

4.6.2 Degeneracy

We have seen with the minimum ratio test that it possible to have a basic feasible solution with at least one basic variable having zero value. These solutions are called *degenerate.*

Degenerate solutions can cause problems as different degenerate basic feasible solutions can correspond to the same vertex. To show this, consider a basic feasible solution which is degenerate for some row, index $\hat{\imath}$, so that
$$(x_B)_{\hat{\imath}} = \bar{b}_{\hat{\imath}} = 0. \tag{4.50}$$
Assume we want to increase the value of the non-basic variable x_s, $s \in N$, which starts from zero. As usual we need
$$\bar{b}_{\hat{\imath}} - \bar{a}_{\hat{\imath}s} x_s \geq 0 \tag{4.51}$$
to ensure that $(x_B)_{\hat{\imath}} \geq 0$. If the problem is not unbounded then the minimum ratio test gives
$$\theta^* = \min_{i \in B \colon \bar{a}_{is} > 0} \frac{\bar{b}_i}{\bar{a}_{is}} = 0, \tag{4.52}$$
as $\hat{\imath} \in B$ and $\bar{b}_{\hat{\imath}} = 0$.

Thus the non-basic variable x_s enters the basis but remains at value zero, whilst the basic variable $(x_B)_{\hat{\imath}}$ leaves the basis and also remains at value zero. So, in spite of changing the basis B the values of x are unchanged. Therefore we have not actually moved to a different vertex.

4.6.3 Cycling and Bland's rule

If there is a degenerate solution the simplex method (using the minimum ratio test to move from one vertex to another) my enter an endless loop in which the the same basic and non-basic variables are swapped in and out of the basis forever. B and N would be changing but x would remain the same.

The problem arises as there are multiple choices of indexes to move out of the basis. We need a rule to choose which indexes to use, so that this rule ensures a cycle cannot occur.

Bland's rule is the following method:

- Among all $s \in \{1, \dots, n\}$ with $\bar{c}_s < 0$, choose the smallest s.
- Among all $r \in \operatorname{argmin}_{i \in B \colon \bar{a}_{is} > 0} \frac{\bar{b}_i}{\bar{a}_{is}}$, choose the smallest r.

Theorem 4.6.2. *With Bland's rule, the simplex method converges in a finite number of iterations.*

We will not show the proof here - there is some discussion in Chvátal.

4.7 Summary

The simplex method with Bland's rule starts from the linear program

$$\begin{array}{lrcl} \min & c_B A_B^{-1} b + \bar{c}_N x_N + \bar{c}_B x_B & & \\ \text{subject to} & x_B + \bar{A} x_N & = & \bar{b} \\ & x_B & \geq & 0 \\ & x_N & \geq & 0. \end{array} \tag{4.53}$$

Start the algorithm with a basis $B \subseteq \{1, \dots, n\}$ with a basic matrix A_B yielding a basic feasible solution $x_B = A_B^{-1} b \geq 0$ with $x_N = 0$.

At each step of the simplex method, then

1. Compute
   - A_B^{-1},
   - $\bar{A} = A_B^{-1} A_N$,
   - $\bar{b} = A_B^{-1} b$,
   - $\bar{c} = c - c_B A_B^{-1} A$.
2. If $|\{s \in \{1, \dots, n\} : \bar{c}_s < 0\}| > 0$, then
   - Pick the smallest index $s \in \{1, \dots, n\}$ with $\bar{c}_s < 0$;
   - Pick the smallest index r where

     $$r \in \operatorname*{argmin}_{i \in B \colon \bar{a}_{is} > 0} \frac{\bar{b}_i}{\bar{a}_{is}}. \tag{4.54}$$

     If there is no such index then the problem is unbounded;
   - Let $B \to B \setminus \{r\} \cup \{s\}$.
3. Else we have reached the optimal solution. Stop, returning $x = (x_B, x_N)$ with $x_B = A_B^{-1}$ and $x_N = 0$.

4.8 Tableau form

The simplex method as summarised above is a working algorithm. However, it is not in the most efficient form to implement on a computer. In part this is because there are many different variables to compute and manipulate. In part it is because some of the calculations (such as the explicit matrix inversion of A_B) are expensive.

We can build on a number of concepts from linear algebra to make the algorithm more efficiently implemented.

4.8.1 The problem as a block matrix

We again start from the standard basic form linear program

$$\begin{array}{lrcl} \min & c_B A_B^{-1} b + \bar{c}_N x_N + \bar{c}_B x_B & & \\ \text{subject to} & x_B + \bar{A} x_N & = & \bar{b} \\ & x_B & \geq & 0 \\ & x_N & \geq & 0. \end{array} \tag{4.55}$$

We transfer the problem to the *tableau*

$$\begin{array}{|c|c|c|} \hline -c_B A_B^{-1} b & \bar{c}_B & \bar{c}_N \\ \hline \bar{b} & I & \bar{A} \\ \hline \end{array} \tag{4.56}$$

The objective function is contained in the first row, row 0. The value in position $(0, 0)$ is the (negative of the) objective function value of the current basic solution. Rows 1 to m contain the constraints. The right hand side $\bar{b}$ is in column 0. The m rows and columns corresponding to the basic variables are those of the identity matrix I, as the constraints have been written as $x_B + \bar{A}x_N = \bar{b}$ in this form of the linear program. Each row corresponds to a specific basic variable.

To complete the simplex method we need to apply Bland's rule and the minimum ratio test and then swap our chosen basic and non-basic variables.

1. Look at row zero, all columns except column zero. This contains the reduced costs. If any entry is negative we are not optimal and must continue.

2. Using Bland's rule, set s as the index of the first column containing a negative reduced cost.

3. Using Bland's rule and the minimum rate test, set r as the row index with the minimum ratio, only considering rows i where $\bar{a}_{is} > 0$, and taking the ratio with $\bar{b}_i$, which is in column zero of the tableau.

4. Perform a *pivot* operation so that the tableau entry $\bar{a}_{rs} = 1$ and every other entry in column s, corresponding to $\bar{a}_{is}$, is set to zero. This swaps the basis and non-basis entries.

4.8.2 Pivoting

Pivoting is the operation used in Gaussian Elimination to solve linear systems. It is highly efficient allowing us to move from one basic solution to another without explicitly constructing a lot of intermediate terms. Pivoting consists of two steps. Given a matrix A and a pair of row-column indexes (r, s) with $a_{rs} \neq 0$, the steps are

1. Divide the r^{th} row by a_{rs};

2. For each row with index $i \neq r$, subtract the r^{th} row multiplied by a_{is}

The first step ensures that $a_{rs} \to 1$. The second step ensures that $a_{is} \to 0$ for all $i \neq r$.

After the pivoting operation the column with index s becomes equal to a column of the identity matrix. That means that the variable s is now basic. Because the basic variables have columns that match those in the identity matrix, the pivoting operation will only change the contents of column r. Therefore the pivoting operation swaps $(x_B)_r$ and x_s as expected.

4.9 Two-phase simplex method

We now, via the simplex method in tableau form, have an efficient algorithm for solving a linear program. However, it is not a general algorithm: it requires that the linear program is in basic form,

$$\begin{array}{lrcl} \min & c_B A_B^{-1} b + \bar{c}_N x_N + \bar{c}_B x_B & & \\ \text{subject to} & x_B + \bar{A}x_N & = & \bar{b} \\ & x_B & \geq & 0 \\ & x_N & \geq & 0, \end{array} \tag{4.57}$$

before it can start. We need to work out how to get a linear program in, for example, the standard form

$$\begin{array}{lrcl} \min & cx & & \\ \text{subject to} & Ax & = & b \\ & x & \geq & 0, \end{array} \tag{4.58}$$

into the basic form before we can use the simplex method.

The two-phase simplex method is the approach to use. This

1. iteratively constructs *some* basic feasible solution and from it the associated basic feasible tableau;

2. applies the tableau simplex method to find the optimal solution.

An important part of the first phase is that, by constructing *some* basic feasible solution it shows that the problem is feasible. If the first phase fails it means the problem is infeasible.

4.9.1 Auxiliary problem

Define the vector $e = (1, \ldots, 1)$ with m components. Starting from the linear program in standard form,

$$\begin{array}{lrcl} \min & cx & & \\ \text{subject to} & Ax & = & b \\ & x & \geq & 0, \end{array} \tag{4.59}$$

define the *auxiliary problem*

$$\begin{array}{lrcl} \min & ey & & \\ \text{subject to} & Ax + Iy & = & b \\ & x & \geq & 0, \\ & y & \geq & 0. \end{array} \tag{4.60}$$

Note that $ey = \|y\|_1$ as $y \geq 0$. This problem is trying to find the "smallest" y (in the one norm) such that $Ax + Iy = b$.

Theorem 4.9.1. *The auxiliary problem admits an optimal solution (x^*, y^*) with $ey^* = 0$ if, and only if, the original problem is feasible.*

Proof. For the implication, work by contradiction. Assume there is an optimal solution (x^*, y^*) with $ey^* > 0$. Therefore there is no solution (x', y') with $x' \geq 0$ and $y' = 0$ satisfying $Ax' + Iy' = Ax' = b$. This is because any such solution would have objective function value $ey' = 0 < ey^*$, which is a contradiction.

Finally, if the original problem admits a feasible solution x' then the pair (x', y') with $y' = 0$ is feasible for the auxiliary problem. It is also optimal as the objective function $ey' = 0$, which is the minimum as $e, y' \geq 0$. □

Note that it immediately follows from the proof that if the auxiliary problem does not admit an optimal solution the linear program is infeasible.

4.9.2 Phase 1 tableau

The auxiliary problem as constructed automatically contains an identity matrix corresponding to the columns associated with y. It immediately follows that y is basic. Therefore we can immediately write down the tableau

0	0	$1, \ldots, 1$
b	A	I

(4.61)

This is, however, not completely in the tableau form that we require for the simplex method. The issue is the reduced cost vector of the basic variables. To start the simplex method we need the reduced costs to be 0: here they are 1. This has been caused by expressing the objective function in terms of the basic variables. We instead want to express the objective function solely in terms of the non-basic variables. To do this, use

$$Ax + Iy = b \tag{4.62}$$

$$\implies \quad y = b - Ax \tag{4.63}$$

$$\implies \quad ey = eb - eAx. \tag{4.64}$$

By re-writing the objective function in this form we have the tableau

$-eb$	$-eA$	$0, \ldots, 0$
b	A	I

(4.65)

We can now apply the simplex method to the auxiliary problem.

4.9.3 Outline of two-phase method

Assume we are starting from the standard form of the linear program,

$$\begin{array}{lrcl} \min & cx & & \\ \text{subject to} & Ax & = & b \\ & x & \geq & 0, \end{array} \tag{4.66}$$

construct the tableau for the auxiliary problem as

$$\begin{array}{|c|c|c|} \hline -eb & -eA & 0,\ldots,0 \\ \hline b & A & I \\ \hline \end{array} \tag{4.67}$$

Then

1. Solve the auxiliary problem. Let (x^*, y^*) be an optimal solution.
   (a) If the optimal objective function value $ey^* > 0$ the original problem is infeasible. Stop at this point.
   (b) If $ey^* = 0$ then
      i. If there is some basic variable y_i with zero value the auxiliary problem is degenerate. Each such basic y_i can be made non-basic by pivoting on any non-zero component of its row.
      ii. Delete all columns corresponding to the basic y_i variables.
      iii. Re-write the objective function in terms of the non-basic x_j variables.
2. Rewrite the objective function of the original problem in terms of only the non-basic variables.
3. Write the basic feasible tableau for the original problem using the basic solution found in phase 1. Apply the simplex method to it.

The issue of degeneracy in the auxiliary problem needs more explanation. If there is a degeneracy for y_i then the tableau is in the form

$$\begin{array}{c|c|c|c|} & & x_1 \ldots x_j \ldots x_n & y_1 \ldots y_i \ldots y_m \\ \hline & 0 & \cdots & \cdots \\ \hline & & & 0 \\ & & & \vdots \\ y_i & 0 & \bar{a}_{i1} \ldots \bar{a}_{ij} \ldots \bar{a}_{in} & 1 \\ & & & \vdots \\ & & & 0 \\ \hline \end{array} \tag{4.68}$$

Here we have added information about which rows and columns correspond to which variables at the top and left. This is the tableau once the simplex method has been run on the auxiliary problem, and the objective function is zero meaning a feasible solution exists.

We can pivot on any $\bar{a}_{ij}$ that is non-zero. The reason is that the value changes by $\bar{b}_i/\bar{a}_{ij}$, and $\bar{b}_i = 0$. This pivot operation ensures that x_j enters the basis with value zero, and y_i leaves the basis (also with value zero).

If, however, there is no entry $\bar{a}_{ij}$ that is non-zero, then the matrix A is not full rank. In this case the entire row (and hence variable y_i) can be removed from the tableau immediately.

4.10 Efficiency and complexity

Our aim in deriving the simplex method was to avoid enumerating all the vertexes of the feasible region. The reason is that enumerating all vertexes has computational complexity $\mathcal{O}(n!)$ which is not tractable.

The simplex method works by finding one vertex and then moving to neighbouring vertexes one at a time. If the simplex method has to visit every vertex to find the optimal solution then it also has complexity $\mathcal{O}(n!)$. Unfortunately there are "pathological" cases where this can occur.

Theorem 4.10.1 (Klee and Minty, 1972)**.** *The linear program*

$$\begin{array}{lrcll} \max & \sum_{j=1}^{n} 10^{n-j} x_j & & & \\ \textit{subject to} & \left(2\sum_{j=1}^{i-1} 10^{i-j} x_j\right) + x_i & \leq & 100^{i-1}, & i \in \{1,\dots,n\} \\ & x_j & \geq & 0, & j \in \{1,\dots,n\}, \end{array} \tag{4.69}$$

requires $2^n - 1$ *iterations to solve with the simplex method.*

Therefore the worst case analysis of the simplex method shows that it is intractable. *However*, when applied to "generic" problems in practical use the simplex method is (nearly always) fast. Empirically the computational complexity of the average case is $\mathcal{O}(m \log(n))$.