# Chapter 15

# MSM2M2 Introduction To Management Mathematics

## (15.1) Combinatorial Methods

### (15.1.1) Algorithms & Complexity

At the heart of any combinatorial method is an algorithm; a process which is (supposed) to yield the most efficient or profitable solution so some situation. A simple and obvious way to do this is as follows. In an attempt to solve this, the following could be used.

Algorithm 1 (The Greedy Algorithm) *Choose the most attractive feasible option available.*

However, it is easy to construct situations where this does not work — a number of conditions about the problem must be satisfied in order the the Greedy Algorithm to work. In search of a better solution to the one supplied using the Greedy Algorithm, it may be possible to find all permutations then choose the best. However, even powerful computers may not be able to solve even simple problems given many years. Two areas of interest are apparent,

- does a given algorithm find a correct solution.
- how efficient is the algorithm.

In a mathematical sense, the efficiency can be expressed in terms of how many elementary operations — addition, multiplication, etc. — must be performed in order to achieve the result. However, the 'input' must also be taken into account as clearly an algorithm that takes 10 operations on 3 'inputs' is less efficient than one that takes 10 operations on 30 'inputs'.

Example 2 *Compute the number of elementary operations needed to evaluate the polynomial function* $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ *using the method*

1. *by calculating the powers of* $x$*, multiplying by the coefficient* $a_i$*, then adding the terms.*
2. *by starting with the lowest power of* $x$*, and 'saving' the powers of* $x$ *for use in calculating the next term, then adding the terms.*
3. *using Horner's Method,* $p(x) = (\ldots((a_n x + a_{n-1})\,x + a_{n-2})\,x + \cdots + a_1)\,x + a_0$

Proof. Solution

1. Evaluation using this method involves

- $i-1$ multiplications for the power of $x$ in the $i$th term and another for the coefficient, giving a total of $\sum_{i=0}^{n} i = \frac{1}{2}n(n+1)$.
- $n$ additions in order to combine the resulting $n+1$ terms.

This gives a total of $\frac{1}{2}n(n+1)+n = \frac{1}{2}n^2+\frac{3}{2}n$.

2. Evaluation using this method involves
   - The term with $x^0$ requires no multiplication. The term with $x^1$ requires one multiplication, the coefficient. Subsequent terms require a multiplication by $x$ and by the coefficient i.e. two each. There are therefore $0+1+2(n-2) = 2n-1$ multiplications.
   - $n$ additions in order to combine the $n+1$ terms.

   This gives a total of $3n-1$.
3. Evaluation using Horner's method involves
   - one multiplication for each power of $x$ — these are the $x$s outside the brackets of which there are $n-1$ and the 'first' $x$ which is multiplied by its coefficient, giving $n$ multiplications.
   - whenever an $x$ appears it is multiplied by something then added to the next coefficient. There are therefore $n$ additions.

   This gives a total of $2n$ operations performed. □

Finding the number of operations is easy enough in this case, and the number of 'inputs' — $n+2$ — is obvious. What is not obvious, though, is how to combine the two.

Definition 3 *The* computational complexity *of an algorithm $A$, $cc(A)$, is the number of operations performed on an input of length $L$. This is expressed as a function of $L$ so that $cc(A) = f(L)$.*

In Example ( 2 ) the computational complexities can now be calculated and are

1. $\frac{(L-2)^2}{2} + \frac{3(L-2)}{2}$.
2. $3L-7$.
3. $2L-4$.

Clearly Horner's method is the most efficient for an input of any given length.

Functions for the computational complexity may vary enormously for the same result but using different processes. It is of course desirable to achieve the computational complexity of lowest value. This is typically done by reducing the power of the expression, or in the linear case by reducing the values of the coefficients.

### (15.1.2) '$O$' Notation

It is convenient at this point to introduce the apparently bizarre ' '$O$' notation'. Consider the set of real functions defined on $\mathbb{R}^+$ which are eventually positive, i.e.

$$F^+ = \{f \mid f\colon \mathbb{R}^+ \to \mathbb{R} \quad \text{with} \quad f(x) > 0 \quad \forall x \geqslant x_0,\ x_0 \in \mathbb{R}^+\}$$

For $g(x) \in F^+$ the following definition is now made.

Definition 4 *The function $O(g)$ for a function $g$ which is eventually positive i.e. $g \in F^+$ is given by the set*

$$O(g) = \{f(x) \in F^+ \mid \exists c > 0 \quad \exists x_0 > 0 \quad \forall x \geqslant x_0 \quad f(x) \leqslant c \times g(x)\}$$

Note that when $f \in O(g)$ it is common to write "$f = O(g)$" or even "$f(x) = (g(x))$". $O(g)$ is the eventually positive functions whose value at $x$ does not exceed $c$ times $g(x)$, eventually. It is readily deduced that.If $f(x) \leqslant g(x)$ then $O(f) \subseteq O(g)$, i.e. $f = O(g)$. As functions can be composed, it follows that some similar process can be performed on the sets $O$.

Definition 5

1. $O(f) + O(g) = \left\{\overline{f}(x) + \overline{g}(x) \mid \overline{f} \in O(f) \quad \textit{and} \quad \overline{g} \in O(g)\right\}$
2. $O(f) \cdot O(g) = \left\{\overline{f}(x) \cdot \overline{g}(x) \mid \overline{f} \in O(f) \quad \textit{and} \quad \overline{g} \in O(g)\right\}$
3. $f + O(g) = \{f(x) + \overline{g}(x) \mid \overline{g} \in O(g)\}$
4. $f \cdot O(g) = \{f(x) \cdot \overline{g}(x) \mid \overline{g} \in O(g)\}$

The abuse of notation "$f(x) = (g(x))$" is extended so that "$\subseteq$" becomes the same as "$\leqslant$". It is readily shown from the definition that if $f \in O(g)$ then $O(f) \leqslant O(g)$, and indeed if $f \leqslant g$ then $O(f) \leqslant O(g)$. Along the same line of thought, the following can be shown from the definitions above.

Theorem 6 *For $f, g \in F^+$ and $k \in \mathbb{R}^+$,*

1. $O(f + g) = O(f) + O(g)$
2. $O(f \cdot g) = O(f) \cdot O(g)$
3. $k \cdot O(f) = O(f)$
4. $g \leqslant f \Rightarrow O(g + f) = O(f)$
5. $g \leqslant f \Rightarrow g + O(f) = O(f)$
6. $f(x) \geqslant \varepsilon > 0 \; \forall x \geqslant x_0 \Rightarrow f + a \in O(f) \quad (a > 0)$

The virtue of '$O$' notation can be seen when considering computational complexity functions. Rather than the actual function, a simplified form of it — its '$O$' set — is usually of interest as it gives an indication of the level of complexity. For Example 2,

- $O\left(\frac{(L-2)^2}{2} + \frac{3(L-2)}{2}\right) = L^2$.
- $O(3L) = L$.
- $O(2L) = L$.

So while Horner's method is the simplest, its computational complexity is comparable to that of the second method, while the first method is particularly more complex.

### (15.1.3) Sorting

A particular class of algorithm relate to sorting, perhaps most commonly putting a sequence of numbers into ascending order. First of all the *Merge* operation is defined.

Definition 7 *Suppose that $(x_1, x_2, \ldots, x_p)$ and $(y_1, y_2, \ldots, y_q)$ are sequences which are already sorted. The Merge operation defines the sequence $(z_1, z_2, \ldots, z_{p+q})$ as follows.*

1. *If $x_1 \leqslant y_1$ then set $z_1 := x_1$. Repeat comparing the first elements of $(x_2, x_3, \ldots, x_p)$ and $(y_1, y_2, \ldots, y_q)$.*
2. *If $x_1 > x_1$ then set $z_1 := y_1$. Repeat comparing the first elements of $(x_1, x_2, \ldots, x_q)$ and $(y_2, y_3, \ldots, y_p)$.*

3. *The process terminates when one of the sequences becomes empty. The remainder of the other sequence is then copied onto the end of the new sequence to give* $(z_1, z_2, \ldots, z_{p+q})$

*In mathematical formulae, merge will be denoted by the binary operation symbol '$\circ$'.*

The *Merge* operation takes two elementary operations for each iteration, of which there is one for every element of both of the sequences being merged. In the worst case $p = q$, so since the final element is copied, the computational complexity of *Merge* must be $2(p + q - 1)$.
Merging can only be done on sorted sequences, which begs the question as to how to sort.

Algorithm 8 (*MergeSort*) *Take an unsorted set of elements, treat each element as a sequence and apply Merge on pairs of the single element sequences. Repeat applying Merge until a single sorted sequence is obtained.*

The *MergeSort* algorithm can be expressed diagrammatically.

$$\underbrace{\underbrace{(\bullet_1, \bullet_2, \ldots, \bullet_k) \circ (\bullet_1, \bullet_2, \ldots, \bullet_k)}_{(\bullet_1, \bullet_2, \ldots, \bullet_{2k}) \circ \ldots} \cdots \underbrace{(\bullet_1, \bullet_2, \ldots, \bullet_k) \circ (\bullet_1, \bullet_2, \ldots, \bullet_k)}_{\cdots \circ (\bullet_1, \bullet_2, \ldots, \bullet_{2k})}}_{(\bullet_1, \bullet_2, \ldots, \bullet_{nk}) \quad \text{for some} \quad n \in \mathbb{N}}$$

Suppose that initially there are $n = 2^r$ elements to sort, for some $r \in \mathbb{N}$.

- Initially there are $2^n$ sorted sequences — the single element sequences.
- After the first iteration there are $\frac{n}{2}$ sorted sequences, and there were $\frac{n}{2}$ applications of *Merge*.
- After $k$ iterations there are $\frac{n}{2^k}$ sorted sequences, and there were $\frac{n}{2^k}$ applications of *Merge*.

When there is one sorted sequence the process terminates, at which time it must be the case that $\frac{n}{2^k} = 1$. This gives the number of iterations to be $k = \log_2 n$. Now, the computational complexity of *Merge* is $2(p + q - 1)$, and at each stage there are $n$ elements being merged. Therefore

$$\begin{aligned} cc\,(\text{ all merges }) &< 2n \\ \text{and so} \quad cc\,(Mergesort) &< 2n \log_2 n \\ &= O(n \log_2 n) \end{aligned}$$

### (15.1.4) Graph Theory

Most of the results covered here are also covered in (**??**). Indeed, it is necessary to know the same results.

Definition 9 *A graph is an ordered pair, $\Gamma = (V, E)$ where $V$ is a non-empty finite set, and $E$ is a set of two-subsets of $V$.*

The elements of $V$ are called vertices or nodes, and the elements of $E$ are called edges or arcs. Graphs can be readily drawn, as is shown in Fig.**??**. This graph has $V = \{1, 2, 3, 4\}$ and $E = \{\{1, 2\}, \{3, 4\}, \{1, 4\}\}$.

It does not matter how the graph is drawn — in particular the edges do not have to be straight — and two graphs that 'look' different but are in fact the same graph are called isomorphic.

There are, however, the limitations that there cannot be any 'loops', and edges can only meet at a vertex. Also, many of the graph theory results require that the graph is finite, i.e. $|V| \in \mathbb{N}$.

Definition 10 *For a graph with vertices $u, v$ and edges $e, f$,*

- *if $u$ is one of the vertices at the end of $e$, then $e$ is incident with $u$.*
- *$u$ and $v$ are adjacent if the edge $\{u, v\}$ is in $\Gamma$.*
- *$e$ and $f$ are adjacent if they are incident with the same vertex.*

**Definition 11** *For a graph $\Gamma = (V, E)$ with $|V| = n$ and $|E| = m$ then,*

- *$\Gamma$ is complete if any vertex is adjacent with any other vertex. Such a graph is written $K_n$, and clearly $m = \sum_{i=1}^{n} i = \frac{n(n-1)}{2}$.*
- *$\Gamma$ is trivial if $m = 0$ i.e. $E = \emptyset$.*
- *$\Gamma$ is sparse if $m = O(n)$. If $\Gamma$ is not sparse then it is dense.*

Some such graphs are shown in Fig.2

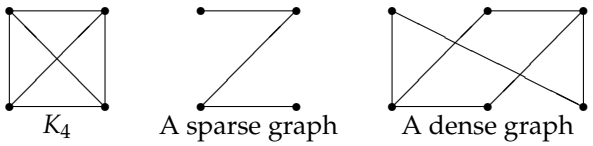


Figure 2: Different kinds of graph

Clearly complete graphs are dense.

**Definition 12** *A graph $\Gamma = (V, E)$ is bipartite if there exists $V_1 \subseteq V$ and $V_2 \subseteq V$, both of which are non-empty, such that $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$ and no two vertices of $V_1$ are adjacent and no two vertices of $V_2$ are adjacent.*

A complete bipartite graph — where all vertices of $V_1$ are connected to all vertices of $V_2$ but no vertices in $V_1$ — is denoted as $K_{pq}$ where $|V_1| = p$ and $|V_2| = q$.

**Definition 13** *A graph $\Gamma' = (V', E')$ is a subgraph of the graph $\Gamma = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. It is common to write $\Gamma' \subseteq \Gamma$.*

**Definition 14** *The degree of a vertex is the number of edges incident with that vertex.*

So if $\Gamma = (V, E)$ and $v \in V$ then $\deg_\Gamma v = |\{u \mid \{u, v\} \in E\}|$. Since each edge is incident with exactly two vertices, it is 'obvious' that $\sum_{v \in V} \deg v = 2|E|$. A formal proof can be given by induction, or by considering the incidence matrix of a graph.

**Definition 15** *For a graph $\Gamma = (V, E)$, a path in $\Gamma$ is a sequence of vertices $(v_1, v_2, \ldots, v_k)$ such that $v_i$ is adjacent to $v_{i+1}$ for $1 \leqslant i \leqslant k-1$, and $v_i \neq v_j \; \forall i \neq j$.*

Informally put, a path is a sequence of distinct non-adjacent vertices. A '$u$–$v$ path' is a path which starts at $u$ and ends at $v$. A graph is called connected if there is a $u$–$v$ path for all $u, v \in V$.

**Definition 16** *A cycle is a path $(v_1, v_2, \ldots, v_k)$ for which $k \geqslant 3$ and $v_1 = v_k$.*

A graph with no cycles is called acyclic, and a connected acyclic graph is a tree.

**Theorem 17** *If $\Gamma$ is a tree, then there is at least one vertex which has degree 1.*

An equivalent to this theorem is to say that if all vertex degrees are at least 2, then $\Gamma$ contains a cycle, and indeed the proofs are to all intents and purposes, 'the same'.

**Proof.** Suppose that the result is false, so that $\Gamma$ is a tree and all vertex degrees are at least 2.
Consider the path of longest length, say $P = (v_1, v_2, \ldots, v_k)$ which must exist since $\Gamma$ is a finite graph.
Since $\deg v_k \geqslant 2$ there must exist some vertex $v_i \neq v_{k-1}$ which is adjacent to $v_k$.
But since $P$ was the path of longest length, it must be the case that $\exists j < k-1$ such that $v_i = v_j$ and hence $(v_i, v_{i+1}, \ldots, v_k, v_i$ is a cycle in $\Gamma$.
Contradiction since $\Gamma$ is a graph. □

**Theorem 18** *If a graph $\Gamma = (V, E)$ is a tree, then $|E| = |V| - 1$.*

**Proof.** The result clearly holds when $|V| = 1$.
Suppose that $|V| \geqslant 1$ and that the result holds.
Consider adding a vertex $v$ to $\Gamma$ to give the graph $\Gamma^*$. In order not to create a cycle and hence maintain the status of the graph as a tree, $v$ must be connected to $\Gamma$ by exactly one edge. Observe that

$$\begin{aligned} |E| &= |V| - 1 \\ |E^*| - 1 &= (|V^*| - 1) - 1 \\ |E^*| &= |V^*| - 1 \end{aligned}$$

So $\Gamma^*$ is a tree and the property holds. □

The following extension to Theorem 17 can now be made.

**Theorem 19** *If $\Gamma = (V, E)$ is a tree, then at least two vertices have degree 1.*

**Proof.** Suppose the result is false.

If no vertices have degree 1, then since $\Gamma$ is a tree it must be connected so all vertices have degree at least 2. This contradicts Theorem 17.

The remaining case is that one vertex, say $u$, has degree 1, and all others have degree at least 2.
For the vertices of degree at least 2,

$$\begin{aligned} \sum_{\substack{v \in V \\ v \neq u}} \deg v &\geqslant 2\,|V \setminus \{u\}| \\ 2|E| &\geqslant 2\,|V \setminus \{u\}| \\ &\geqslant 2\,(|V| - 1) \\ |E| &\geqslant |V| - 1 \end{aligned}$$

However, this excluded the vertex of degree 1. Adding this in,

$$\begin{aligned} |E| + 1 &\geqslant |V| - 1 \\ |E| &\geqslant |V| \end{aligned}$$

But this is a contradiction since by Theorem 18 $|E| = |V| - 1$. Hence $\Gamma$ is not a tree, and hence by contradiction the theorem must hold. □

So far it has been implicit that all the vertices of the graph are connected together by edges in some way. However, this need not be the case, as a graph may have many different connected component.

**Definition 20** *A graph $\Gamma_1$ is a connected component of the graph $\Gamma$ if*

- *$\Gamma_1$ is a subgraph of $\Gamma$.*
- *$\Gamma_1$ is connected.*
- *$\Gamma_1$ is a subgraph of $\Gamma_2$ is a subgraph of $\Gamma$, then $\Gamma_1 = \Gamma_2$.*

So a connected component is part of a graph which is a graph in its own right. An example is given in Figure 3.

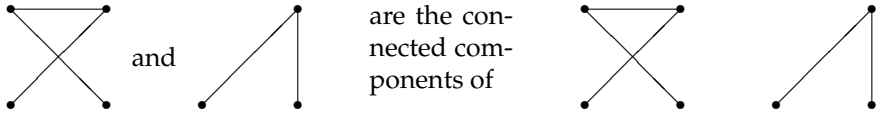


Figure 3: Connected components of a graph

**Theorem 21** *Suppose that $\Gamma$ is a connected graph with a cycle. If an edge of the cycle is removed, then the resulting graph is still connected.*

**Proof.** Let $e = \{v_1, v_2\}$ be an edge of the cycle in $\Gamma$, and let $\Gamma \setminus \{e\} = \Gamma^*$.
Since $\Gamma$ is connected, for any two vertices $u_1$ and $u_k$ there exists a path $P$ between them.
If $P$ does not traverses $e$, then $P$ is a path in $\Gamma^*$.
Suppose that the cycle in $\Gamma$ be $(v_1, v_2, v_3, \ldots, v_n, v_1)$. Since a cycle can be 'started' at any point, this can be done. If $P$ traverses $e$, say $P = (u_1, u_2, \ldots, v_1, v_2, \ldots, u_k)$. Then $(u_1, u_2, \ldots, v_1, v_n, v_{n-1}, \ldots, v_3, v_2, \ldots, u_k)$ is a path connecting $u_1$ and $u_k$ in $\Gamma^*$.
Either way, any two vertices of $\Gamma^*$ are connected, hence the result. □

**Theorem 22** *If $\Gamma$ is an acyclic graph, then $|E| = |V| - p(\Gamma)$ where $p(\Gamma)$ is the number of connected components of $\Gamma$.*

**Proof.** Since $\Gamma$ is acyclic, then for each connected component, $|E_1| = |V_i| - 1$. Summing these up produces the required result. □

**Theorem 23** *For a graph $\Gamma$ the following are equivalent.*

1. *$\Gamma$ is a tree.*
2. *$\Gamma$ is acyclic and $|E| = |V| - 1$.*
3. *$\Gamma$ is connected and $|E| = |V| - 1$.*
4. *For any two vertices, there exists a unique path between them.*

The proof of this theorem is quite straightforward, and it follows directly from (4) that if a new edge is added to $\Gamma$ then a unique cycle is created. If any edge of this cycle is now removed, then the graph is once again a tree.

**Definition 24** *If $\Gamma$ is a connected graph, then a subgraph $\Gamma^*$ is a spanning tree for $\Gamma$ if it is a tree and has the same vertex set as $\Gamma$.*

It is clear to see that the spanning tree of a graph with $n$ vertices has $n - 1$ edges.

**Theorem 25** *A graph $\Gamma^*$ is a spanning tree for a graph $\Gamma$ if and only if $\Gamma^*$ is a minimal connected spanning subgraph of $\Gamma$.*

A 'minimal connected spanning subgraph' is the spanning subgraph with the least edges.

**Proof.** Let $|V| = n$ and let $\Gamma^*$ be an arbitrary connected spanning subgraph of $\Gamma$.
If $\Gamma^*$ is acyclic, then it is a spanning tree so $|E(\Gamma^*)| = n - 1$.
If $\Gamma^*$ is not acyclic, then remove an edge from one of the cycles. The graph is then still connected by Theorem 21. This process can continue until there are no cycles in the resulting graph, $\Gamma^{**}$ say. So $|E(\Gamma^{**})| = n - 1$.□

**Theorem 26** *A graph $\Gamma^*$ is a spanning tree for a graph $\Gamma$ if and only if $\Gamma^*$ is a maximal acyclic subgraph of $\Gamma$.*

**Proof.** Let $\Gamma^*$ be an acyclic subgraph of $\Gamma$. By Theorem 22,

$$\begin{aligned} |E(\Gamma^*)| &= |V(\Gamma^*)| - p(\Gamma) \\ &\leqslant |V(\Gamma)| - 1 \end{aligned}$$

So when $\Gamma^*$ is connected — and so spans $\Gamma$ — it must have $|E(\Gamma^*)| = |V(\Gamma)| - 1$ i.e. be a tree. □

**Theorem 27 (Cayley's Theorem)** *The graph $K_n$ has $m^{m-2}$ spanning trees, where $m \geqslant 2$.*

This is readily verified by considering a few examples, but the proof is not given.

**Definition 28** Comp $x$ *is the set of all vertices of a graph which are in the same connected component as the vertex $x$.*

#### Some Basic Algorithms

Enough theory has been covered at this point to introduce another algorithm.

**Algorithm 29 (Search)** *For a graph $\Gamma = (V, E)$ with a vertex $u \in V$, the algorithm* Search *identifies all vertices of the connected component containing $u$ i.e. finds* Comp $u$*. The algorithm is*

1. *Define the set $P$ such that when ever a vertex $x$ is 'marked', the operation $P := P \cup \{x\}$. Initially $P = \emptyset$.*
2. *Define the set $Q := \{u\}$ and mark $u$.*
3. *Consider any $x \in Q$ and set $Q := Q \setminus \{x\}$.*
4. *For all $y \notin P$ such that $\{x, y\} \in E$, mark $y$, and set $Q := Q \cup \{y\}$. Unless $y$ is already marked.*
5. *If $Q = \emptyset$ then the algorithm has finished. Otherwise repeat from (3).*

For the 'proof' of an algorithm, it is required to show its correctness and its computational complexity.

**Proof.** The correctness of Search is obvious.
For the computational complexity each step is considered in turn.

1. No operation is performed here.
2. One operation in assigning $Q$, and another in marking $u$. Therefore 2.
3. Every vertex will be processed exactly once, therefore these two operations will be performed $|\text{Comp } u|$ times, contributing $2|\text{Comp } u|$ operations.
4. There are three operations here, and this step will be performed once for each edge $\{x, y\} \in E_u$, giving a total of $3|E_u|$.

5. This comparison is performed once for each vertex, since only when all vertices have been processed will $Q = \emptyset$. Therefore $|\operatorname{Comp} u|$.

Summing all these up and since $|E_u| \geqslant |\operatorname{Comp} u| - 1$,

$$\begin{aligned} cc(\operatorname{Search}) &= 2 + 3|\operatorname{Comp} u| + 3|E_u| \\ &\leqslant 5 + 6|E_u| \quad \text{since } |\operatorname{Comp} u| \leqslant |E_u| + 1 \\ &= O(|E_u|) \end{aligned}$$ □

It can be said, therefore, that all the connected components of a graph "can be identified in $O(|E|)$ time".

An obvious application of Search is to the question of connectedness. This may be formulated as "Given any two vertices of a graph, are they connected" i.i. does there exist a path between them. Since Search finds connected components, it can provide a solution to this problem.

#### The Minimal Spanning Tree Problem

Each edge of a graph may be given a weight by some function $w\colon E \to \mathbb{R}$. It is then of interest to find the spanning tree of minimal or maximal weight. The greedy algorithm, Algorithm 1 has already been stated, and in this context is interpreted as

attractive means the edge of least weight.

feasible means an edge that will not create a cycle.

**Theorem 30 (The Greedy Algorithm For The Minimal Spanning Tree Problem)** *Let $\Gamma = (V, E)$ be a connected graph, and $w\colon E \to \mathbb{R}$ be a weight function. Let $(V, F)$ be a subgraph of $\Gamma$. The Greedy Algorithm works for the minimal spanning tree problem in that if $(V, F)$ is a subgraph of a minimal spanning tree, then so is $(V, F \cup \{e\})$ where $e \in E \setminus F$ is chosen by the Greedy algorithm.*

**Proof.** Let $e = \{u, v\}$ be an edge of minimal weight such that $(V, F \cup \{e\})$ is acyclic, i.e. an edge chosen by the Greedy Algorithm.
Observe that $u$ and $v$ must be in different connected components of $(V, F)$, since otherwise $e$ would create a cycle.
Suppose that $T = (V, E^*)$ is a minimal spanning tree.

i. If $e \in E^*$ then $(V, F \cup \{e\}) \subseteq T$ and the statement holds.

ii. If $e \notin E^*$ then $(V, E^* \cup \{e\})$ contains a unique cycle, $\sigma$ say.
Now, $e$ joins connected components, so since it also creates a cycle the must exist an edge $f$ which also joins the same connected components. $f$ cannot be in $(V, F)$ since if it were, these connected components would already be connected together. Hence take $(V, F \cup \{f\})$ which must be acyclic, so at this stage the Greedy algorithm could have chosen $e$ or $f$.
Since $e$ was chosen, $w(e) \leqslant w(f)$.
Let $T' = (V, (E^* \cup \{e\}) \setminus \{f\})$ which must also be a spanning tree for $\Gamma$ — because an edge in a cycle has been added and another removed.
Now, $w(T') = w(T) + w(e) - w(f) \leqslant w(T)$.
But $T$ was a minimal spanning tree, hence $w(T') \geqslant w(T)$.
The only possibility is that $w(T) = w(T')$, and hence $e$ is an edge of a minimal spanning tree. Hence this second case holds.

Both cases hold, so the theorem is proven. □

This has shown that the Greedy Algorithm is correct, but what remains to be shown is its computational complexity.

**Theorem 31** *The computational complexity of the greedy algorithm for the minimal spanning tree problem is $O(|E|^2)$. For a graph $\Gamma = (V, E)$ with weight function $w\colon E \to \mathbb{R}$ the greedy algorithm is formulated as.*

1. *Set $F := \emptyset$ and $p := 0$.*
2. *Use the greedy algorithm to find $e \in E \setminus F$ such that $w(e) = \min\left(w(f) \mid f \in E \setminus F\right)$.*
3. *If $(V, F \cup \{e\})$ is acyclic, then set $F := F \cup \{e\}$ and $p := p + 1$ and continue. Otherwise, just continue.*
4. *If $p = |V| - 1$ then then stop. Otherwise repeat from (2).*

**Proof.** The number of operations at each state is considered in turn.

1. In the initialisation there are only 2 operations.
2. From all the edges available in $E \setminus F$ it is necessary to find the one with the least weight. Each of these edges, say $e_1, e_2, \ldots, e_k$ is assessed in turn.

$$\begin{array}{lll} & a_1 & r := 1 \quad \text{1 operation} \\ \text{if} & a_2 < a_r & r := 2 \quad \text{at most 2 operations} \\ \text{if} & a_3 < a_r & r := 3 \quad \text{at most 2 operations} \\ & \vdots & \vdots \\ \text{if} & a_k < a_r & r := k \quad \text{at most 2 operations} \end{array}$$

   The required edge will now have been found. The number of operations used is now assessed.

$$\begin{aligned} 2(k-1) + 1 &= 2k - 1 \\ &= O(k) \\ &= O\left(E \setminus F\right) \\ &= O(|E|) \quad \text{since } E \setminus F \subseteq E \end{aligned}$$

   There are two other operations in (2) where the set $E$ is adjusted. This gives a total of $2 + O(|E|)$.

3. It is now necessary to check if $(V, F \cup \{e\})$ is acyclic and for this it is first necessary to show that $(N, F \cup \{e\})$ is acyclic $\Leftrightarrow$ where $e = \{u, v\}$, $u$ is not connected to $v$ in $(N, F)$. It is easier to prove the negation.

   $\Rightarrow$ Suppose that $(N, F \cup \{e\})$ has a cycle $\sigma$, then $e$ is an edge in $\sigma$. It is clear therefore that $u$ is connected to $v$ in $(N, F)$.

   $\Leftarrow$ If $u$ is connected to $v$ in $(N, F)$ then a path between them exists in $(N, F)$. Hence $(N, F \cup \{e\})$ has a cycle.

   The Search algorithm provides a way to check for connectedness between two vertices, so this condition can be verified with $O(|F|)$ operations.

   But $|F| \leqslant |E| - 1$ since otherwise the tree $(V, F)$ would be the original graph $\Gamma = (V, E)$ and the algorithm would have finished. Since there are 4 other operations in this step, the computational complexity provided here is $4 + O(|E - 1|)$.

4. The final step (and hence the loop) can be repeated at most $|E|$ times.

Adding up the computational complexities from each stage,

$$\begin{aligned} \text{computational complexity} &\leqslant 2 + |E|\,(O|E| + O(|E| - 1) + 6) \\ &\leqslant |E|O(|E|) \\ &\leqslant O\left(|E|^2\right) \end{aligned}$$

□

#### (15.1.5) Assignment Problems

Assignment problems involve creating a one to one mapping, say of workers to jobs, that maximises efficiency, say, or minimises some quantity.

The total number of options can be represented by a complete bipartite graph, and the weight of the edges represents the efficiency or whatever of the assignment it represents.

The problem can also be represented by a matrix, where the $ij$ element is the weight of the edge $\{i, j\}$.

**Example 32** *Three taxis are located at distances from three customers as shown in table 32.*

| | $c_1$ | $c_2$ | $c_3$ |
|---|---|---|---|
| $t_1$ | 1 | 5 | 2 |
| $t_1$ | 4 | 4 | 3 |
| $t_1$ | 2 | 2 | 4 |

Table 1: Distances from taxis to customers

*Which taxi should go to which customer in order to minimise the total distance traveled by the taxis?*

**Proof.** Solution An exhaustive method is adopted for illustrative purposes. The permutation "taxi $i$ goes to customer $j$" is represented by $\pi(i) = j$, and this is expressed using disjoint cycles.

| $\pi$ | $w(A, \pi)$ | $\pi$ | $w(A, \pi)$ |
|---|---|---|---|
| (1)(2)(3) | 1+4+4=8 | (3)(12) | 5+4+4=13 |
| (1)(23) | 1+3+2=6 | (123) | 5+3+2=10 |
| (2)(13) | 2+4+2=8 | (132) | 2+4+4=10 |

Table 2: Possible solutions to the taxi problem

□

The table shows all possible permutations, and it is clear that (1)(23) is the best solution to this problem.

#### Classical Assignment

Example 32 is an instance of the classical assignment problem, which may be formulated as follows.

**Definition 33** *Given a square $(n \times n)$ matrix, find the elements such that*

1. *no two of the chosen elements are in the same row or column.*
2. *the sum of the elements is minimal (maximal).*

Where $P_n$ denotes the set of all permutations of $n$ items and $\pi \in P_n$,

$$w(A, \pi) = \sum_{i=1}^{n} a_{i,\,\pi(i)}$$

It is then necessary to find

$$\min_{\pi \in P_n} w(A, \pi) = \min A$$

The set $ap(A)$ is the set of optimal permutations, and $AP^{\min}$ or $AP^{\max}$ represent the chosen solution, depending on whether it is of the maximal or minimal variety.

Without loss of generality, from this point on only the minimisation problem will be considered. The maximisation problem can be solved by considering the matrix $-A$.

The solution to classical assignment utilises the following property.

- If a constant $\alpha$ is added to every element of a single row or a single column of a matrix $A$ to give $A'$, then $w(A', \pi) = w(A, \pi) + \alpha$ and so $ap(A') = ap(A)$.

Definition 34 *A $n \times n$ matrix is in normal form if $a_{ij} \geqslant 0\ \forall a_{ij}$, and $A$ contains $n$ independent zeros i.e. it is possible to find $n$ zeros in $A$ which do not lie in the same row or column.*

Notice that when a matrix is in normal form, $\min A = 0$ and since there are no negative elements, it is simple to read off the optimal permutation.

Theorem 35 (König-Egervary) *Where a 'line' is either a row or column, the maximal number of independent zeros in a matrix is equal to the minimal number of lines required to cover all the zeros of the matrix.*

Algorithm 36 (The Hungarian Method* ) *The Hungarian method works by transforming the $n \times n$ matrix, say $A$, into normal form as follows.*

1. *(a) Subtract from every column of $A$ the constant that is the least element of that column, giving matrix $A_1$*
   *(b) Subtract from every row of $A_1$ the constant that is the lease element of that row, giving matrix $A_2$.*
2. *Say there are at most $k$ independent zeros in $A_2$ — this needs to shown using Theorem 35.*
   *(a) if $k = n$ then stop.*
   *(b) if $k < n$ proceed as follows.*
      *i. Cover all zeros in the matrix by the $k$ lines found. Suppose that there are $k_r$ rows and $k_c$ columns, so that $k = k_r + k_c$.*
      *ii. Of all uncovered elements, find the least, $t$.*
      *iii. Subtract $t$ from each uncovered row.*
      *iv. Add $t$ to each covered column*
      *v. consider the matrix created, $A_3$, and repeat the process from 2.*

There is no guarantee that the number of independent zeros will increase at every iteration, so it is important to shown that the Hungarian method is finite.

Theorem 37 *When the matrix in question is integer, the Hungarian method for solving the classical assignment problem is finite in that it terminates after a finite number of iterations.*

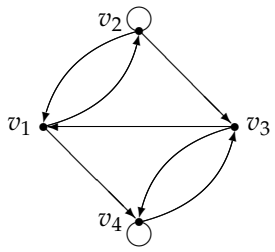


Figure 4: A digraph

Proof. From the general step, where the new $(n \times n)$ matrix created is $A'$,

$$\begin{aligned} w(A', \pi) &= w(A, \pi) + tk_c - t(n - k_r) \\ &= w(A, \pi) + t(k_c + k_r - n) \end{aligned}$$

but $k_c + k_r = k < n$, so

$$< w(A, \pi)$$

if $A$ is integer, i.e. all its entries are integers, then since $w(A', \pi) < w(A, \pi)$, it must be the case that

$$w(A', \pi) \leqslant w(A, \pi) - 1$$

But this relationship must hold for each iteration. Hence

$$\begin{aligned} \text{Initially} \quad & w(A, \pi) = \min A \\ \text{After the first iteration} \quad & w(A', \pi) \leqslant \min A - 1 \\ \text{After the second iteration} \quad & w(A', \pi) \leqslant \min A - 2 \\ \vdots \qquad & \vdots \\ \text{After the } n\text{th iteration} \quad & w(A', \pi) \leqslant \min A - n \end{aligned}$$

The algorithm stops when $\min A' = 0$, which can take at most $\min A$ iterations.
Hence the algorithm is finite and the theorem is proved. □

It can be shown that the computational complexity of the Hungarian method is $O(n^2)$.

### (15.1.6) Digraphs — Directed Graphs

A directed graph is identical to a normal graph with the exception that the edges are ordered pairs of vertices, say $(e, f)$ instead of $\{e, f\}$. This means that the edges have direction.

The directed edges makes the possibility of loops meaningful. Some of the terms from graph theory are modified for use in a digraph. A path may now contain repeated vertices, although a 'normal' path may exist and is now called an elementary path. Similarly, cycles may contain repeated vertices, and a 'normal'

cycle is called an elementary cycle.

For example, in the digraph in Figure 4, $(v_1, v_2, v_2, v_3, v_4)$ is a path, and $(v_4, v_3, v_1, v_2, v_3, v_1, v_4)$ is a cycle.

For a sizable digraph with a not inconsiderable number of vertices it is clear that the shortest path from one vertex to another is not a trivial matter. Add to this the complexities imposed by a weighting function, and the problem soon becomes rather unwieldy. For a digraph $F = (V, E)$ with weighting function $d\colon E \to \mathbb{R}$, the weight of a path $P = (v_0, v_1, \ldots, v_k)$ is $d(P) = \sum_{i=1}^{k} d(v_{i-1}, v_i)$. The shortest path problem is to find the path of minimal weight (or length) between two given vertices.

A solution is found by finding the shortest path from the start vertex $s$ to all other vertices. For a set of vertices $W \subseteq V$, define $d_W(v)$ to be length of the shortest $sv$ path with all intermediate vertices in $W$. This is the temporary shortest distance. By extending $W$ from $\{s\}$ to $V$ all the shortest paths can be found and hence solve the problem.

Theorem 38 *Choose $x \in V \setminus W$ such that $d_W(x) = \min\{d_W(v) \mid v \in V \setminus W\}$ i.e. the shortest $sv$ path. Let $W' = W \cup \{x\}$ then for $y \in V \setminus W$, $d_{W'}(y) = \min\{d_W(y),\ d_W(x) + d(x, y)\}$*

In order to prove this, the following lemma is required

Lemma 39 *If $x$ is a vertex selected by the process in Theorem 38, then $d_W(x) = d_V(x)$.*

Proof. Proof Of Lemma 39 A temporary shortest distance must be greater or equal to the actual shortest distance, hence $d_W(x) \geqslant d_V(x)$.
Suppose that $d_W(x) > d_V(x)$. Then there exists a path $P$ with $d(P) = d_V(x)$ that has at least one intermediate vertex not in $W$. Let $z$ be the first vertex of $P$ that is not in $W$, then $P$ can be written as $P' + P''$ where $P'$ is as $sz$ path in $W$ and $P''$ is the rest of the path.
Hence $d(P') + d(P'') = d(P) < d_W(x)$. But $d(P'') > 0$ so $d(P') < d_W(P)$
Now, $x$ was the vertex (not in $W$) that had the temporary shortest distance. But $z \notin W$ and $d(P') < d_W(P)$ where by construction $P'$ is an $sz$ path with intermediate vertices in $W$ which is shorter.
Hence by contradiction the lemma holds. □

Proof. Proof Of Theorem 38 Consider $y \in V \setminus W$. $d_W(y)$ and $d_W(x) + d(x, y)$ are the lengths of two $sy$ paths with intermediate vertices in $W'$, and so

$$d_{W'}(y) \leqslant \min\{d_W(y),\ d_W(x) + d(x, y)\}$$

Now, let $P$ be the shortest $sy$ path with intermediate vertices in $W'$, so $d(P) = d_{W'}(y)$. The following three cases correspond to those shown in Figure 5.

case 1 If $x \notin P$ then $P$ has all intermediate vertices on $W$ and hence $d(P) = d_W(y)$.

case 2 (a) If $x$ is the penultimate vertex of $P$, then let $P' = (s, \ldots, x)$ which is the shortest $sx$ path with intermediate vertices in $W$. Hence $d(P') = d_W(x)$ and so $d(P) = d_W(x) + d(x, y)$.

(b) Let $P$ be the shortest $sy$ path with all intermediate vertices in $W'$. Since $x$ is not penultimate, $P$ may be expressed as $P = P' + P''$ where

- $P'$ is an $sx$ path with all intermediate vertices in $W$.
- $P''$ is an $sy$ path with all intermediate vertices in $W$.

Say the penultimate vertex on $P''$ is $z$, then $z \in W$. This means that $z$ must have been chosen earlier, so there exists a shorter $sz$ path than using $P$. This contradicts that $P$ is the shortest $sy$ path, so in this case $d_{W'}(y) = d_W(y)$.

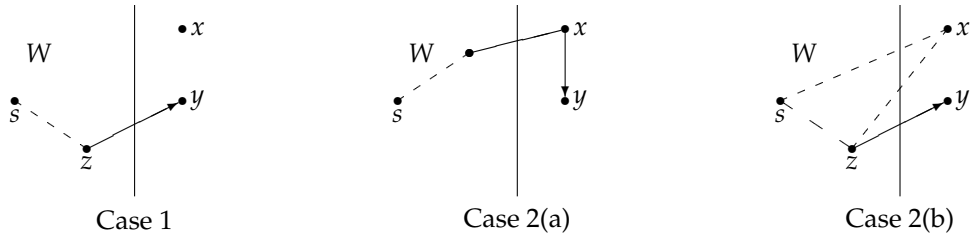


Figure 5: Pictorial representation of Theorem 38.

In all cases, the shortest $sy$ path is one of $d_W(y)$ and $d_W(x) + d(x, y)$. Hence

$$d_{W'}(y) = \min\{d_W(y),\ d_W(x) + d(x, y)\}$$

and so the theorem is proved. □

Having shown that this process of finding shortest paths works, it is now possible to formulate the algorithm for putting this result into practice.

Algorithm 40 *Dijkstra For a digraph $F = (V, E)$ and weight function $d\colon E \to \mathbb{R}$ the shortest paths from the vertex $s$ are found as follows.*

1. (a) *Set $W := \{s\}$.*
   (b) *Set $d(x) := d(s, x)$ if $(s, x) \in E$ else $d(x) = \infty$.*
2. *$\forall x \in V \setminus W$ find $x'$ such that $d_W(x') = \min\{d_W(x)\}$.*
3. *Set $W := W \cup \{x'\}$.*
4. *For all $y \in V \setminus W$ set $d_W(y) := \min\{d_W(y),\ d_W(x') + d(x', y)\}$.*
5. *If $W = V$ then stop, else repeat from 2.*

Proof. The correctness of the algorithm is shown by Theorem 38.

The computational complexity is calculated for each step in turn.

1. 1 for the assignment, and $|V| - 1 = n - 1$ for the checking of vertices.
2. Finding $x'$ can be done in at most $|V \setminus W|$ steps, which is $O(n)$.
3. As assignment and a union, making 2.
4. An addition, a minimum, and an assignment. Each of these must be done for $|V \setminus W|$ vertices, giving $3O(n) = O(n)$.
5. One operation. Note that the loop will be made $n - 1$ times.

Hence the total computational complexity is

$$\begin{aligned}\text{computational complexity} &= 1 + (n-1) + (n-1)\left(O(n) + 2 + O(n)\right)\\ &\leqslant O(n) + (O(n))^2\\ &= O(n) + O(n^2)\\ &= O(n^2)\end{aligned}$$

Hence the computational complexity is $O(n^2)$. □

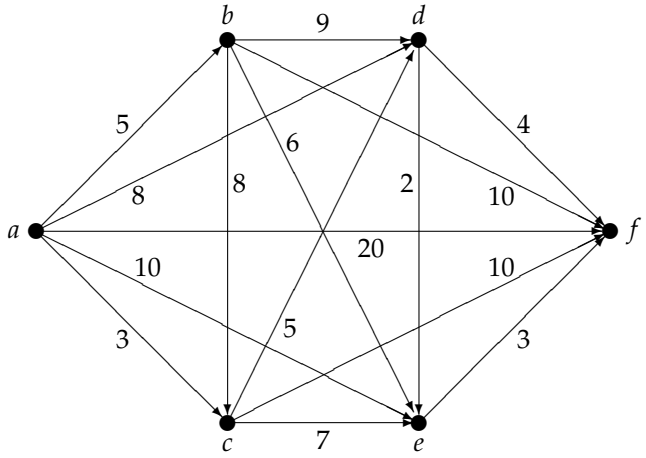


Figure 6: Find the shortest path from $a$ to $f$. See Example 41.

| Iteration | Temporary Shortest Distances | Vertex Chosen |
|---|---|---|
| 1 | | $a$ |
| 2 | $b_5$ $c_3$ $d_8$ $e_{10}$ $f_{20}$ | $c_3$ |
| 3 | $d_8$ $e_{10}$ $f_{13}$ | $b_5$ |
| 4 | $d_{14}$ $e_{11}$ $f_{15}$ | $d_8$ |
| 5 | $e_{12}$ $f_{10}$ | $f_{10}$ |
| 6 | | $e_{10}$ |

Table 3: A run of Dijkstra's Algorithm for Example 41.

Dijkstra's algorithm is relatively complicated, and when it has finished the task remains to actually find the optimal solution. First of all an example is given

Example 41 *Find the shortest path from $a$ to $f$ in the weighted digraph shown in Figure 6.*

Proof. Solution The run of Dijkstra's algorithm is shown in Table 3. The notation $a_b$ is used to show that the temporary shortest distance to vertex $a$ is $b$.

Notice that the algorithm only finishes when all the vertices have been chosen. The remaining task is to actually find the shortest path, though clearly it has length 10. This is done as follows.

(i) The vertices going into $f$ have weights 2, 10, 20, 10, and 3. $10 - 2 = 8$ and one of the temporary shortest distances to a chosen vertex, namely $d$ is 8. Therefore $d$ comes before $f$.

(ii) The vertices going into $d$ have weights 9, 8, and 5. But the shortest distance to $d$ is 8, so the preceding vertex must be $a$/

One shortest path is therefore $a, d, f$, though there may be more. □

### (15.1.7) The Transportation Problem

As the name suggests, the transportation problem concerns how to efficiently move goods from producers to consumers. However, there are many other applications of the problem.

Th information used for the transportation problem may be set out in tabular form, as shown in Table 4. The first column shows the amounts produced by the producers, and the first row shows the amounts

| | $b_1$ | $b_2$ | $\cdots$ | $b_n$ |
|---|---|---|---|---|
| $a_1$ | $c_{11}$ | $c_{12}$ | $\cdots$ | $c_{1n}$ |
| $a_2$ | $c_{21}$ | $c_{22}$ | $\cdots$ | $c_{2n}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $a_m$ | $c_{m1}$ | $c_{m2}$ | $\cdots$ | $c_{mn}$ |

Table 4: Information for the transportation table.

demanded by the consumers. The central values, $c_{ij}$, show the cost of moving a unit from producer $i$ to consumer $j$.

The problem now is to find numbers $x_{ij}$, the amount of goods to be taken from producer $i$ to customer $j$. The problem can be represented on a bipartite graph—bipartite because taking goods from producer to producer or consumer to consumer is clearly not an option. Suppose the producers are $A_i$s with production $a_i$ and similarly the consumers be $B_j$ with demand $b_j$ then clearly the transportation $c_{ij}$ is equivalent to the edge $\{A_i, B_j\}$.

The objective is to minimise the total cost of transportation, i.e. $\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}c_{ij}$. There are however the obvious constraints can a producer cannot exceed capacity, and consumers demand as much as is stated and this demand must be met.

**Model 42** *The transportation problem may be formalised as follows. Minimise the quantity* $f(X) = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}c_{ij}$ *subject to the constraints*

$$\sum_{j=1}^{n} x_{ij} = a_i \quad \textit{so supplies must be used} \tag{43}$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad \textit{so demands must be met} \tag{44}$$

$$x_{ij} \geqslant 0 \quad \textit{for the sake of reality} \tag{45}$$

*It is also assumed that supplies and demands are positive i.e.* $a_i \geqslant 0\ \forall i$ *and* $b_j \geqslant 0\ \forall j$.

$f(X)$ is called the objective function, and $X$ represents the set of $x_{ij}$s of the solution under consideration. A solution satisfying equations (43), (44), and (45) is called feasible, and when $f(X)$ attains its minimum the solution is optimal. Note that $f(X)$ does attain its minimum since the set of all possible solutions is finite. Hence if a feasible solution exists, then an optimal solution exists.

**Theorem 46** *The transportation has a feasible solution if and only if* $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$.

**Proof.** The theorem has two parts, so each is taken in turn,

$\Rightarrow$ If a solution $X$ is feasible then,

$$\begin{aligned}\sum_{i=1}^{m} a_i &= \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \quad \text{by feasibility} \\ &= \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij} \quad \text{property of sums} \\ &= \sum_{j=1}^{n} b_j \quad \text{by feasibility}\end{aligned}$$

Hence the 'only if' part.

$\Leftarrow$ Suppose that $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = t$, say.

$$\begin{aligned}\text{set} \quad & x_{ij} = \frac{a_i b_j}{t} > 0 \quad \text{so (45) holds} \\ \text{then} \quad & \sum_{j=1}^{n} x_{ij} = \frac{a_i}{t}\sum_{j=1}^{n} b_j = a_i \quad \text{so (44) holds} \\ \text{and} \quad & \sum_{i=1}^{m} x_{ij} = \frac{b_j}{t}\sum_{i=1}^{m} a_i = b_j \quad \text{so (43) holds}\end{aligned}$$

Furthermore, this proof has revealed a feasible solution to be $x_{ij} = \dfrac{a_i b_j}{t}$. □

Clearly there is a problem if $\sum_{i=1}^{m} a_i \neq \sum_{j=1}^{n} b_j$. In this case a 'dummy producer' or a 'dummy consumer' can be introduced with the required capacity or demand. The costs of transportation to these are zero. Hence without loss of generality it is assumed that $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$.

The task of finding a feasible solution and in turn an optimal solution is now set about. Note the following definitions.

**Definition 47** *A solution to the transportation problem is*

*(i) basic if the associated graph is acyclic.*

*(ii) degenerate if the associated graph is not connected.*

*(iii) non-degenerate if the associated graph is a spanning tree.*

First of all a basic feasible solution is sought. Note that a degenerate solution can be extended to a non-degenerate one by adding edges of zero weight.

**Definition 48** *Let $X$ be a basic feasible solution. The base of $X$ is the set $\mathcal{B} \subseteq (1,2,\ldots m) \times (1,2,\ldots n)$ such that*

*(i)* $x_{ij} = 0 \ \forall (i,j) \notin \mathcal{B}$.

*(ii) The set of vertices* $\left\{ A_i B_j \mid (i,j) \in \mathcal{B} \right\}$ *form a spanning tree for the associated graph, $K_{nm}$.*

If a basic feasible solution $X$ is already a spanning tree then the base is unique, but if it is degenerate then it may be possible to extend to a non-degenerate solution in many ways. A solution can be checked for cycles in the transportation table directly—without having to draw the associated graph. Since the graph is bipartite it is readily deduced that a cycle must join non-zero cells in the table by alternating vertical and horizontal lines.

**Definition 49** *The dual solution to the transportation problem is the vector* $\begin{pmatrix}\alpha_1 & \alpha_2 & \dots & \alpha_m & \beta_1 & \beta_2 & \dots & \beta_n\end{pmatrix}$ *where* $\alpha_i + \beta_j = c_{ij} \forall (i,j) \in \mathcal{B}$.

By fixing $\beta_n = 0$ the dual solution can be easily found.

**Theorem 50 (The Optimality Criterion)** *Let* $X$ *be a basic feasible solution, and* $\begin{pmatrix}\alpha_1 & \alpha_2 & \dots & \alpha_m & \beta_1 & \beta_2 & \dots & \beta_n\end{pmatrix}$ *be the dual solution. If* $\alpha_i + \beta_j \leqslant c_{ij}\ \forall i\ \forall j$ *then* $X$ *is an optimal solution.*

**Proof.** Consider some arbitrary feasible solution $Y$.

$$\begin{aligned}
f(Y) &= \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}y_{ij} \\
&\geqslant \sum_{i=1}^{m}\sum_{j=1}^{n} (\alpha_i + \beta_j)y_{ij} \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n} \alpha_i y_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{n} \beta_j y_{ij} \\
&= \sum_{i=1}^{m} \alpha_i a_i + \sum_{j=1}^{n} \beta_j b_j \quad \text{since } Y \text{ is feasible} \\
&= \sum_{i=1}^{m} \alpha_i \sum_{j=1}^{n} x_{ij} + \sum_{j=1}^{n} \beta_j \sum_{i=1}^{m} x_{ij} \quad \text{since } X \text{ is feasible} \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n} (\alpha_i + \beta_j)x_{ij} \\
&= \sum_{(i,j)\in\mathcal{B}(X)} (\alpha_i + \beta_j)x_{ij} + \sum_{(i,j)\notin\mathcal{B}(X)} (\alpha_i + \beta_j)x_{ij} \\
&= \sum_{(i,j)\in\mathcal{B}(X)} c_{ij}x_{ij} + \sum_{(i,j)\notin\mathcal{B}(X)} (\alpha_i + \beta_j)x_{ij} \\
&= \sum_{(i,j)\in\mathcal{B}(X)} c_{ij}x_{ij} \quad \text{since } (i,j) \notin \mathcal{B}(X) \Rightarrow x_{ij} = 0 \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\
f(Y) &\geqslant f(X)
\end{aligned}$$

Hence $X$ is an optimal solution since its cost is at most the cost of any other feasible solution. □

Knowing the optimality criterion allows feasible solutions to be checked for being optimal. If they are not then it is necessary to find a different one and check that. Indeed, a method for finding a single feasible solution has not jet been exhibited.

**Algorithm 51 (The North-West Corner Rule)** *The north-west corner rule finds a basic feasible solution as follows*

1. *Define* $(r,s)$ *as the top left most 'available' cell in the transportation table.*
2. *Assign* $x_{rs} := \max(a_r, b_s)$.
3. *Let* $a_r := a_r - x_{rs}$ *and make unavailable row* $r$ *if* $a_r = 0$.
4. *Let* $b_s := b_s - x_{rs}$ *and make unavailable columns* $s$ *if* $b_s - x_{rs} = 0$.
5. *If no cell is available the algorithm stops, else repeats from 1.*

*It is claimed that this process will always find a basic feasible solution.*

Proof. By construction the solution is feasible. What remains to be shown is that the solution is basic i.e. the associated graph is acyclic.
Suppose the solution found has a cycle, and let $A_iB_j$ be the top left most edge of the cycle. There must be edges $A_rB_j$ and $A_iB_s$ with non-zero entries in the transportation table in order for a cycle to exist, note $r > i$ and $s > j$.
By hypothesis $A_iB_j$ is chosen first, and either row $i$ or column $j$ or both are made unavailable. Hence at least one of $A_rB_j$ and $A_iB_s$ cannot also be chosen.
Hence by contradiction the theorem is proved. □

Having found a basic feasible solution and found it not to be optimal, a method must now be found to move to an optimal solution. Since the optimality criterion is violated, $\exists i\ \exists j \alpha_i + \beta_j > c_{ij}$. It is now sought to include the offending cell in the solution, it will then contribute to the $\alpha$s and $\beta$s and so help get rid of the violation of the optimality criterion.

By analogy with the associated graph, observe that an edge can be added and one from the resulting cycle removed leaving another spanning tree. By adding an edge $A_kB_l$ say, one of the edges $A_rB_l$ and $A_kB_s$ must be removed, hence let $x_{kl} = \theta$ be the minimum of all the $x_{ij}$ on the cycle and subtract $\theta$ from each of the co-linear and co-columnar edges to $A_kB_l$. But then the solution nolonger works, so $\theta$ has to be added to the vertices co-linear or co-columnar to those from which $\theta$ was subtracted... The following algorithm therefore arises

$$x'_{ij} = \begin{cases} x_{ij} - \theta & \text{if } A_iB_j \text{ is an even arc on the cycle} \\ x_{ij} + \theta & \text{if } A_iB_j \text{ is an odd arc on the cycle} \\ x_{ij} & \text{otherwise} \end{cases}$$

Finally, the transportation problem is illustrated by means of a short example.

Example 52 *Three factories have capacities 100, 70, and 50. Four cities demand the product in the amounts 30, 50, 50, and 90. The costs of transport are set out in Table 5.*

| | 30 | 50 | 50 | 90 |
|---|---|---|---|---|
| 100 | 5 | 6 | 8 | 3 |
| 70 | 7 | 6 | 2 | 3 |
| 50 | 6 | 7 | 4 | 3 |

Table 5: Table of costs for Example 52

*Find the optimal way to transport the goods to the cities.*

Proof. Solution Applying the north-west corner rule, Table 6 is produced. The small numbers in the top right of each cell are the transport costs, and the main entries are the $x_{ij}$s.

| | 30 | 50 | 50 | 90 | |
|---|---|---|---|---|---|
| 100 | 30 $^{5}$ | 50 $^{6}$ | 20 $^{8}$ | $\star$ $^{3}$ | $\alpha_1 = 9$ |
| 70 | $^{7}$ | $^{6}$ | 30 $^{2}$ | 40 $^{3}$ | $\alpha_2 = 3$ |
| 50 | $^{6}$ | $^{7}$ | $^{4}$ | 50 $^{3}$ | $\alpha_3 = 3$ |
| | $\beta_1 = -4$ | $\beta_2 = -3$ | $\beta_3 = -1$ | $\beta_4 = 0$ | |

Table 6: First stage in the solution of the transportation problem

Observe that the optimality criterion fails in the cell marked with a '$\star$'. Consider the $(1, 4)$ cell, then taking $\theta = 20$ the solution is modified to that shown in Table 7

| | 30 | 50 | 50 | 90 | |
|---|---|---|---|---|---|
| 100 | 30 $^{5}$ | 50 $^{6}$ | $^{8}$ | 20 $^{3}$ | $\alpha_1 = 2$ |
| 70 | $^{7}$ | $\star$ $^{6}$ | 50 $^{2}$ | 20 $^{3}$ | $\alpha_2 = 3$ |
| 50 | $^{6}$ | $^{7}$ | $^{4}$ | 50 $^{3}$ | $\alpha_3 = 3$ |
| | $\beta_1 = 3$ | $\beta_2 = 4$ | $\beta_3 = -1$ | $\beta_4 = 0$ | |

Table 7: First modified solution

In Table 7 the optimality criterion fails again. Now consider the $(2, 2)$ cell, and let $\theta = 20$. The results of this modification are shown in Table 8

| | 30 | 50 | 50 | 90 | |
|---|---|---|---|---|---|
| 100 | 30 $^{5}$ | 30 $^{6}$ | $^{8}$ | $^{3}$ | $\alpha_1 = 3$ |
| 70 | $^{7}$ | 20 $^{6}$ | 50 $^{2}$ | 40 $^{3}$ | $\alpha_2 = 3$ |
| 50 | $^{6}$ | $^{7}$ | $^{4}$ | 50 $^{3}$ | $\alpha_3 = 3$ |
| | $\beta_1 = 2$ | $\beta_2 = 3$ | $\beta_3 = -1$ | $\beta_4 = 0$ | |

Table 8: Second modified solution

In Table 8 the optimality criterion holds, so the solution is complete. □

## (15.2) Linear Programming

### (15.2.1) Formulation & Terminology

The purpose of linear programming is to find the best solution to a single linear equation, subject to a number of constraints on the variables. When there is only one equation in two unknowns the problem can be solved graphically in the plane, but for more unknowns the problem requires a theoretical solution.

While linear programming problems may appear in many forms, it is preferable to convert them to a standard form, which can then be solved without loss of generality.

**Model 53 (The Linear Programming Problem)** *Minimise the equation*

$$c_1 x_1 + c_2 x_2 + \dots c_n x_n$$

*subject to the constraints*

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\gtreqless b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\gtreqless b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\gtreqless b_m \end{aligned}$$

*where '$\gtreqless$' is one of '$\geqslant$', '$=$', and '$\leqslant$'. Furthermore, $x_i \geqslant 0 \ \forall i$*

This can be expressed in matrix form, as minimise $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$ subject to the constraints $A\mathbf{x} = \mathbf{b}$. However, clearly equality on the latter equation is a problem since some constraints involve inequality. Also, it is not necessarily the case that all variable will be non-negative, as is another requirement.

To overcome these problems, 'slack' variables are introduced. This works as follows,

$$g_i(\mathbf{x}) \leqslant b_i \;\Leftrightarrow\; g_i(\mathbf{x}) + x_{n+1} = b_i \quad \text{for some} \quad x_{n+1} \geqslant 0$$
$$g_i(\mathbf{x}) \geqslant b_i \;\Leftrightarrow\; g_i(\mathbf{x}) - x_{n+1} = b_i \quad \text{for some} \quad x_{n+1} \geqslant 0$$

Similarly, the non-negativity criterion can be solved by re-defining

$$x_k = x_{n+1} - x_{n+2} \qquad 1 \leqslant k \leqslant n$$

Having standardised the problem in this way, a number of new variables will have been introduced. It is often convenient to re-define the subscripts so they run from 1 to some new value of $n$.

Definition 54 *For a linear programming problem in standard form,*

1. $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$ *is called the objective function.*
2. *Every* $\mathbf{x} \in \mathbb{R}^n$ *satisfying* $A\mathbf{x} = \mathbf{b}$ *and* $x_i \geqslant 0\ \forall i$ *is called a feasible solution.*
3. *The set of feasible solutions is denoted by* $M$*, and* $M^{\text{opt}}$ *is the set of all optimal solutions. Hence*

$$M^{\text{opt}} = \{\mathbf{x} \in M \mid f(\mathbf{x}) \leqslant f(\mathbf{z})\ \forall \mathbf{z} \in M\}$$

4. $\mathbf{a}_i$ *is the column vector whose entries are the ith row of the matrix* $A$*.*
5. $\mathbf{A}_j$ *is the column vector whose entries are the jth column of the matrix* $A$*.*

Notice that maximisation can be performed by minimising the negation of the objective function. Recall that $A$ is an $m \times n$ matrix. From the above definitions it is evident that

$$A\mathbf{x} = \mathbf{b} \quad \Leftrightarrow \quad \sum_{j=1}^{n} x_j\mathbf{A}_j = \mathbf{b} \quad \Leftrightarrow \quad \sum_{j=1}^{n} a_{ij}x_j = b_j\ \forall i \quad \Leftrightarrow \quad \mathbf{a}_i^T\mathbf{x} = b_i\ \forall i$$

### Finding A Solution

There is nothing to say that $A\mathbf{x} = \mathbf{b}$ must have only one solution—if $m < n$ then there are necessarily many solutions. However, it may be assumed that the rank of $A$ is $m$ since linearly dependent rows could be removed. From this it follows that there must exist a non-singular $m \times m$ submatrix of $A$, $B$ say. Hence $B = \begin{pmatrix}\mathbf{A}_{j_1} & \mathbf{A}_{j_2} & \dots & \mathbf{A}_{j_m}\end{pmatrix}$ for some $1 \leqslant j \leqslant n - m$.

Definition 55 *Where* $B$ *is a non-singular submatrix of* $A$*,* $B = \begin{pmatrix}\mathbf{A}_{j_1} & \mathbf{A}_{j_2} & \dots & \mathbf{A}_{j_m}\end{pmatrix}$ *then,*

1. *The indices* $j_1, j_2, \dots, j_m$ *are defined by the function* $\mathcal{B}(i) = j_i$ *for* $1 \leqslant i \leqslant m$*.*
2. *The set* $\mathcal{B} = \left\{\mathbf{A}_{j_1}, \mathbf{A}_{j_2}, \dots, \mathbf{A}_{j_m}\right\}$ *is called a basis for* $A$*, as the vectors span the column space of* $A$*.*
3. *The indices of the basis, i.e.* $j_1, j_2, \dots, j_m$ *are referred to as 'basic'.*
4. $\mathbf{x}_{\mathcal{B}} = \begin{pmatrix}x_{j_1} & x_{j_2} & \dots & x_{j_m}\end{pmatrix}^T$ *is the component of* $\mathbf{x}$ *with basic indices.*
5. $\mathbf{c}_{\mathcal{B}} = \begin{pmatrix}c_{j_1} & c_{j_2} & \dots & c_{j_m}\end{pmatrix}^T$ *is the component of* $\mathbf{c}$ *with basic indices.*

Using this notation, it is evident that

$$\mathbf{b} = A\mathbf{x} = \sum_{j=1}^{n} \mathbf{A}_j x_j = \sum_{\mathbf{A}\in\mathcal{B}} \mathbf{A}_j x_j + \sum_{\mathbf{A}\notin\mathcal{B}} \mathbf{A}_j x_j = B\mathbf{x}_{\mathcal{B}} + \sum_{\mathbf{A}\notin\mathcal{B}} \mathbf{A}_j x_j$$

Hence a basic solution $\mathbf{x}$ satisfies $B\mathbf{x}_{\mathcal{B}} = \mathbf{b}$ and $x_j = 0$ for $\mathbf{A}_j \notin \mathcal{B}$. If also the entries of $\mathbf{x}_{\mathcal{B}}$ are non-negative then it is a basic feasible solution, and if $x_{\mathcal{B}(i)} > 0$ then it is non-degenerate. In summary

- A basic feasible solution has zero entries bar for the $m$ entries in the base positions. If any of these $m$ entries are zero, then the basic feasible solution is degenerate.

Having found a solution, its optimality now needs to be assessed.

**Theorem 56 (Optimality Criterion)** *Let* $z_j = \mathbf{c}_{\mathcal{B}}^T B^{-1}\mathbf{A}_j$ *and let* $\mathbf{z} = \begin{pmatrix} z_1 & z_2 & \dots & z_n \end{pmatrix}^T$. *If* $\mathbf{x}$ *is a basic feasible solution and* $c_j - z_j \geqslant 0 \forall j$ *then* $\mathbf{x} \in M^{\text{opt}}$.

**Proof.** Let $\mathbf{y}$ be a feasible solution, so $A\mathbf{y} = \mathbf{b}$ and $y_j \geqslant 0 \forall j$.
By hypothesis $c_j - z_j \geqslant 0 \forall j$, so $\mathbf{c} - \mathbf{z} \geqslant \mathbf{0}$.
Hence $(\mathbf{c} - \mathbf{z})\mathbf{y}^T \geqslant 0$ and so $\mathbf{c}\mathbf{y}^T \geqslant \mathbf{z}\mathbf{y}^T$. This gives

$$\begin{aligned} f(\mathbf{y}) = \mathbf{c}^T\mathbf{y} &\geqslant \mathbf{z}^T\mathbf{y} \\ &= \sum_{j=1}^{n} z_j y_j \\ &= \sum_{j=1}^{n} \mathbf{c}_{\mathcal{B}}^T B^{-1}\mathbf{A}_j y_j \\ &= \mathbf{c}_{\mathcal{B}}^T B^{-1} \sum_{j=1}^{n} \mathbf{A}_j y_j \\ &= \mathbf{c}_{\mathcal{B}}^T B^{-1}\mathbf{b} \\ &= \mathbf{c}_{\mathcal{B}}^T \mathbf{x}_{\mathcal{B}} \\ &= \mathbf{c}^T\mathbf{x} \\ &= f(\mathbf{x}) \end{aligned}$$

So $f(\mathbf{y}) \geqslant f(\mathbf{x})$, and since $\mathbf{y}$ was any feasible solution, $\mathbf{x}$ must be optimal. □

Note that this optimality criterion is a sufficient but not a necessary condition. It is therefore possible for optimal solutions to exist which do not obey this criterion.

**Corollary 57** *If* $\mathbf{A}_j \in \mathcal{B}$ *then* $c_j - z_j = 0$.

**Proof.** By definition, $B^{-1}\mathbf{A}_{\mathcal{B}(i)} = \mathbf{e}_i$, one of the standard ordered basis vectors. Hence

$$z_j = \mathbf{c}_{\mathcal{B}}^T B^{-1}\mathbf{A}_j = \begin{pmatrix} c_{\mathcal{B}(1)} & c_{\mathcal{B}(2)} & \dots & c_{\mathcal{B}(m)} \end{pmatrix} \mathbf{e}_i = c_{\mathcal{B}(i)} = c_j$$

Hence the result. □

#### Finding A Better Solution

Finding a solution and assessing its viability is all very well, but if the solution is not optimal then another solution must be found and tried. Ideally a new solution will be better than the old one in that the value of the objective function will be nearer its minimum.

From this point it will be necessary to distinguish between bases, and so the notation $\mathcal{B}_1, \mathcal{B}_2, \dots$ will be used. In the case of the first basis,

$$B\mathbf{x}_{\mathcal{B}_1} = \sum_{i=1}^{m} \mathbf{A}_{\mathcal{B}_1(i)} x_{\mathcal{B}_1(i)} = \mathbf{b} \tag{58}$$

where $\mathbf{A}_{\mathcal{B}_1(i)}$ represents the $i$th column of $B$.

The matrix $B^{-1}A$ has an important role, so for convenience its columns will be denoted by $\mathbf{A}'_j = \begin{pmatrix} a'_{1j} & a'_{2j} & \dots & a'_{mj} \end{pmatrix}^T$. Hence $A' = B^{-1}A \Leftrightarrow A = BA'$ gives

$$B\mathbf{A}'_j = \mathbf{A}_j \;\Leftrightarrow\; \sum_{i=1}^{n} \mathbf{A}_{\mathcal{B}_1(i)} a'_{ij} = \mathbf{A}_j$$

where $\mathbf{A}_{\mathcal{B}_1(i)}$ represents the $i$th column of $B$. This matrix will become $A$ in the next iteration.

Now, since this is leading to finding another solution, it is assumed that the optimality criterion has been violated. Say this is so for column $l$, so $c_l - z_l < 0$, $1 \leqslant l \leqslant n$. In particular the above now gives

$$\sum_{i=1}^{n} \mathbf{A}_{\mathcal{B}_1(i)} a'_{il} = \mathbf{A}_l \tag{59}$$

Now consider a parameter $\theta$, and find the difference between the solution as it stands, and $\theta$ times the offending column i.e. (58)$-\theta$(59) giving

$$\theta\mathbf{A}_l + \sum_{i=1}^{m} \mathbf{A}_{\mathcal{B}_1(i)} \left( x_{\mathcal{B}_1(i)} - \theta a'_{il} \right) = \mathbf{b}$$

The expression on the left hand side represents a new solution, with the existing basic solutions being adjusted, another ($l$) being added, and the others remaining zero. The new solution is given by

$$\mathbf{x}' = \begin{cases} x_{\mathcal{B}_1(i)} \mapsto x_{\mathcal{B}_1(i)} - \theta a'_{il} & 1 \leqslant i \leqslant m \\ x_l \mapsto \theta & \\ x_j \mapsto 0 & j \neq l \quad \mathbf{A}_j \notin \mathcal{B} \end{cases} \tag{60}$$

However, this isn't a new solution yet: a value for $\theta$ has to be found.

If $\theta = 0$ then $\mathbf{x}' = \mathbf{x}$, the old solution[†]. In order to maintain feasibility, it is clear that $\theta \geqslant 0$, so start from 0 and increase to make $\theta$ as big as possible. The point when $\theta$ will be maximal is determined by the top line of (60) when for some $k$, $x_{\mathcal{B}_1(k)} - \theta a'_{kl} = 0$. Hence

$$\theta = \frac{x_{\mathcal{B}_1(k)}}{a'_{kl}} = \min \left( \frac{x_{\mathcal{B}_1(i)}}{a'_{il}} \mid 1 \leqslant i \leqslant m \quad a'_{il} > 0 \right)$$

Notice that since $x_{\mathcal{B}_1(k)} - \theta a'_{kl} = 0$, the vector $\mathbf{A}_k$ drops out of the basis and is replaced by $\mathbf{A}_l$. However, a problem still remains: in order for $B^{-1}$ to exist the new basis vectors must be linearly independent, since it is used to define $\mathbf{x}'$.

Theorem 61 *The new basis, $\mathcal{B}_2 = \left( \mathcal{B} \setminus \left\{ \mathbf{A}_{\mathcal{B}(k)} \right\} \right) \cup \left\{ \mathbf{A}_{\mathcal{B}(l)} \right\}$ is linearly independent.*

It can be shown that $\mathbf{c}^T\mathbf{x}' = \mathbf{c}^T\mathbf{x} + \theta(c_l - z_l)$ from which it is obvious that the value of the objective function has been reduced. A new and better basic feasible solution has now been found, but there remain two issues to be addressed.

Firstly, if $\theta = 0$ then $\mathbf{x}$ does not change. The fact that $\theta = 0$ at all means that $\mathbf{x}$ was degenerate. However, the basis is still changed by "replacing one zero by another".

[†]It would be more correct to say 'basic feasible solution' rather than just 'solution'.

Secondly, there is a problem if $a'_{il} \leqslant 0 \ \forall i$. In this case $\theta \to -\infty$ and this gives

$$\mathbf{c}^T\mathbf{x}' = \mathbf{c}^T\mathbf{x} + \theta(c_l - z_l) \to -\infty$$

negative because column $l$ violates the optimality criterion. It may be possible to choose some sufficiently large value of $\theta$ so that the objective function has smaller value. On the other hand the linear programming problem may have no solution.

#### Solving The Linear Programming Problem

At this point enough theory has been covered to actually solve a linear programming problem. This theory can be neatly packaged into the 'Simplex Method'. However, it is necessary to make a few assumptions, which will be eliminated in the following section.

Given a linear programming problem in the standard form

$$f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \to \min \qquad A\mathbf{x} = \mathbf{b} \qquad x_j \geqslant 0 \ \forall j$$

assume that $A$ contains the $m \times m$ identity matrix as a submatrix, and $b_j \geqslant 0 \ \forall j$. These assumptions may seem trivial to achieve, but meeting both conditions at the same time is rather more difficult than meeting just one of them.

The value of the assumption is that $B = I$ so that $\mathbf{A}'_j = B^{-1}\mathbf{A}_j = \mathbf{A}_j$. Also, $\mathbf{x}_\mathcal{B} = \mathbf{b}$. The identity matrix in $A$ is there because of the slack variables. If $A\mathbf{x} > \mathbf{b}$ or $A\mathbf{x} < \mathbf{b}$ then one slack variable must be added for each of the $m$ constraints. This produces a constraint equation of the form $A\mathbf{x} + I\mathbf{w} = \mathbf{b}$. These slack variables make no appearance in the objective function, so it follows that $\mathbf{c}_{\mathcal{B}(i)} = 0 \ \forall i$. This means that $\mathbf{z} = \mathbf{0}$ so for the basis subscripts, the relative costs must be 0.The assorted information is usually displayed in a 'Simplex Table' as shown in Table 15.2.1.

| | 0 | $c_1$ | $c_2$ | ... | $c_{\mathcal{B}(1)}$ | ... | $c_{\mathcal{B}(m)}$ | ... | $c_n$ |
|---|---|---|---|---|---|---|---|---|---|
| | $-z_0$ | $c_1 - z_1$ | $c_2 - z_2$ | ... | 0 | ... | 0 | ... | $c_n - z_n$ |
| $c_{\mathcal{B}(1)}$ <br> $\vdots$ <br> $c_{\mathcal{B}(m)}$ | $\mathbf{x}_\mathcal{B}$ | $\mathbf{A}'_1$ | $\mathbf{A}'_2$ | ... | identity matrix | | | ... | $\mathbf{A}'_n$ |

Table 9: General form of a 'Simplex Table'.

Note that it is usual to omit the first row and first column. Note also that the quantity $-z_0$ is calculated as $c_0 - z_0 = 0 - \mathbf{c}^T_\mathcal{B}\mathbf{x}$, this is the same as $f(\mathbf{x})$.

From the Simplex Table it is a trivial matter to check the optimality criterion, and whether there is in fact no solution. Assuming the optimality criterion fails but there is a solution, a new solution can be constructed using the methods described above. But there is a better way.

In the equation $BA' = A$, $B$ is the matrix whose columns are the vectors of the basis. The columns of $A'$ are therefore the coefficients which produce the corresponding columns of $A$ from the basis vectors. In order to keep the standard form of the Simplex table, $\mathbf{A}_l$, the vector replacing $\mathbf{A}_k$ in the basis is required be the $k$th standard basis vector. The basis therefore needs to be transformed. Recall that

$$\theta = \frac{x_{\mathcal{B}(k)}}{a'_{kl}} = \min\left(\frac{x_i}{a'_{il}} \mid 1 \leqslant i \leqslant m\right)$$

and the chosen $x_{kl}$ is called the pivot. In order to transform the basis,

- divide row $k$ by the pivot.
- subtract from each row that row's entry ($x_{il}$) times row $k$ for $i \neq l$.

This is really just Gaussian elimination, and results in the formula

$$a''_{ij} = \begin{cases} \frac{a'_{kj}}{a'_{kl}} & i = k \\ a'_{ij} - a'_{il}\frac{a'_{kj}}{a'_{kl}} & i \neq k \end{cases} \tag{62}$$

While this is all very well for the main body of the table, there is still the matter of $\mathbf{x}_\mathcal{B}$ and the adjusted costs (and $z_0$). Now, if $\mathbf{x}$ is any feasible solution,

$$f(\mathbf{x}) = \sum_{\mathbf{A}_j \notin \mathcal{B}} c_j x_j + \sum_{i=1}^{m} c_{\mathcal{B}(i)} x_{\mathcal{B}(i)} = \sum_{\mathbf{A}_j \notin \mathcal{B}} c_j x_j + \mathbf{c}_\mathcal{B}^T \mathbf{x}_\mathcal{B} \tag{63}$$

and every feasible solution also satisfies

$$\begin{aligned} \mathbf{b} &= B\mathbf{x}_\mathcal{B} + \sum_{\mathbf{A}_j \notin \mathcal{B}} \mathbf{A}_j x_j \\ \text{so} \quad B^{-1}\mathbf{b} &= \mathbf{x}_\mathcal{B} + \sum_{\mathbf{A}_j \notin \mathcal{B}} B^{-1}\mathbf{A}_j x_j \\ \text{and} \quad \mathbf{c}_\mathcal{B}^T B^{-1}\mathbf{b} &= \mathbf{c}_\mathcal{B}^T \mathbf{x}_\mathcal{B} + \sum_{\mathbf{A}_j \notin \mathcal{B}} \mathbf{c}_\mathcal{B}^T B^{-1}\mathbf{A}_j x_j \\ \text{i.e.} \quad z_0 &= \mathbf{c}_\mathcal{B}^T \mathbf{x}_\mathcal{B} + \sum_{\mathbf{A}_j \notin \mathcal{B}} z_j x_j \end{aligned} \tag{64}$$

Comparing equations (63) and (64) it is evident that

$$\begin{aligned} f(\mathbf{x}) - z_0 &= \sum_{\mathbf{A}_j \notin \mathcal{B}} (c_j - z_j) x_j \\ z_0 &= \sum_{\mathbf{A}_j \notin \mathcal{B}} (c_j - z_j) x_j - \sum_{\mathbf{A}_j \notin \mathcal{B}} c_j x_j - \mathbf{c}_\mathcal{B}^T \mathbf{x}_\mathcal{B} \\ &= -\sum_{\mathbf{A}_j \notin \mathcal{B}} z_j x_j - \mathbf{c}_\mathcal{B}^T \mathbf{x}_\mathcal{B} \quad \text{but } x_j = 0 \text{ for } j \notin \mathcal{B}, \text{ so} \\ &= -\mathbf{c}_\mathcal{B}^T \mathbf{x}_\mathcal{B} \end{aligned}$$

But this is the definition of $z_0$, so this method also works in the first row of the Simplex table. This rather confusing notation becomes rather more transparent in the following example.

Example 65 *Given the Simplex table shown (Table 65), complete the second step.*

| costs | $-1$ | $-2$ | 0 | 0 |
|---|---|---|---|---|
| adjusted | $-1$ | $-2$ | 0 | 0 |
| $x_3 = 7$ | **2** | 3 | 1 | 0 |
| $x_4 = 6$ | 1 | 4 | 0 | 1 |

Table 10: First Simplex table with pivot in bold.

Proof. Solution From Table 65 it is evident that the optimality criterion is violated in column 1. Hence

$$\theta = \min\left(\frac{7}{2}, \frac{6}{1}\right) = \frac{7}{2}$$

so it is $x_3$ that is eliminated in favour of $x_1$.
Here $k$ is the first row of $x_3$ and $l$ is the first column. For example in the second column,

- from the first case in (62),
$$a''_{12} = \frac{3}{2}$$
- from the second case in (62),
$$c_2 - z_2 = -2 - (-1)\frac{3}{2} = \frac{-1}{2} \quad \text{and} \quad a''_{22} = 4 - 1\frac{3}{2} = \frac{5}{2}$$
This can be continued for the other 2 columns. The 'zero-th' column is treat in the same way. The two entries are
$$x_1 = \theta = \frac{7}{2} \quad \text{and} \quad x_4 = 6 - 1\frac{7}{2} = \frac{5}{2}$$

The new Simplex table is therefore as shown in Table 24

| costs | $-1$ | $-2$ | 0 | 0 |
|---|---|---|---|---|
| adjusted | 0 | $\frac{-1}{2}$ | $\frac{1}{2}$ | 0 |
| $x_2 = \frac{7}{2}$ | 1 | $\frac{3}{2}$ | $\frac{1}{2}$ | 0 |
| $x_4 = \frac{5}{2}$ | 0 | $\frac{5}{2}$ | $\frac{-1}{2}$ | 1 |

Table 11: Second Simplex table.

□

So it is now possible to solve a standard linear programming problem. However, a number of issues arise.

- In moving from one basis to another, will an already used basis be returned to—will the solution cycle? This is only an issue if the solutions are degenerate, since otherwise the objective function always decreases.
- Anti-cycling rules exist, and if used the method must terminate since there are finitely many bases — $\binom{n}{m}$ to be precise. From this finiteness it follows that either the linear programming problem has a solution, or $f \to -\infty$.
- If there are two columns violating the optimality criterion which one should be chosen? As it happens, random choice is as good a strategy as any.
- How complex is the simplex method? Each pivoting requires a polynomial number of operations, but the number of bases is $\binom{n}{m}$. There are rare cases that can be solved in 'time' which is a polynomial in $m$ and $n$, but generally this is not so.
- Does a method of polynomial order for solving the linear programming problem exist?

#### Transformation To Standard Form

For the Simplex method to work it is required that $A$ has as a submatrix the $m \times m$ identity matrix, and $b_i \geqslant 0 \ \forall i$. Satisfying the latter condition is simply a case of negating the offending equation, so this on its own can be assumed without loss of generality. The problem now is to include the former criterion at the same time.

In transforming any given problem into standard form it is necessary to introduce at most one slack variable for each constraint equation, but some of them may be negative, so this does not really work. Once the problem is in standard form, add one slack variable to each auxiliary equation, the problem then has the form $A\mathbf{x} + I\mathbf{w} = \mathbf{b}$.

However, economies can be made since if $A$ already has some identity matrix columns, $k$ say, then only $m-k$ extra slack variables need be added (and possibly variable re-numbered). Also in the case $A\mathbf{x}=\mathbf{b}$ the slack variables added to produce standard from will provide some identity matrix columns.

The auxiliary problem is to minimise $\sum_{i=1}^{m} w_i$ i.e.

$$\begin{pmatrix}0 & 0 & \dots & 0 & 1 & 1 & \dots & 1\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{w}\end{pmatrix}\to \min \qquad \begin{pmatrix}A & I\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{w}\end{pmatrix}=\mathbf{b} \qquad \begin{pmatrix}\mathbf{x}\\ \mathbf{w}\end{pmatrix}\geqslant 0$$

Since $\begin{pmatrix}\mathbf{x}\\ \mathbf{w}\end{pmatrix}=\begin{pmatrix}\mathbf{0}\\ \mathbf{b}\end{pmatrix}$ is always a basic feasible solution which has objective function bounded below by $0$, the Simplex method must terminate after a finite number of steps. Clearly this problem can be solved using the Simplex method, and there are two main conclusions.

1. If there are no $w_i$s in the solution, so $\mathbf{w}=\mathbf{0}$ then the columns in the Simplex table for $\mathbf{w}$ can be removed and the resulting '$A$' matrix has an identity submatrix. The Simplex method can now be used on this problem.
2. If it is not possible to eliminate some of the $w_i$s then the problem is infeasible and cannot be solved. This is because from the definition of the auxiliary problem the optimal solutions have objective function value of zero. A non-zero $w_i$ contradicts this.

The problem just solved has cost co-efficients $\begin{pmatrix}0 & 0 & \dots & 0 & 1 & 1 & \dots & 1\end{pmatrix}$ which is clearly not the original problem. Assuming $\mathbf{w}=\mathbf{0}$ the original problem is solved by deleting the columns of the auxiliary simplex table corresponding to $\mathbf{w}$. However, it is necessary to re-calculate the row of adjusted costs using the original cost coefficients. The following technique is used, and is known as the two-phase method.

1. Transform the problem into standard form.
2. Add one slack variable, $w_i$, to each constraint equation.
3. Use the simplex method to solve the auxiliary problem, this produces the required identity submatrix.
4. Use the simplex method to solve the original problem.

The two-phase method and simplex method are actually a way to solve simultaneous linear inequalities: Linear programming is simply one particular application of it.

### (15.2.2) Dual Solution To Linear Programming

#### Formulating The Dual Solution

The dual linear programming problem to a given linear programming problem is constructed as shown in Table 12.

The method for converting from the primal to the dual is, therefore,

1. For every equation in the primal there is a free variable in the dual.
2. For every inequality in the primal there is a restricted variable in the dual.
3. For each restricted primal variable construct an inequality with reversed sense (i.e. '$\leqslant$' instead of '$\geqslant$') as follows,

| Primal Problem | Description Of Indices | Dual Problem |
|---|---|---|
| $\mathbf{c}^T\mathbf{x} \to \min$ | | $\mathbf{b}^T\mathbf{ß} \to \max$ |
| $\mathbf{a}_i^T\mathbf{x} = b_i$ | $i \in I$ | $\pi_i$ free |
| $\mathbf{a}_i^T\mathbf{x} \geqslant b_i$ | $i \in \overline{I}$ | $\pi_i \geqslant 0$ |
| $x_j \geqslant 0$ | $j \in J$ | $\mathbf{ß}^T\mathbf{A}_j \leqslant c_j$ |
| $x_j$ free | $j \in \overline{J}$ | $\mathbf{ß}^T\mathbf{A}_j = c_j$ |

Table 12: Correspondence between the primal and dual linear programming problems.

(a) the coefficients of the dual variables are the same as the coefficients of the chosen primal variable in the primal constraint equations and inequalities.

(b) the right hand side is the cost coefficient of the primal variable under consideration.

4. For each free primal variable construct an equation as follows,
   (a) the coefficients of the dual variables are the same as the coefficients of the chosen primal variable in the primal constraint equations and inequalities.
   (b) the right hand side is the cost coefficient of the primal variable under consideration.
5. The new objective function is to be maximised (where before it was minimised) and the cost coefficients of the dual variables are the right hand sides of the constraint equations and inequalities of the primal.

From Table 12 it is evident that the matrix $A$ can be expressed as

$$A = \begin{pmatrix} \mathbf{a}_i^T & i \in I \\ \hline \mathbf{a}_i^T & i \in \overline{I} \end{pmatrix} = \left( \mathbf{A}_j \quad j \in J \quad | \quad \mathbf{A}_j \quad j \in \overline{J} \right)$$

Theorem 66 *The dual problem of a linear programming problem which is itself a dual problem, is the primal problem, i.e. the dual to the dual is the primal.*

Proof. Consider a linear programming problem and its dual solution, as shown in the second ad third columns of Table 25. Now, it is not possible to apply the method described above for finding the dual directly, so in the fourth column an equivalent problem to the dual is constructed. The dual can then be found—in the fifth column—and some more algebra produces the required result in the sixth column.

| Indices | Primal | Dual | $\text{Dual}_1$ | Dual Of $\text{Dual}_1$ | Dual Of Dual |
|---|---|---|---|---|---|
| | $\mathbf{c}^T\mathbf{x} \to \min$ | $\mathbf{ß}^T\mathbf{b} \to \max$ | $\mathbf{ß}^T(-\mathbf{b}) \to \min$ | $(-\mathbf{c})^T\mathbf{ø} \to \max$ | $\mathbf{c}^T\mathbf{ø} \to \min$ |
| $i \in I$ | $\mathbf{a}_i^T\mathbf{x} = b_i$ | $\pi_i$ free | $\pi_i$ free | $\left(-\mathbf{a}_i^T\right)\mathbf{ø} = -b_i$ | $\mathbf{a}_i^T\mathbf{ø} = b_j$ |
| $i \in \overline{I}$ | $\mathbf{a}_i^T\mathbf{x} \geqslant b_i$ | $\pi_i \geqslant 0$ | $\pi_i \geqslant 0$ | $\left(-\mathbf{a}_i^T\right)\mathbf{ø} \leqslant -b_j$ | $\mathbf{a}_i^T \geqslant b_j$ |
| $j \in J$ | $x_j \geqslant 0$ | $\mathbf{ß}^T\mathbf{A}_j \leqslant c_j$ | $\mathbf{ß}^T\left(-\mathbf{A}_j\right) \geqslant -c_j$ | $\tau_j \geqslant 0$ | $\tau_j \geqslant 0$ |
| $j \in \overline{J}$ | $x_j$ free | $\mathbf{ß}^T\mathbf{A}_j = c_j$ | $\mathbf{ß}^T\left(-\mathbf{A}_j\right) = -c_j$ | $\tau_j$ free | $\tau_j$ free |

Table 13: Calculating the dual of a dual.

Clearly the second and last columns of Table 25 are equivalent, hence the theorem is proved. □

This theorem shows that is is meaningful to speak of a "primal-dual pair" rather than distinguishing between a primal problem and a dual problem. The notation $M_P$ and $M_P^{\text{opt}}$ refers to the set of feasible and optimal solutions to the primal problem, and similarly for the dual.

### Information Provided By The Dual

**Theorem 67 (Weak Duality Theorem)** $\forall \mathbf{x} \in M_P \quad \forall \mathfrak{ß} \in M_D \quad \mathbf{c}^T\mathbf{x} \geqslant \mathfrak{ß}^T\mathbf{B}$ *i.e. for all feasible solutions,* $f(\mathbf{x}) \geqslant \phi(\mathfrak{ß})$.

**Proof.** Now, specifically for $j \in J$, $c_j \geqslant \mathfrak{ß}^T\mathbf{A}_j$. However, when $j \in \bar{J}$ there is equality, so the relationship holds in general. Similarly $b_i \leqslant \mathbf{a}_i^T\mathbf{x}$. Hence,

$$\begin{aligned} f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} &= \sum_j c_j x_j & \phi(\mathfrak{ß}) = \mathfrak{ß}^T\mathbf{b} &= \sum_i \pi_i b_i \\ &\geqslant \sum_j \mathfrak{ß}^T\mathbf{A}_j x_j & &\leqslant \sum_i \pi_i \mathbf{a}_i^T\mathbf{x} \\ &= \mathfrak{ß}^T A\mathbf{x} & &= \mathfrak{ß}^T A\mathbf{x} \end{aligned}$$

□

From this result it is readily seen that

1. If $\mathbf{x} \in M_P$, if $\mathfrak{ß} \in M_D$, and if $f(\mathbf{x}) = \phi(\mathfrak{ß})$ then $\mathbf{x} \in M_P^{\text{opt}}$ and $\mathfrak{ß} \in M_D^{\text{opt}}$.
2. (a) If $\min\limits_{\mathbf{x}\in M_P} f(\mathbf{x}) = -\infty$ then $M_D = \emptyset$.
   (b) If $\max\limits_{\mathfrak{ß}\in M_D} \phi(\mathfrak{ß}) = \infty$ then $M_P = \emptyset$.

**Theorem 68 (Strong Duality Theorem)** $M_P^{\text{opt}} \neq \emptyset \Leftrightarrow M_D^{\text{opt}} \neq \emptyset$ *and when this is so,* $\min\limits_{\mathbf{x}\in M_P} f(\mathbf{x}) = \max\limits_{\mathfrak{ß}\in M_D} \phi(\mathfrak{ß})$.

The information gleaned thus far about the relationship between the primal and dual problems is shown in Table 14.

| Dual Solution (this column) | Primal Problem | | |
|---|---|---|---|
| | $M_P^{\text{opt}} \neq \emptyset$ | $\min\limits_{\mathbf{x}\in M_P} \mathbf{c}^T\mathbf{x} = -\infty$ | $M_P = \emptyset$ |
| $M_D^{\text{opt}} \neq \emptyset$ | $\Leftrightarrow$ | impossible | impossible |
| $\max\limits_{\mathfrak{ß}\in M_D} \mathfrak{ß}^T\mathbf{b} = \infty$ | impossible | impossible | row $\Rightarrow$ column |
| $M_D = \emptyset$ | impossible | column $\Rightarrow$ row | possible |

Table 14: Possibilities for solutions to the primal and dual problems.

The bottom right hand entry in Table 14 says simply that the combination is possible: finding, say, $M_D = \emptyset$ gives no information about $M_P$. However, the table is very useful in determining whether a given linear programming problem can be solved. Indeed, given the large amount of work needed to solve a linear programming problem and the relatively small amount of work needed to check these criteria, it is prudent to use them.

For example, suppose it is found that $M_D = \emptyset$—this is plausible since sometimes in the translation to the dual the constraint equations become inconsistent. It must then be the case that either $M_P = \emptyset$, or $f(\mathbf{x})$ is not bounded below. Exhibiting a single feasible solution to the prime will then show that $f(\mathbf{x})$ is not bounded below.

### The Dual Simplex Method

From the strong duality theorem, the solution to the primal is the same as the solution to the dual, so there is a clear advantage if the dual is an easier problem to solve.

The simplex method already seen works by finding a feasible solution, then swapping from feasible solution to feasible solution until the optimality criterion is satisfied. The dual simplex method on the other hand works by starting with an infeasible solution for which the optimality criterion holds. In both cases the same simplex table is used, and hence there is no need to formulate the dual.

Beginning with an infeasible solution is very useful. The problem

$$\mathbf{c}^T\mathbf{x} \to \min \quad \text{with} \quad A\mathbf{x} \geqslant \mathbf{b} \quad \text{and} \quad \mathbf{c} \geqslant \mathbf{0} \quad \text{and} \quad \mathbf{x} \geqslant \mathbf{0}$$

can be converted into standard form to give

$$\mathbf{c}^T\mathbf{x} + \mathbf{0}^T\mathbf{w} \to \min \quad \text{with} \quad -A\mathbf{x} + I\mathbf{w} = -\mathbf{b} \quad \text{and} \quad \mathbf{c} \geqslant \mathbf{0} \quad \text{and} \quad \mathbf{x} \geqslant \mathbf{0} \quad \text{and} \quad \mathbf{w} \geqslant \mathbf{0}$$

Clearly $\mathbf{w} = -\mathbf{b}$ is an infeasible solution which obeys the optimality criterion because $\mathbf{c} \geqslant \mathbf{0}$. The pivot is found as

$$\theta = \frac{c_l}{a_{kl}} = \max\left(\frac{c_j}{a_{kj}} \mid 1 \leqslant j \leqslant n \text{ and } a_{kj} < 0\right)$$

For an offending row $k$ i.e. $x_k < 0$ find the maximum of the cost coefficients divided by the corresponding matrix entry, provided that entry is negative.

Finding the new simplex table is done in exactly the same way, as given by equation (62). Again, this also works for the top row—the adjusted costs.

The number obtained from the maximisation calculation is not '$\theta$'—it has been calculated in a completely different way. However, clearly $\theta = \frac{x_k}{a_{kl}}$ from which equation (60) can be used. Observe that this is effectively the same as equation (62).

Clearly the dual simplex method provides a way to solve the primal problem, but an optimal solution to the dual problem is also contained in the final optimal simplex table.

- If a linear programming problem in canonical form ($A\mathbf{x} \geqslant \mathbf{b}$) is transformed to standard form using slack variables, the constraining matrix will gain a unit submatrix. The optimal solution to the dual is the adjusted costs of the slack variables.
- If a linear programming problem already has a unit submatrix then the adjusted costs of the columns in the final simplex table which were the unit matrix in the first simplex table are $-\mathbf{ß}$.

The change in sign occurs because when a problem in canonical form is transformed into standard form it becomes $-A\mathbf{x} + I\mathbf{w} = -\mathbf{b}$.

## (15.3) Game Theory

### (15.3.1) Matrix Games

Consider a matrix with the following interpretation;

- The rows correspond to different options (strategies) available to player 1.
- The columns correspond to different options available to player 2—but the same options as are available to player 1.
- On a given row, the matrix entries are the gain to player 1 by taking the strategy of that row, in each case of the strategy of player 2.

- On a given column each matrix entry represents the gain to player 1 if player 2 takes the strategy of that column.

Clearly the objective of player 1 is to maximise the minimum gain it can guarantee by choosing the maximum of the row minima.
Player 2 on the other hand wishes to minimise the gain of player 1 by choosing the strategy which minimises the column maxima.

It is assumed that whatever strategy is finally taken, the gain to player 1 and to player 2 is a constant sum—zero when the game is in standard form. It is also assumed that each player knows what the matrix—the payoff matrix–is, and that they could make the best decision for themselves if they know what the other player was going to do.

For example, Table 15.3.1 gives details of two companies which are in competition for a market of 100,000 people. The strategies they may take are to advertise of television, on radio, or in newspapers. The payoff matrix gives the number of sales made by company 1.

| | company 2 | | | |
|---|---|---|---|---|
| company 1 | television | radio | newspapers | row. min. |
| television | 50 | 70 | 80 | 50 |
| radio | 40 | 60 | 50 | 40 |
| newspapers | 20 | 30 | 40 | 40 |
| col. max. | 50 | 70 | 80 | |

Table 15: Customers ('000) of company 1 out of 100,000 depending on strategies chosen.
Company 1 will choose television advertising as it maximises their minimum number of customers—their gain floor. Company 2 will choose to advertise on television also, as it minimises the gain which can be made by company 1—the loss ceiling.

Table 15.3.1 gives an example of a two person constant sum game, which can be transformed to standard form by subtracting $\frac{100}{2} = 50$ from each element in the payoff matrix. If $S$ is the constant sum, then the gain to player 2 used to be $S - a_{ij}$. In the standard matrix this becomes $-a_{ij}$ and $S = 0$.

Definition 69 *For a two player constant sum matrix game with payoff matrix $A$,*

- *The gain floor is the maximum of the row minima, hence*

$$\textit{gain floor} = v_1 = \max_i \min_j a_{ij}$$

- *The loss ceiling is the minimum of the column maxima, hence*

$$\textit{loss ceiling} = v_2 = \min_j \max_i a_{ij}$$

- *If $v_1 = v_2$ then the game is said to have a solution is pure strategies. $v = v_1 = v_2$ is called the value of the game.*
- *If there is an element of $A$ which is the maximum of its column and the minimum of its row, i.e. $a_{is} \leqslant a_{rs} \leqslant a_{rj}$ for all $i$ and $j$, then it is called a saddle point.*

The example in Table 15.3.1 has a solution in pure strategies: both companies advertise on television. Doing so meets the criteria that company 1 maximises its gain floor, and company 2 minimises its loss ceiling. Observe that a gain to company 1 is a loss to company 2. The saddle is the $(1, 1)$ element, 50.

**Theorem 70** $v_1 \leqslant v_2$.

**Proof.**

$$\min_j a_{ij} \leqslant a_{is} \quad \text{for all } i \text{ and any } s \text{ chosen}$$

$$\text{so} \quad \max_i \min_j a_{ij} \leqslant \max_i a_{is} \quad \text{for any } s$$

$$v_1 \leqslant \max_i a_{is} \quad \text{for any } s$$

$$v_1 \leqslant \min_s \max_i a_{is} = v_2$$

□

**Theorem 71** *If $a_{rs}$ is a saddle point, then $v_1 = a_{rs} = v_2$.*

**Proof.**

$$v_2 = \min_j \max_i a_{ij} \leqslant \max_i a_{is} = a_{rs} = \min_j a_{rj} \leqslant \max_i \min_j a_{ij} = v_1 \leqslant v_2$$

This rather involved series of inequalities proves the theorem. □

**Theorem 72** *If $a_{rs}$ and $a_{kl}$ are saddle points, then so are $a_{rl}$ and $a_{ks}$.*

**Proof.** Since $a_{kl}$ is its row minimum and $a_{rs}$ is its column maximum,

$$a_{kl} \leqslant a_{ks} \leqslant a_{rs}$$

But $a_{rs} = a_{kl}$, hence the result. □

If $v_1 = v_2 = v$ it does not follow that any element of $A$ which is equal to $v$ is a saddle point—it must also be its row minimum and column maximum.

**Theorem 73** *If $v_1 = v_2 = v$ then a saddle point exists, and for every saddle point $a_{pq}$, $a_{pq} = v$.*

**Proof.**

$$\text{Say that } v_1 = \max_i \min_j a_{ij} = \min_j a_{kj} = a_{kl}$$

$$\text{and } v_2 = \min_j \max_i a_{ij} = \max_i a_{is} = a_{rs}$$

This gives

$$a_{ks} \geqslant a_{kl} = v \quad \text{and} \quad a_{ks} \leqslant a_{rs} = v$$

So it must be the case that $a_{ks} = a_{kl} = a_{rs}$ and hence

$$a_{ks} = \min_j a_{kj} = \max_i a_{is}$$

i.e. $a_{ks}$ is a saddle point. □

### (15.3.2) Solution In Mixed Strategies

It is quite possible that the maximum of the row minima, and the minimum of the column maxima do not coincide i.e. there is no saddle point. In this case there is no sure-fire strategy for either of the players.

Definition 74 *The mixed strategy of player 1 is the probability distribution on the set of pure strategies which yield—in the long term—the maximum profit.*

Instead of picking the same strategy each time, the players will now choose different strategies, and it is necessary to calculate the frequency with which they choose the various strategies. Say

- Player 1 chooses strategy $i$ with probability $x_i$.
- Player 2 chooses strategy $j$ with probability $y_j$.

This means that any particular combination will occur is $x_i y_j$. These are independent events because player 1 does not know in advance how player 2 will play, and vice versa. The expected gain for player 1 is then

$$\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}x_i y_j = \mathbf{x}^T A\mathbf{y}$$

Define $S_k = \left\{ (z_1, z_2, \ldots, z_k) \mid z_i \geqslant 0 \ \forall i \text{ and } \sum_{i=1}^{k} z_i = 1 \right\}$ so that

$$v_1 = \max_{\mathbf{x}\in S_m}\min_{\mathbf{y}\in S_n} \mathbf{x}^T A\mathbf{y} \quad \text{the gain floor}$$
$$v_2 = \min_{\mathbf{y}\in S_n}\max_{\mathbf{x}\in S_m} \mathbf{x}^T A\mathbf{y} \quad \text{the loss ceiling}$$

As with solutions in pure strategies, $v_1 \leqslant v_2$, and in fact $v_1 = v_2$ for all matrices. A solution $(\mathbf{x}^*, \mathbf{y}^*) \in S_m \times S_n$ must satisfy

$$\min_{\mathbf{y}\in S_n} (\mathbf{x}^*)^T A\mathbf{y} = v_1 = v_2 = \max_{\mathbf{x}\in S_m} \mathbf{x}^T A\,(\mathbf{y}^*)^T$$

As might be expected, a solution—the probabilities—are found using linear programming.

Assertion 75 *A matrix game $A = (a)_{ij}$ can be solved in mixed strategies with probability distribution $\mathbf{x}$ by solving*

$$\sum_{i=1}^{m} x_i' \to \min \quad \textit{subject to} \quad \sum_{i=1}^{m} a_{ij}x_i' \geqslant 1 \quad \textit{and} \quad x_i' \geqslant 0$$

*where $x_i' = \frac{x_i}{z}$ where $\frac{1}{z}$ is the optimal value of the objective function.*

The above is for player 1, and for player 2 the linear programming problem is in fact the dual to this,

$$\sum_{j=1}^{n} y_j' \to \max \quad \text{subject to} \quad \sum_{j=1}^{n} a_{ij}y_i' \leqslant 1 \quad \text{and} \quad y_i' \geqslant 0$$

where $y_j' = \frac{y_j}{z}$ where $\frac{1}{z}$ is the optimal value of the objective function.

From the linear programming problem for player 1, transforming to standard form gives

$$\sum_{i=1}^{m} x_i' \to \min \quad \text{subject to} \quad -A^T\mathbf{x}' + I\mathbf{u} = \begin{pmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{pmatrix} \quad \text{and} \quad \mathbf{x}' \geqslant 0 \ \mathbf{u} \geqslant 0$$

which can be solved using the dual simplex method. Having found the solution for $\mathbf{x}'$ it is simply a case of dividing by the optimal value of the objective function i.e. multiplying by $z$.