1 Finding an initial basic feasible solution

Recall our discussion from last time about how to find an initial basic feasible solution of a linear program. Suppose we want to find a basic feasible solution of

\[ \min c^T x \]
\[ \text{s.t.} \quad Ax = b \]
\[ x \geq 0. \]

We modify the LP so that there is an easy choice of basic solution. We start by solving

\[ \min e^T z \]
\[ \text{s.t.} \quad Ax + Iz = b \]
\[ x \geq 0 \]
\[ z \geq 0, \]

where \( e \) is the vector of all ones, and \( b \geq 0 \) (if not, then we can multiply the constraints by \(-1\) to achieve this). The \( z \) variables are called artificial variables, and the \( x \)'s are called real variables. Define \( x' := [x \ z]^T \) and \( A' := [A \ I] \) so that the constraints of the modified LP can be written as \( A'x' = b, \ x' \geq 0. \)

Let \( B \) be the indices of the artificial variables. Then \( B \) is a basis, since the corresponding columns of \( A' \) are \( I \), the identity, and thus linearly independent. The corresponding basic feasible solution is \( x = 0, \ z = b \). We use this to initialize the simplex algorithm.

The simplex method can be one of two possible results (note that the modified LP is never unbounded: since \( z \geq 0 \), the objective function is bounded from below by 0.)

**Case (1):** The value of the LP is non-zero (and thus strictly greater than zero). Then there are no feasible solutions for the original LP, i.e., there are no \( x \) such that \( Ax = b \). Indeed, if there were, we could take \( z = 0 \) and thus obtain a new feasible solution to the modified LP with value 0, a contradiction.

**Case (2):** The value of the LP is zero. Then there are two subcases:

(i) The Good Case: All artificial variables are non-basic. Then \( A'_B = A_B \), so that \( B \) is a basis also for the original problem: \( x'_B = (A'_B)^{-1}b, \ x'_N = 0 \) is feasible, so \( x_B = A_B^{-1}b, \ x_N = 0 \) is a basic feasible solution. for \( Ax = b \).

We can now run the simplex method for the original problem, starting with the basis \( B \).

(ii) The Bad Case: Some artificial variables are in the basis.

In the bad case, we know that all the artificial variables \( z_i = 0 \). Therefore, the idea is that we should perform pivots, taking artificial variables out of basis, putting “real” variables in.
Recall: \( \bar{A}' = (A'_B)^{-1} A'_N \)

Now we again have two cases. We fix the index \( i \in B \) corresponding to artificial variables.

**Case (1):** Suppose there exists a “real” variable \( j \in N \) such that \( \bar{A}_{ij} \neq 0 \) for artificial variable \( i \in B \). Consider pivot \( \hat{B} \leftarrow B \setminus \{i\} \cup \{j\} \).

**Claim 1** Current solution \( x' \) is also a solution associated with \( \hat{B} \)

**Proof:** All we need to show is that \( x' \) satisfies \( A'x' = b \) and \( x'_k = 0 \) \( \forall k \notin \hat{B} \). For \( k \notin \hat{B} \), \( x'_k = 0 \) \( \forall k \notin \hat{B} \). For \( k \notin B \), \( x'_k = 0 \) (same as before). For \( k = i \), \( x'_i = 0 \) (since \( i \) is an artificial variable). \( \square \)

**Claim 2** \( \hat{B} \) is a basis

**Proof:** We use the same proof we used to show that a pivot leads to a new basis. We have

\[
A'_B = A'_B (A'^{-1}_B A'_B) \\
= A'_B \begin{bmatrix} 1 & 1 & (A'_j) \\ \vdots & & \end{bmatrix} \\
\uparrow \text{ith column}
\]

where \( A'_B \) is non-singular (it was a basis), and the next matrix is also non-singular (because its determinant value is \( \bar{A}_{ij} \neq 0 \) by assumption. \( \square \)

**Case (2):** Suppose for artificial variable \( i \in B \), for all real \( j \in N \), \( \bar{A}_{ij} = 0 \). Let \( \alpha_i \) be \( i^{th} \) row of \( (A'_B)^{-1} \). Then for each real \( j \in N \)

\[
\alpha_i A'_j = \bar{A}_{ij} = 0.
\]

For each real \( j \in B \)

\[
\alpha_i A'_j = 0
\]

since \( (A'_B)^{-1} A_B = I \), and \( i \neq j \) since \( j \) real and \( i \) artificial. So then, \( \alpha_i A = 0 \), which implies that the rows of \( A \) not linearly independent. Either this violates an assumption (if we assumed that \( A \) has linearly independent rows) or we can find a linearly dependent row and eliminate it. Get rid of constraints linearly dependent on others and continue.

**Definition 1** Finding an initial basic feasible solution an associate basis is called Phase I of the simplex method. Finding an optimal solution given the initial basic feasible solution is called Phase II.
2 The complexity of a pivot

We now turn to thinking about the complexity (number of arithmetic operations) needed to perform a single pivot. Assume we have a basic feasible solution $x$ and associated basis $B$. Recall the steps of a pivot:

- **Step 1:** Solve $A_B^T y = c_B$ for $y$.

- **Step 2:** Compute $\bar{c} = c - A^T y$. If $\bar{c} \geq 0$, stop. Else find $\bar{c}_j < 0$

- **Step 3:** Solve $A_B d = A_j$ for $d$. This computes column $d = \begin{pmatrix} \bar{A}_{1j} \\ \vdots \\ \bar{A}_{mj} \end{pmatrix}$ of $\bar{A} = (A_B^{-1}) A_N$.

- **Step 4:** Compute $\max \epsilon$ s.t. $\epsilon d \leq \bar{b} = x_B$

- **Step 5:** Update solution to $\hat{x}$ where $\hat{x}_j = \epsilon$. $\hat{x}_B = x_B - \epsilon d$, Basis $\hat{B} = B - \{i^*\} \cup \{j\}$

Let’s now consider the total work involved:

- Step 1 and 3: need to solve $m \times m$ system of equations. : $O(m^3)$ (this is faster if $A_B$ is sparse, lots of zeros)

- Step 4 and 5: check $O(m)$ inequalities: Check $O(m)$ inequatities or update $O(m)$ components $O(m)$ work

- In Step 2, to compute any component of $\bar{c}$ is $O(m)$ work, but there are $n$ of them. Overall, $O(mn)$ times if we look through all entries.

Therefore, the overall work involved is $O(m^3 + mn)$ per pivot.

Suppose we do one pivot step with input $x, B$, output $x', B'$. The next pivot involves $A_{\hat{B}}, c_{\hat{B}}, A_{\hat{N}}$ and $|B \cap \hat{B}| = n - 1$. So linear system solving should not be too different in next pivot.

Suppose initially $A_B = I$. (If not true, we can multiply the constraints by $A_B^{-1}$ to make it true). Suppose $B_0 = B, B_1, B_2, \cdots B_k$ be bases in a sequence of $k$ pivots.

Recall that

$$A_{B_{i+1}} = A_{B_i} \begin{bmatrix} 1 & 1 & \vdots & d \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

called an eta matrix

Let $E_i$ be $i^{th}$ eta matrix. Given that this, is the case how hard is it to solve the systems

$$A_{B_i} x = b \text{ for } x$$
\[ A_{B_1}^T y = c_{B_1} \quad \text{for } y \]
\[ A_{B_1} d = A_j \quad \text{for } d \]

We know that \( A_{B_1} = E_1 \) for \( E_1 \) an eta matrix. So \( A_{B_1} x = b \) is equivalent to

\[
\begin{bmatrix}
1 & 1 \\
1 & 1
\end{bmatrix}
\begin{bmatrix}
d \\
x
\end{bmatrix}
= \begin{bmatrix}
b
\end{bmatrix}
\]

\( j \text{th} \)

This implies

\[ x_i + d_i x_j = b_i \quad (i \neq j) \quad \text{and} \quad d_j x_j = b_j \quad (i = j). \]

Then to solve this system, set \( x_j = \frac{b_j}{d_j} \), and then \( x_i = b_i - \frac{d_i b_j}{d_j} \). Solving this then takes \( O(m) \) time.

Now consider solving \( A_{B_1}^T y = c_{B_1} \) for \( y \). Then

\[
\begin{bmatrix}
1 & 0 & \cdots & 0 & 0 \\
0 & 1 & \cdots & 0 & 0 \\
d & 0 & \cdots & 1 & 0 \\
0 & 0 & \cdots & 0 & 1
\end{bmatrix}
\begin{bmatrix}
y
\end{bmatrix}
= \begin{bmatrix}
c_{B_1}
\end{bmatrix}
\]

This implies

\[ y_i = c_i \quad i \neq j \quad \text{and} \quad \sum_{i=1}^{n} d_i y_i = c_j, \]

which we can easily solve in \( O(m) \) time.

In the general case, we want to solve equations of the form \( A_{B_k} x = b \). Note that we can solve \((E_1E_2\ldots E_k)x = b\) if we solve \((E_2\ldots E_k)x = b\). Let \( x_1 \) denote the product \( E_2 \cdot \cdots \cdot E_k x \) (where we still don’t know \( x \)). Then \( E_1 x_1 = b \). We can solve this system for \( x_1 \) in \( O(m) \) time. Now we iteratively solve \( E_2 \ldots E_k x = x_1 \) for \( x \). Thus we can solve for \( x \) in \( O(km) \) time.

Hence in general, after \( k \) pivots, we can perform a pivot in \( O(km + mn) \) time. Note that this running time gets larger after we have performed a large number of pivots, so in practice, after some number of iterations, we recompute \( A_B^{-1} \), make the current basis \( I \), and start over again.