Contents

1 Linear Programming

Recall that in general, an optimisation problem can be formulated as

$$\begin{aligned} &\min_{\boldsymbol{x}\in X} f(\boldsymbol{x}) \\ &\text{s.t. } g_i(\boldsymbol{x}) = 0 \quad \text{for } i = 1, 2, \cdots, p \\ &\qquad h_j(\boldsymbol{x}) \leq 0 \quad \text{for } j = 1, 2, \cdots, m, \end{aligned}$$

where f is known as the *objective function*, g_i's are known as *equality constraints* and h_j's are known as *inequality constraints*. The set of all $\boldsymbol{x}$ that satisfy all constraints is called the *feasible set*, where each of such $\boldsymbol{x}$ is a *feasible solution*. The vector which minimises f is known as the *optimal solution* and is denoted by $\boldsymbol{x}^*$, with $f(\boldsymbol{x}^*)$ being the *optimal value*.

The concept of a *linear program* is intuitive to understand: it is simply an optimisation problem whose objective function and constraint functions are all linear. In a 2-dimensional plane, we see that any region bounded by linear functions is a polygon. We will abstract this idea and generalise it for any finite-dimensional space.

1.1 Geometry of Linear Programming

In any Euclidean space $\mathbb{R}^n$, a linear function can be written as

$$f(x_1, x_2, \cdots, x_n) = a_0 + \sum_{i=1}^{n} a_i x_i,$$

where the a_i's are real coefficients. In matrix notations, this becomes

$$f(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + a_0$$

for some $\boldsymbol{c} \in \mathbb{R}^n$. Let $f(\boldsymbol{x}) = a_0 + b$, then we have $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = b$. Note that this level set equation gives a linear function in $\mathbb{R}^{n-1}$, because obviously $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = c_n x_n + \bar{\boldsymbol{c}}^{\mathrm{T}}\boldsymbol{x}'$, so

$$x_n = \frac{b}{c_n} - \frac{1}{c_n}\bar{\boldsymbol{c}}^{\mathrm{T}}\boldsymbol{x}'.$$

Apparently, $-\frac{1}{c_n}\bar{\boldsymbol{c}} \in \mathbb{R}^{n-1}$, so x_n is a linear function of $\boldsymbol{x}'$. This means every straight line in $\mathbb{R}^n$ is defined by an equation

$$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = b$$

for some $\boldsymbol{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Take $\boldsymbol{x}_0 \in \mathbb{R}^n$ which is on this line an let $\boldsymbol{d} \in \mathbb{R}^n$ be the direction vector of the line, then for every $\boldsymbol{x}$ with $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = b$, we also have

$$\boldsymbol{x} = \boldsymbol{x}_0 + \lambda \boldsymbol{d}$$

for some $\lambda \in R$. However, this implies that for any such $\boldsymbol{x}$, we have

$$\boldsymbol{a}^{\mathrm{T}}(\boldsymbol{x}_0 + \lambda \boldsymbol{d}) = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = b,$$

but $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_0 = b$, so we have to have $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{d} = 0$. Therefore, the set

$$A_{\perp} \coloneqq \left\{\boldsymbol{x} : \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = b\right\}$$

in fact is a set of vectors orthogonal to $\boldsymbol{a}$ in $\mathbb{R}^n$.

Definition 1.1.1 ▶ Hyperplane

Let $\boldsymbol{a} \in \mathbb{R}^n$ be a vector. For any $b \in \mathbb{R}$, the set

$$H_b \coloneqq \left\{\boldsymbol{x} : \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = b\right\}$$

is said to be a **hyperplane** with normal vector $\boldsymbol{a}$.

It is easy to see that in $\mathbb{R}^2$, a hyperplane is a straight line perpendicular to $\boldsymbol{a}$ and in $\mathbb{R}^3$, it is a plane whose normal vector is parallel to $\boldsymbol{a}$.

Intuitively, a hyperplane partitions the space $\mathbb{R}^n$ into 2 halves. Therefore, we refer to the set

$$\left\{\boldsymbol{x} : \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \leq b\right\}$$

as a *half-space*. Intuitively, if we have m half-spaces, then their intersection is a polyhedron in the space, i.e., we define the set

$$P \coloneqq \bigcap_{i=1}^{m} \left\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \leq b_i\right\}$$

as a *polyhedral set*.

Remark. Note that for P to be a polyhedron, the intersection must be finite. Otherwise, consider the counter example of the bounded set whose boundary is defined by all hyperplanes at a distance d away from a fixed point Q, which is a sphere.

We can let $\boldsymbol{a}_i^{\mathrm{T}}$ be the i-th row of the matrix $\boldsymbol{A}$ and define a column vector $\boldsymbol{b}$ whose i-th entry is b_i, then we can re-write the above intersection using matrix multiplication.

Definition 1.1.2 ▶ Polyhedron

A **polyhedron** is defined as the set

$$P \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$.

1.2 Standard Form of Linear Programs

Definition 1.2.1 ▶ Linear Programming Problem

A **linear programming** (LP) problem is an optimisation problem where the objective function f is linear and the feasible set P is a polyhedron.

Note that each linear constraint corresponds to a half-space, so we can formulate a linear programming problem as

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^n} \ & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{s.t. } & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \leq b_i \quad \text{for } i = 1, 2, \cdots, p \\ & \boldsymbol{a}_j^{\mathrm{T}}\boldsymbol{x} = b_j \quad \text{for } i = 1, 2, \cdots, m, \\ & \boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{x} \neq b_k \quad \text{for } i = 1, 2, \cdots, q. \end{aligned}$$

where $\boldsymbol{c} \in \mathbb{R}^n$ is called the *cost* or *profit* vector, $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}$, $\boldsymbol{a}_j^{\mathrm{T}}\boldsymbol{x}$ and $\boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{x}$ are called the *constraints* and $\boldsymbol{x}$ is known as *decision variables*. Note that the number of constraints must be finite, or else we may have a feasible set which is not a polyhedron.

Given any linear program, it may present one of the following possibilities:

1. The program has a unique optimal solution.
2. The program has infinitely many optimal solution (but still a unique optimal value).
3. The program has no optimal solution.

4. The program has no feasible solution.

The first 2 cases are trivial. Suppose a linear program with objective function f has no optimal solution, then it means for every feasible $\boldsymbol{x}$, we can find a different feasible $\boldsymbol{x}'$ such that $f(\boldsymbol{x}') < f(\boldsymbol{x})$.

Suppose a linear program with objective function f has no feasible solution, then this means that the feasible set is empty. In this case, we define the optimal value to be ∞. The reasoning is as follows.

Suppose f and g are objective functions of 2 optimisation problems with feasible set S_1 and S_2 respectively such that $S_1 \subseteq S_2$. Clearly, $\min f(\boldsymbol{x}) \geq \min g(\boldsymbol{x})$. Note that if $S_1 = \varnothing$, then for every S_2, we have the above inequality, which means that

$$\min_{\boldsymbol{x} \in S_1} f(\boldsymbol{x}) \geq y$$

for all $y \in \mathbb{R}$. Therefore, $\min_{\boldsymbol{x} \in S_1} f(\boldsymbol{x}) = \infty$.

For each inequality constraint in the form of $\boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} \leq b_i$, we can introduce a *slack variable* $s_i \geq 0$ such that $\boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} + s_i = b_i$. For each constraint on x_i in the form of $x_i \leq 0$, we can replace every occurrence of x_i by $-x_i^- = x_i$ such that $x_i^- \geq 0$. For each free variable x_j, we can express it as

$$x_j = x_j^+ - x_j^- \quad \text{for some } x_j^+, x_j^- \geq 0.$$

For instance, we can take $x_i^+ = 0$ and $x_i^- > 0$ whenever $x_i < 0$ and vice versa for $x_i > 0$. Note that this correspondence is not unique.

After the above transformations, we see that every constraint is equivalent to either an equality constraint $\boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} + s_i = b_i$ or an inequality constraint in the form of $x_i \geq 0$. Therefore, we define the following as the *standard form* of a linear program:

$$\begin{aligned} &\min_{\boldsymbol{x} \in \mathbb{R}^n} \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\qquad x_i \geq 0, \quad \text{for } i = 1, 2, \cdots, m. \end{aligned}$$

One should realise that a linear program in the standard form can be more easily solved by using linear algebra to find the optimal solution. Note that not every optimisation problem is given in the standard form. Fortunately, we can always convert a linear program into the standard form.

1.3 Convex Sets and Functions

Intuitively, we describe two types of shapes in natural languages: the shapes which, if you choose any of its edges, lies in the same side of that edge, and the shapes which span across both sides from some chosen edge of its.

Graphically, this means that some shapes are "convex" to all directions, where as some other shapes are "concave". We shall define this rigorously as follows:

Definition 1.3.1 ▸ Convex Set

A set $D \subseteq \mathbb{R}^n$ is said to be **convex** if for all $\boldsymbol{x}, \boldsymbol{y} \in D$ and for all $\lambda \in [0,1]$,

$$\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in D.$$

Analogously, we might want to say that a function is convex if, for any 2 points on its graph, the line segment joining the 2 points "lies within" the graph of the function. We can define convexity over functions as follows:

Definition 1.3.2 ▸ Convex Function

A function $f : D \to \mathbb{R}^n$ is said to be **convex** if for all $\boldsymbol{x}, \boldsymbol{y} \in D$ and for all $\lambda \in [0,1]$,

$$f\left(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}\right) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}).$$

Definition 1.3.3 ▸ Concave Function

A function $f : D \to \mathbb{R}^n$ is said to be **concave** if for all $\boldsymbol{x}, \boldsymbol{y} \in D$ and for all $\lambda \in [0,1]$,

$$f\left(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}\right) \geq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}).$$

From another perspective, we can see that for any convex function f, the tangent plane to the graph of f at any point will lie below the graph.

> *Remark.* A function which is not convex must be concave. However, a function which is convex may not be non-concave (consider $f(x) = x$).

Therefore, it is easy to see that all functions in the form of $f(\boldsymbol{x}) = d + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$ are both convex and concave. Such functions are said to be *affine* functions. Equivalently, this means that all affine functions are neither strictly convex nor strictly concave.

In the above definitions, the expression $\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}$ for $\lambda \in [0,1]$ is known as a *convex combination*. This notion can be generalised for any finite number of terms.

Proposition 1.3.4 ▸ Generalised Convex Combination

Let $k \in \mathbb{N}^+$ and let $f : S \to \mathbb{R}$ be a convex function on the convex set $S \subseteq \mathbb{R}^n$ and let $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_k \in S$, then

$$f\left(\sum_{i=1}^{k} \lambda_i \boldsymbol{x}_i\right) \leq \sum_{i=1}^{k} \lambda_i f(\boldsymbol{x}_i),$$

where $\sum_{i=1}^{k} \lambda_i = 1$ and $\lambda_i \geq 0$ for $i = 1, 2, \cdots, k$.

Using the idea of convex combinations, we can define the notion of a *convex hull*.

Definition 1.3.5 ▸ Convex Hull

The **convex hull** of $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$ is defined as the set of all convex combinations of the vectors, denoted by

$$\operatorname{conv}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) := \left\{ \boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x} = \sum_{i=1}^{n} \lambda_i \boldsymbol{x}_i, \lambda_i \in [0,1], \sum_{i=1}^{n} \lambda_i = 1 \right\}.$$

Note that $\operatorname{conv}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$ is the smallest convex set containing all of $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$.

Now we consider the following proposition:

Proposition 1.3.6 ▸ Maximum of Convex Functions Is Convex

Let $f_1, f_2, \cdots, f_m : \mathbb{R}^n \to \mathbb{R}$ be convex functions, then the function

$$f(\boldsymbol{x}) := \max_{i=1,2,\cdots,m} f_i(\boldsymbol{x})$$

is convex.

Proof. Take any $\boldsymbol{x} \neq \boldsymbol{y} \in \mathbb{R}^n$ and consider $\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}$ for some $\lambda \in [0,1]$. Note that for each of the f_i's, we have

$$f_i\big(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}\big) \leq \lambda f_i(\boldsymbol{x}) + (1-\lambda) f_i(\boldsymbol{y}),$$

and so

$$\max_{i=1,2,\cdots,m} f_i\big(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}\big) \leq \max_{i=1,2,\cdots,m} \left[\lambda f_i(\boldsymbol{x}) + (1-\lambda) f_i(\boldsymbol{y})\right].$$

Therefore,

$$\begin{aligned} f\big(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}\big) &= \max_{i=1,2,\cdots,m} f_i\big(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}\big) \\ &\leq \max_{i=1,2,\cdots,m} [\lambda f_i(\boldsymbol{x}) + (1-\lambda) f_i(\boldsymbol{y})] \\ &= \lambda \max_{i=1,2,\cdots,m} f_i(\boldsymbol{x}) + (1-\lambda) \max_{i=1,2,\cdots,m} f_i(\boldsymbol{y}) \\ &= \lambda f(\boldsymbol{x}) + (1+\lambda) f(\boldsymbol{y}). \end{aligned}$$

□

An immediate corollary of Proposition 1.3.6 allows us to define a piece-wise convex affine function.

Corollary 1.3.7 ▶ Piece-wise Affine Functions Are Convex

The piece-wise affine function

$$f(\boldsymbol{x}) = \max_{i=1,2,\cdots,n} \left(\boldsymbol{c}_i^{\mathrm{T}} \boldsymbol{x} + d_i\right)$$

is convex.

2 The Simplex Method

2.1 Basic Feasible Solutions

Recall that given any linear program, its feasible set is a polyhedron P. Note that in $\mathbb{R}^n$, the smallest possible number of hyperplanes intersecting at a point is n. Intuitively, any polyhedron can be completely defined by all of such "corner points" of itself. Here we provide three equivalent definitions.

Definition 2.1.1 ▸ Extreme Point

Let P be a polyhedron, a point $\boldsymbol{x}^* \in P$ is said to be an **extreme point** if whenever there are $\boldsymbol{y}, \boldsymbol{z} \in P$ with $\boldsymbol{x}^* = \lambda \boldsymbol{y} + (1-\lambda)\boldsymbol{z} = \boldsymbol{x}^*$ for some $\lambda \in (0,1)$, we have $\boldsymbol{y} = \boldsymbol{z} = \boldsymbol{x}^*$.

We can interpret the definition as follows: suppose $\boldsymbol{x}^*$ is not a corner point, then there are 2 possibilities. If $\boldsymbol{x}^*$ is an internal point, then there is some $\delta > 0$ such that the neighbourhood $V_\delta(\boldsymbol{x}^*) \subseteq P$. Therefore, we can always find $\boldsymbol{y}, \boldsymbol{z} \in V_\delta(\boldsymbol{x}^*)$ with $\boldsymbol{y} \neq \boldsymbol{z} \neq \boldsymbol{x}^*$ such that $\boldsymbol{x}^* \in (\boldsymbol{y}, \boldsymbol{z})$. Otherwise, $\boldsymbol{x}^*$ is on the boundary, i.e.,

$$\boldsymbol{x}^* \in H \coloneqq \left\{\boldsymbol{x} : \ \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \leq b\right\}.$$

Clearly, we can also find $\boldsymbol{y}, \boldsymbol{z} \in V_\delta(\boldsymbol{x}^*) \cap H$ with $\boldsymbol{y} \neq \boldsymbol{z} \neq \boldsymbol{x}^*$ such that $\boldsymbol{x}^* \in (\boldsymbol{y}, \boldsymbol{z})$.

Alternatively, we consider a hyperplane defined by $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = b$ such that $\boldsymbol{c}$ is not orthogonal to any of the boundaries of the polyhedron P. Clearly, $\boldsymbol{c}$ is the gradient vector of the affine function

$$f(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}.$$

Therefore, by translating the hyperplane in the direction of $\boldsymbol{c}$, we are able to maximise b. Note that this hyperplane is not parallel to any boundary of the polyhedron, so when b reaches the maximum, the hyperplane will have a unique intersection with the polyhedron, which must be a corner point.

Definition 2.1.2 ▸ Vertex

Let P be a polyhedron, a point $\boldsymbol{x}^* \in P$ is said to be a **vertex** if there exists some $\boldsymbol{c}$ such that $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^* > \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{y}$ for all $\boldsymbol{y} \in P - \{\boldsymbol{x}^*\}$.

Note that here we require the inequality to be strict, because otherwise $\boldsymbol{x}^*$ and $\boldsymbol{y}$ may both be internal points of some boundary hyperplane of P.

Note that any boundary of a polyhedron P is uniquely determined by a constraint $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \leq b_i$. Let $\boldsymbol{x}^* \in P$, it is clear that $\boldsymbol{x}^*$ is "on the boundary" if and only if $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} = b_i$ for some i. In such cases, we say that the corresponding constraint is *active/binding/tight* at $\boldsymbol{x}^*$.

Geometrically, a corner point of a polyhedron P in $\mathbb{R}^n$ is the intersection of n boundary hyperplanes which are pair-wise non-parallel. Let $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} = b_i$ and $\boldsymbol{a}_j^{\mathrm{T}}\boldsymbol{x} = b_j$ be any of the n hyperplanes, then it is clear that $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ must be linearly independent for the two hyperplanes to be non-parallel. This leads to the notion of *basic feasible solutions*.

Definition 2.1.3 ▶ Basic Feasible Solution

Let $P \subseteq \mathbb{R}^n$ be a polyhedron. $\boldsymbol{x}^* \in P$ is said to be a **basic feasible solution** if there are n linearly independent constraints which are active at $\boldsymbol{x}^*$.

We can generalise Definition 2.1.3 to deal with even infeasible points. First we introduce some terminologies.

Definition 2.1.4 ▶ Rank

Let $P \subseteq \mathbb{R}^n$ be a polyhedron with constraints $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \leq b_i$ for $i = 1, 2, \cdots, m$. For any $\boldsymbol{x} \in \mathbb{R}^n$, the **rank** of $\boldsymbol{x}$ is defined as

$$\operatorname{rank}(\boldsymbol{x}) \coloneqq \dim\left(\operatorname{span}\left\{\boldsymbol{a}_j : \boldsymbol{a}_j^{\mathrm{T}}\boldsymbol{x} = b_j\right\}\right).$$

Clearly, $\boldsymbol{x}^* \in \mathbb{R}^n$ is a basic feasible solution if and only if $\operatorname{rank}(\boldsymbol{x}^*) = n$ and $\boldsymbol{x}^* \in P$. Notice that not all $\boldsymbol{x}$ with rank n is feasible, so we might consider the following definition:

Definition 2.1.5 ▶ Basic Solution

Let $P \subseteq \mathbb{R}^n$ be a polyhedron. A vector $\boldsymbol{x} \in \mathbb{R}^n$ is a **basic solution** if $\operatorname{rank}(\boldsymbol{x}) = n$.

One important thing to note here is that $\operatorname{rank}(\boldsymbol{x}) = n$ does not necessarily imply that there are exactly n constraints active at $\boldsymbol{x}$. Intuitively, there can be more than n hyperplanes in $\mathbb{R}^n$ which intersect at a point $\boldsymbol{x}$. However, if $\operatorname{rank}(\boldsymbol{x}) = n$, then some hyperplanes are "redundant", i.e., their normal vectors can be expressed as linear combinations of the gradients of some n linearly independent constraints.

Definition 2.1.6 ▶ Degeneracy

A basic solution $\boldsymbol{x} \in \mathbb{R}^n$ is said to be **degenerate** if there are more than n constraints active at $\boldsymbol{x}$.

Let $P \subseteq \mathbb{R}^n$ be a polyhedron defined by m constraints. Clearly, for each basic feasible solution, we require at least n different constraints to be active. Therefore, the number of distinct basic feasible solutions in P is at most $\binom{m}{n}$. This justifies the fact that any polyhedron in $\mathbb{R}^n$ determined by less than n constraints has no basic feasible solution, and any polyhedron in a finite-dimensional space must have finitely many basic feasible solutions.

Suppose we are given a standard linear program, then we can write its feasible set as the polyhedron

$$P \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\},$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. Obviously, P has no basic feasible solution if $m < n$. Suppose $m \geq n$, we will devise a way to compute the basic feasible solutions systematically.

Theorem 2.1.7 ▸ Basic Solution Characterisation

Let

$$P \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$$

be a polyhedron for some $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with $m \geq n$. A vector $\boldsymbol{x}^* \in \mathbb{R}^n$ is a basic solution if and only if

- $\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{b}$, *and*
- *There exists an index set $B \subseteq \{1, 2, \cdots, n\}$ such that the set*

$$\{\boldsymbol{A}_i : i \in B\}$$

is linearly independent and $x_j^* = 0$ for all $j \notin B$, where

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 & \cdots & \boldsymbol{A}_{\boldsymbol{n}} \end{bmatrix}.$$

Proof. Write $B = \{B(1), B(2), \cdots, B(m)\}$ and define

$$N = \{N(1), N(2), \cdots, N(n-m)\} \coloneqq \{1, 2, \cdots, n\} - B.$$

For each $i \in N$, since $x_i^* = 0$, we have $\boldsymbol{e}_i^{\mathrm{T}} \boldsymbol{x}^* = 0$. Therefore, the matrix representation for the active constraints is

$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{e}_{N(1)}^{\mathrm{T}} \\ \boldsymbol{e}_{N(2)}^{\mathrm{T}} \\ \vdots \\ \boldsymbol{e}_{N(n-m)}^{\mathrm{T}} \end{bmatrix} \boldsymbol{x}^* = \boldsymbol{0}.$$

Re-arranging the columns, the above matrix can be re-written as

$$\begin{bmatrix} \boldsymbol{A}_B & \boldsymbol{A}_N \\ \boldsymbol{0} & \boldsymbol{I}_N \end{bmatrix} \bar{\boldsymbol{x}}^* = \boldsymbol{0},$$

where $\bar{\boldsymbol{x}}^*$ is obtained by re-arranging the rows of $\boldsymbol{x}^*$ accordingly. Note that the columns of $\boldsymbol{A}_B$ is linearly independent, so $\det(\boldsymbol{A}_B) \neq 0$. Therefore,

$$\begin{vmatrix} \boldsymbol{A}_B & \boldsymbol{A}_N \\ \boldsymbol{0} & \boldsymbol{I}_N \end{vmatrix} = \det(\boldsymbol{A}_B)\det(\boldsymbol{I}_N) \neq 0,$$

and so the matrix is invertible. Therefore, the rows of the matrix are linearly independent. This means that there are n linearly independent constraints active at $\boldsymbol{x}^*$. Therefore, $\boldsymbol{x}^*$ is a basic feasible solution.

Suppose conversely that $\boldsymbol{x}^*$ is a basic feasible solution, then clearly $\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{b}$. Since there are m equality constraints, then we must have $(n-m)$ active active inequality constraints at $\boldsymbol{x}^*$, indexed by $N = \{N(1), N(2), \cdots, N(n-m)\}$, such that the constraints are linearly independent. Therefore, the matrix

$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{e}_{N(1)}^{\mathrm{T}} \\ \boldsymbol{e}_{N(2)}^{\mathrm{T}} \\ \vdots \\ \boldsymbol{e}_{N(n-m)}^{\mathrm{T}} \end{bmatrix}$$

is invertible and that for all $i \in N$, $x_i^* = 0$. Let

$$B = \{B(1), B(2), \cdots, B(m)\} \coloneqq \{1, 2, \cdots, n\} - N$$

be an index set, then the above matrix can be re-arranged as

$$\begin{bmatrix} \boldsymbol{A}_B & \boldsymbol{A}_N \\ \boldsymbol{0} & \boldsymbol{I}_N \end{bmatrix},$$

which is invertible. Therefore, $\{\boldsymbol{A}_{B(1)}, \boldsymbol{A}_{B(2)}, \cdots, \boldsymbol{A}_{B(m)}\}$ is linearly independent. □

Note that according to Theorem 2.1.7, for every $i \in N$, $x_i = 0$. Note that $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_B & \boldsymbol{A}_N \end{bmatrix}$,

so the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is equivalent to $\boldsymbol{A}_B\boldsymbol{x}_B = \boldsymbol{b}$. Therefore, given any standard polyhedron $P := \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$, we can construct a basic solution in P using the following procedures:

1. Choose m linearly independent columns from $\boldsymbol{A}$ with B being the index set for the columns to form the set $\{\boldsymbol{A}_{B(1)}, \boldsymbol{A}_{B(2)}, \cdots, \boldsymbol{A}_{B(m)}\}$;
2. For each $i \in N := \{1, 2, \cdots, n\} - B$, set $x_i = 0$. The vector consisting of all of these zero entries is denoted by $\boldsymbol{x}_N$;
3. Solve the linear system $\boldsymbol{A}_B\boldsymbol{x}_B = \boldsymbol{b}$ to obtain $\boldsymbol{x}_B = \boldsymbol{A}_B^{-1}\boldsymbol{b}$;
4. Let x_i be the i-th entry of $\boldsymbol{x}$, then

$$x_i = \begin{cases} 0, & \text{if } i \in N \\ (\boldsymbol{x}_B)_i & \text{if } i \in B \end{cases}.$$

The $\boldsymbol{x}$ obtained this way is alternatively denoted as $\boldsymbol{x} := (\boldsymbol{x}_B, \boldsymbol{x}_N)$.

Geometrically, 2 distinct basic feasible solutions of P are *adjacent* if there is an edge on the boundary joining the 2 points. Here we give an equivalent definition algebraically.

Definition 2.1.8 ▶ Adjacency

Let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be distinct basic solutions with respect to polyhedron P. $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are said to be **adjacent** if there are exactly $(n-1)$ linearly independent constraints active at both points, or their corresponding bases only contain 1 different basic column.

Recall that not all polyhedrons have basic feasible solutions. Informally, we can see that if a polyhedron does not contain any basic feasible solution, it contains at least 2 "openings" which allows us to place a straight line into the polyhedron. This observation is summarised rigorously as follows:

Theorem 2.1.9 ▶ Conditions for the Existence of Basic Feasible Solution

Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ *and*

$$P := \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\} \neq \varnothing,$$

then the following statements are equivalent:

1. *P does not contain any straight line.*
2. *P has a basic feasible solution.*
3. *P has n linearly independent constraints.*

Proof. Suppose P has a basic feasible solution $\boldsymbol{x}^*$, then there are n linearly independent constraints active at $\boldsymbol{x}^*$. Therefore, it is trivial that P must contain at least n linearly independent constraints.

We shall prove that (3) implies (1) by considering the contrapositive statement. Suppose that P contains a straight line $\{\boldsymbol{x}_0 + \lambda \boldsymbol{d} : \lambda \in \mathbb{R}\}$ for some fixed point $\boldsymbol{x}_0 \in P$ and direction vector $\boldsymbol{d} \in \mathbb{R}^n$, then for any $\lambda \in \mathbb{R}$, we have $\boldsymbol{A}(\boldsymbol{x}_0 + \lambda \boldsymbol{d}) \leq \boldsymbol{b}$. Notice that $\boldsymbol{A}\boldsymbol{x}_0 \leq \boldsymbol{b}$, so $\boldsymbol{A}\boldsymbol{d} = \boldsymbol{0}$. However, $\boldsymbol{d} \neq \boldsymbol{0}$, so $\boldsymbol{A}$ does not contain n independent rows, which implies that P does not have n linearly independent constraints.

Suppose that P does not contain any straight line. Take some $\boldsymbol{x} \in P$ and let

$$I(\boldsymbol{x}) \coloneqq \left\{\boldsymbol{a}_i : \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} = b_i\right\}$$

be the set of gradient vectors of all active constraints at $\boldsymbol{x}$. If $I(\boldsymbol{x})$ contains n linearly independent vectors, then we are done. Otherwise, $I(\boldsymbol{x})$ does not span $\mathbb{R}^n$, so there is some $\boldsymbol{d} \in \mathbb{R}^n$ with $\boldsymbol{d} \neq 0$ such that it is normal to $\operatorname{span}\big(I(\boldsymbol{x})\big)$. Since P contains no straight lines, there exists some constraints with gradient vector $\boldsymbol{a}_j \notin I(\boldsymbol{x})$ such that we can find some $\mu \in \mathbb{R}$ such that $\boldsymbol{a}_j^{\mathrm{T}}(\boldsymbol{x} + \mu \boldsymbol{d}) = b_j$. Set $\boldsymbol{x}' = \boldsymbol{x} + \mu \boldsymbol{d}$. Note that for all $\boldsymbol{a}_i \in I(\boldsymbol{x})$, we have $\boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{d} = 0$, and so $\boldsymbol{a}_i^{\mathrm{T}}(\boldsymbol{x} + \mu \boldsymbol{d}) = b_i$. Therefore, $\boldsymbol{a}_i \in I(\boldsymbol{x}')$, and so we have $I(\boldsymbol{x}) \cup \{\boldsymbol{a}_j\} \subseteq I(\boldsymbol{x}')$. We claim that $\boldsymbol{a}_j$ linearly independent with $I(\boldsymbol{x})$. Otherwise, $\boldsymbol{a}_j^{\mathrm{T}} \boldsymbol{d} = 0$, which means that the boundary defined by $\boldsymbol{a}_j^{\mathrm{T}} \boldsymbol{x} = b_j$ is parallel to $\boldsymbol{d}$ and so if the constraint is active at $\boldsymbol{x}'$, it must also be active at $\boldsymbol{x}$, which is a contradiction. Therefore, $I(\boldsymbol{x}')$ has more linearly independent vectors than $I(\boldsymbol{x})$. Repeat this process and we will eventually obtain a set of n linearly independent gradient vectors, $I(\boldsymbol{x}^*)$, where $\boldsymbol{x}^*$ is a basic feasible solution. □

Consider the level set defined by $f(\boldsymbol{x}) = k$ for some affine function f. Suppose the level set has an intersection with some polyhedron P, then by translating the level set in the direction of $-\nabla f$, it will eventually intersect P at a basic feasible solution such that any further translation will make the intersection empty. Therefore, a reasonable guess is that an optimal solution for any linear program is closely linked to the basic feasible solutions of the feasible set.

Theorem 2.1.10 ▸ Optimality of Basic Feasible Solutions

Consider the linear program

$$\min_{\boldsymbol{x} \in P} f(\boldsymbol{x})$$

where $f(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$ and P is a polyhedron. If P has a basic feasible solution and the linear program has an optimal solution, then there exists a basic feasible solution of P which is an optimal solution to the linear program.

Proof. Let the optimal value of f be v^*, then the set of optimal solutions is

$$Q \coloneqq P \cap \left\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = v^*\right\}.$$

Note that Q is a polyhedron. By Theorem 2.1.9, since P contains a basic feasible solution, it does not contain a straight line. Therefore, $Q \subseteq P$ cannot contain a straight line and so it contains a basic feasible solution.

Let $\boldsymbol{x}^*$ be any basic feasible solution of Q. We claim that $\boldsymbol{x}^*$ is a basic feasible solution of P. Suppose on contrary that $\boldsymbol{x}^*$ is not a basic feasible solution of P, then there exists some $\boldsymbol{d} \in \mathbb{R}^n$ with $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{d} = 0$ such that $\boldsymbol{x}^* + \lambda\boldsymbol{d} \in Q$ is an internal point of P for some $\lambda \in \mathbb{R}$. However, this means that there exists some $\mu > 0$ such that $\boldsymbol{x}^* - \boldsymbol{c} \in P$. Note that $\boldsymbol{c}^{\mathrm{T}}(\boldsymbol{x}^* - \boldsymbol{c}) < v*$, which is impossible. Therefore, $\boldsymbol{x}^*$ must be a basic feasible solution of P. □

Note that this essentially implies that Q is the convex hull of basic feasible solutions of P. Therefore, any optimal solution of the original linear program can be expressed as a convex combination of basic feasible solutions of P, and so it suffices to first find all basic feasible solutions if our feasible set contains any.

2.2 The Simplex Method

Note that for any linear program, we can convert it to a standard linear program with feasible region

$$P \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}.$$

Let $\{\boldsymbol{x}_0 + \lambda\boldsymbol{d} : \lambda \in \mathbb{R}\}$ be any straight line, then clearly we can find some $\lambda \in \mathbb{R}$ such that there is some entry x_i of $\boldsymbol{x}$ with $x_i < 0$. Therefore, P does not contain any straight line, and so P always contains at least one basic feasible solution if $P \neq \varnothing$. Therefore, as long as a linear program has a finite optimal value, it has optimal solutions which are basic feasible solutions.

Therefore, to find the optimal solution of a linear program, we only need to start at any basic feasible solution and try to reach an adjacent basic feasible solution which can improve our objective value. Continue this search and eventually we will be able to collect all optimal solutions.

Given any basic feasible solution $\boldsymbol{x}$, we wish to find a direction $\boldsymbol{d}$ such that $\boldsymbol{x} + \theta\boldsymbol{d}$ is some adjacent basic feasible solution for some $\theta \in \mathbb{R}$. First, we need to ensure that $\boldsymbol{x} + \theta\boldsymbol{d}$ is still feasible.

Definition 2.2.1 ▶ Feasible Direction

Let P be a polyhedron and $\boldsymbol{x} \in P$ be a feasible point. A vector $\boldsymbol{d}$ is a **feasible direction** if $\boldsymbol{x} + \lambda\boldsymbol{d} \in P$ for some $\lambda > 0$.

Notice that $\boldsymbol{x} + \lambda\boldsymbol{d}$ is still feasible, so we must have $\boldsymbol{A}(\boldsymbol{x} + \lambda\boldsymbol{d}) = \boldsymbol{b}$. However, $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, which implies that $\boldsymbol{A}\boldsymbol{d} = \boldsymbol{0}$. We can write $\boldsymbol{d} = (\boldsymbol{d}_B, \boldsymbol{d}_N)$ with respect to $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$. We need $\boldsymbol{x} + \lambda\boldsymbol{d} \geq \boldsymbol{0}$, but $\boldsymbol{x}_N = \boldsymbol{0}$, so we must also have $\boldsymbol{d}_N \geq \boldsymbol{0}$.

Suppose $\boldsymbol{d}$ is a feasible direction and $\boldsymbol{x}$ is a basic feasible solution. Consider $\boldsymbol{x}' = \boldsymbol{x} + \theta\boldsymbol{d} \in P$ for some $\theta \in \mathbb{R}$. If $\boldsymbol{x}'$ is on an edge, then clearly there is exactly one less active constraint at $\boldsymbol{x}'$ than at $\boldsymbol{x}$.

Theorem 2.2.2 ▶ Characterisation of A Direction Connecting Basic Feasible Solutions

Let $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$ with $\boldsymbol{x}_B \geq \boldsymbol{0}$ and $\boldsymbol{x}_N = \boldsymbol{0}$ be a basic feasible solution, then a direction that connects $\boldsymbol{x}$ to an adjacent basic feasible solution is in the form of $\boldsymbol{d}^j = \left(\boldsymbol{d}_B^j, \boldsymbol{d}_N^j\right)$ for some $j \in N$, such that $\boldsymbol{d}_N^j = \boldsymbol{e}_j$ and $\boldsymbol{d}_B^j = -\boldsymbol{A}_B^{-1}\boldsymbol{A}_j$.

Proof. Note that for any feasible $\boldsymbol{x}' = \boldsymbol{x} + \theta\boldsymbol{d}^j$ which is not a basic feasible solution, we have $\boldsymbol{A}\boldsymbol{x}'_B = \boldsymbol{b}$. This means that all constraints corresponding to B are active along the edge. Therefore, the edge frees one constraint of the form $x_j \geq 0$ for some $j \in N$. This means that the index set of active constraints along the edge is given by $B \cup N - \{j\}$. Note that

$$\boldsymbol{x} + \theta\boldsymbol{d}^j = \left(\boldsymbol{x}_B + \theta\boldsymbol{d}_B^j, \boldsymbol{x}_N + \theta\boldsymbol{d}_N^j\right).$$

Notice that $\boldsymbol{x}_N = \boldsymbol{0}$ and $\boldsymbol{x}_N + \theta\boldsymbol{d}_N^j = \theta\boldsymbol{e}_j$, so $\boldsymbol{d}_N^j = \boldsymbol{e}_j$. Since $\boldsymbol{x} + \theta\boldsymbol{d}^j$ is feasible, we have

$$\boldsymbol{A}\left(\boldsymbol{x} + \theta\boldsymbol{d}^j\right) = \boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}.$$

Therefore, $\boldsymbol{A}\boldsymbol{d}^j = \boldsymbol{0}$. However, note that

$$\begin{aligned}\boldsymbol{A}\boldsymbol{d}^j &= \begin{bmatrix}\boldsymbol{A}_B & \boldsymbol{A}_N\end{bmatrix}\begin{bmatrix}\boldsymbol{d}_B^j \\ \boldsymbol{d}_N^j\end{bmatrix} \\ &= \boldsymbol{A}_B\boldsymbol{d}_B^j + \boldsymbol{A}_N\boldsymbol{e}_j \\ &= \boldsymbol{A}_B\boldsymbol{d}_B^j + \boldsymbol{A}_j.\end{aligned}$$

Since $\boldsymbol{A}_B$ has linearly independent columns, it is invertible, so $\boldsymbol{d}_B^j = -\boldsymbol{A}_B^{-1}\boldsymbol{A}_j$. □

Recall that the feasible polyhedron is actually the convex hull of all of its basic feasible solutions. Naturally, we may conjure that any feasible direction is a linear combination of all the $\boldsymbol{d}^j$'s for $j \in N$.

Proposition 2.2.3 ▸ Feasible Directions as Linear Combinations

Let $\boldsymbol{x} \coloneqq (\boldsymbol{x}_B, \boldsymbol{x}_N)$ be a basic feasible solution, then any feasible direction at $\boldsymbol{x}$ can be expressed as

$$\boldsymbol{d} = \sum_{j \in N} \lambda_j \boldsymbol{d}^j$$

for $\lambda_j \geq 0$.

Proof. Note that since $\boldsymbol{d}$ is a feasible direction, $\boldsymbol{A}\boldsymbol{d} = \boldsymbol{0}$. Since $\boldsymbol{A}\boldsymbol{d} = \boldsymbol{A}_B\boldsymbol{d}_B + \boldsymbol{A}_N\boldsymbol{d}_N$, this implies that

$$\boldsymbol{A}_B\boldsymbol{d}_B = -\boldsymbol{A}_N\boldsymbol{d}_N = -\sum_{j \in N} d_j \boldsymbol{A}_j.$$

Note that $\boldsymbol{A}_B$ is invertible, so by Theorem 2.2.2,

$$\boldsymbol{d}_B = -\sum_{j \in N} d_j \boldsymbol{A}_B^{-1}\boldsymbol{A}_j = \sum_{j \in N} d_j \boldsymbol{d}_B^j.$$

Since $\boldsymbol{d}_N^j = \boldsymbol{e}_j$, we have $\boldsymbol{d}_N = \sum_{j \in N} d_j \boldsymbol{d}_N^j$. Take $\lambda_j = d_j$, we have

$$\boldsymbol{d} = \sum_{j \in N} \lambda_j \boldsymbol{d}^j.$$

□

Simply traversing between basic feasible solutions is not very useful, because our ultimate goal is to minimise our objective function, i.e., we wish to find a direction $\boldsymbol{d}^j$ such that for some $\theta > 0$,

$$\boldsymbol{c}^{\mathrm{T}}\left(\boldsymbol{x} + \theta \boldsymbol{d}^j\right) < \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}.$$

Clearly, we require $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{d}^j < 0$. Note that

$$\begin{aligned}
\boldsymbol{c}^{\mathrm{T}}\boldsymbol{d}^j &= (\boldsymbol{c}_B, \boldsymbol{c}_N)^{\mathrm{T}} \begin{bmatrix} \boldsymbol{d}_B^j \\ \boldsymbol{d}_N^j \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{c}_B^{\mathrm{T}} & \boldsymbol{c}_N^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} -\boldsymbol{A}_B^{-1}\boldsymbol{A}_j \\ \boldsymbol{e}_j \end{bmatrix} \\
&= -\boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_j + c_j,
\end{aligned}$$

so our target is simply $c_j - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_j < 0$.

Definition 2.2.4 ▶ Reduced Cost

Let $\boldsymbol{x}$ be a basic feasible solution with respect to objective function $f(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$. Let $\boldsymbol{c} = (\boldsymbol{c}_B, \boldsymbol{c}_N)$. For each $j = 1, 2, \cdots, n$, the **reduced cost** of variable x_j is defined as

$$\bar{c}_j = c_j - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_j.$$

Remark. For any $i \in B$, note that $\boldsymbol{A}_i = \boldsymbol{A}_B\boldsymbol{e}_i$, we have $\boldsymbol{A}_B^{-1}\boldsymbol{A}_i = \boldsymbol{e}_i$. Therefore, $\overline{c_i} = 0$.

We say that a direction $\boldsymbol{d}^j$ is an *improving direction* if and only if $\bar{c}_j < 0$. Furthermore, notice that by Theorem 2.2.2, we have

$$\begin{aligned}
\bar{c}_j &= c_j - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_j \\
&= \boldsymbol{c}^{\mathrm{T}}\boldsymbol{e}_j - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_j \\
&= \boldsymbol{c}_N^{\mathrm{T}}\boldsymbol{d}_N^j - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{d}_B^j \\
&= \boldsymbol{c}^{\mathrm{T}}\boldsymbol{d}^j.
\end{aligned}$$

Combining with Proposition 2.2.3, the above gives us a quicker way to compute the reduced cost given any feasible direction at a point $\boldsymbol{x}$.

Next, we only need to determine a $\overline{\theta_j} > 0$ such that $\boldsymbol{x} + \overline{\theta_j}\boldsymbol{d}^j$ gives us another basic feasible solution. Since we already know that $\boldsymbol{A}\left(\boldsymbol{x} + \overline{\theta_j}\boldsymbol{d}^j\right) = \boldsymbol{b}$ for all $\overline{\theta_j} > 0$, a natural idea is to take $\overline{\theta_j}$ to be the greatest positive real number such that $\boldsymbol{x} + \overline{\theta_j}\boldsymbol{d}^j \geq \boldsymbol{0}$, i.e., we will move in the direction $\boldsymbol{d}^j$ until we can barely stay in the feasible polyhedron. Consider

$$\boldsymbol{x} + \overline{\theta_j}\boldsymbol{d}^j = \left(\boldsymbol{x}_B + \overline{\theta_j}\boldsymbol{d}_B^j, \boldsymbol{x}_N + \overline{\theta_j}\boldsymbol{d}_N^j\right).$$

Note that $\boldsymbol{x}_N \geq \boldsymbol{0}$ and $\overline{\theta_j}\boldsymbol{d}_N^j = \overline{\theta_j}\boldsymbol{e}_j \geq \boldsymbol{0}$, so it suffices to check that $\boldsymbol{x}_B + \overline{\theta_j}\boldsymbol{d}_B^j \geq \boldsymbol{0}$. This implies that for each $i \in B$, we have $\overline{\theta_j} \geq -\frac{x_i}{d_i^j}$. Therefore, we only need to take

$$\overline{\theta_j} = \min\left\{-\frac{x_i}{d_i^j} : i \in B, d_i^j < 0\right\}.$$

Note that here we do not consider those d_i^j's with $d_i^j > 0$ because in such cases $x_i + \overline{\theta_j}d_i^j \geq 0$ for all $\overline{\theta_j} > 0$. Now, we would verify that the new point we reach is indeed another basic feasible solution.

Proposition 2.2.5 ▸ $\boldsymbol{x} + \bar{\theta}_j \boldsymbol{d}^j$ Is A Basic Feasible Solution

If $\left\{i \in B : d_i^j < 0\right\} \neq \varnothing$, *then* $\boldsymbol{x} + \overline{\theta_j}\boldsymbol{d}^j$ *is a basic feasible solution adjacent to* $\boldsymbol{x}$.

Proof. Note that $\boldsymbol{x} + \overline{\theta_j}\boldsymbol{d}^j$ is feasible, so $\boldsymbol{A}\left(\boldsymbol{x} + \overline{\theta_j}\boldsymbol{d}^j\right) = \boldsymbol{b}$. Note that by the definition of $\overline{\theta_j}$, we have

$$\overline{\theta_j} = -\frac{x_\ell}{d_\ell^j}$$

for some $\ell \in B$. Therefore, we have $\left(\boldsymbol{x} + \overline{\theta_j}\boldsymbol{d}^j\right)_\ell = 0$. Notice that at $\boldsymbol{x} \coloneqq (\boldsymbol{x}_B, \boldsymbol{x}_N)$, we have $\ell \in B$ and $j \in N$. Consider $\overline{B} \coloneqq (B - \{\ell\}) \cup \{j\}$ and $\overline{N} \coloneqq (N - \{j\}) \cup \{\ell\}$. One may check that $\overline{B}$ is linearly independent, so by Theorem 2.1.7, $\boldsymbol{x} + \overline{\theta_j}\boldsymbol{d}^j$ is a basic feasible solution. □

By now, we already have devised a systematic method to reach another basic feasible solution from a given basic feasible solution such that our objective value improves. The next goal is to determine a condition by which we should terminate our search and declare the current basic feasible solution to be optimal.

Before we do that, we need to first deal with an edge case where some basic feasible solution we find might be degenerate. Note that in the standard form, we have exactly m equality constraints represented by $\boldsymbol{A}_B\boldsymbol{x}_B = \boldsymbol{b}$ which are always active, and at least $(n - m)$ active inequality constraints in the form of $\boldsymbol{x}_N = \boldsymbol{0}$. However, note that $\boldsymbol{x} \geq \boldsymbol{0}$ represents n inequality constraints, so in total we actually have $(m + n)$ constraints. An implication here is that if any entry of $\boldsymbol{x}_B$ is 0, then we will have more than n active constraints at $\boldsymbol{x}$.

Definition 2.2.6 ▸ Degeneracy in Standard Form

Let $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$ be a basic feasible solution. We say that $\boldsymbol{x}$ is **degenerate** if there is some entry of $\boldsymbol{x}_B$ being 0.

Remark. This also concludes that $\boldsymbol{x}$ is non-degenerate if $\boldsymbol{x}_B = \boldsymbol{A}_B^{-1}\boldsymbol{b} > \boldsymbol{0}$.

Recall that along an improving direction $\boldsymbol{d}^j$, the reduced cost for x_j is $\bar{c}_j < 0$. A natural observation here is that if we cannot find any improving direction at a point $\boldsymbol{x}^*$, then this point must be optimal. Therefore, we are tempted to conclude that $\bar{c}_j \geq 0$ for all $j \in N$.

Theorem 2.2.7 ▸ Optimality Conditions for Simplex Method

Let $\boldsymbol{x}^* = (\boldsymbol{x}_B, \boldsymbol{x}_N)$ *be a basic feasible solution to some standard linear program. Let* $\bar{\boldsymbol{c}}$ *be the vector of reduced costs associated with* $\boldsymbol{x}^*$, *then*

1. *If* $\bar{\boldsymbol{c}} \geq \boldsymbol{0}$, *then* $\boldsymbol{x}^*$ *is optimal;*

2. *If $\boldsymbol{x}^*$ is optimal and non-degenerate, then $\bar{\boldsymbol{c}} \geq \boldsymbol{0}$.*

Proof. Suppose that $\bar{\boldsymbol{c}} \geq \boldsymbol{0}$. Let $\boldsymbol{y}$ be any feasible solution, then $\boldsymbol{y} - \boldsymbol{x}^*$ is a feasible direction. By Proposition 2.2.3,

$$\boldsymbol{y} - \boldsymbol{x}^* = \sum_{j \in N} \lambda_j \boldsymbol{d}^j,$$

where $\lambda_j \geq 0$ for all $j \in N$. Therefore,

$$\begin{aligned} \boldsymbol{c}^{\mathrm{T}} \boldsymbol{y} &= \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x}^* + \sum_{j \in N} \lambda_j \boldsymbol{c}^{\mathrm{T}} \boldsymbol{d}^j \\ &= \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x}^* + \sum_{j \in N} \lambda_j \bar{c}_j \\ &\geq \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x}^*. \end{aligned}$$

Therefore, $\boldsymbol{x}^*$ is an optimal solution.

Suppose that $\boldsymbol{x}^*$ is a non-degenerate optimal solution. For each $j \in N$, consider the adjacent basic feasible solution $\boldsymbol{x} + \overline{\theta_j} \boldsymbol{d}^j$. Notice that

$$\boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} \leq \boldsymbol{c}^{\mathrm{T}} \left(\boldsymbol{x} + \overline{\theta_j} \boldsymbol{d}^j \right) = \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} + \overline{\theta_j} \bar{c}_j$$

for all $j \in N$. Therefore, we must have $\overline{\theta_j} \bar{c}_j \geq 0$ for all $j \in N$. Since $\boldsymbol{x}$ is non-degenerate, for each $j \in N$ we have

$$\overline{\theta_j} = -\frac{x_\ell}{d_\ell^j} > 0,$$

and so $\bar{c}_j \geq 0$. Note that for all $i \in B$, we have $\overline{c_i} = 0$, so $\bar{\boldsymbol{c}} \geq \boldsymbol{0}$. □

Recall that $\bar{\boldsymbol{c}}_B = \boldsymbol{0}$, so it suffices to check that $\bar{\boldsymbol{c}}_N \geq \boldsymbol{0}$ for the $(n - m)$ inequality constraints to determine whether $\boldsymbol{x}^*$ is an optimal solution.

With the above preliminary works done, we are now able to develop the algorithm for simplex method.

Technique 2.2.8 ▸ Simplex Method

Let $f : \mathbb{R}^n \to \mathbb{R}$ defined by $f(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x}$ be the objective function of a standard linear program with feasible set

$$P \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}.$$

The simplex method finds the optimal solution using the following procedures:

1. Initialise $\boldsymbol{x}_0$ to be any basic feasible solution.
2. At the k-th iteration, choose an index set B_k such that the columns of $\boldsymbol{A}_{B_k}$ are a basis for the column space of $\boldsymbol{A}$.
3. Let $N_k := \{1, 2, \cdots, n\} - B_k$. For each $j \in N_k$, compute the reduced cost

$$\bar{c}_j = c_j - \boldsymbol{c}_{B_k}^{\mathrm{T}} \boldsymbol{A}_{B_k}^{-1} \boldsymbol{A}_j.$$

4. If $\bar{c}_j \geq 0$ for all $j \in N_k$:
   - $\boldsymbol{x}_k$ is an optimal solution.
5. Otherwise:
   (a) Take some $j \in N_k$ such that $\bar{c}_j < 0$. $(\boldsymbol{x}_k)_j$ is called an **entering variable**.
   (b) Compute $\boldsymbol{d}_{B_k}^j = -\boldsymbol{A}_{B_k}^{-1} \boldsymbol{A}_j$.
   (c) If $\boldsymbol{d}_{B_k}^j \geq \boldsymbol{0}$:
      - the problem is **unbounded**.
   (d) Otherwise:
      i. Let $\ell \in B_k$ be such that $\overline{\theta_j} = \min\left\{-\frac{x_i}{d_i^j} : i \in B, d_i^j < 0\right\} = \frac{x_\ell}{d_\ell^j}$. $(\boldsymbol{x}_k)_\ell$ is called a **leaving variable**.
      ii. Update $B_{k+1} := (B - \{\ell\}) \cup \{j\}$.
      iii. Update $\boldsymbol{x}_{k+1}$ by

$$(\boldsymbol{x}_{k+1})_i = \begin{cases} (\boldsymbol{x}_k)_i + \overline{\theta_j} \boldsymbol{d}_i^j & \text{if } i \in B - \{\ell\} \\ \overline{\theta_j} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

2.3 Tableau Implementation

A common way to run simplex method is by tabulating the relevant variables, solutions and reduced costs so that we can keep track of their values conveniently. A generalised tableau looks like the following:

Basic	$\boldsymbol{x}$	Solution
$\bar{\boldsymbol{c}}$	$\boldsymbol{c}^T - \boldsymbol{c}_B^{\mathrm{T}} \boldsymbol{A}_B^{-1} \boldsymbol{A}$	$-\boldsymbol{c}_B^{\mathrm{T}} \boldsymbol{A}_B^{-1} \boldsymbol{b}$
$\boldsymbol{x}_B$	$\boldsymbol{A}_B^{-1} \boldsymbol{A}$	$\boldsymbol{A}_B^{-1} \boldsymbol{b}$

If we expand the tableau with more detailed information, we get the following representation:

Basic	x_1	$\cdots$	x_n	Solution
$\bar{\boldsymbol{c}}$	$\bar{c}_1$	$\cdots$	$\bar{c}_n$	$-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}_B$
$x_{B(1)}$				
$\vdots$				
$x_{B(i)}$	$\boldsymbol{A}_B^{-1}\boldsymbol{A}_1$	$\cdots$	$\boldsymbol{A}_B^{-1}\boldsymbol{A}_n$	$\boldsymbol{A}_B^{-1}\boldsymbol{b}$
$\vdots$				
$x_{B(m)}$				

Notice that $\boldsymbol{A}_B^{-1}\boldsymbol{A}_B = \boldsymbol{I}$, we may wish to re-arrange the columns of the tableau into the following form:

Basic	$\boldsymbol{x}_B$	$\boldsymbol{x}_N$	Solution
$\bar{\boldsymbol{c}}$	$\boldsymbol{0}$	$\boldsymbol{c}_N^T - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_N$	$-\boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{b}$
$\boldsymbol{x}_B$	$\boldsymbol{I}$	$\boldsymbol{A}_B^{-1}\boldsymbol{A}_N$	$\boldsymbol{A}_B^{-1}\boldsymbol{b}$

Note that at each iteration, we need to "swap" the ℓ-th and j-th columns to update B. Essentially, suppose $\ell = B(i)$, then this is equivalent to performing row operations on the matrix

$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{c}_N^T - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_N & -\boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{b} \\ \boldsymbol{I} & \boldsymbol{A}_B^{-1}\boldsymbol{A}_N & \boldsymbol{A}_B^{-1}\boldsymbol{b} \end{bmatrix}$$

such that the j-th column becomes $\boldsymbol{e}_i$. Then, we will swap the row label $\boldsymbol{x}_\ell$ with the column label $\boldsymbol{x}_j$. We repeat this process until $\bar{\boldsymbol{c}} \geq \boldsymbol{0}$, where we will extract $\boldsymbol{A}_B^{-1}\boldsymbol{b}$ from the table as our $\boldsymbol{x}_B$.

2.4 Finding An Initial Basic Feasible Solution

Recall that to start the simplex method algorithm, we need to initialise $\boldsymbol{x}_0$ to be some basic feasible solution in the feasible region. However, this may not be straight-forward. Consider the standard linear program

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

for some $\boldsymbol{A} \in \mathbb{R}^{m\times n}$.

We can choose the index set B such that $\boldsymbol{A}_B$ is invertible and so $\boldsymbol{A}^{-1}\boldsymbol{b}$ is a basic solution. However, we might not be so lucky that this basic solution is feasible, i.e., it is not easy to

ensure that $\boldsymbol{A}^{-1}\boldsymbol{b} \geq \boldsymbol{0}$. To address this issue, we consider an *auxiliary linear program* as follows:

First, note that we can change $\boldsymbol{A}$ by multiplying some of its rows by -1 so that $\boldsymbol{b} \geq \boldsymbol{0}$. Then, we will introduce a new variable $\boldsymbol{y} \in \mathbb{R}^m$ and consider the linear program

$$\begin{aligned} \min_{\boldsymbol{y}\in\mathbb{R}^m} & \sum_{i=1}^{m} y_i \\ \text{s.t. } & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \\ & \boldsymbol{y} \geq \boldsymbol{0}. \end{aligned}$$

Now, it is obvious that $\boldsymbol{x} = \boldsymbol{0}$ and $\boldsymbol{y} = \boldsymbol{b}$ is a basic feasible solution to the problem. Solving this auxiliary minimisation problem will try to force $\boldsymbol{y} = \boldsymbol{0}$. If this is possible, i.e., the optimal value of the auxiliary program is 0, then it is clear that we have found some $\boldsymbol{x} \geq \boldsymbol{0}$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, which is exactly a basic feasible solution to our original program!

Technique 2.4.1 ▸ Two-Phase Method

Consider a standard linear program

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n} & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{s.t. } & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0}. \end{aligned}$$

The two-phase method solves the linear program with the following procedures:

1. Multiplying some rows of $\boldsymbol{A}$ by -1 wherever necessary such that $\boldsymbol{b} \geq \boldsymbol{0}$.
2. Construct the auxiliary linear program

$$\begin{aligned} \min_{\boldsymbol{y}\in\mathbb{R}^m} & \sum_{i=1}^{m} y_i \\ \text{s.t. } & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \\ & \boldsymbol{y} \geq \boldsymbol{0}. \end{aligned}$$

3. Run simplex method on the auxiliary linear program to obtain its optimal solution $(\boldsymbol{y}^*, \boldsymbol{x}^*)$ and optimal value v^*.
4. If $v^* > 0$:
   - The original problem has empty feasible region.

5. Otherwise, $v^* = 0$:
   - Run simplex method on the original problem with $\boldsymbol{x}_0 = \boldsymbol{x}^*$.

Note that in the two-phase method, we need to solve 2 linear programs. This can be computationally expensive, so we wish to find a way to combine the two phases. Here we use the idea of a penalty term. Consider the linear program

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{y}\in\mathbb{R}^m} & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + M\sum_{i=1}^{m} y_i \\ \text{s.t. } & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \\ & \boldsymbol{y} \geq \boldsymbol{0} \end{aligned}$$

where $M > 0$ is an arbitrarily large constant. We can see that if this augmented objective function has a finite optimal value, then it will also force $\boldsymbol{y} = \boldsymbol{0}$. The corresponding $\boldsymbol{x}$ will be the optimal solution of the original problem.

Technique 2.4.2 ▸ Big-M Method

Consider a standard linear program

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n} & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{s.t. } & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0}. \end{aligned}$$

The big-M method solves the linear program with the following procedures:

1. Multiplying some rows of $\boldsymbol{A}$ by -1 wherever necessary such that $\boldsymbol{b} \geq \boldsymbol{0}$.
2. Augment the original problem with some arbitrarily large constant $M > 0$ into

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{y}\in\mathbb{R}^m} & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + M\sum_{i=1}^{m} y_i \\ \text{s.t. } & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \\ & \boldsymbol{y} \geq \boldsymbol{0}. \end{aligned}$$

3. Run simplex method on the augmented linear program.
4. If an optimal solution $(\boldsymbol{y}^*, \boldsymbol{x}^*)$ can be found with $\boldsymbol{y}^* = \boldsymbol{0}$:
   - $\boldsymbol{x}^*$ is an optimal solution to the original problem.
5. Otherwise:

- The original problem has empty feasible set or is unbounded.

2.5 Special Cases

Next, we discuss a few special cases when using the simplex method. First, recall that in Definition 2.1.6, we know that a feasible basic solution $\boldsymbol{x}$ is degenerate if there is some redundant active constraint. However, note that this implies that more than one basic variables can leave. Algebraically, this means that there are different $\ell_1, \ell_2 \in B$ such that

$$\overline{\theta_j} = \min\left\{-\frac{x_i}{d_i^j} : i \in B, d_i^j < 0\right\} = -\frac{x_{\ell_1}}{d_{\ell_1}^j} = -\frac{x_{\ell_2}}{d_{\ell_2}^j}.$$

Without loss of generality, suppose we take x_{ℓ_1} as the leaving variable and try to update x_{ℓ_2} to x'_{ℓ_2}. By Proposition 2.2.5, we have

$$x'_{\ell_2} = x_{\ell_2} + \overline{\theta_j} d_{\ell_2}^j = x_{\ell_2} - \frac{x_{\ell_1}}{d_{\ell_1}^j} d_{\ell_2}^j = x_{\ell_2} - \frac{x_{\ell_2}}{d_{\ell_2}^j} d_{\ell_2}^j = 0,$$

so x_{ℓ_2} is indeed degenerate by Definition 2.2.6.

Note also that in some cases, we may have more than one optimal solution for the linear program. Intuitively, this occurs when at an optimal basic feasible solution, we can still find some feasible direction along which the objective value does not deteriorate. Therefore, **an alternative optimal solution can be found if and only if some reduced cost** $\bar{c}_j = 0$ at an optimal solution. There are 2 specific cases:

1. If we can find optimal solutions $\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^k$, then the set of optimal solutions is bounded and can be constructed as

$$\operatorname{conv}\{\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^k\}.$$

2. If at an optimal solution $\boldsymbol{x}^*$, there is $\bar{c}_j = 0$ but $\boldsymbol{d}^j \geq \boldsymbol{0}$, it means that $\boldsymbol{x}^* + \theta \boldsymbol{d}^j \geq \boldsymbol{0}$ is feasible for any $\theta \geq 0$, so the set of optimal solutions given by

$$\{\boldsymbol{x}^* + \theta \boldsymbol{d}^j : \theta \geq 0\}$$

is unbounded.

Similar to case 2 above, if at any basic feasible solution $\boldsymbol{x}^*$, there is some $j \in N$ such that $\bar{c}_j < 0$ and $\boldsymbol{d}^j \geq \boldsymbol{0}$, then any $\boldsymbol{x}^* + \theta \boldsymbol{d}^j$ where $\theta > 0$ is feasible and improves the objective value. This implies that the optimal value can be **improved indefinitely** and so the

program is unbounded.

Lastly, suppose we have some linear program with an empty feasible set. We have seen in Section 2.4 that we can use either the two-phase method or the big-M method to detect this, because if the feasible set were to be empty, we will not be able to find an optimal solution for the auxiliary problem such that $\boldsymbol{y}^* = \mathbf{0}$.

3

Duality Theory

3.1 The Dual Problem

A *dual problem* can be seen as an alternative formulation of some linear program which is known as the *primal problem*. The motivation for the dual problem comes from the following observation: suppose we have a well-ordered set S and we wish to find $\inf S$. If it is difficult to minimise S directly, we may first consider a "relaxed" lower bound t, i.e., fix some t such that $t \leq s$ for all $s \in S$.

Then, by collecting all such lower bounds t into a set denoted as T, we know that for all $t \in T$, $t \leq s$ for all $s \in S$. Now, by **maximising** t, we obtain a **greatest possible lower bound** for S. If we can formulate the set T such that $S \cap T \neq \varnothing$, i.e., there is no x such that $\sup T < x < \inf S$, then this greatest lower bound is exactly $\inf S$. Consider the linear program

$$\begin{aligned} &\min_{\boldsymbol{x} \in \mathbb{R}^n} \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\qquad \boldsymbol{x} \geq \boldsymbol{0}. \end{aligned}$$

The constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ can be re-written as $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$. Now, we augment a "penalty" term $\boldsymbol{p}^{\mathrm{T}}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})$ to the original objective function for some $\boldsymbol{p} \in \mathbb{R}^m$.

Proposition 3.1.1 ▸ Lower Bound for Optimal Value

Let $\boldsymbol{x}^*$ be an optimal solution to the linear program

$$\begin{aligned} &\min_{\boldsymbol{x} \in \mathbb{R}^n} \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} \\ &\textit{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\qquad \boldsymbol{x} \geq \boldsymbol{0}. \end{aligned}$$

Define

$$g(\boldsymbol{p}) \coloneqq \min_{\boldsymbol{x} \geq \boldsymbol{0}} \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} + \boldsymbol{p}^{\mathrm{T}}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}),$$

then $g(\boldsymbol{p}) \leq \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x}^*$.

Proof. Since $\boldsymbol{x}^* \geq \mathbf{0}$, we have

$$g(\boldsymbol{p}) = \min_{\boldsymbol{x}\geq \mathbf{0}} \left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{p}^{\mathrm{T}}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})\right) \leq \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^* + \boldsymbol{p}^{\mathrm{T}}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*).$$

Notice that $\boldsymbol{x}^*$ is feasible to the original problem, so $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^* = \mathbf{0}$. Therefore,

$$g(\boldsymbol{p}) \leq \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^* + \boldsymbol{p}^{\mathrm{T}}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^*.$$

□

Proposition 3.1.1 demonstrates a construction for a function g with respect to every linear program (P) such that g always bounds the objective value of (P) below.

Definition 3.1.2 ▸ Dual Problem

For any linear program

$$\begin{aligned} (P) \quad & \min_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ & \text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \quad \boldsymbol{x} \geq \mathbf{0}, \end{aligned}$$

its **dual problem** is defined as

$$(D) \quad \max_{\boldsymbol{p}\in\mathbb{R}^m} \min_{\boldsymbol{x}\geq \mathbf{0}} \left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{p}^{\mathrm{T}}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})\right),$$

and (P) is known as the **primal problem**.

Remark. We know that $g(\boldsymbol{p}) \coloneqq \min_{\boldsymbol{x}\geq \mathbf{0}} \left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{p}^{\mathrm{T}}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})\right)$ is an lower bound for the objective value of (P), so (D) will try to search for the greatest lower bound of the optimal value of (P).

Let us study this lower bound function $g(\boldsymbol{p})$ more closely. Observe that

$$g(\boldsymbol{p}) = \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} + \min_{\boldsymbol{x}\geq \mathbf{0}} \left(\boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}\right)\boldsymbol{x}.$$

Now we consider 2 cases. If $\boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A} \geq \mathbf{0}$, then it is clear that $\min_{\boldsymbol{x}\geq \mathbf{0}} \left(\boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}\right)\boldsymbol{x} = \mathbf{0}$ by taking $\boldsymbol{x} = \mathbf{0}$. Otherwise, $(\boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A})_j < 0$ for some $j \in \mathbb{Z}^+$, then $\left(\boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}\right)\boldsymbol{x}$ is unbounded because clearly for any feasible $\boldsymbol{x}$, we have $\left(\boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}\right)(\boldsymbol{x} + \boldsymbol{e}_j) < \left(\boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}\right)\boldsymbol{x}$ and $\boldsymbol{x} + \boldsymbol{e}_j$ is obviously still feasible.

Therefore, we can essentially reduce $g(\boldsymbol{p})$ to

$$g(\boldsymbol{p}) = \begin{cases} \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} & \text{if } \boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A} \geq \boldsymbol{0} \\ -\infty & \text{otherwise} \end{cases}.$$

Clearly, in order for us to be able to maximise $g(\boldsymbol{p})$, we need $\boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A} \geq \boldsymbol{0}$ to be satisfied by the dual problem. This means that we can have the following systematic construction for the dual problem:

Technique 3.1.3 ▸ Formulation of the Dual Problem

Intuitively, if we try to formulate the dual problem of the dual problem, then we should just return to the primal problem.

Proposition 3.1.4 ▸ Dual of the Dual Is the Primal

Let (P) be a primal problem with the dual problem (D). If (D') is the dual problem of (D), then (D') is equivalent to (D).

3.2 Duality Theorems

We see that a valid dual problem is such that $\boldsymbol{p}^{\mathrm{T}}\boldsymbol{A} \leq \boldsymbol{c}^{\mathrm{T}}$.

Theorem 3.2.1 ▸ Weak Duality

Let

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

be a primal problem and consider its dual

$$\begin{aligned} &\max_{\boldsymbol{p}\in\mathbb{R}^m} \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} \\ &\text{s.t. } \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A} \leq \boldsymbol{c}^{\mathrm{T}}, \end{aligned}$$

then

$$\sup \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} \leq \inf \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}.$$

Proof. Let $\boldsymbol{p}$ and $\boldsymbol{x}$ be feasible, then $\boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} = \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$, so $\sup \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} \leq \inf \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$. □

The above theorem justifies that the maximum of the dual objective is indeed an lower bound for the primal optimal value. We wish the primal and the dual problems to have an equal optimal value, so that solving one problem is equivalent to the other.

Corollary 3.2.2 ▶ Necessary Condition for Equal Optimal Values

Let $\boldsymbol{x}^*$ and $\boldsymbol{p}^*$ be feasible solutions to a primal problem (P) and its dual problem (D). If $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^* = (\boldsymbol{p}^*)^{\mathrm{T}}\boldsymbol{b}$, then $\boldsymbol{x}^*$ and $\boldsymbol{p}^*$ are the optimal solutions.

Proof. By Theorem 3.2.1,

$$(\boldsymbol{p}^*)^{\mathrm{T}}\boldsymbol{b} \leq \sup \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} \leq \inf \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \leq \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^*.$$

Since $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^* = (\boldsymbol{p}^*)^{\mathrm{T}}\boldsymbol{b}$, this implies that

$$(\boldsymbol{p}^*)^{\mathrm{T}}\boldsymbol{b} = \sup \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} = \inf \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^*.$$

Therefore, $\boldsymbol{x}^*$ and $\boldsymbol{p}^*$ are the optimal solutions for the primal and dual problems respectively. □

Another corollary helps us determine the boundedness and feasibility of a problem.

Corollary 3.2.3 ▶ Primal Is Unbounded If and Only If Dual Is Infeasible

If a primal problem (P) is unbounded, then its dual problem (D) is infeasible.

Proof. Suppose (P) is unbounded, then for every feasible $\boldsymbol{x}$, there exists some $\boldsymbol{x}'$ which is feasible such that $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}' < \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$. Suppose on contrary that there is a feasible solution $\boldsymbol{p}$ for (D), then by Theorem 3.2.1, $\boldsymbol{p}^T\boldsymbol{b} < \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$ for all feasible $\boldsymbol{x}$. However, this means that $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$ has an infimum, which is a contradiction. □

The opposite statement of the above corollary also holds, i.e., the primal is infeasible if its dual is unbounded.

Recall that in general, the duality gap between the primal and dual optimal values may not be 0, but in fact, all linear programs also gets for free *strong duality*.

Theorem 3.2.4 ▸ Strong Duality

If an linear program has an optimal solution, then so does its dual. Both the primal and the dual problems have the same optimal value.

Proof. Suppose that (P) is a primal linear program with an optimal solution $\boldsymbol{x}^*$, then there is some basis B such that $\boldsymbol{x}_B^* = \boldsymbol{A}_B^{-1}\boldsymbol{b}$. Since B is an optimal basis, we have

$$\bar{\boldsymbol{c}}^{\mathrm{T}} = \boldsymbol{c}^{\mathrm{T}} - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A} \geq \boldsymbol{0}.$$

Take $\boldsymbol{p}^{\mathrm{T}} = \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}$, then $\boldsymbol{p}^{\mathrm{T}}\boldsymbol{A} \leq \boldsymbol{c}^{\mathrm{T}}$ and so $\boldsymbol{p}$ is feasible to the dual problem (D). Note that $\boldsymbol{x}_N = \boldsymbol{0}$, so

$$\boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} = \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{b} = \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{x}_B^* = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^*.$$

By Corollary 3.2.2, $\boldsymbol{p}$ is an optimal solution for (D), and so the dual optimal value is just $\boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^*$, which is the primal optimal value. □

We would like to further characterise the primal and dual optimal solutions to a linear program. For that, we introduce the notion of *complementary slackness*.

Theorem 3.2.5 ▸ Complementary Slackness Conditions

Let (P) be a primal linear program with objective function $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$, constraints $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \leq b_i$, $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \geq b_i$ or $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} = b_i$. Let $\boldsymbol{x}$ and $\boldsymbol{p}$ be feasible solutions to (P) and the dual problem (D) respectively, then $\boldsymbol{x}$ and $\boldsymbol{p}$ are optimal if and only if

$$p_i\left(\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} - b_i\right) = 0, \qquad \left(c_j - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}_j\right)x_j = 0$$

for all i, j.

Proof. Consider

$$\begin{aligned}\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} &= \boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} \\ &= \left(\boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}\right)\boldsymbol{x} + \boldsymbol{p}^{\mathrm{T}}\left(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\right) \\ &= \sum_j \left(c_j - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}_j\right)x_j + \sum_i p_i\left(\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} - b_i\right).\end{aligned}$$

Note that $\left(c_j - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}_j\right)x_j, p_i\left(\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} - b_i\right) \geq 0$ for all i, j. If $\boldsymbol{x}$ and $\boldsymbol{p}$ are optimal, by Theorem 3.2.4, $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} = 0$ and so $\left(c_j - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}_j\right)x_j = p_i\left(\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} - b_i\right) = 0$. If $\left(c_j - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{A}_j\right)x_j = p_i\left(\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} - b_i\right) = 0$, then $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b} = 0$ and so by Theorem 3.2.1, $\boldsymbol{x}$ and $\boldsymbol{p}$ are optimal soultions. □

Complementary slackness enables us to better characterise a primal optimal solution.

Proposition 3.2.6 ▶ Characterisation of Primal Optimal Solutions

Let $\boldsymbol{x}$ be a feasible solution to a primal linear program (P), then $\boldsymbol{x}$ is optimal if and only if there is a feasible dual solution $\boldsymbol{p}$ satisfying complementary slackness conditions.

Proof. Suppose that $\boldsymbol{x}$ is optimal, then by Theorem 3.2.4, there exists an optimal solution $\boldsymbol{p}^*$ to the dual problem (D). By Theorem 3.2.5, $\boldsymbol{p}^*$ satisfies complementary slackness. The converse is just Theorem 3.2.5. □

3.3 Dual Simplex Method

In the previous section, we see that a dual problem helps us better characterise the optimal solutions for the primal problem. Consider a primal linear program (P) with its dual (D) in standard form. For any basis index set B, we have

$$\boldsymbol{x} = \left(\boldsymbol{A}_B^{-1}\boldsymbol{b}, \boldsymbol{0}\right), \qquad \boldsymbol{p}^{\mathrm{T}} = \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}.$$

Note that although $\boldsymbol{x}$ and $\boldsymbol{p}$ may not be both feasible, we still have

$$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{b} = \boldsymbol{p}^{\mathrm{T}}\boldsymbol{b}.$$

By Corollary 3.2.2, $\boldsymbol{x}$ and $\boldsymbol{p}$ will be the optimal solutions if they are both feasible. This gives us inspirations to find the optimal solutions of a primal-dual pair, by either

1. maintain feasibility of $\boldsymbol{x}$ and work towards feasibility of $\boldsymbol{p}$, or
2. maintain feasibility of $\boldsymbol{p}$ and work towards feasibility of $\boldsymbol{x}$.

The two approaches are known as P-Algorithm and D-Algorithm respectively.

To maintain feasibility of $\boldsymbol{x}$, it is equivalent to maintaining $\boldsymbol{A}_B^{-1}\boldsymbol{b} \geq \boldsymbol{0}$. Now, for $\boldsymbol{p}$ to be feasible, we need $\boldsymbol{p}^{\mathrm{T}}\boldsymbol{A} \leq \boldsymbol{c}^{\mathrm{T}}$, which implies that

$$\bar{\boldsymbol{c}} = \boldsymbol{c}^T - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A} \geq \boldsymbol{0}.$$

This gives the implementation for the *primal algorithm*, which is exactly Technique 2.2.8.

Conversely, by maintaining $\bar{\boldsymbol{c}} \geq \boldsymbol{0}$, we only need to achieve $\boldsymbol{A}_B^{-1}\boldsymbol{b} \geq \boldsymbol{0}$ to make $\boldsymbol{x}$ feasible, because $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ always holds. This leads to the *dual simplex method*.

Technique 3.3.1 ▸ Dual Simplex Method

Let $f : \mathbb{R}^n \to \mathbb{R}$ defined by $f(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$ be the objective function of a standard linear program with feasible set

$$P \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}.$$

The dual simplex method finds the optimal solution using the following procedures:

1. Manipulate the constraints and add slack variables so that the right-most portion of $\boldsymbol{A}$ becomes an identity matrix.
2. Run simplex method on the transformed problem, while maintaining $\bar{\boldsymbol{c}} \geq \boldsymbol{0}$.
   (a) At the k-th iteration, if there is no negative basic variable, then an optimal primal solution has been found.
   (b) Otherwise, select some $x_\ell < 0$ as the leaving variable.
      i. If $d_\ell^j \geq 0$ for all $j \in N_k$, then the primal problem is infeasible.
      ii. Otherwise, take
      $$i = \underset{j \in N_k}{\operatorname{argmin}} \left\{ \frac{\bar{c}_j}{\left|d_\ell^j\right|} : d_\ell^j < 0 \right\}$$
      as the index of the entering variable.
3. Terminate the search when we have obtained $\boldsymbol{x}_B \geq \boldsymbol{0}$.

3.4 Sensitivity Analysis

Suppose that we have found an optimal solution $\boldsymbol{x}^*$ for some linear program, we are interested to know if the original problem is altered, how far away will $\boldsymbol{x}^*$ become from the new optimal solution. Specifically, we want to examine the sensitivity of

1. feasibility $\boldsymbol{A}_B^{-1}\boldsymbol{b} \geq \boldsymbol{0}$, and
2. optimality $\boldsymbol{c}^{\mathrm{T}} - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A} \geq \boldsymbol{0}$.

Suppose $\boldsymbol{b}$ is changed to $\boldsymbol{b} + \delta\boldsymbol{e}_i$. Note that the optimality condition is independent of $\boldsymbol{b}$ and so is unaffected. If $\boldsymbol{x}^*$ were still to be feasible, we must have

$$\boldsymbol{0} \leq \boldsymbol{A}_B^{-1}(\boldsymbol{b} + \delta\boldsymbol{e}_i) = \boldsymbol{x}_B^* + \delta\left(\boldsymbol{A}_B^{-1}\boldsymbol{e}_i\right).$$

The above inequality yields a range of values for δ such that $\boldsymbol{x}^*$ remains optimal and feasible in the altered problem. The optimal value in the altered problem is given by

$$\begin{aligned}\boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}(\boldsymbol{b}+\delta\boldsymbol{e}_i) &= \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{b}+\delta\boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{e}_i\\ &= \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{b}+\delta\left(\boldsymbol{p}^*\right)^{\mathrm{T}}\boldsymbol{e}_i\\ &= \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{b}+\delta p_i^*.\end{aligned}$$

We see that the change in objective value is closely tied to the dual optimal solution, such that a small change of δ in b_i results in a change of δp_i^* in the optimal value.

Definition 3.4.1 ▶ Marginal Cost

Let (P) be a primal problem with dual problem (D). If the dual problem has an optimal solution $\boldsymbol{p}*$, then p^* is called the **marginal cost** or **shadow cost** of b_i for minimisation (P), and the **marginal profit** or **shadow price** of b_i for maximisation (P).

Suppose that $\boldsymbol{c}$ is changed to $\boldsymbol{c}+\delta\boldsymbol{e}_j$ for some j. Note that the feasibility condition is independent of $\boldsymbol{c}$. If x_j is non-basic, the altered reduced cost is

$$\bar{c}_j' = c_j+\delta-\boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A} = \bar{c}_j+\delta.$$

Clearly, optimality remains if and only if $\delta \geq -\bar{c}_j$. If x_j is basic, then for each $i \in N$, we need

$$\begin{aligned}c_i-\left(\boldsymbol{c}_B+\delta\boldsymbol{e}_j\right)^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_i &= c_i-\boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_i-\delta\boldsymbol{e}_j^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_i\\ &= \bar{c}_j-\delta\boldsymbol{e}_j^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_i\\ &\geq 0.\end{aligned}$$

If $\boldsymbol{e}_j^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_i > 0$, we have $\delta \leq \frac{\bar{c}_j}{\boldsymbol{e}_j^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_i}$. If $\boldsymbol{e}_j^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_i < 0$, we have $\delta \geq \frac{\bar{c}_j}{\boldsymbol{e}_j^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_i}$. If $\boldsymbol{e}_j^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_i = 0$, it is clear that $\bar{c}_j$ is unchanged. Therefore, let $\bar{a}_{i,j} = \boldsymbol{e}_j^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_i$, we have

$$\max_{\bar{a}_{i,j}<0}\frac{\bar{c}_j}{\bar{a}_{i,j}} \leq \delta \leq \min_{\bar{a}_{i,j}>0}\frac{\bar{c}_j}{\bar{a}_{i,j}}.$$

Suppose that some entry a_{ij} of a non-basic column $\boldsymbol{A}_j$ is changed to $a_{ij}+\delta$ for some $\delta \neq 0$. Note that $j \notin B$, so this change does not affect the feasibility $\boldsymbol{A}_B^{-1}\boldsymbol{b} \geq \boldsymbol{0}$. In terms of

optimality, the only reduced cost affected is

$$\begin{aligned}\bar{c}'_j &= c_j - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\left(\boldsymbol{A}_j + \delta\boldsymbol{e}_i\right)\\ &= \bar{c}_j - \delta\boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{e}_i\\ &= \bar{c}_j - \delta p_i.\end{aligned}$$

Clearly, if $\bar{c}_j - \delta p_i \geq 0$, the original optimal solution remains optimal in the altered problem, so we obtain a condition on δ to maintain the optimality.

Suppose we add a new variable x_{n+1}, i.e., we raise the dimension of the problem by 1, then we obtain a new problem

$$\begin{aligned}&\min \boldsymbol{c}^{\mathrm{T}} + c_{n+1}x_{n+1}\\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} + x_{n+1}\boldsymbol{A}_{n+1} = \boldsymbol{b},\\ &\qquad \boldsymbol{x} \geq \boldsymbol{0}, \qquad x_{n+1} \geq 0.\end{aligned}$$

Note that $(\boldsymbol{x}^*, 0)$ is still a basic feasible solution to the new problem by taking $x_{n+1} = 0$, so it suffices to consider the optimality at $(\boldsymbol{x}^*, 0)$ by computing

$$\bar{c}_{n+1} = c_{n+1} - \boldsymbol{c}_B^{\mathrm{T}}\boldsymbol{A}_B^{-1}\boldsymbol{A}_{n+1}.$$

If $\bar{c}_{n+1} \geq 0$, this new basic feasible solution is still an optimal solution. Otherwise, this implies that x_{n+1} can be an entering variable, and we should use the simplex method to find the optimal solution for the update problem.

Suppose we add a new constraint $\boldsymbol{a}_{m+1}^{\mathrm{T}}\boldsymbol{x} \leq b_{m+1}$, we will actually implicitly introduce a new variable x_{n+1} as a slack variable. Note that the objective function remains unchanged, so if $\boldsymbol{x}^*$ satisfies this new constraint, there is nothing to do — it is still an optimal solution!

Otherwise, $\boldsymbol{x}^*$ becomes infeasible. Note that now we should have one more basic solution due to the new constraint. To find the new optimal, we first consider the new constraint in standard form:

$$\boldsymbol{a}_{m+1}^{\mathrm{T}}\boldsymbol{x} + x_{n+1} = b_{m+1}.$$

Suppose we add x_{n+1} as a new basic variable. Since $\boldsymbol{x}^*$ is infeasible now in the altered problem, we have $\boldsymbol{a}_{m+1}^{\mathrm{T}}\boldsymbol{x} > b_{m+1}$, so $x_{n+1} < 0$ is infeasible.

4 Network Optimisation

4.1 Network Flow Problems

In network flow problems, we study directed movement of resources between nodes. It is easy to see that such problems can be easily modelled with *directed graphs*.

Definition 4.1.1 ▶ Directed Graph

A **directed graph** $G = (V, E)$ consists of a vertex set $V(G)$ and an edge set $E(G)$, where $E(G)$ consists of ordered pairs of vertices in $V(G)$.

Every directed graph G induces an underlying undirected graph G' by defining

$$V(G') \coloneqq V(G), \qquad E(G') \coloneqq \{uv : \{(u,v),(u,v)\} \cap E(G) \neq \varnothing\}.$$

We say that G is *connected* if and only if G' is connected.

Definition 4.1.2 ▶ Network

A **network** is a directed graph G with a mapping $x : E(G) \to \mathbb{R}_0^+$ known as the **flow**, a mapping $c : E(G) \to \mathbb{R}_0^+$, known as the **cost**, and a mapping $b : V(G) \to \mathbb{R}$, known as the **external supply/demand**.

If $x_{ij} = x(u_iu_j) \leq u_{ij}$ for some *upper bound* u_{ij}, we say that the network is *capacitated*. Otherwise, we say that the corresponding problem is *uncapacitated*. For any $v \in V(G)$, we say that v is a

- *supply node* if $b(v) > 0$;
- *demand node* if $b(v) < 0$;
- *trans-shipment node* if $b(v) = 0$.

We also denote $b(v) = b_v$.

Note that the above construction can be easily applied to directed multigraphs by merging all edges (u, v) and consider the aggregate flow between the vertices.

In network optimisation, we wish to maintain *flow balance*, i.e., whatever flows out of a node (total flow out plus external demand) must equal the sum of total flow in and external

supply.

Definition 4.1.3 ▶ Flow Balance Constraint

Let G be a network. For each $v \in V(G)$, denote

$$O(v) \coloneqq \{u \in V(G) : (v,u) \in E(G)\}, \qquad I(v) \coloneqq \{u \in V(G) : (u,v) \in E(G)\},$$

then the **flow balance constraint** is formulated as

$$\sum_{u \in O(v)} x_{vu} - \sum_{u \in I(v)} x_{uv} = b_v$$

for all $v \in V(G)$.

We say that a network has a feasible flow if it satisfies the flow balance constraint and the *capacity constraint*. Intuitively, this means that

$$\sum_{v \in V(G)} b_v = 0.$$

We can prove this equality more rigorously by considering the following representation:

Definition 4.1.4 ▶ Node-Arc Incidence Matrix

Let G be a simple directed graph. Label the vertices

$$V(G) = \{v_1, v_2, \cdots, v_n\}$$

and edges

$$E(G) = \{e_j = (u_j, w_j) : j = 1, 2, \cdots, m\}.$$

The matrix $\boldsymbol{A}$ defined by

$$a_{ij} = \begin{cases} 1 & \text{if } v_i = u_j \\ -1 & \text{if } v_i = w_j \\ 0 & \text{otherwise} \end{cases}$$

is said to be the **node-arc incidence matrix** of G.

Clearly, the flow for every $e_j \in E(G)$ is just $x_{u_j w_j}$. Collect the flows on all edges into a $1 \times m$ column vector

$$\boldsymbol{x} = \begin{bmatrix} x_{u_1 w_1} & x_{u_2 w_2} & \cdots & x_{u_m w_m} \end{bmatrix}^{\mathrm{T}},$$

then $b_{v_i} = \left(\boldsymbol{A}^{\mathrm{T}}\right)_i \boldsymbol{x}$, where $\left(\boldsymbol{A}^{\mathrm{T}}\right)_i$ is the i-th row of $\boldsymbol{A}$. Observe that the column sum of $\boldsymbol{A}$ is $\boldsymbol{0}$,

which naturally implies that

$$\sum_{v \in V(G)} b_v = \left(\sum_{i=1}^{n} \left(\boldsymbol{A}^{\mathrm{T}} \right)_i \right) \boldsymbol{x} = 0.$$

Notice that by using the node-arc incidence matrix, we can actually formulate a network flow problem as the linear program

$$\begin{aligned} &\min_{\boldsymbol{x} \in \mathbb{R}^n} \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\qquad \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}, \end{aligned}$$

where $\boldsymbol{x}$ is the flow vector, $\boldsymbol{c}$ is the cost, $\boldsymbol{b}$ is the external supply vector, $\boldsymbol{A}$ is the node-arc incidence matrix and $\boldsymbol{u}$ is the upper bound for flow.

In general, there are some types of linear programs which can be converted to a network flow problem. For example, if the constraint of a linear program is of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{A}$ is an $m \times n$ 0-1 matrix such that the 1's in each column appear consecutively, then we can obtain a corresponding node-arc incidence matrix by

$$\boldsymbol{A}' = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 \\ 0 & 0 & 0 & \cdots & 0 & -1 \end{bmatrix} \boldsymbol{A}.$$

4.1.1 Single-Source Shortest Path Problem

In graph theory, the *single-source shortest path problem* (SSSP) is a classic and easy-to-solve optimisation problem. The problem statement is as follows:

> Let G be a directed graph and $w : E(G) \to \mathbb{R}$ be a weight mapping on edges. Fix a source vertex s. For any destination vertex $t \in V(G)$, find an (or all) s-t path(s) P_{st} in G such that
>
> $$\sum_{v \in V(P_{st})} w(v)$$
>
> is minimised.

In programming, an SSSP can be easily solved by deploying one of many greedy graph traversal algorithms, such as Bellman-Ford Algorithm, Dijkstra's Algorithm and A* Algorithm, provided that the graph does not contain any negatively weighted cycle, where the problem is unsolvable.

How to formulate an SSSP as a linear program? Suppose s is our source and t is our destination, the shortest s-t path is essentially such that **the sum of flow costs over the edges in the path is minimised**. Therefore, it suffices to send **a unit flow from** s **to** t and minimise the total cost $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$.

Proposition 4.1.5 ▸ SSSP as a Linear Program

Let G be a network with source and destination vertices s and t respectively. Let $\boldsymbol{x}^*$ be an optimal solution to the linear program

$$\begin{aligned}
&\min_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\
&s.t. \sum_{u\in O(v)} x_{vu} - \sum_{u\in I(v)} x_{uv} = \begin{cases} 1 & \text{if } v = s \\ -1 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} \\
&\boldsymbol{x} \geq \boldsymbol{0}.
\end{aligned}$$

such that $x_i^* \in \{0,1\}$, then the shortest s-t path is induced by $\{e_i \in E(G) : x_i^* = 1\}$.

Proof. The case where $s = t$ is trivial. Assume $s \neq t$. Let P be the subgraph of G induced by $E(P) := \{e_i \in E(G) : x_i^* = 1\}$. We first prove that P is an s-t walk in G. It is easy to see that $t \in V(P)$, because otherwise

$$\sum_{u\in O(t)} x_{tu} - \sum_{u\in I(t)} x_{ut} = 0.$$

Similarly, $s \in V(P)$. We claim that s and t are the only end vertices in P. Suppose on contrary that there exists some $r \in V(P)$ with $r \neq s, t$ and either $O(r) = \varnothing$ or $I(r) = \varnothing$. However,

$$\sum_{u\in O(r)} x_{ru} - \sum_{u\in I(r)} x_{ur} = 0 \neq -1,$$

so r must be an isolated vertex in P, which is a contradiction because P is induced on a subset of $E(G)$, meaning that every connected component of P must contain an edge. Therefore, P is an s-t walk. Suppose on contrary that P is not a shortest s-t path,

then there exists a shorter s-t path P' in G. Consider $\boldsymbol{x}^{**}$ defined by

$$x_i^{**} = \begin{cases} 1 & \text{if } e_i \in E(P') \\ 0 & \text{otherwise} \end{cases},$$

then $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^{**} < \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^{*}$, which contradicts the optimality of $\boldsymbol{x}^{*}$. Therefore, P is a shortest s-t path in G. □

An alternative perspective to view Proposition 4.1.5 is as follows: label the edge set

$$E(G) \coloneqq \{e_1, e_2, \cdots, e_m\}.$$

Define a mapping $E(G) \mapsto \mathcal{B}$, where $\mathcal{B}$ is the family of all boolean arrays of length m, as follows: for every edge $e_i \in E(G)$, mark it as "taken" by mapping it to 1 and "not taken" by mapping it to 0. The corresponding boolean array $\boldsymbol{x}$ optimal to the minimisation problem essentially indicates all edges included in the shortest s-t path in G.

In addition, we can re-write the SSSP problem in the following form:

$$\begin{aligned} &\min \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{e}_s - \boldsymbol{e}_t \\ &\boldsymbol{x} \geq \boldsymbol{0}. \end{aligned}$$

the dual problem to an SSSP can be formulated as

$$\begin{aligned} &\max \boldsymbol{p}^{\mathrm{T}}(\boldsymbol{e}_s - \boldsymbol{e}_t) \\ &\text{s.t. } \boldsymbol{A}^{\mathrm{T}}\boldsymbol{p} \leq \boldsymbol{c}, \end{aligned}$$

which can be further simplified to

$$\begin{aligned} &\max p_s - p_t \\ &\text{s.t. } p_i - p_j \leq c_{(i,j)} \quad \text{for all } (i,j) \in E(G). \end{aligned}$$

This means that $\boldsymbol{p}$ and $\boldsymbol{p} + \boldsymbol{\alpha}$ are equivalent in the dual problem and so we can safely set $p_t = 0$ by defining $\alpha = -p_t$.

We now discuss a few applications of SSSP programs.

Three Jug Puzzle

Three Jug Puzzle

Given a a-litre jug filled with water and two empty jugs of capacity b and c litres respectively such that $a = b + c$, find the minimum number of steps to measure out exactly d litres of water by transferring water between the jugs only.

We first define a state in the problem by an ordered 3-tuple (a_i, b_i, c_i) indicating the amount of water in the three jugs respectively. Clearly, $a_i + b_i + c_i = a$ and so a_i is actually implicitly determined by b_i and c_i Therefore, every state in the problem can be represented by an ordered pair (b_i, c_i). Now, construct a network with vertex set

$$V \coloneqq \{(i, j) : \ i = 0, 1, \cdots, b, j = 0, 1, \cdots, c\}.$$

We define the edges as follows:

- $(i, j) \to (b, j)$: transferring water from the first jug to the second jug;
- $(i, j) \to (0, j)$: transferring water from the second jug to the first jug;
- $(i, j) \to (i, c)$: transferring water from the first jug to the third jug;
- $(i, j) \to (i, 0)$: transferring water from the third jug to the first jug;
- $(i, j) \to (\min\{0, i + j - c\}, \max\{i + j, c\})$: transferring water from the second jug to the third jug;
- $(i, j) \to (\max\{i + j, b\}, \min\{0, i + j - b\})$: transferring water from the third jug to the second jug;

Now we only need to find the shortest $(0, 0)$-(i, j) path such that either $i = d$ or $j = d$ or $a - i - j = d$.

Dynamic Lot Sizing

Dynamic Lot Sizing

Suppose a factory has T production periods with demand d_i for the i-th period. Let x_i be the output of the i-th period and I_i be the inventory at the end of the $(i - 1)$-th period. The cost per unit of production is c_i and the cost per unit of inventory holding is h_i in the i-th period, and a set-up cost of K_i is incurred in the i-th period whenever $x_i > 0$. Determine a production plan $\boldsymbol{x} = (x_1, x_2, \cdots, x_T)$ to meet the demands and minimise the total cost.

First, notice that $I_i = I_{i-1} + x_i - d_i$. To minimise the total cost, we should not produce any redundant unit of goods, and so $I_0 = I_T = 0$. Moreover, the following proposition is true:

Proposition 4.1.6 ▸ Optimality Condition of Dynamic Lot Sizing

If $(x_1, x_2, \cdots, x_T)$ is an optimal production plan, then $I_{i-1}x_i = 0$ and $x_i = \sum_{k=i}^{j_i} d_k$ for some $j_i \geq j - 1$.

This means that to achieve minimal production cost, we should not both use the inventory and produce new goods in a period, and each production should be prudently scheduled to meet future demands for several periods. Therefore, suppose there are m periods, labelled as $j_1, j_2, \cdots, j_m$, where production is carried out, then

$$x_{j_k} = \sum_{i=j_k}^{j_{k+1}-1} d_i.$$

Therefore, if we define the T periods as a vertex set, then a 0-T path indicates the periods where production should take place, where an edge (i, j) connects two consecutive production periods. Therefore, we can define the flow cost on the edge by

$$c_{(i,j)} = K_i + c_i x_i + \sum_{k=i}^{j-1} h_k I_k,$$

where

$$I_k = x_i - \sum_{r=i}^{k} d_r = \sum_{r=k+1}^{j-1} d_r.$$

Since we do not carry over I_{j-1} to the j-th period, we need $I_{j-1} = 0$ to minimise wastage. Therefore,

$$c_{(i,j)} = K_i + c_i \sum_{k=i}^{j-1} d_k + \sum_{k=i}^{j-2} \left(h_k \sum_{r=k+1}^{j-1} d_r \right).$$

Now, we only need to find the shortest 0-T path.

Project Management

Project Management

Given a list of activities $A_1, A_2, \cdots, A_n$. Let t_i be the duration for A_i and $P(A_i)$ be the required activities which must be completed before A_i starts. For each activity, find the maximum possible time to delay without affecting the project completion time.

For each activity A_i, write $A_i := \left(s_{E_i}, e_{E_i}, s_i, i, s_{L_i}, e_{L_i}\right)$ as a node, where s_{E_i}, e_{E_i} are the earli-

est start and end time and s_{L_i}, e_{L_i} are the latest start and end time, such that $s_i = s_{L_i} - s_{E_i}$. s_i is known as the *slack* of A_i. Connect $A_i \to A_j$ if $A_i \in P(A_j)$. Take a node with the lowest topological ordering as the source node. We will first process the nodes in the forward direction and set

$$s_{E_i} = \max\left\{e_{E_j} : A_j \in P(A_i)\right\},$$

i.e., take the latest start time for each activity. After all nodes are processed, we backtrack from the node with the highest topological ordering and set

$$e_{L_i} = \min\left\{s_{L_j} : A_i \in P(A_j)\right\}.$$

Eventually, there will be a path such that every node in the path has zero slack. This path is known as the *critical path*.

4.1.2 Maximum Flow Problem

The maximum flow problem has the following statement:

> Let G be a capacitated network with u_{ij} as the upper bound for the flow on edge (i, j). For a fixed source vertex s and a fixed destination vertex t, find the maximum external supply b_s at s such that we can transport b_s units of flow from s to t.

Suppose $b_s = b$, then the maximum flow problem can be formulated as the linear program

$$\begin{aligned}
&\min_{b\geq 0} b \\
&\text{s.t.} \sum_{u\in O(v)} x_{vu} - \sum_{u\in I(v)} x_{uv} = \begin{cases} b & \text{if } v = s \\ -b & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} \\
&\quad \mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{u}.
\end{aligned}$$

The flow balance constraint can be written as $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{d}f$, where f is the amount of flow from s to t and $\boldsymbol{d} = \boldsymbol{e}_s - \boldsymbol{e}_t$.

A variant of the maximum flow problem is the multi-source maximum flow problem, where we are given n source nodes $s_1, s_2, \cdots, s_n$ and m destination nodes $t_1, t_2, \cdots, t_m$. The aim is to find the maximum total flow from the sources to the destinations. This can be easily converted back to the single-source case by adding a new source s and a new destination t, and join $s \to s_i$ for all $i = 1, 2, \cdots, n$ and $t_j \to t$ for all $j = 1, 2, \cdots, m$ using uncapacitated arcs.

A problem closely related to the maximum flow problem is the *minimum cut problem*, which states the following:

> Let G be a directed graph with $w : E(G) \to \mathbb{R}_0^+$ as a weight map over the edges. For two fixed vertices $s, t \in V(G)$, find an edge cut $X \subseteq E(G)$ with minimum total weight such that s and t are in different connected components in $G - X$.

We introduce a few relevant definitions to the minimum cut problem.

Definition 4.1.7 ▸ s-t Cut

Let G be a graph and $s, t \in V(G)$. An **s-t cut** in G is a bipartition $\{S_1, S_2\}$ of $V(G)$ such that $s \in G[S_1]$ and $t \in G[S_2]$. The cut is minimal if $|E(S_1, S_2)|$ is minimised.

Remark. Clearly, the total capacity of the minimal s-t cut in G upper bounds the maximum flow from s to t in G.

Therefore, this motivates us to view a maximum flow problem from the perspective of a minimum cut problem. To formulate a minimum s-t cut problem, we can assign boolean flags to the vertices, such that $y_i = 1$ if $u_i \in S_1$ and $y_i = 0$ if $u_i \in S_2$. It is clear that $y_s = 1$ and $y_t = 0$.

For every edge u_iu_j, we again assign boolean flags to them such that $z_{ij} = 1$ if u_1u_j is an S_1-S_2 edge and $z_{ij} = 0$ otherwise. Additionally, note that for every edge u_iu_j, if $z_{ij} = 1$, we can without loss of generality assume that $y_i = 1$ and $y_j = 0$, and so $y_i - y_j = 1$. Otherwise, $z_{ij} = 0$ and $y_i = y_j$. Either way, we have $y_i - y_j \leq z_{ij}$. However, if $\boldsymbol{A}$ is the node-arc incident matrix, $y_i - y_j$ is exactly the row corresponding to u_iu_j in $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}$.

Therefore, we can formulate the minimum s-t cut problem as follows:

$$\begin{aligned} \min\ & \boldsymbol{u}^{\mathrm{T}}\boldsymbol{z} \\ \text{s.t. } & \boldsymbol{d}^{\mathrm{T}}\boldsymbol{y} = 1 \\ & -\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{z} \leq \boldsymbol{0} \\ & \boldsymbol{z} \geq \boldsymbol{0} \end{aligned}$$

With some careful work, it can be verified that this is exactly the dual problem of a maximum flow problem. By Theorem 3.2.4, this means that the total capacity of the minimum s-t cut gives the maximum s-t flow.

Proposition 4.1.8 ▸ Maximum Flow Problem and Minimum Cut Problem

The maximum s-t flow in a network G is equal to the total capacity of the minimum s-t cut in G.

4.2 Network Simplex Method

Consider an uncapacitated minimum cost flow problem formulated as

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^m} \ & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{s.t. } & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0}. \end{aligned}$$

Note that here, $\boldsymbol{A}$ is a node-arc incidence matrix. By Definition 4.1.4, we see that $\boldsymbol{A}$ has a very nice structure, which makes such network problems to be solvable very efficiently. We will modify the simplex method to solve such problems.

We first characterise basic feasible solutions in the problem. Take any index set B. Notice that the row sum of $\boldsymbol{A}_B$ is always 0, so any $\boldsymbol{A}_B$ is never invertible. Therefore, we can never find a basis for the network problem and thus cannot compute the basic solution using $\boldsymbol{x}_B = \boldsymbol{A}_B^{-1}\boldsymbol{b}$.

Definition 4.2.1 ▸ Truncated Node-Arc Incidence Matrix

Let $\boldsymbol{A}$ be a node-arc incidence matrix, the **truncated node-arc incidence matrix** $\widetilde{\boldsymbol{A}}$ is obtained by deleting any row from $\boldsymbol{A}$.

It is easy to check that the rows of $\widetilde{\boldsymbol{A}}$ are always linearly independent, provided that the underlying directed graph has no isolated vertex. Correspondingly, we can obtain a truncated constraint $\widetilde{\boldsymbol{A}}\boldsymbol{x} = \widetilde{\boldsymbol{b}}$. Now, for any network G, we consider its spanning trees.

Definition 4.2.2 ▸ Tree Solution

Let G be a network with node-arc incidence matrix $\boldsymbol{A}$. A flow vector $\boldsymbol{x} \in \mathbb{R}^{e(G)}$ is a **tree solution** if $\widetilde{\boldsymbol{A}}\boldsymbol{x} = \widetilde{\boldsymbol{b}}$ and there exists some spanning tree T of G such that $x_i = 0$ for all $e_i \notin E(T)$. A **feasible tree solution** is a tree solution $\boldsymbol{x}$ with $\boldsymbol{x} \geq \boldsymbol{0}$.

Proposition 4.2.3 ▸ Spanning Tree Induces Basis on $\tilde{A}$

Let $\boldsymbol{A}$ be the node-arc incidence matrix for a network G. An index set B is a basis of $\widetilde{A}$ if and only if the set $E_B := \{e_i \in E(G) : i \in B\}$ induces a spanning tree of G.

Proof. Without loss of generality, suppose $\widetilde{\boldsymbol{A}}$ is the first $(n-1)$ rows of $\boldsymbol{A}$. Suppose E_B induces a spanning tree of G, then it suffices to prove that $\widetilde{\boldsymbol{A}}_B$ is invertible. We shall prove this by showing that the rows of $\widetilde{\boldsymbol{A}}$ are linearly independent. Suppose

$\boldsymbol{w} \in \mathbb{R}^{n-1}$ is such that $\boldsymbol{w}^{\mathrm{T}}\widetilde{\boldsymbol{A}}_B = \boldsymbol{0}$, then $\boldsymbol{w}^{\mathrm{T}}\widetilde{\boldsymbol{A}}_i = 0$ for all $i \in B$. Let $e_i = (v_k, v_h)$. Observe that if $v_n \notin e_i$, then $w_k = w_h$. If $v_k = v_n$, then $w_h = 0$ and vice versa. Since E_B induces a spanning tree T, there is some $j \in B$ such that e_j is incident to v_n, and so there is some $w_r = 0$. Note that for all neighbours v_m connected to v_r in T, $w_m = 0$. This means $\boldsymbol{w} = 0$. □

Now, we consider the reduced cost. Recall that $\boldsymbol{p}$ such that $\boldsymbol{p}^{\mathrm{T}} = \boldsymbol{c}_B^{\mathrm{T}}\widetilde{\boldsymbol{A}}_B^{-1}$ is an optimal solution to the dual problem. Therefore, the reduced costs are

$$\bar{\boldsymbol{c}}^{\mathrm{T}} = \boldsymbol{c}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}}\widetilde{\boldsymbol{A}}.$$

Since the n-th row of $\boldsymbol{A}$ is removed in the truncated matrix, we have

$$\bar{c}_{(i,j)} = \begin{cases} c_{(i,j)} - (p_i - p_j) & \text{if } i, j \neq n \\ c_{(i,j)} - p_i & \text{if } j = n \\ c_{(i,j)} + p_j & \text{if } i = n \end{cases}.$$

By defining $p_n = 0$, we obtain $\bar{c}_{(i,j)} = c_{(i,j)} - (p_i - p_j)$ for all $(i, j) \in E(G)$. Let B be a basis inducing a spanning tree T on the network G. Note that if $(i, j) \in B$, we must have $\bar{c}_{(i,j)} = 0$, so we have

$$\begin{aligned} &p_i - p_j = c_{(i,j)} \quad \text{for all } (v_i, v_j) \in E(T), \\ &p_n = 0. \end{aligned}$$

Note that $V(T) = V(G)$, so solving the above system of equations gives the values of the p_i's for all $i = 1, 2, \cdots, n$, which then allows us to compute $\bar{\boldsymbol{c}}$.

Suppose we take a basis B corresponding to the spanning tree T of a network G. If there is some $(i, j) \notin B$ such that $\bar{c}_{(i,j)} < 0$, it means the edge (v_i, v_j) should be added to improve the flow. Note that there is an undirected v_i-v_j path in T, so $T + (v_i, v_j)$ contains a cycle C containing v_i and v_j. We define the direction of (v_i, v_j) as the forward direction and let C_f and C_b be the sets of forward and backward edges in C respectively.

To improve the cost, we need to increase the flow in (v_i, v_j) as much as possible. Suppose we set $x_{(i,j)} = \theta^*$, then the amount of forward flow in every $C_f \in C_f$ increases by θ^*, which means we need to reduce the backward flow in every $e_b \in C_b$ by θ^* as well. Intuitively, we can take

$$\theta^* = \min_{(v_k, v_\ell) \in C_b} x_{k\ell}.$$

In particular, if $C_b = \varnothing$, $\theta^* = \infty$ and the optimal cost is $-\infty$. Otherwise, if $x_{pq} = \theta^*$, we will choose (p, q) to leave the basis.

Technique 4.2.4 ▸ Network Simplex Method

Consider an uncapacitated minimum cost flow problem on a network G with node-arc incidence matrix $\boldsymbol{A}$. The network simplex method finds the optimal flow vector $\boldsymbol{x}^*$ by the following procedures:

1. Take a spanning tree $T_0 \subseteq G$ and find a basis B from $E(T_0)$ and a feasible tree solution $\boldsymbol{x}_0$.
2. At the k-th iteration, compute the dual vector $\boldsymbol{p}_k^{\mathrm{T}} = \boldsymbol{c}_{B_k}^{\mathrm{T}} \widetilde{\boldsymbol{A}}_{B_k}^{-1}$. Define $p_{n_k} = 0$.
3. For each edge $u_i u_j \in E(G)$, compute the reduced cost $\bar{c}_{(i,j)} = c_{(i,j)} - (p_{i_k} - p_{j_k})$.
4. If $\bar{c}_{(i,j)} \geq 0$ for all $(i, j) \in E(G)$, then $\boldsymbol{x}_k$ is optimal.
5. Otherwise, choose $(i, j) \notin B$ with $\bar{c}_{(i,j)} < 0$.
6. Let C_k be the cycle contained in $T + u_i u_j$ and C_f and C_b be the forward and backward edges in C_k respectively, where $u_i u_j$ induces the forward direction.
   (a) If $C_b = \varnothing$, then the optimal cost is unbounded.
   (b) Otherwise, compute

$$\theta^* \coloneqq x_{pq} = \min_{(u_k, u_\ell) \in C_b} x_{k\ell}.$$

7. Update $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1}$ by

$$\widehat{x_{k\ell}} = \begin{cases} x_{k\ell} + \theta^* & \text{if } u_k u_\ell \in C_f \\ x_{k\ell} - \theta^* & \text{if } u_k u_\ell \in C_b \\ x_{k\ell} & \text{otherwise} \end{cases}.$$

8. Update $T_{k+1} = \big(T - u_p u_q\big) \cup u_i u_j$.

Similar to Technique 2.4.1, we can also deploy a two-phase method to initialise a feasible tree solution.

Technique 4.2.5 ▸ Network Two-Phase Method

Consider an uncapacitated minimum cost flow problem on an n-vertex network G with node-arc incidence matrix $\boldsymbol{A}$ and supply vector $\boldsymbol{b}$. The network two-phase method finds the optimal flow vector $\boldsymbol{x}^*$ by the following procedures:

1. Construct an auxiliary network G' on $V(G)$ with

$$E(G') \coloneqq E(G) \cup \{u_i u_n \notin E(G) : \ b_i \geq 0\} \cup \{u_n u_j \notin E(G) : \ b_j < 0\}.$$

2. Initialise flow costs on G' by

$$c'_{(i,j)} = \begin{cases} 0 & \text{if } u_i u_j \in E(G) \\ 1 & \text{otherwise} \end{cases}.$$

3. Take $B \coloneqq \{u_i u_n \in E(G') : \ b_i \geq 0\} \cup \{u_n u_j \in E(G') : \ b_j < 0\}$ as an initial basis. Run network simplex method on G' until a basis B_0 containing no edge from $G' - G$ is found.
4. Take B_0 as an initial basis to run network simplex method on G.

4.3 Integrality

Network problems exhibit an interesting feature that most quantities and parameters of interest are integers if the network data is given in integers.

Theorem 4.3.1 ▶ Integrality of Network Optimisation Problems

Let G be an n-vertex network with a connected underlying graph, then $\widetilde{\boldsymbol{A}}_B^{-1}$ has only integer entries for any basis B.

Proof. For each vertex $v_i \in V(G)$ where $1 \leq i \leq n-1$, Consider a network flow problem (P_i) defined on a network $H_i \subseteq G$ with $V(H_i) = V(G)$ and $E(H_i) = B$, such that $b_i = 1$ and $b_n = -1$, while all other nodes are transshipment nodes with no external supply nor demand. Clearly, the flow balance equation for (P_i) is

$$\boldsymbol{A}_B \boldsymbol{x}_B = \boldsymbol{b}, \quad \text{where } \boldsymbol{b} = \boldsymbol{e}_i - \boldsymbol{e}_n.$$

Truncating the equation by taking the first $(n-1)$ rows yields $\widetilde{\boldsymbol{A}}_B \boldsymbol{x}_B = \boldsymbol{e}_i$. Therefore, we have $\widetilde{\boldsymbol{A}}_B^{-1} \boldsymbol{e}_i = \boldsymbol{x}_B$. By Proposition 4.2.3, H_i is a spanning tree of G and so there is a unique path between any pair of vertices. Therefore, the flow is either 1 or -1 depending on the directions of the edges. This implies that $\widetilde{\boldsymbol{A}}_B^{-1} \boldsymbol{e}_i$ has only integer entries, and so $\widetilde{\boldsymbol{A}}_B^{-1}$ is an integer matrix. □

From Theorem 4.3.1, we see that if the supply vector $\boldsymbol{b}$ is also integer, then every basic solution $\boldsymbol{x}$ must be integer since $\boldsymbol{x}_B = \widetilde{\boldsymbol{A}}_B^{-1} \boldsymbol{b}$ is now integer. Similarly, the dual basic solution $\boldsymbol{p} = \boldsymbol{c}_B^{\mathrm{T}} \widetilde{\boldsymbol{A}}_B^{-1}$ is also integer if the cost vector $\boldsymbol{c}$ is integer. This leads to the following easy corollary:

Corollary 4.3.2 ▶ Integrality of Optimal Solution in Network Flow Problems

Let G be an n-vertex network with a connected underlying graph. Consider a network flow problem (P) such that an optimal solution exists. If the supply vector $\boldsymbol{b}$ consists of purely integer entries, then there is an integer optimal solution; if the cost vector $\boldsymbol{c}$ consists of purely integer entries, then there is an integer dual optimal solution.