# Mathematics of Operations Research

*Part III Course, Michaelmas 2009*

Revision Notes

Daniel Guetta

guetta@cantab.net

# Linear Programming

## Lagrangian Methods

- Let $P(\boldsymbol{b})$ denote the optimization problem

$$\text{Minimize } f(\boldsymbol{x}) \text{ subject to } \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{b}, \boldsymbol{x} \in X$$

We say that $x$ is feasible if $x \in X(b) = \left\{x \in X : h(x) = b\right\}$. We define the **Lagrangian** as

$$L\left(\boldsymbol{x}, \boldsymbol{\lambda}\right) = f\left(x\right) - \boldsymbol{\lambda}^T \left(\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{b}\right)$$

$\boldsymbol{\lambda}$ is called a **Lagrangian multiplier** and typically

- $X \subseteq \mathbb{R}^n$
- $\boldsymbol{b}, \boldsymbol{\lambda} \in \mathbb{R}^m$
- $h : \mathbb{R}^n \mapsto \mathbb{R}^m$

The following theorem concerns such problems:

> **Theorem (Lagrangian Sufficiency Theorem – LST)**: If $\overline{\boldsymbol{x}} \in X(\boldsymbol{b})$ (ie: $\overline{\boldsymbol{x}}$ is feasible for $P(\boldsymbol{b})$) and there exists $\overline{\boldsymbol{\lambda}}$ such that
>
> $$\inf_{\boldsymbol{x} \in X} L\left(\boldsymbol{x}, \overline{\boldsymbol{\lambda}}\right) = L\left(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}}\right)$$
>
> Then $\overline{\boldsymbol{x}}$ is optimal for $P(b)$.

***Proof***: For all $\boldsymbol{\lambda}$ and feasible $\boldsymbol{x}$ (ie: $X \in X(\boldsymbol{b})$ ), we have

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\lambda}^T \left(\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{b}\right) = f(\boldsymbol{x})$$

By assumption

$$L(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}}) \leq L(\boldsymbol{x}, \overline{\boldsymbol{\lambda}}) \qquad \forall\ \boldsymbol{x} \in X$$

If we restrict our attention to $\boldsymbol{x} \in X(\boldsymbol{b}) \subseteq X$ , the only possible $\boldsymbol{x}$ that can solve our problem, we find that

$$f(\overline{\boldsymbol{x}}) = L(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\lambda}}) \leq L(\boldsymbol{x}, \overline{\boldsymbol{\lambda}}) = f(x) \qquad \forall\ \boldsymbol{x} \in X(\boldsymbol{b})$$

**Thus,** $\overline{\boldsymbol{x}}$ is optimal for $P(b)$ ■

This leads to a method for solving such optimisation problems

- Minimize $L\left(\boldsymbol{x}, \boldsymbol{\lambda}\right)$ subject to $\boldsymbol{x} \in X$ . The answer $\boldsymbol{x}^*(\boldsymbol{\lambda})$ will, of course, depend on $\boldsymbol{\lambda}$ .
- Find a $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ such that $\boldsymbol{x}^*\left(\boldsymbol{\lambda}^*\right)$ is feasible.

This method effectively involves relaxing the constraint but associating a penalty to deviations from the constraint. For some value of the penalty, the maximum will be just right.

Note, however, that the converse of the theorem above is not true, and so this method might not always work.

- We define

$$\text{Solution value to } P(b) = \phi(\boldsymbol{b}) = \inf_{\boldsymbol{x}\in X(\boldsymbol{b})} f(\boldsymbol{x})$$
$$g(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x}\in X} L(\boldsymbol{x},\boldsymbol{\lambda})$$

It turns out that

**Theorem (Weak Duality)**: $\phi(\boldsymbol{b}) \geq g(\boldsymbol{\lambda})\,,\ \forall\ \boldsymbol{\lambda},\boldsymbol{b} \in \mathbb{R}^m$

***Proof***: First, we have that if $\boldsymbol{x} \in X(\boldsymbol{b})$ , then $L(\boldsymbol{x},\boldsymbol{\lambda}) = f(\boldsymbol{x})$ for all $\boldsymbol{\lambda}$ , so

$$\phi(\boldsymbol{b}) = \inf_{\boldsymbol{x}\in X(\boldsymbol{b})} f(\boldsymbol{x}) = \inf_{\boldsymbol{x}\in X(\boldsymbol{b})} L(\boldsymbol{x},\boldsymbol{\lambda})$$

Clearly,

$$\inf_{\boldsymbol{x}\in X(\boldsymbol{b})} L(\boldsymbol{x},\boldsymbol{\lambda}) \geq \inf_{\boldsymbol{x}\in X} L(\boldsymbol{x},\boldsymbol{\lambda}) = g(\boldsymbol{\lambda})$$

Because the first infimum is taken over a smaller set. This proves the Theorem. ■

Clearly, therefore, $g$ is a lower bound on the solution value of $P(b)$. It makes sense to try and make this lower bound as large as possible. In fact, we define:

**Definition (Dual Problem)**: The **dual problem** of $P(\boldsymbol{b})$ is

$$\text{maximize } g(\boldsymbol{\lambda}) \text{ subject to } \boldsymbol{\lambda} \in Y$$

Where $Y = \{\boldsymbol{\lambda} : g(\boldsymbol{\lambda}) > -\infty\}$

The **Primal Problem** is simply finding $\phi(\boldsymbol{b})$.

**Definition (Strong Lagrangian)**: We say that $P(\boldsymbol{b})$ is **Strong Lagrangian** if there exists $\boldsymbol{\lambda}$ such that

$$\phi(\boldsymbol{b}) = g(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x}\in X} L\Big(\boldsymbol{x},\boldsymbol{\lambda}\Big)$$

In other words, problems for which:

1. The Lagrangian Sufficiency Theorem applies
2. The min of the primal = max of the dual

Determining whether the LST applies to a given problem reduces to determining whether the problem is Strong Lagrangian.

- Geometrically, what is $g(\boldsymbol{\lambda})$? We first present the following calculation

$$\begin{aligned} g(\boldsymbol{\lambda}) &= \inf_{\boldsymbol{x}\in X} L(\boldsymbol{x},\boldsymbol{\lambda}) \\ &= \inf_{\boldsymbol{c}\in\mathbb{R}^m} \inf_{\boldsymbol{x}\in X(\boldsymbol{c})} \left[ f(\boldsymbol{x}) - \boldsymbol{\lambda}^T \left( \boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{b} \right) \right] \\ &= \inf_{\boldsymbol{c}\in\mathbb{R}^m} \phi(\boldsymbol{c}) - \boldsymbol{\lambda}^T \left( \boldsymbol{c} - \boldsymbol{b} \right) \\ &= \sup \left\{ \beta : \beta + \boldsymbol{\lambda}^T \left( \boldsymbol{c} - \boldsymbol{b} \right) \le \phi(\boldsymbol{c}) \qquad \forall \boldsymbol{c} \in \mathbb{R}^m \right\} \\ &= \beta_{\boldsymbol{\lambda}} \end{aligned}$$

Geometrically, this implies that $g(\boldsymbol{\lambda}) = \beta_{\boldsymbol{\lambda}}$ is obtained by

- Drawing a graph of $\phi(\boldsymbol{c})$
- Drawing a **hyperplane** of gradient(s) $\boldsymbol{\lambda}$ under this curve, and pushing it up as snugly as possible against the curve
- Reading off the intercept of this curve when $\boldsymbol{c} = \boldsymbol{b}$

It's therefore clear that the **dual problem** simply consists of **tweaking $\boldsymbol{\lambda}$** to find the **maximum value of** $\beta_{\boldsymbol{\lambda}}$. The specification that $\boldsymbol{\lambda} \in \{\boldsymbol{\lambda} : g(\boldsymbol{\lambda}) > -\infty\}$ simply ensures that the resulting hyperplane is not **vertical**.

We will only obtain $\max \beta_{\boldsymbol{\lambda}} = \phi(\boldsymbol{b})$ (the solution to the problem) if one of those "snugly fitting hyperplanes" touches the point $\phi(\boldsymbol{b})$ of the curve $\phi$. Formally, we require there to be a **supporting hyperplane**:

> **Definition (Supporting hyperplane)**: A supporting hyperplane $(\boldsymbol{c},\alpha)$ to $\phi$ at the point $\boldsymbol{b}$ is of the form
>
> $$\alpha(\boldsymbol{c}) = \phi(\boldsymbol{b}) - \boldsymbol{\lambda}^T(\boldsymbol{b} - \boldsymbol{c})$$
>
> Where
>
> $$\phi(\boldsymbol{c}) \ge \alpha(\boldsymbol{c}) \qquad \forall \boldsymbol{c} \in \mathbb{R}^m$$
>
> In other words, the hyperplane touches the curve snugly at $\phi(\boldsymbol{b})$ and lies below $\phi$ everywhere else.

And in fact

> **Theorem**: The following are equivalent
>
> 1. There exists a (non-vertical) supporting hyperplane to $\phi(\boldsymbol{c})$ at $\boldsymbol{c} = \boldsymbol{b}$.

2. The problem is Strong Lagrangian (ie: $\max \beta_{\boldsymbol{\lambda}} = \phi(\boldsymbol{b})$ )

***Proof***: Suppose 1 is true. This means that there exists $\boldsymbol{\lambda}$ such that

$$\phi(\boldsymbol{c}) \geq \phi(\boldsymbol{b}) - \boldsymbol{\lambda}^T(\boldsymbol{b} - \boldsymbol{c}) \qquad \forall \boldsymbol{c} \in \mathbb{R}^m$$

This implies that

$$\begin{aligned} \phi(\boldsymbol{b}) &\leq \inf_{\boldsymbol{c} \in \mathbb{R}^m} \left[\phi(\boldsymbol{c}) + \boldsymbol{\lambda}^T(\boldsymbol{b} - \boldsymbol{c})\right] \\ &= \inf_{\boldsymbol{c} \in \mathbb{R}^m} \inf_{\boldsymbol{x} \in X(\boldsymbol{c})} \left[f(\boldsymbol{x}) + \boldsymbol{\lambda}^T(\boldsymbol{b} - h(\boldsymbol{x}))\right] \\ &= \inf_{\boldsymbol{x} \in X} L(\boldsymbol{x}, \boldsymbol{\lambda}) \\ &= g(\boldsymbol{\lambda}) \end{aligned}$$

However, we know that $g(\boldsymbol{\lambda}) \leq \phi(\boldsymbol{b})$, and so we must have $g(\boldsymbol{\lambda}) = \phi(\boldsymbol{b})$ and $P(b)$ is Strong Lagrangian.

Conversely, if the problem is Strong Lagrangian, then there exists $\boldsymbol{\lambda}$ such that

$$\begin{aligned} \phi(\boldsymbol{b}) &= \inf_{\boldsymbol{x} \in X} L(\boldsymbol{x}, \boldsymbol{\lambda}) \\ \phi(\boldsymbol{b}) &\leq L(\boldsymbol{x}, \boldsymbol{\lambda}) \\ \phi(\boldsymbol{b}) &\leq f(\boldsymbol{x}) - \boldsymbol{\lambda}^T\left(\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{b}\right) \end{aligned} \qquad \forall \boldsymbol{x} \in X$$

We can minimize the RHS over $\boldsymbol{x} \in X(\boldsymbol{c})$, and get

$$\phi(\boldsymbol{b}) \leq \phi(\boldsymbol{c}) - \boldsymbol{\lambda}^T\left(\boldsymbol{c} - \boldsymbol{b}\right)$$

Thhis is true for all $\boldsymbol{c}$, and hence

$$\phi(\boldsymbol{b}) - \boldsymbol{\lambda}^T\left(\boldsymbol{b} - \boldsymbol{c}\right) \leq \phi(\boldsymbol{c}) \qquad \forall \boldsymbol{c} \in \mathbb{R}^m$$

And therefore $\phi$ has a non-vertical supporting hyperplane at $\boldsymbol{b}$. ■

This theorem is particularly useful in conjunction with the following:

**Theorem (Supporting hyperplane)**: If $\phi$ is convex and $\boldsymbol{b}$ lies in the interior of the set of points where $\phi$ is finite, there exists a (non-vertical) supporting hyperplane to $\phi$ at $\boldsymbol{b}$.

Note also the following definitions

**Definition (Convexity)**:

1. A set $S$ is a **convex set** if for all $\lambda \in [0,1]$

$$x, y \in S \Rightarrow \lambda x + \left(1-\lambda\right) y \in S$$

2. A real-valued function $f$ defined over a convex set $S$ is a **convex function** if for all $x, y \in S$ and $\lambda \in [0,1]$

$$\lambda f(x) + \left(1-\delta\right) f(y) \geq f\left(\lambda x + (1-\delta) y\right)$$

**Definition (Extreme Point)**: A point $x$ is an **extreme point** of $S$ whenever, for some $x, y \in S$ and $\lambda \in [0,1]$

$$x = \lambda y + \left(1-\lambda\right) z \Leftrightarrow x = y = z$$

- The discussion above has made it clear that we are interested in problems in which $\phi$ is convex. The following theorem is of use in that respect

**Theorem**: Consider the problem $P(b)$ defined as

$$\text{minimize } f(\boldsymbol{x}) \text{ subject to } \boldsymbol{x} \in X, h(\boldsymbol{x}) \leq \boldsymbol{b}$$

If $X$ is a convex set and $f$ and $h$ are convex, then $\phi$ is convex.

***Proof***: Take

- $\boldsymbol{b}_1, \boldsymbol{b}_2$ such that $\phi$ is defined, and $\boldsymbol{b} = \lambda \boldsymbol{b}_1 + \left(1-\lambda\right) \boldsymbol{b}_2$ for $\lambda \in (0,1)$.
- $\boldsymbol{x}_i$ to be feasible for $P(\boldsymbol{b}_i)$ and $\boldsymbol{x} = \lambda \boldsymbol{x}_1 + \left(1-\lambda\right) \boldsymbol{x}_2$

Then $\boldsymbol{x}_1, \boldsymbol{x}_2 \in X$ and $X$ convex implies $\boldsymbol{x} \in X$. Also, $h$ convex gives

$$\begin{aligned} h(\boldsymbol{x}) &= h\left(\lambda \boldsymbol{x}_1 + \left(1-\lambda\right) \boldsymbol{x}_2\right) \\ &\leq \lambda h(\boldsymbol{x}_1) + \left(1-\lambda\right) h(\boldsymbol{x}_2) \\ &\leq \lambda \boldsymbol{b}_1 + \left(1-\lambda\right) \boldsymbol{b}_2 \\ &= \boldsymbol{b} \end{aligned}$$

So $\boldsymbol{x}$ is feasible for $P(\boldsymbol{b})$. This means that

$$\phi(\boldsymbol{b}) \leq f(\boldsymbol{x})$$

If $f$ is convex

$$\phi(\boldsymbol{b}) \leq \lambda f\left(\boldsymbol{x}_1\right) + \left(1-\lambda\right) f\left(\boldsymbol{x}_2\right)$$

This holds for all $\boldsymbol{x}_i \in X(\boldsymbol{b}_i)$ and so taking infimums

$$\phi(\boldsymbol{b}) \leq \lambda \phi\left(\boldsymbol{b}_1\right) + \left(1-\lambda\right) \phi\left(\boldsymbol{b}_2\right)$$

So $\phi$ is indeed convex. ■

Note that the constraint $h(\boldsymbol{x}) = \boldsymbol{b}$ is equivalent to the two constraints $h(\boldsymbol{x}) \leq \boldsymbol{b}$ and $-h(\boldsymbol{x}) \leq -\boldsymbol{b}$, so $\phi$ is also convex under that constraint provided $h$ and $-h$ are convex.

## Linear Programs

- We consider problems of the form

**Definition (Linear program)**:

$$\text{minimize } \left\{ \underset{1\times n}{\boldsymbol{c}^T} \underset{n\times 1}{\boldsymbol{x}} : \underset{m\times n}{A}\underset{n\times 1}{\boldsymbol{x}} \geq \underset{m\times 1}{\boldsymbol{b}}, \boldsymbol{x} \geq 0 \right\}$$

Problems that involve **free variables** $x$ can be modified to fit this form by replacing $x$ with $x^+ - x^-$, where $x^+$ and $x^-$ are two nonnegative variables.

- These problems can be cast into **standard form** by the addition of **slack variables** $\boldsymbol{z}$ to form the problem

**Definition (Linear program in standard form)**:

$$\text{minimize } \left\{ \boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} - \boldsymbol{z} = \boldsymbol{b}, \boldsymbol{z} \geq 0, \boldsymbol{x} \geq 0 \right\}$$

- Let's clearly summarise the variables involved in a standard form LP

| | **Original form** $\min\left\{ \boldsymbol{c}^T\boldsymbol{x} : \begin{matrix} A\boldsymbol{x} \geq \boldsymbol{b} \\ \boldsymbol{x} \geq 0 \end{matrix} \right\}$ | **Standard form** $\min\left\{ \boldsymbol{c}^T\boldsymbol{x} : \begin{matrix} A\boldsymbol{x} - \boldsymbol{z} = \boldsymbol{b} \\ (\boldsymbol{x}, \boldsymbol{z}) \geq 0 \end{matrix} \right\}$ |
|---|---|---|
| #Variables | $n'$ | $n = n' + m$ |
| #Constraints | $m$ | $m$ |
| #Non-neg constraints | $n'$ | $n = n' + m$ |
| **Total constraints** | $n' + m$ | $n' + 2m = n + m$ |

Geometrically, these two problems are somewhat different

- The original problem defines a polyhedron, over which we need to find the maximum of a given function. It turns out that

**Theorem**: If an LP has a finite optimum, then it occurs at an **extreme point** of the **feasible set**.

Since there are $n'$ variables, we need $n'$ equations to characterise an extreme point. Thus, we require $n'$ of the inequalities to be tight, from the $n'+m$ in the original problem, and there are therefore ${}^{n'+m}C_{n'} = {}^{n}C_{n-m}$ extreme points.

Finding extreme points involves choosing which inequalities to make tight.

- The normal form problem involves a higher dimensional system of equations, the solutions of which completely cover the polyhedron in the original problem.

It should be pretty clear here that

- The slack variable $z_i$ is 0 only if the corresponding inequality is active.
- The variable $x_i$ is 0 only if the corresponding non-negativity constraint is active.

So finding the extreme points simply reduces to setting $n' = n - m$ variables to 0. To make our lives easier, we introduce the following notation

$$\underset{m\times m}{A_B}\boldsymbol{x}_B + \underset{m\times(m-n)}{A_N}\boldsymbol{x}_N = \underset{m\times 1}{\boldsymbol{b}}$$

$$\underset{m\times 1}{\boldsymbol{x}_B} \quad \underset{(m-n)\times 1}{\boldsymbol{x}_N}$$

We're now ready to characterise our extreme points algebraically

**Definition** (**Basic solution**):

- A **basic solution** to $A\boldsymbol{x} = \boldsymbol{b}$ is one with at least $n-m$ zero variables. In other words, one in which $\boldsymbol{x}_N = \boldsymbol{0}$.
- The $m$ non-zero variables $\boldsymbol{x}_B$ are called **basic** and form the **basis**. The others are called **non-basic**.

- Such a solution is **non-degenerate** if **exactly** $n - m$ of these variables are 0 (ie: if no component of $\boldsymbol{x}_B$ is 0).
- If a **basic solution** satisfies $\boldsymbol{x} \geq 0 \Leftrightarrow \boldsymbol{x}_B \geq 0$ then it is called a **basic feasible solution**.

**Theorem**: The **basic feasible solutions** are the **extreme points** of the feasible set.

- The idea behind the simplex algorithm is simply to start with a BFS, test whether it is optimal and move to another one if it isn't. We make the following assumptions (if they do not hold, then they will for a small perturbation of the data)
  - The matrix $A$ has linearly independent rows (ie: rank $A = m$ ); otherwise, we have "duplicate constraints".
  - Any $m \times m$ matrix formed from $m$ columns of $A$ is non-singular; otherwise, we might have some trouble finding one of the BFS.
  - All basic solutions have exactly $m$ non-zero variables (ie: the BFS is non-degenerate).
- Imagine we find ourselves at a BFS $\boldsymbol{x}$, with $\boldsymbol{x}_B = A_B^{-1}\boldsymbol{b}$. How can we check whether this BFS is optimal, and where do we go if it isn't?

  Imagine moving away from that BFS to a new point $\boldsymbol{x} + \theta \boldsymbol{d}^j$ by increasing a non-basic variable $j$. The variables in the basis will then also have to change to keep the solution feasible. Algebraically, we have

$$\theta d_i^j = \theta \begin{cases} 1 & i = j \\ 0 & i \in N \setminus \{j\} \\ \left(d_B^j\right)_i & i \in B \end{cases}$$

  We can find $\boldsymbol{d}_B^j$ by requiring that $\boldsymbol{x} + \theta \boldsymbol{d}^j$ be feasible (and remembering that $\boldsymbol{x}$ is feasible, so $A\boldsymbol{x} = \boldsymbol{b}$ )

$$A\left(\boldsymbol{x} + \theta \boldsymbol{d}^j\right) = \boldsymbol{b}$$
$$A\boldsymbol{d}^j = \boldsymbol{0}$$
$$A_B \boldsymbol{d}_B^j + A_{\bullet j} = 0$$
$$\boxed{\boldsymbol{d}_B^j = -A_B^{-1}\boldsymbol{A}_{\bullet j}}$$

Moving along this direction will change the objective function by the **reduced cost** $\overline{c}_j = \boldsymbol{c}^T \boldsymbol{d}^j = c_j - \boldsymbol{c}_B^T \boldsymbol{d}_B^j$. We can write this as

$$\boxed{\overline{c}_j = c_j - \boldsymbol{c}_B^T A_B^{-1} \boldsymbol{A}_{\bullet j}}$$

(And note that for $j$ basic, $\boldsymbol{c}_B^T A_B^{-1} \boldsymbol{A}_{\bullet j} = c_j$ and so $\overline{c}_j = 0$ )[1].

But consider that for any $\boldsymbol{x}$ with $A\boldsymbol{x} = \boldsymbol{b}$, we have

$$A_B \boldsymbol{x}_B + A_N \boldsymbol{x}_N = \boldsymbol{b}$$
$$\boldsymbol{x}_B = A_B^{-1}\left(\boldsymbol{b} - A_N \boldsymbol{x}_N\right)$$

Therefore

$$\begin{aligned} \text{Objective function} = \boldsymbol{c}^T \boldsymbol{x} &= \boldsymbol{c}_B^T \boldsymbol{x}_B + \boldsymbol{c}_N^T \boldsymbol{x}_N \\ &= \boldsymbol{c}_B^T A_B^{-1}\left(\boldsymbol{b} - A_N \boldsymbol{x}_N\right) + \boldsymbol{c}_N^T \boldsymbol{x}_N \\ &= \boldsymbol{c}_B^T A_B^{-1} \boldsymbol{b} + \overline{\boldsymbol{c}}_N \boldsymbol{x}_N \end{aligned}$$

Now, imagine a BFS $\boldsymbol{X}$ with $\overline{\boldsymbol{c}}_N \leq 0$ for all $N$, and, by definition, $\boldsymbol{x}_N = \boldsymbol{0}$. Moving away from this BFS necessarily implies increasing $\boldsymbol{x}_N$ (or keeping it at 0) and therefore decreasing the objective function. Thus

$$\text{Objective function} \leq \boldsymbol{c}_B^T A_B^{-1} \boldsymbol{b} = \boldsymbol{c}^T \boldsymbol{X}$$

And so $\boldsymbol{X}$ is optimal. In fact

**Theorem**: A basis matrix $A_B$ is said to be **optimal** if

$$A_B^{-1} \boldsymbol{b} = \boldsymbol{x}_B \geq 0 \qquad (\textit{Feasibility})$$
$$\overline{\boldsymbol{c}} = \boldsymbol{c} - \boldsymbol{c}_B A_B^{-1} A \geq \boldsymbol{0} \qquad (\textit{Optimality})$$

(Note that if the BFS is degenerate, it *could* be optimal but nevertheless have negative reduced costs. This is because if those negative reduced costs coincide with a variable in the basis that is 0, we cannot move in that direction without violating the non-negativity constraints. So the direction only *appears* to be improving).

If we find that some of the reduced costs are negative, then we choose the non-basic variable $j$ with **most negative reduced cost** (ie: from which we can gain the most) and make that variable **enter the basis**. In other words, we move in the direction $\boldsymbol{d}^j$ until one of the previously basic variables becomes 0 (ie: **leaves the basis**) – past that point, we would be violating

[1] To see why, note that $A_B^{-1} A_B = I$, which implies that $A_B^{-1}\left(A_B\right)_{\bullet j} = I^j$, where $I_i^j = \delta_{ij}$. Thus, if $j \in B$ then $\boldsymbol{c}_B^T A_B^{-1} \boldsymbol{A}_{\bullet j} = \boldsymbol{c}_B^T A_B^{-1}\left(A_B\right)_{\bullet j} = \boldsymbol{c}_B^T I^j = c_j$.

non-negativity constraints. If we move a distance $\theta \boldsymbol{d}^j$, then each variable $i$ changes by $\theta d_i^j$. If it's positive, then no problem. If it's negative, then we require

$$x_i + \theta d_i^j \geq 0 \Rightarrow \theta \leq -\frac{x_i}{d_i^j} \qquad \forall i \in \left\{ i : \theta d_i^j < 0 \right\}$$

This makes it clear that we should choose

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

$$\boxed{\theta = \min \left\{ \frac{x_i}{\left( A_B^{-1} A_{\bullet j} \right)_i} : \left( A_B^{-1} A_{\bullet j} \right)_i > 0, i \in B \right\}}$$

The variable which ends up determining $\theta$ will be the one to leave the basis.

(**Note**: if every $d_i^j = -\left( A_B^{-1} A_{\bullet j} \right)_i > 0$, then $\theta = \infty$, because we can improve in that direction infinitely without ever violating the non-negativity constraints. So the problem is unbounded).

- We now need a practical method to carry out simplex. Unfortunately, carrying out the method above naively leads to a very inefficient algorithm, because we need to evaluate $A_B^{-1}$ at every iteration.

  A move efficient method carries this information through each iteration. We store the information stored at each step in the following **tableau**:

| $A_B^{-1} A$ | *Value of basic variables* |
|---|---|
| *Reduced costs* | *−Objective function* |

Mathematically, the tableau looks like:

| $\vdots$<br>$\cdots \quad A_B^{-1} A \quad \cdots$<br>$\vdots$ | $\vdots$<br>$A_B^{-1} \boldsymbol{b} = \boldsymbol{x}_B$<br>$\vdots$ |
|---|---|
| $\cdots \quad \boldsymbol{c}^T - \boldsymbol{c}_B^T A_B^{-1} A \quad \cdots$ | $-\boldsymbol{c}_B^T A_B^{-1} \boldsymbol{b} = -\boldsymbol{c}_B^T \boldsymbol{x}_B$ |

We often represent the Tableau by an array $a$

| $\left( a_{ij} \right)$ | $a_{i0}$ |
|---|---|
| $a_{0j}$ | $a_{00}$ |

The simplex method then proceeds as follows:

- Begin by choosing a **basic feasible solution**. This is often easily done by **setting all the slack variables to $\boldsymbol{b}$** and all the "real" variables to 0 (see later for cases where this doesn't work). In that case, $A_B = A_B^{-1} = I$ (where some of the entries might be negative, depending on the type of inequality), and the tableau can easily be filled in:

| | $x_1$ | $x_2$ | … | $z_m$ | $\boldsymbol{b}$ |
|---|---|---|---|---|---|
| $z_1$ basic | $a_{11}$ | | | | $b_1$ |
| ⋮ | | | | | ⋮ |
| $z_m$ basic | | | | $a_{mn}$ | $b_m$ |
| | $c_1$ | $c_2$ | … | 0 | 0 |

  Calculate the **reduced cost** and enter them into the last row. Often, the objective function coefficients will only include non-basic variables. In that case, $\boldsymbol{c}_B^T = \boldsymbol{0}$ and so $\boldsymbol{c} = \overline{\boldsymbol{c}}$ and our work is done. Otherwise, proceed as follows:

  - **Multiply** each **row** by the corresponding **objective function coefficient** (eg: multiply the first row above by $c_{z_1}$).
  - **Subtract each row** from **the last row**.

  In terms of our tableau, this looks like:

$$\overline{a}_{0j} = a_{0j} - a_{ij}a_{0j}$$

  Finally, work out the objective function value to put into the bottom right-hand corner.

- **Choose a variable to enter the basis** Choose a **pivot column** with the smallest $a_{0j} = \overline{c}_j < 0$, or, in case of ties, the one with the smallest $j$. This is the **nonbasic variable** with the **most negative reduced cost**.

> **Termination check**: If all $\overline{c}_j > 0$, then there is no improving direction. Done.

  In terms of our tableau:

$$j = \text{Entering basis} = \text{Pivot column} = \underset{j}{\operatorname{argmin}}\left\{a_{0j} : a_{0j} < 0\right\}$$

- **Choose a variable to leave the basis** by finding $\theta$:

$$\theta = \min\left\{\frac{x_i}{\left(A_B^{-1}A_{\bullet j}\right)_i} : \left(A_B^{-1}A_{\bullet j}\right)_i > 0, i \in B\right\}$$

In this case, we know what $j$ is; we've already chosen the pivot column. Since the tableau contains $A_B^{-1}A$, then the **pivot column** contains $A_B^{-1}A_{\bullet j}$, and we know that the **last column** contains $x_B$. So:

- **Divide each item** in the **last column** by the corresponding item in the **pivot column**; only when the item in the pivot column is **positive**.
- **Choose** the **least one of those** – the corresponding variable will end up **leaving the basis**, and the corresponding row is the **pivot row**.

In terms of our tableau:

$$i = \text{Leaving basis} = \text{Pivot row} = \underset{i}{\operatorname{argmin}}\left\{\frac{a_{i0}}{a_{ij}} : a_{ij} > 0\right\}$$

> **Degeneracy check**: If there is more than one $i$ that minimizes the above, then two variables will simultaneously become 0 and the problem is degenerate (because more than $n$ constraints will be tight at that point).

> **Termination check**: If all $a_{ij} = \left(A_B^{-1}A_{\bullet j}\right)_i < 0$, then the problem is unbounded, and we can improve along that direction to infinity. Done.

- **Update the tableau (pivot on element $a_{ij}$)** by moving along that direction. Effectively, this involves updating the matrix $A_B^{-1}$ to reflect the fact the basis has now changed. Assume variable $j$ has entered the basis and variable $i$ has left it. The old and new matrices look like

$$A_B = \begin{bmatrix} A_{\bullet ?} & \cdots & \boldsymbol{A}_{\bullet i} & \cdots & A_{\bullet ?} \end{bmatrix}$$
$$\overline{A}_B = \begin{bmatrix} A_{\bullet ?} & \cdots & \boldsymbol{A}_{\bullet j} & \cdots & A_{\bullet ?} \end{bmatrix} \qquad ? \in B$$

Now, since $A_B^{-1}A_B = I$, we have that $A_B^{-1}\left(A_B\right)_{\bullet j} = I^j$, where $I_i^j = \delta_{ij}$.

This implies that

$$A_B^{-1}\overline{A}_B = \left[I^1 \quad \cdots \quad A_B^{-1}\boldsymbol{A}_{\bullet j} \quad \cdots \quad I^?\right]$$

Now, imagine we find a matrix $Q$ such that

$$QA_B^{-1}\overline{A}_B = Q\left[I^1 \quad \cdots \quad A_B^{-1}\boldsymbol{A}_{\bullet j} \quad \cdots \quad I^?\right] = I$$

$$\boxed{QA_B^{-1} = \overline{A}_B^{-1}}$$

So we simply need to find this magical matrix of row operations $Q$, apply it to our tableau (which contains $A_B^{-1}$) and we'll get $\overline{A}_B^{-1}$.

These row operations are as follows:

- **Divide** the **pivot row** by the **pivot element** (so we get 1 on the diagonal)
- For **each row** $\alpha \neq i$, **subtract** $a_{\alpha j}$ **times the pivot row**, where $A_{ki}$ is the value of the pivot column in that row (so we get 0 everywhere else).

In terms of our tableau

$$\overline{a}_{\alpha\beta} = \begin{cases} a_{\alpha\beta} / a_{ij} & \text{for the pivot row (ie: } \alpha = i) \\ a_{\alpha\beta} - a_{\alpha j}a_{i\beta} & \text{for every other row (ie: } \alpha \neq i) \end{cases}$$

It turns out this rule also applies to the last row of the tableau. To see why, consider that originally, the last row is (the vertical lines indicate jumping from cell to cell in the tableau)

$$\left[\boldsymbol{c}^T \mid 0\right] - \boldsymbol{c}_B^T A_B^{-1}\left[A \mid \boldsymbol{b}\right]$$

When we add a multiple of the pivot row, we are adding a linear combination of $\left[A \mid \boldsymbol{b}\right]$. So our result will be of the form

$$\left[\boldsymbol{c}^T \mid 0\right] - \boldsymbol{T}\left[A \mid \boldsymbol{b}\right]$$

where $\boldsymbol{T}$ is some linear transformation. But notice that after these row operations

- We end up with a 0 in the pivot column, by design.
- We end up with a 0 in every other column $k \neq j$ that stays in the basis, because (1) the original value there was 0, being the reduced cost of a basic variable (2) the entry in the pivot row for that column is also 0, because for any

basic variable $k$, $A_B^{-1}\boldsymbol{A}_{\bullet k} = I^k$, but $k \neq j$, so the entry in the pivot row is indeed 0.

So we end up with a 0 in every column corresponding to the *new* basis. Thus

$$\boldsymbol{c}_B^T - \boldsymbol{T}\overline{A}_B = 0$$
$$\boldsymbol{T} = \boldsymbol{c}_B^T \overline{A}_B^{-1}$$

And so we end up with a bottom row of

$$\left[\boldsymbol{c}^T \mid 0\right] - \boldsymbol{c}_B^T \overline{\boldsymbol{A}}_B^{-1}\left[A \mid \boldsymbol{b}\right]$$

As required.

- Sometimes, we do not have any obvious basic feasible solution (this often occurs when we have equality constraints, $\geq$ *something positive* or $\leq$ *something negative*, in which case the "obvious" BFS can make the slack variables negative).

  For example, consider the problem

  $$\min\left\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0\right\}$$

  The trick here is to

  - Multiply the equalities by –1 as needed to make $b_i$ positive.
  - Introduce a vector of **artificial variables** $\boldsymbol{y}$ and solve the problem

    $$\min\left\{y_1 + \cdots + y_m : \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{b}, \boldsymbol{x} \geq 0, \boldsymbol{y} \geq 0\right\}$$

  Several possibilities

  - If the optimal solution has $\boldsymbol{y} \neq 0$, then the original problem was not feasible.
  - If the optimal solution has $\boldsymbol{y} = \boldsymbol{0}$ and the method terminates with a basis matrix $A_B$ consisting exclusively of columns of $A$, then we're good – we can simply drop the columns corresponding to artificial variables and go from there.
  - If the optimal solution has $\boldsymbol{y} = \boldsymbol{0}$ but some of the artifical variables are still in the basis, then the original problem has a degenerate basic feasible solution. We must **drive the artificial variables out of the basis**.

    To do that, assume that the $\ell^{\text{th}}$ basic variable is an artificial variable (in the basis at 0 level, since $\boldsymbol{y} = \boldsymbol{0}$).

- Find a column $j$ corresponding to one of the original non-artificial variables with a non-zero entry in that row – ie: $\left(A_B^{-1}\boldsymbol{A}_{\bullet j}\right)_\ell \neq 0$.
- This column is linearly independent from every other column $A_B^{-1}A_{\bullet k}$ where $k \in B$, because for these columns, $\left(A_B^{-1}A_{\bullet k}\right)_i = \delta_{ik}$, and since $k < \ell$ (since $\ell$ is an artificial variable) every one of these columns have a 0 entry in the $\ell^{\text{th}}$ row.
- We can therefore make that variable $j$ enter the basis instead of $\ell$ [note that the pivot element might be negative].

If, however, every element in the $\ell^{\text{th}}$ row of $A_B^{-1}A$ [ie: every entry corresponding to the original non-artificial variables in the tableau] is 0, then the above method does not work, because it means that the original matrix $A$ has a rank $< m$. In other words, the matrix $A$ has some linearly dependent rows, and some of the constraints in the original problem are redundant. In that case, we can simply delete that $\ell^{\text{th}}$ row. To see why, consider

- The $\ell^{\text{th}}$ row of $A_B^{-1}A$ is equal to 0, so $\boldsymbol{T}^T A = 0$, where $\boldsymbol{T}^T$ is the $\ell^{\text{th}}$ row of $A_B^{-1}$.
- Since the problem is feasible, we must also have $\boldsymbol{T}^T\boldsymbol{b} = 0$.
- This means that the constraint $\boldsymbol{T}^T A\boldsymbol{x} = \boldsymbol{T}^T\boldsymbol{b}$ is redundant, and can be eliminated. Since this constraint is the information provided in the $\ell^{\text{th}}$ row, we can simply delete that row, and continue from there.

In general, if a variable appears in a single constraint with a positive coefficient, we can always let that variable be in the initial basis, and we do not need to associate an artificial variable with that constraint.

Note that the two methods can be combined into a single one, by minimizing

$$\boldsymbol{c}^T\boldsymbol{X} + M\sum y_i$$

Where $M$ is a very large number (in fact, we don't even need to ascribe a number to $M$ – we can simply leave it as an unknown, and assume it is bigger than any number in our tableau when we need to make comparisons).

## Duality Theory

- Recall that:

> **Definition (Linear program in standard form)**:
> $$\text{minimize } \left\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} - \boldsymbol{z} = \boldsymbol{b}, \boldsymbol{z} \geq 0, \boldsymbol{x} \geq 0\right\}$$

The permissible set $X \subset \mathbb{R}^{m+n}$ is given by

$$X = \left\{\left(\boldsymbol{x}, \boldsymbol{z}\right) : x_i \geq 0, z_i \geq 0\right\}$$

The **Lagrangian** for the problem is then

$$\begin{aligned} L\left((\boldsymbol{x}, \boldsymbol{z}); \boldsymbol{\lambda}\right) &= \boldsymbol{c}^T\boldsymbol{x} - \boldsymbol{\lambda}^T\left(A\boldsymbol{x} - \boldsymbol{z} - \boldsymbol{b}\right) \\ &= \left(\boldsymbol{c}^T - \boldsymbol{\lambda}^T A\right)\boldsymbol{x} + \boldsymbol{\lambda}^T\boldsymbol{z} + \boldsymbol{\lambda}^T\boldsymbol{b} \end{aligned}$$

Now, recall that

$$g(\boldsymbol{\lambda}) = \inf_{(\boldsymbol{x}, \boldsymbol{z}) \in X} L\left((\boldsymbol{x}, \boldsymbol{z}); \boldsymbol{\lambda}\right)$$

In this case, we note that

- $L$ only has a finite minimum in $X$ if both the coefficients of $\boldsymbol{x}$ and $\boldsymbol{z}$ are positive, so
$$\boldsymbol{\lambda} \in Y = \left\{\boldsymbol{\lambda} : \boldsymbol{\lambda} \geq 0, \boldsymbol{c}^T - \boldsymbol{\lambda}^T A \geq 0\right\}$$
Otherwise, we could simply take $\boldsymbol{x}$ and/or $\boldsymbol{z}$ to infinity and make the function smaller and smaller as we go.
- This minimum occurs when both the first two terms in $L$ disappear, so that
$$g(\boldsymbol{\lambda}) = \inf_{(\boldsymbol{x}, \boldsymbol{z}) \in X} L\left((\boldsymbol{x}, \boldsymbol{z}); \boldsymbol{\lambda}\right) = \boldsymbol{\lambda}^T\boldsymbol{b}$$

This motivates the definition of the **dual linear program**:

> **Definition (Dual program)**:
> $$\text{maximize } \left\{\boldsymbol{\lambda}^T\boldsymbol{b} : A^T\boldsymbol{\lambda} \leq \boldsymbol{c}^T, \boldsymbol{\lambda} \geq 0\right\}$$

By similar logic, we can construct dual programs for all kinds of primal constraints and negativity constraints. In fact

| Primal | Dual |
|---|---|

| minimize $\boldsymbol{c}^T\boldsymbol{x}$ | maximize $\boldsymbol{\lambda}^T\boldsymbol{b}$ |
|---|---|
| $\boldsymbol{A}_{i\bullet}^T\boldsymbol{x} \geq b_i$ | $\lambda_i \geq 0$ |
| $\boldsymbol{A}_{i\bullet}^T\boldsymbol{x} \leq b_i$ | $\lambda_i \leq 0$ |
| $\boldsymbol{A}_{i\bullet}^T\boldsymbol{x} = b_i$ | $\lambda_i$ free |
| $x_j \geq 0$ | $\boldsymbol{\lambda}^T\boldsymbol{A}_{\bullet j} \leq c_j$ |
| $x_j \leq 0$ | $\boldsymbol{\lambda}^T\boldsymbol{A}_{\bullet j} \geq c_j$ |
| $x_j$ free | $\boldsymbol{\lambda}^T\boldsymbol{A}_{\bullet j} = c_j$ |

- We note also that we can give an interpretation to the $\boldsymbol{\lambda}$. Since this optimization problem is convex, we have

$$\frac{\partial \phi}{\partial b_i} = \frac{\partial g(\boldsymbol{\lambda})}{\partial b_i} = \lambda_i$$

The variables $\lambda_i$ are therefore **shadow prices** or **marginal costs** – they indicate by how much our solution $\phi$ would increase if we increased one of the constraints $b_i$.

This provides another interpretation of duality. $\boldsymbol{c}$ is the cost of adding 1 to variables in the primal problem. $\boldsymbol{\lambda}$ is the cost of adding 1 to the RHS of each constraint. These are two ways of accounting for the total cost; we can either see it as a cost imposed by buying stuff, or a cost imposed by constraints. Duality requires $\boldsymbol{c}^T\boldsymbol{x} = \boldsymbol{\lambda}^T\boldsymbol{b}$; in other words, it requires the cost to be the same regardless of the accounting method.

- We can quickly prove the following theorem

> **Theorem (Strong duality)**: If a linear programming problem has an optimal solution, so does its dual, and the respective optimal costs are equal.

***Proof***: We can first translate any problem to its standard form equivalent

$$\text{minimize } \left\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0\right\}$$

At the optimal solution, the reduced costs must be non-negative, so

$$\boldsymbol{c}^T - \boldsymbol{c}_B^T A_B^{-1} A \geq \boldsymbol{0}^T$$

If we let $\boldsymbol{\lambda}^T = \boldsymbol{c}_B^T A_B^{-1}$, we have $\boldsymbol{\lambda}^T A \leq \boldsymbol{c}^T$. So $\boldsymbol{\lambda}$ is feasible for the dual

$$\text{maximize } \left\{\boldsymbol{\lambda}^T\boldsymbol{b} : A^T\boldsymbol{\lambda} \le \boldsymbol{c}^T, \boldsymbol{\lambda} \ge 0\right\}$$

Furthermore $\boldsymbol{\lambda}^T\boldsymbol{b} = \boldsymbol{c}_B^T A_B^{-1}\boldsymbol{b} = \boldsymbol{c}_B^T\boldsymbol{x}_B = \boldsymbol{c}^T\boldsymbol{x}$ , and so by weak duality $\boldsymbol{\lambda}$ is also optimal for the dual. ■

The theorem above provides an interesting insight into what the simplex algorithm does. We see that the optimality condition $\boldsymbol{c}^T - \boldsymbol{c}_B^T A_B^{-1} A \ge \boldsymbol{0}^T$ implies $\boldsymbol{\lambda}^T A \le \boldsymbol{c}^T$ , which is dual feasibility. So effectively, the primal simplex algorithm maintains primal feasibility while searching for dual feasibility.

- The **dual simplex method** does the opposite – it starts with a dual feasible solution, and searches for primal feasibility. Let $A_B$ be a basis and consider the corresponding simplex tableau

| | |
|---|---|
| $\cdots \quad A_B^{-1}A \quad \cdots$ | $A_B^{-1}\boldsymbol{b} = \boldsymbol{x}_B$ |
| $\cdots \quad \boldsymbol{c}^T - \boldsymbol{c}_B^T A_B^{-1} A \quad \cdots$ | $-\boldsymbol{c}_B^T A_B^{-1}\boldsymbol{b} = -\boldsymbol{c}_B^T\boldsymbol{x}_B$ |

This time, we no longer require $\boldsymbol{x}_B \ge 0$, and so the solution is basic but might not be feasible. However, we require that $\overline{\boldsymbol{c}} \ge 0$, which means that he solution is dual feasible. If it so happens that $\boldsymbol{x}_B \ge 0$, then the solution is also primal feasible with the same cost, and optimal solutions to both problems have been found. Otherwise:

  - We choose a row with $\left(x_B\right)_i < 0$; our **pivot row** $i$
  - For each **negative item** $v_k = \left(A_B^{-1}\boldsymbol{A}_{\bullet k}\right)_i < 0$ in that row, we calculate $\overline{c}_k \,/\, \left|v_k\right|$ and let $j$ be the index that minimizes this; our **pivot column**

$$j = \underset{\{k|v_k<0\}}{\operatorname{argmin}} \frac{\overline{c}_k}{\left|v_k\right|}$$

    If every item in this row is non-negative, then we can move in that direction for ever and keep the dual problem feasible. Thus, the problem is unbounded.
  - We then let the **pivot column enter the basis** and the **pivot row leave the basis** in the usual way. As a result, the $0^{\text{th}}$ row for every $k$ now reads

$$\overline{c}_k + v_k \frac{\overline{c}_j}{\left|v_j\right|}$$

We chose $j$ precisely to ensure that none of these values fall below 0. So the result is still dual feasible.

Furthermore, the cost cell will decrease, which means that the objective cost of the dual will increase, and the algorirthm will eventually terminate.

Note that if the pivot column entry in the $0^{\text{th}}$ row is 0, then moving in that direction does not change the cost; we have degeneracy.

In general, the value of the dual variables can be found in the last row of the tableau, under the slack variables (the sign will vary depending on the type of inequality).

- This method is particularly useful in two situations:
  - When it's easy to find a basic solution with $\bar{\boldsymbol{c}} > 0$, but not so easy to find a BFS. For example, if it involves making all the slack variables negative.
  - When we need to add a new constraint $\boldsymbol{\alpha}^T \boldsymbol{x} \geq \beta$ to an already-solved simplex. In that case, we create a new slack variable for that problem, and let it enter the basis. The determine what the new tableau will look like, we note that:

$$\bar{A} = \begin{bmatrix} A & \boldsymbol{0} \\ \boldsymbol{\alpha}^T & -1 \end{bmatrix} \qquad \bar{A}_B = \begin{bmatrix} A_B & 0 \\ \boldsymbol{\alpha}_B^T & -1 \end{bmatrix}$$

We then have

$$\bar{A}_B^{-1} = \begin{bmatrix} A_B^{-1} & \boldsymbol{0} \\ \boldsymbol{\alpha}_B^T A_B^{-1} & -1 \end{bmatrix}$$

And so

$$\bar{A}_B^{-1}\bar{A} = \begin{bmatrix} A_B^{-1}A & \boldsymbol{0} \\ \boldsymbol{\alpha}_B^T A_B^{-1}A - \boldsymbol{\alpha}^T & 1 \end{bmatrix}$$

And the reduced costs remain the same, with the last reduced cost equal to 0.

This last method is particular useful for **integer linear programming**, where we can apply **Gomory's cutting plane method**. Consider a solved tableau in which, for each basic variable $i$, we are left with the constraints

$$x_i + \sum_{j \in N} a_{ij} x_j = a_{i0}$$

Since all the variables are positive, we must also have

$$x_i + \sum_{j \in N} \lfloor a_{ij} \rfloor x_j \le x_i + \sum_{j \in N} a_{ij} x_j = a_{i0}$$

Where $\lfloor a_{ij} \rfloor$ is the integer just below $a_{ij}$. If, however, $x_j$ should be an integer (ie: if we have an ILP), then the LHS is an integer and can be no more than the integer right below $a_{i0}$. We then have

$$x_i + \sum_{j \in N} \lfloor a_{ij} \rfloor x_j \le \lfloor a_{i0} \rfloor$$

This new constraint is called a **cutting plane**, and can be added to the problem. Repeatedly applying these cuts eventually provides us with an integer solution.

- The following important result underlies much of the theory above

> **Theorem (Complementary slackness)**: For the LP problem
>
> $$\text{minimize } \left\{ \boldsymbol{c}^T \boldsymbol{x} : A\boldsymbol{x} \ge \boldsymbol{b}, \boldsymbol{x} \ge 0 \right\}$$
>
> $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ are primal and dual optimal respectively if and only if $\boldsymbol{x}$ is **primal feasible**, $\boldsymbol{\lambda}$ is **dual feasible** and
>
> $$\left( \boldsymbol{c}^T - \boldsymbol{\lambda}^T A \right)_i \boldsymbol{x}_i = 0$$
> $$\boldsymbol{\lambda}_j \left( A\boldsymbol{x} - \boldsymbol{b} \right)_j = 0$$

***Proof***: Consider the two vectors

$$\boldsymbol{v}_i = \left( \boldsymbol{c}^T - \boldsymbol{\lambda}^T \boldsymbol{A} \right)_i \boldsymbol{x}_i$$
$$\boldsymbol{u}_j = \boldsymbol{\lambda}_j \left( \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \right)_j$$

Note that

- The form of the dual problem requires the sign of $\boldsymbol{\lambda}_j$ to be the same as that of $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$, so $\boldsymbol{u}_j \ge 0$.
- Likewise for $\boldsymbol{v}$; $\boldsymbol{v}_i \ge 0$.

Furthermore, not that $\sum \boldsymbol{v}_i = \left( \boldsymbol{c}^T - \boldsymbol{\lambda}^T \boldsymbol{A} \right) \boldsymbol{x}$ and $\sum \boldsymbol{u}_j = \boldsymbol{\lambda}^T \left( \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \right)$, so

$$\sum \boldsymbol{v}_i + \sum \boldsymbol{u}_j = \boldsymbol{c}^T \boldsymbol{x} - \boldsymbol{\lambda}^T \boldsymbol{b}$$

And finally, note that if $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ are primal and dual feasible, then by strong duality

$$\sum \boldsymbol{v}_i + \sum \boldsymbol{u}_j = \boldsymbol{c}^T \boldsymbol{x} - \boldsymbol{\lambda}^T \boldsymbol{b} = 0$$

But since none of the components can be negative, this means that every component is 0. ■

The second of these constraints is automatically satisfied for any optimal solution of the problem $A\boldsymbol{x} = \boldsymbol{b}$. If the problem is not in standard form, then the constraint simply states that if a constraint is not tight, the reduced cost for that constraint is 0 (which makes sense; changing the constraint would do nothing to our optimal solution).

The first constraint is more interesting. It states that

$$\left(\boldsymbol{c}^T - \boldsymbol{\lambda}^T A\right)_i \boldsymbol{x}_i = 0$$
$$\boldsymbol{c}_i^T \boldsymbol{x}_i = \boldsymbol{\lambda}^T \boldsymbol{A}_{\bullet i} \boldsymbol{x}_i$$

If the problem is primal feasible (hence the first constraint)

$$\boldsymbol{c}_i^T \boldsymbol{x}_i = \boldsymbol{\lambda}^T \boldsymbol{b}$$

The LHS is the cost of the primal problem. The RHS is the cost of the dual. So complimentary slackness effectively states that the cost of the primal and the dual have to be the same.

This gives us yet another insight into what the simplex method does; it maintains primal feasibility and complementary slackness, and seeks dual feasibility. The dual simplex algorithm maintains dual feasibility and complimentary slackness, seeks primal feasibility.

## Integer Linear Programming

- Consider a linear program in which every variable has to be an integer. The best integer solution is *not* necessarily the closest to the best non-integer solution, so rounding won't always give the best solution. In fact, the closest integer solution might not even be feasible!
- One possible strategy is to try every solution in the (finite) set of possibilities and compare them. An efficient way of doing this is using the **branch and bound technique**:

  Suppose we want to solve the problem

  $$\min\left\{f(x)\right\} \qquad \text{s.t. } x \in X$$

  We divide this problem into sub-problems, the $i^{\text{th}}$ of which is $\min\left\{f(x)\right\}$ s.t. $x \in X_i$. We continue breaking those into sub-problems,

until we find out that is easy to solve. We also suppose that for any sub-problem we can calculate a lower bound $\ell\left(X_i\right) \leq \min_{x \in X_i} f(x)$. The steps are then:

- **Initialise**: Set $U = \infty$, discard any obviously infeasible solutions and treat the rest as one subset.
- **Branch**: Use some branch rule to select one of the remaining subsets, and break it into two or more subsets. Two common rules are
  - **Best bound rule** – partition the subset with the lowest bound, in the hope that this gives the best chance of an optimal solution and of being able to discard other larger subsets by the fathom test.
  - **Newest bound rule** – we partition the most recently created subset, breaking ties with the best bound rule. This has book-keeping advantages because it doesn't involve jumping round the tree so much.
- **Bound**: For each new subset $Y$, calculate $\ell\left(Y\right)$
- **Fathom step**:
  - If $\ell\left(Y\right) \geq U$, delete the subset
  - If $Y$ contains no feasible solution, delete the subset
  - If $Y$ can be solved to find an optimal solution $y \in Y$, then $\ell(Y) = f(y)$. If $\ell(Y) \geq U$, we eliminate the subset. If $\ell(Y) < U$, reset $U \leftarrow \ell\left(Y\right)$, store $y$ as the best solution so far, and re-apply the fathom step to all subsets.
- **Stopping rule**: if there are no remaining active subsets, stop. The best solution obtained so far is optimal. Otherwise, return to the branch step.

An example of that method is the knapsack problem, in which the lower bound might be obtained by noting that at least one item is needed in the knapsack.

- **Dakin's Method** applies to **mixed integer programs** (where only some of the variables are constrained to be integers) as well as **pure integer**

**programs**. It is a form of branch-and-bound in which we use the **linear programming relaxation** as our **lower bound**.

- **Initialise**: Set $U = \infty$ and solve the LP relaxation with the primal simplex method. If the optimal solution $\hat{\boldsymbol{x}}$ has $\hat{x}_j \in \mathbb{Z}\ \forall j$, then stop – the solution is optimal and feasible.
- **Branch**: Pick a variable that should be an integer but isn't. Partition into two subsets by adding one or another of the constraints

  $$x_j \le \lfloor x_j \rfloor \qquad\qquad x_j \ge \lceil x_j \rceil$$

  Use the newest bound rule for greatest efficiency.
- **Bound**: Solve the resulting LP relaxation of the problem with the new constraint to find $\ell$. This works best using the dual simplex.
- **Fathom step**: for each sub-problem $Y$
  - If $\ell(Y) \ge U$, delete the subset
  - If the dual simplex indicate that $Y$ is infeasible
  - If $Y$ has an optimal solution with integer values of the variables and $\ell(Y) < u$, reset $U \leftarrow \ell(Y)$ and store $\boldsymbol{x}$ as the incumbent solution.

## Complexity

- An **instance** of an optimization problem is defined by its input data, and the **instance size** is the number of bits required to define the instance. Ignoring the details of the implementation, we might expect the running time of an algorithm to be proportional to the number of arithmetic operations involved.
- We define
  - $f(n) = O(g(n))$ if there exists a $c$ such that $f(n) \le cg(n)\ \forall n$.
  - $f(n) = \Omega(g(n))$ if there exists a $c$ such that $f(n) \ge cg(n)\ \forall n$.
  - $f(n) = \Theta(g(n))$ if $f(n)$ is both $O(g(n))$ and $\Omega(g(n))$.
- Turing proved that the class of things that can be computed is the class of things that can be computed by a deterministic turing machine (DTM). When a DTM is given an input $x$, it runs for a certain number of steps (its **running time**) and outputs an answer $f(x)$. There are many Turing

Machines that can run a problem; let $T_M(n)$ be the **worse running time** of a given DTM over inputs of size $|x| = n$. We say $f(x)$ is **computable in polynomial time** if there exists a machine that can calculate $f(x)$ within $|x|^k$ steps (for some fixed $k$). The definition is robust because different DTMs can replicate each other by at most squaring or cubing the number of operations. In constrast, if $T_M(n) = \Omega(2^{cn})$ for all $m$, then $f(x)$ is said to be computable in **exponential time**.

- There are broadly three types of problems
  - **Optimization**; for example "find the shortest tour"
  - **Evaluation**; for example "find the length of the shortest tour"
  - **Decision**; for example "is there a tour with length $\leq L$?"

  We focus on decision problems, which have less potential complications.
- A decision problem is in $\mathcal{P}$ if its answer is calculable in polynomial time (ie: given input $x$, there exists a DTM can compute an answer in a number of steps bounded by $|x|^k$).
- A decision problem belongs to $\mathcal{NP}$ if and only if there exists a **checking function** $r(x,y)$ such that the answer is yes if and only if there exists a $y$ (called a **certificate**) such that $r(x,y) = 1$ and $r(x,y)$ can be calculated in polynomial time.

  For the decision TSP, for example, a certificate might be the order in which the nodes are visited. It takes $O(n)$ time to check the length of the path from $y$.

  $\mathcal{NP}$ stands for **nondeterministic polynomial**, and an alternative definition is that it contains problems which can be solved by a **nondeterministic Turing machine**, consisting of many DTMS working in parallel, any one of which can answer "yes" in polynomial time without consulting the others. For example, for the decision TSP, we could use $(n - 1)!$ machines checking $r(x, y)$ for a different $y$.
- Clearly, $\mathcal{P} \subseteq \mathcal{NP}$. It is believed that $\mathcal{P} \subset \mathcal{NP}$, but this is a major unsolved problem.
- We say that problem $\Pi_1$ **reproduces** $\Pi_2$ and is no harder than it if

  - We can make a polynomial time transformation of $\Pi_1$ into an instance of $\Pi_2$.
  - Apply some algorithm to solve an instance of $\Pi_2$.
  - Make a polynomial time transformation of this solution of $\Pi_2$ into a solution of $\Pi_1$.
- A problem $\Pi$ is known as $\mathcal{NP}$-hard if every problem in $\mathcal{NP}$ can be reduced to it. It is said to be $\mathcal{NP}$-complete if, in addition, $\Pi \in \mathcal{NP}$. Thus, all $\mathcal{NP}$-complete problems can be reduced to one another and are as difficult as all problems in $\mathcal{NP}$.

  To show that a new problem is $\mathcal{NP}$-complete, we must
  - Show that it is in $\mathcal{NP}$
  - Find another $\mathcal{NP}$-complete problem that reduces to it.
- Examples of $\mathcal{NP}$-complete problems:
  - **ILPs**: LPs were values of variables are restricted to $\{0, 1\}$.
  - The **TSP**
  - **Satisfiability**: given a logical expression involving several variables, can we find an assignment that makes the whole expression true?
  - **Hamiltonian circuit**: given a graph $G$ with $n$ edges is there a set of edges forming a tour of all vertices (equivalent to a decision TSP asking "is there a tour with length $\leq n$" on a saturated TSP with edge cost 1 if the edge exists in $G$ and 2 otherwise)
  - **Subgraph isomorphism**: Given two graphs $G$ and $G'$, is there a subgraph of $G$ isomorphic to $G'$?
  - **Clique decision problem**: given a graph $G$, does the it contain a clique of size $k$ ($k$ vertices all pairs of which are connected together)
  - **Vertex cover decision problem**: given a graph $G$, is there a set of $k$ vertices such that every edge starts or finishes at one of them?

# Graphs, Networks & All that Jazz

## Terminology

- A **graph** $G = \left(N, \mathcal{A}\right)$ consists of a set of **nodes**, $N$, and a set of **arcs**, $\mathcal{A}$.
- In an **undirected graph**, the arcs are **unordered pairs of nodes** $\left\{i, j\right\} \in \mathcal{A}$ and $i, j \in N$. In a **directed graph** or **network** the arcs are **ordered pairs of nodes** $(i, j)$.
- A **walk** is an ordered list of nodes $i_1 \cdots i_t$ such that $\left\{i_k, i_{k+1}\right\} \in \mathcal{A}$. A walk is a **path** if each of the $i$ are distinct. A walk is a **cycle** if each of the $i_1 \cdots i_{t-1}$ are distinct and $i_1 = i_t$. A graph is **connected** if there is a path connecting every pair of nodes.
- A network is **acyclic** if it contains no cycles. A network is a **tree** if it is **connected** and **acyclic**.
- A network $\left(N', \mathcal{A}'\right)$ is a **subnetwork** of $\left(N, \mathcal{A}\right)$ if $N' \subset N$ and $\mathcal{A}' \subset \mathcal{A}$. A sub-network $\left(N', \mathcal{A}'\right)$ is a **spanning tree** if it is a tree and $N' = N$.

## The Minimum Cost Flow Problem

- Let
  - $f_{ij}$ denote the flow of some material along arc $\left(i, j\right) \in \mathcal{A}$
  - $b_i$ denote the amount of flow that enters the network at node $i \in N$. If $b_i > 0$, the node is a **source**. If $b_i < 0$, it is a **sink**.
  - $m_{ij}$ and $M_{ij}$ denote the minimum and maximum flow possible along arc $\left(i, j\right) \in \mathcal{A}$. The special case of **uncapacitated flows** has $m_{ij} = 0$ and $M_{ij} = \infty$.
  - $c_{ij}$ be the cost of unit flow on arc $\left(i, j\right) \in \mathcal{A}$

  Then the **minimum cost flow problem** is

$$\left.\begin{array}{c} \min\left\{\sum_{(i,j)\in A} c_{ij} f_{ij}\right\} \\ \sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)\in A} f_{ji} = b_i \qquad \forall i \in N \\ m_{ij} \le f_{ij} \le M_{ij} \qquad \forall \left(i, j\right) \in A \end{array}\right\} \Leftrightarrow \begin{cases} \min\left\{\boldsymbol{c}^T \boldsymbol{f}\right\} \\ A\boldsymbol{f} = \boldsymbol{b} \\ \boldsymbol{m} \le \boldsymbol{f} \le \boldsymbol{M} \end{cases}$$

  Where the vector $\boldsymbol{f}$ has dimensions $\left|\mathcal{A}\right|$ and the matrix $A$ is given by

$$A_{ik} = \begin{cases} 1 & i \text{ is the start node of } k^{\text{th}} \text{ arc} \\ -1 & i \text{ is the end node of } k^{\text{th}} \text{ arc} \\ 0 & \text{otherwise} \end{cases}$$

Thus, every column has exactly two nonzero entries – a 1 and a –1.

- We now characterise basic solutions

> **Definition (Spanning tree solution)**: A **spanning tree solution** $\boldsymbol{f}$ to the problem above is one that can be constructed as follows:
>
> 1. Pick a set $T \subset \mathcal{A}$ of $n - 1$ arcs that form a tree when their direction is ignored.
> 2. Partition the rest of the arcs into two disjoint subsets $L$ and $U$. Set
>
> $$f_{ij} = \begin{cases} m_{ij} & (i,j) \in L \\ M_{ij} & (i,j) \in U \end{cases}$$
>
> 3. Solve the flow equations for the remaining variables. This can be done by starting at the leaves of the tree and then working upwards.

> **Theorem**: A flow vector is a **spanning tree solution** if and only if it is a **basic solution** of the minimum cost flow problem.

- We can also characterise our dual feasibility and complementary slackness conditions. Consider the Lagrangian for this problem

$$\begin{aligned} L(\boldsymbol{f}, \boldsymbol{\lambda}) &= \sum_{(i,j)\in\mathcal{A}} c_{ij} f_{ij} - \sum_{i \in N} \lambda_i \left( -b_i + \sum_{j:(i,j)\in\mathcal{A}} f_{ij} - \sum_{j:(j,i)\in\mathcal{A}} f_{ji} \right) \\ &= \sum_{(i,j)\in\mathcal{A}} \left( c_{ij} - \lambda_i + \lambda_j \right) f_{ij} + \sum_{i \in N} \lambda_i b_i \end{aligned}$$

Minimizing over the allowed $f$ gives the following conditions

$$\overline{c}_{ij} = c_{ij} - \lambda_i + \lambda_j \begin{cases} = 0 & \text{if} \quad m_{ij} < f_{ij} < M_{ij} \\ > 0 & \text{if} \quad f_{ij} = m_{ij} \\ < 0 & \text{if} \quad f_{ij} = M_{ij} \end{cases}$$

We also note that $\overline{c}_{ij} = 0$ for every basic variable (ie: for every arc in $T$). We can therefore solve for the $\lambda$ by setting

$$\lambda_n = 0 \qquad \lambda_i - \lambda_j = c_{ij} \quad \forall (i,j) \in T$$

In practice, we begin at the root note and set $\lambda = 0$ there and we then proceed to find $\lambda$ for every other node in the tree.

- The network simplex algorithm then proceeds as follows:
  - **Initiation**: start with a feasible spanning tree solution. A simple way to do this is as follows:
    - If $\boldsymbol{m} \neq \boldsymbol{0}$, replace the problem with the following one
      $$\boldsymbol{A}\boldsymbol{f}' = \left(\boldsymbol{b} - A\boldsymbol{m}\right)$$
      In which $\boldsymbol{m} = \boldsymbol{0}$.
    - Reduce the problem to one with neither source nor sink, by adding a "master source" which feeds all the sources and a "master sink" which collects from all the sinks and then joining them to each other.
    - The solution $\boldsymbol{f} = \boldsymbol{0}$ is then basic for that problem.
  - Compute the reduced costs for each arc
    $$\overline{c}_{ij} = c_{ij} - \lambda_i + \lambda_j$$
  - **Termination test**: If
    $$\overline{c}_{ij} \begin{cases} \geq 0 & \forall (i,j) \in L \\ \leq 0 & \forall (i,j) \in U \end{cases}$$
    Then decreasing the cost would imply pushing the flow past allowed limits. Thus, the feasible solution is optimal.
  - **Choose a variable to enter the basis**: – in other words, choose an arc $(i,j) \notin T$ such that
    $$\begin{array}{lll} \text{either} & \overline{c}_{ij} \leq 0 & (i,j) \in L \\ \text{or} & \overline{c}_{ij} \geq 0 & (i,j) \in U \end{array}$$
    This arc together with $T$ will form a unique cycle. We'll be pushing as much flow as possible round that cycle.
  - **Update the basis**: Push as much flow as possible round that cycle without exceeding the capacity of any arc.

  Effectively, the algorithm looks for negative-cost cycles and pushes as much flow around them as possible.
- Note that:
  - The matrix $A$ contains only entries of +1 or –1. The columns of the basis matrix $A_B$ can be re-ordered so that every element on the

diagonal is a +1 or –1. The determinant of the matrix is therefore $\pm$ 1, and by Crammer's Rule, $A_B^{-1}$ has integer entries.

- $\boldsymbol{f} = A_B^{-1}\boldsymbol{b}$, so if $\boldsymbol{b}$ contains integer entries, every basic solution has integer coordinates.
- $\boldsymbol{\lambda} = \boldsymbol{c}_B^T A_B^{-1}$, and so provided $\boldsymbol{c}$ contains integer entires, every dual basic solution has integer coordinates.

So for every network flow problem with integer data, every basic solution assigns an integer flow to every arc.

## Transportation & Assignment Problems

- In a transportation problem, there are $m$ suppliers of a good each with supply $s_i$, and $n$ customers each with demand $d_j$, such that $\sum s_i = \sum d_j$ The cost of transport from supplier $i$ to customer $j$ is $c_{ij}$. Our problem is

$$\min\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}f_{ij}\right\}$$

$$\sum_{i=1}^{m} f_{ij} = d_j \ \forall j \qquad \sum_{j=1}^{n} f_{ij} = s_i \ \forall i$$

$$f_{ij} \geq 0 \ \forall i, j$$

It is a special case of the minimum cost flow problem over a **bipartite graph**, in which the nodes divide into two disjoint sets of **suppliers** ($S$) and customers ($C$) and $\mathcal{A} \subset S \times C$.

> **Lemma**: Every minimum cost flow problem is equivalent to a transportation problem.

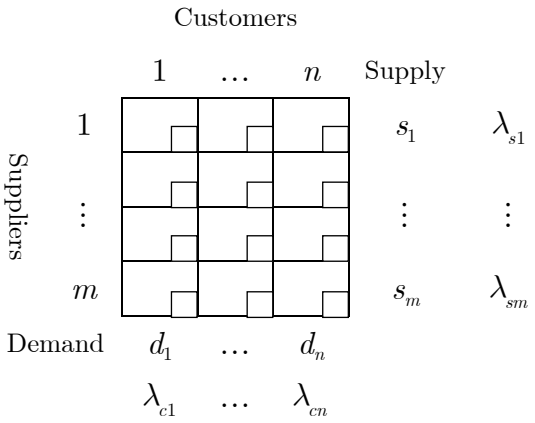
***Proof***: Transform the problem as described above to make $\boldsymbol{m} = \boldsymbol{0}$ and $\boldsymbol{M} < \infty$. Then

- For every arc ($i$, $j$) in the original problem, construct a source node with supply $M_{ij}$.
- For every node, construct a sink node with demand $\sum_{k:(i,k)\in\mathcal{A}} M_{ik} - b_i$ (which gives the absolute maximum that could come out of that node).

- Connect source node $(i, j)$ to the sink nodes $i$ and $j$ with infinite upper bound of capacity, and with $c'_{(i,j),i} = 0$ and $c'_{(i,j),j} = c_{ij}$.

There is then a 1-1 correspondence between optimal flows in the two problems, and the flows have the same cost. ■

- The information contained in a transformation contained is often displayed in a specialised **tableau**:




Each cell contains the amount of product flowing along that route, and the small insets contain the cost of flow among that route.

The steps of the algorithm are then:

- Calculate the $\lambda$, by insisting that for any cell in the basis,
$$\lambda_i - \lambda_j = c_{ij}$$
(Remember to start with $\lambda_{s1} = 0$ ).
- Calculate reduced costs for all empty cells
$$\overline{c}_{ij} = c_{ij} - \lambda_i + \lambda_j$$
- **Termination**: if all reduced costs are positive, we have reached an optimum.
- Choose a single empty cell to enter the basis (usually the one with most negative reduced cost).
- Find a cycle through which flow can be pushed. In terms of the tableau, the requirement for such a cycle is that
  - Consecutive cells are in the same column or row (but not necessarily adjacent).
  - No more than two cells per row or column

- Every cell basic, except for the cell entering the basis

We then increment our first cells by $+\theta$, our second by $-\theta$, etc… The value of $\theta$ is determined by the largest basic cell from which we need to subtract some flow.

- An **assignment problem** is a transportation problem in which the sources (people) all have $b = 1$, the sinks (tasks) all have $b = -1$ and the flows are restricted to be $\pm$ 1. Replacing the integer constraint with $0 \le f_{ij} \le 1$ gives the **LP-relaxation** of the problem, and it is a feature of the spanning-tree (simplex) algorithm that the result will also be an integer, and therefore also solve the integer problem.

## Maximum Flow Problems

- Consider a network with a single source and sink node, upper bounds $\boldsymbol{C}$ on all the arcs, and $\boldsymbol{m = 0}$. The problem of finding the **maximum flow** through this network is

$$\max\{\delta\}$$

$$\sum_{j:(i,j)\in\mathcal{A}} f_{ij} - \sum_{j:(j,i)\in\mathcal{A}} f_{ij} = \begin{cases} \delta & \text{if } i = 1 \\ -\delta & \text{if } i = n \\ 0 & \text{otherwise} \end{cases}$$

$$0 \le f_{ij} \le C_{ij}$$

This problem can be converted to a **minimum cost flow** problem by

- Adding an arc $(n, 1)$ to the network, with cost –1 and $m_{1n} = 0, M_{1n} = \infty$.
- Setting the cost of all the other arcs in the network to 0 (but leaving capacities as they are).
- Finding the minimum cost flow through the network

Since the only arc with non-zero cost has negative cost, the algorithm will circulate as much flow as possible subject to capacity constraints.

- For $S \subset N$, we define the **capacity** of the cut $\left[S, N \setminus S\right]$ as

$$C\left(S, N \setminus S\right) = \sum_{i\in S, j\notin S} C_{ij}$$

**Theorem (Max-flow min-cut)**:

$$\text{Max flow } \delta = \text{Min cut capacity} = \min_{S:1\in S, n\notin S} C\left(S, N \setminus S\right)$$

***Proof***: We first prove that **value of any flow** $\leq$ **capacity of any cut**. We first define a function that calculates all the flow from one set to another

$$f\left(X,Y\right)=\sum_{i\in X,j\in Y:(i,j)\in\mathcal{A}}f_{ij}$$

We then note that if $1\in S$ and $n\notin S$, any flow is simply given by the net amount leaving $S$:

$$\begin{aligned}\delta&=\sum_{i\in S}\left(\sum_{j:(i,j)\in\mathcal{A}}f_{ij}-\sum_{j:(j,i)\in\mathcal{A}}f_{ij}\right)\\&=f\left(S,N\right)-f\left(N,S\right)\\&=f\left(S,S\right)+f\left(S,N\setminus S\right)-f\left(N\setminus S,S\right)-f\left(S,S\right)\\&=f\left(S,N\setminus S\right)-f\left(N\setminus S,S\right)\\&\leq f\left(S,N\setminus S\right)\\&=\sum_{i\in S,j\notin S}f_{ij}\\&\leq\sum_{i\in S,j\notin S}C_{ij}\\&=C\left(S,N\setminus S\right)\end{aligned}$$

For the second part of the proof, we develop the **Ford-Fulkerson Algorithm**. Suppose $f_{ij}$ is optimal, and recursively define $S\subset N$ as follows:

- Start with $1\in S$
- If $i\in S$ and $f_{ij}<C_{ij}$, then $j\in S$
- If $i\in S$ and $f_{ji}>0$, then $j\in S$
- Scan each node newly added to $S$, until every node has been scanned (or, for the purposes of this algorithm, until $n$ has been reached).

So $S$ is simply the set of nodes to which we can increase flow. If $n\in S$, then we have an **augmenting path** – we can increase the flow from 1 to $n$, and we do as much as we can. Otherwise, $\left[S,N\setminus S\right]$ is a cut with $1\in S$ and $n\notin S$. But if $i\in S$, $j\notin S$, then we must have $f_{ij}=C_{ij}$, $f_{ji}=0$ (otherwise, more nodes would have been added to $S$ by the procedure above). As such

$$\delta=f\left(S,N\setminus S\right)-f\left(N\setminus S,S\right)=C\left(S,N\setminus S\right)$$

Thus, the maximum flow is indeed equal to the minimum cut. We also note that if the capacities and initial flow are all integers, then each step increases the flow by an integer amount. Thus, the algorithm will converge to an integer solution. ■

- We can recast our maximum-flow problem in dual form:

$$\min\left\{-f_{n1}\right\}$$
$$\sum_{j:(i,j)\in\mathcal{A}} f_{ij} - \sum_{j:(j,i)\in\mathcal{A}} f_{ij} = 0$$
$$0 \le f_{ij} \le C_{ij} \qquad f_{n1} \ge 0$$

The Lagrangian in its usual form (with dual variables $\lambda_i$) gives, for optimality on arc $(n, 1)$:

$$\overline{c}_{n1} = -1 - \lambda_n + \lambda_1 = 0 \Rightarrow \lambda_1 = 1 + \lambda_n$$

On every other arc, the costs are 0, and

$$\lambda_j - \lambda_i > 0 \qquad \text{if } f_{ij} = 0$$
$$\lambda_j - \lambda_1 < 0 \qquad \text{if } f_{ij} = C_{ij}$$

So

$$\lambda_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \in N \setminus S \end{cases}$$

- **Critical path analysis** – consider a project consisting of a number of jobs, where job $i$ takes time $\tau_i$ to complete. We consider a graph with
  - A node for each project
  - An arc $(i, j)$ if project $i$ needs to be completed before job $j$.
  - A source node $s$ and a sink node $s'$, each of 0 duration. The source node connects to every project, and every project is connected to the sink node.

Our problem is then

$$\min\left\{t_{s'} - t_s\right\}$$
$$t_j - t_i \ge \tau_i \qquad \forall (i,j) \in \mathcal{A}$$

With dual

$$\max\left\{\sum_{(i,j)\in\mathcal{A}}\tau_i f_{ij}\right\}$$

$$\sum_{j:(j,i)\in\mathcal{A}} f_{ji} - \sum_{j:(i,j)\in\mathcal{A}} f_{ij} = -\begin{cases}1 & i=s\\ -1 & i=s'\\ 0 & \text{otherwise}\end{cases}$$

$$f_{ij}\geq 0 \qquad \forall (i,j)\in\mathcal{A}$$

This is a minimum cost flow problem with arc costs $-\tau_i$. The path for which $f_{ij}=1$ defines the **critical path**.

## Shortest Path Problems

- Consider a network and choose a node $n$ as a root node. Put a demand of $n-1$ at that node (ie: $b_n=-(n-1)$) and a supply of one unit on every other node. Let the cost of each arc be equal to its length. Solve the network flow problem. The shortest part from any node $i$ to $n$ is given by following the arcs of the spanning tree from $i$ to $n$. If $v_i$ is the shortest distance from $i$ to $n$ these quantities are known as **labels**. **Label-setting algorithms** systematically determine their values in some order. **Label-correcting algorithms** find their values through a sequence of iterations.
- Consider the solution above, and suppose the $\lambda_i$ are the optimal dual variables associated with the optimal spanning tree solution. On every arc through which $f_{ij}>0$, we have

$$\lambda_i = c_{ij}+\lambda_j$$

Taking $\lambda_n=v_n=0$ (for the root node) and adding these equalities along a path from $i$ to $n$, we conclude that $\lambda_i=v_i$. Furthermore, as $b_1=\cdots=b_{n-1}=1$, the dual is

$$\max\left\{\sum_{i=1}^{n-1}\lambda_i\right\} \qquad \lambda_i\leq c_{ij}+\lambda_j \quad (i,j)\in\mathcal{A}$$

It follows that if all the other $\lambda$ are fixed, $\lambda_i$ satisfies (with $\lambda_n=0$)

$$\lambda_i=\min_{k:(i,k)\in\mathcal{A}}\left\{c_{ik}+\lambda_k\right\} \qquad i=1,\cdots,n-1$$

Intuitively, this means that if the shortest path from $i$ to $n$ contains node $k$, then the path from $k$ to $n$ should also be optimal.

The idea is that if we're looking for a path from $i$ to $n$, then we should choose the segment from $(i,\ k)$ by minimizing over path lengths $c_{ik} + \lambda_k$. This is known as **dynamic programming**.

- The **Bellman-Ford Algorithm** – let $v_i(t)$ be the length of the shortest path from $i$ to $n$ which uses **at most** $t$ arcs. $v_n(t) = 0 \ \ \forall t$ and $v_i(0) = \infty \ \ \forall i \neq n$. Then the **label-correcting Bellman-Ford algorithm** is defined by

$$v_i\left(t+1\right) = \min_{k:(i,k)\in\mathcal{A}} \left\{c_{ik} + v_k(t)\right\} \qquad i = 1,\cdots,n-1$$

Note that $v_i\left(t+1\right) \leq v_i(t)$, because increasing the number of arcs we allow, we increase the possible paths. If there are no negative length cycles, there exists a shortest path which has at most $n$ – 1 arcs. Thus, $v_i\left(n-1\right) = v_i$, and we cannot reduce the total length. Thus, if $\boldsymbol{v}\left(n-1\right) = \boldsymbol{v}\left(n\right)$, we have found an optimal solution.

The algorithm has running time $O(mn)$ since there is a maximum of $n$ iterations, and each iteration examines each arc once.

We can book-keep the Bellman-Ford algorithm as follows:

| $t$ | $v_1(t)$ | $v_2(t)$ | … | $v_n(t)$ |
|---|---|---|---|---|
| 0 | $+\infty$ | $+\infty$ | $+\infty$ | 0 |
| ⋮ | | | | 0 |
| $n$ | | | | 0 |

| $t$ | $d_1$ | $d_2$ | … | $d_n$ |
|---|---|---|---|---|
| 1 | | | | – |
| ⋮ | | | | |
| $n$ | | | | |

At every step, update the $v$ using the formula above. The $d$ correspond to the latest **successor** of any given node (ie: the node that was used for the minimization in updated the $v$).

- **Dijkstra's Algorithm** is a lebel-setting algorithm; it can only be applied when all the arc lengths are positive. To each exposition, we suppose all arcs are present, setting $c_{ij} = \infty$ where no arc exists.

> **Theorem**: Suppose that $c_{ij} \geq 0$ for all $i$, $j$. Let $\ell \neq n$ be such that

$$c_{\ell n} = \min_{i \neq n} c_{in}$$

Then $v_\ell = c_{\ell n}$ and $v_\ell \leq v_k \ \forall k \neq n$.

***Proof***: A path from node $k$ to $n$ has a last arc $(i,\ n)$ whose length $c_{in}$ is at least $c_{\ell n}$, so $v_k \geq c_{\ell n}$. For node $\ell$, we also have $v_\ell \leq c_{\ell n}$, and so $v_\ell = c_{\ell n} \leq v_k \ \forall k \neq n$.

Note that the proof relies heavily on the fact no arcs have negative cost. ■

The algorithm them simply proceeds as follows:

- Find a node $\ell \neq n$ such that $c_{\ell n} \leq c_{in}$ for all $i \neq n$. Set $v_\ell = c_{\ell n}$.
- For every node $i \neq \ell, n$, set $c_{in} = \min\left\{c_{in}, c_{i\ell} + c_{\ell n}\right\}$
- Remove node $\ell$ from the graph, return to step 1 and apply those steps to the new graph.

The algorithm has running time $O(n^2)$ since there are $n$ iteration, each of which involves a comparison and update of arc lengths from each node.

- Note that if $v_i$ is the shortest path length from node $i$ to node $n$, then from Bellman's equations, we have

$$v_i \leq c_{ij} + v_j$$

This means that the quantities $\overline{c}_{ij} = c_{ij} + v_j - v_i \geq 0$. We can therefore construct a new problem in which the arc lengths $c_{ij}$ are replaced by $\overline{c}_{ij}$.

For such a problem, the length of any path from $i_1$ to $i_t$ is given by

$$\sum_{\tau=1}^{t-1} \overline{c}_{i_\tau i_{\tau+1}} = \sum_{\tau=1}^{t-1}\left(c_{i_\tau i_{\tau+1}} + v_{i_{\tau+1}} - v_{i_\tau}\right) = v_{i_t} - v_{i_1} + \sum_{\tau=1}^{t-1} c_{i_\tau i_{\tau+1}}$$

So the shortest path under the new arc lengths are the same as those under the original (possibly negative) arc lengths. We can now, if we wish, use Dijkstra's algorithm.

This transformation is useful when we wish to solve the all-pairs problem (ie: to find the shortest distance between all pairs of nodes). If we have negative arc lengths, we can first use the Bellman-Ford algorithm to obtain $v_i$ for a given root node and then apply Dijkstra's algorithm to solve the problem with the $n$ – 1 remaining root nodes using the transformed non-negative costs.

Assuming a dense graph, the BF algorithm runs in $O\left(nm\right) \approx O\left(n^3\right)$ and Dijkstra's algorithm runs in $O\left(n^2\right)$, so this procedure runs in

$$O\left(n^3\right) + \left(n-1\right)O\left(n^2\right) = O\left(n^3\right)$$

Which is much better than the $O\left(n^4\right)$ which we'd need if we applied the BF algorithm $n$ times.

- Consider a network $\left(N, \mathcal{A}\right)$ with cost $c_{ij}$ associated with arc $(i,j) \in \mathcal{A}$. The **minimal spanning tree problem** is concerned with finding a spanning tree of least cost through this network. We first prove a theorem:

> **Theorem**: Let $U$ be some proper subset of $N$. If
>
> $$\left(u,v\right) = \underset{u \in U, v \in N \setminus U}{\operatorname{argmin}} c_{uv}$$
>
> then there is a minimal cost spanning tree that includes $(u, v)$ as an arc.

***Proof***: Assume that $T$ is a spanning tree that does not contain $(u, v)$. Adding $(u, v)$ to $T$ produces a cycle. There must therefore be another arc $\left(u', v'\right)$ such that $u' \in U, v' \in N \setminus U$, or else there would be no way for the cycle to "return" to $U$.

Now, imagine deleting $\left(u', v'\right)$ and replacing it with $(u, v)$ to form a new spanning tree $T'$. By assumption, $c_{uv} \leq c_{u'v'}$, and so the cost of $T'$ is $\leq$ the cost of $T$. Thus, $T'$ is also a minimal spanning tree, which contradicts our assumption that no minimal spanning tree contains $(u, v)$. ■

Prim's greedy algorithm constructs a MST as follows:

- Label the nodes $N = \left\{1, 2, \cdots, n\right\}$ and set $U = \left\{1\right\}$
- For the cheapest arc $(u, v)$ connecting $U$ and $N \setminus U$.
- Add $v$ to $U$ and repeat until $U = N$.

Prim's algorithm can easily be implemented on the matrix representation of a graph by crossing out rows corresponding to each vertex added to the spanning tree and choosing the smallest entry in *any* column corresponding to an already-added row.

The algorithm can be made to run in $O(n^2)$ provided that before each step, we already know the shortest distance between $U$ and any $j \notin U$ [say $c_{Uj} = \min_{i \in U} c_{ij}$]. It then takes no more than $n$ comparisons to find the best node to add to our tree. We can then find the shortest distance between $U' = U + \{v\}$ and any $j$ in $N \setminus U'$ by evaluating $c_{U'j} = \min\{c_{vj}, c_{Uj}\}$. Thus, the algorithm requires at most $n$ comparisons, and takes $n - 1$ steps.

## The Travelling Salesman Problem (TSP)

- Given an undirected graph $G = (N, \mathcal{A})$ consisting of $n$ nodes and $m$ arcs together with costs $c_{ij}$ for each arc $\{i, j\} \in \mathcal{A}$, the **travelling salesman problem** (TSP) is to find a tour of minimum cost.
- There are three types of algorithms
  - **Exact algorithms** are guaranteed to find an optimal solution but may take an exponential number of iterations.
  - **Approximations algorithms** have polynomial worse-case time complexity, supplying a suboptimal solution with a guaranteed bound on the degree of suboptimality.
  - **Heuristic algorithms** supply suboptimal solutions without any bound on their quality. They do not guarantee polynomial running times, but often provide a successful tradeoff between optimality and speed.
- One **exact method** is to formulate the problem is an integer linear program and solve it using branch-and-bound.

Set $x_{ij} = 1$ if $\{i, j\} \in \mathcal{A}$ is present in the tour, 0 otherwise. Define

$$\delta(S) = \{\{i, j\} \in \mathcal{A} : i \in S, j \notin S\}$$

For a tour, there must be two arcs incident to every node, so

$$\sum_{\{i,j\} \in \delta(\{i\})} x_{ij} = 2 \qquad i \in N$$

Furthermore, for any partition of the nodes into subsets $S$ and $N \setminus S$, there must be at least two edges connecting $S$ and $N \setminus S$, and so

$$\sum_{\{i,j\} \in \delta(S)} x_{ij} \geq 2 \qquad \forall S \subset N \text{ s.t. } S \neq \varnothing \text{ and } S \neq N$$

The so-called **cutest formulation** of the TSP is therefore

$$\min\left\{\sum_{\{i,j\}\in\mathcal{A}} c_{ij}x_{ij}\right\}$$

Subject to the two constraints above and $x_{ij} \in \{0,1\}$.

Alternatively, the **subtour elimination formulation** of the TSP replaces the second constraint above by

$$\sum_{\{i,j\}:i,j\in S} x_{ij} \le \left|S\right| - 1 \qquad \forall S \subset N \text{ s.t. } S \neq \varnothing \text{ and } S \neq N$$

This ensures there is no cycle involved less than all $n$ nodes.

In both cases, we have an exponential number of constraints, because there are $2^n - 2$ possible subsets of $N$. It turns out that the LP relaxation of both problems has the same feasible set.

- There is also a way to formulate this linear program with a **polynomial** number of constraints. First, we note that each node in the tour must have exactly one node preceding it and one node following it. Thus

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij} = 1 \qquad \forall i \qquad (*)$$

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij} = 1 \qquad \forall j \qquad (*)$$

These constraints, however, do not ensure that the solutions to not consist of several sub-tours, each disconnected from each other. To prevent that, we require that if $x_{ij} = 1$, then $t_j = t_i + 1$ (where $t_i$ is the position of node $i$ in the subtour). In fact, it turns out that the following works just as well:

$$t_j \ge t_i + 1 - n\left(1 - x_{ij}\right) \qquad i \ge 0, j \ge 1, i \neq j \qquad (*)$$

To see why, consider a subtour that does not include node 0 and includes $r$ nodes. Summing the inequalities above for all $x_{ij}$ in this tour gives $0 \ge r$, and so there can be no such smaller subtour.

The TSP can therefore be formulated as an ILP in $n^2 + n$ variables and $2n + n\left(n-1\right)$ constraints. Namely:

$$\min\left\{\sum_{i,j} c_{ij}x_{ij}\right\}$$

Subject to the three stared constraints above, $x_{ij} \in \left\{0,1\right\}$, $t_0 = 0$ and $t_i \in \left\{0,1,\cdots,n-1\right\}$.

- Notice that by relaxing the sub-tours constraint, we are left with an assignment problem, which can efficiently be solved by the network

simplex to provide a lower bound on the optimal solution. We need not worry about integer constraints since the network simplex algorithm will find an optimal solution.

If the said solution contains all cities, then it is optimal for the original TSP. Otherwise, we continue with a branch-and-bound using a branching rule that breaks the problem in two by an additional constraint of the form $x_{ij} = 0$ (ie: setting $c_{ij} = \infty$ ) – perhaps by choosing an arc in a subtour.

This creates a new TSP whose solution will be a subset of the old one, and solve the corresponding assignment problem to find a lower bound on that branch.

- We now move on to **approximation algorithms**

> **Definition (approximation algorithm)**: An **approximation algorithm** for a minimization problem with optimal cost $Z_{\text{opt}}$ runs in polynomial time and returns a feasible solution cost cost $Z_{\text{app}}$ such that
> $$Z_{\text{app}} \le \left(1+\varepsilon\right) Z_{\text{opt}}$$

We consider one such algorithm for $\varepsilon = 1$ and with costs that satisfy the triangle inequality:

$$c_{ij} \le c_{ik} + c_{kj} \qquad \forall i, j, k$$

The algorithm works as follows:

- Suppose $M$ is the cost of the minimal spanning tree (obtained using Prim's algorithm). Consider any starting node and traverse the minimal spanning tree to visit all the nodes. This uses each arc exactly twice, with cost $2M$.
- This can be converted into a tour visiting all the nodes by skipping any intermediate node that has already been visited. By the triangle inequality, this will have cost bounded above by $2M$.
- Also, every tour contains a spanning tree (since dropping one arc leaves a spanning tree) and so has cost at least $M$.

Thus, a straight-forward algorithm based on the MST gives

$$Z_{\text{app}} \le 2M \le 2Z_{\text{opt}}$$

- **Heuristic methods** for the TSP include

- **Nearest neighbour heuristic**: start at some city and then visit the nearest city. Continue until the tour is complete.

  The results are usually bad, but can form a good starting point for local search methods.
- **Cheapest insertion heuristic**: start with a single node and then, one by one, add the node whose insertion makes the smallest increase to the length of the tour.
- **Furthest insertion heuristic**: insert the node whose minimal distance to the existing tour node is greatest. The idea is to determine the overall layout of the tour early on.
- **Savings heuristic**: rank the arcs in ascending order of cost. Add the arcs in this order, so long as they do not violate any constraints, and until all cities have been visited.

- We also consider the concept of a **neighbourhood search**. Consider the general problem

$$\min\left\{c(x)\right\} \text{ s.t. } x \in X$$

  Suppose that for any point $x \in X$ we have a set of **neighbourhood points** $N(x) \subset X$. The basic approach of local search is
  - Select some $x \in X$
  - Evaluate $c(x)$
  - Pick some $y \in N(x)$ and evaluate $c(y)$. If $c(y) < c(x)$, select $y$ as the new value for $x$ and result to the previous step. If there is no such $y$, stop with solution $x$.

  We need to specify $N(x)$ and which $y \in N(x)$ to choose. We might also choose the best $y \in N(x)$, or the best of a few tries, etc... Note also that simplex is a local search method which happens to lead to a global optimum because in linear programming, any local optimum is a local optimum.

  In the TSP, we can define a **neighbour** to a feasible solution $x$ by removing any $k \geq 2$ arcs from the tour, and replacing them with $k$ new ones. When $k = 2$, the method is known as **2OPT**. It can be shown that **3OPT** gives better results than 2OPT, but going beyond 3 doesn't gain

much. Choosing a greater neighbourhood gives a better solution, but makes $N$ bigger and therefore results in a slower algorithm.

In practice, we fix the neighbourhood size and repeat the algorithm starting at different optimal solutions.

- **Simulated annealing** tries to prevent the algorithm getting stuck at a local optimum. It allows the algorithm to jump to worse neighbours originally but slowly becomes more reluctant to allow such jumps.

  A jump from $x$ to $y \in N(x)$ can occur with probability

  $$p_{xy} = \min\left(1, \exp\left[-\frac{c(y)-c(x)}{T}\right]\right)$$

  where $T$ starts large and decreases at each iteration (a common schedule for $T$ is to let it decrease with iteration number as $T(t) = C \,/\, \log t$, with constant $C$).

  It can be shown that if $C$ is sufficiently large,

  $$\lim_{t\to\infty} P\left(\boldsymbol{x}(t) \text{ is optimal}\right) = 1$$

- Finally, **genetic algorithms** can be used in a wide range of problems, but they can often get stuck at a minimum. They run as follows:
  - **Initiate**: Create a random initial state (ie: a set of tours).
  - **Evaluate fitness** of each item (eg: the length of the tour)
  - **Reproduce**: those fitter chromosomes are more likely to reproduce. For example, "greedy crossover" selects the first city of one parent, then the closest next city in either parents, etc… If both cities have already been chosen, choose another random one.
  - **Mutate**: randomly swap a pair of cities.

# Game Theory

## Terminology

- A **move** is either a decision by a player or the outcome of a chance event.
- A **game** is a sequence of moves, some of whicsh may be simultaneous.
- At the end of each game, each player receives a **return** (**payoff**). The payoff to each player is a *real number*. If a move has a *random outcome*, we use an *expected payoff*.
- A **strategy** is a description of the decisions that a player will make at all possible situations that can arise in the game.
- A game is **zero-sum** if the sum of the players' payoffs is always 0.
- A game has **perfect information** if at every move of the game, all players know all the moves that have already been made (including random moves).

## Two-Person, Zero-Sum Games

- **Introduction**
  - Players I and II, each with a finite number of **pure strategies** ($\mathrm{I}_1 \cdots \mathrm{I}_n$ for player I and $\mathrm{II}_1 \cdots \mathrm{II}_m$ for player II).
  - $e_{ij}$ denotes the **payoff** to **player I** when player I uses strategy $\mathrm{I}_i$ and player II uses strategy $\mathrm{II}_j$. We also denote $e_1(i,j) = e_{ij} = -e_2(i,j)$ where $e_1$ and $e_2$ are the payoffs to player I and II.
  - The players can play a pure strategy or a **mixed strategy**. We say that player I adopts strategy **p** and player II adopts strategy **q** if
    $$\mathbb{P}\left(\text{Player I plays } \mathrm{I}_i\right) = p_i \qquad \mathbb{P}\left(\text{Player II plays } \mathrm{II}_j\right) = q_j$$
    In that case, the payoff to player I is
    $$e_1\left(\boldsymbol{p},\boldsymbol{q}\right) = e\left(\boldsymbol{p},\boldsymbol{q}\right) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_i e_{ij} q_j = \boldsymbol{p}^T e \boldsymbol{q}$$
    The **normal form representation** of the game is simply the matrix $\left(e_{ij}\right)$.
- **Maximin Criterion**
  - Player II loses what player I wins. So if player I chooses strategy $\mathrm{I}_i$, player II will play $\mathrm{II}_j$ resulting in the *minimum* gain to I. Player I

can therefore guarantee a gain of at least the **lower value of the game**

$$v_L = \max_i \min_j e_{ij}$$

If player II choses $\text{II}_j$, then I will chose $\text{I}_i$ that **maximises** $e_{ij}$. So II can ensure that it will note lose more than the **upper value of the game**

$$v_U = \min_j \max_i e_{ij}$$

If $v_L = v_U$, then this is the **solution of the game**.

- In a game with **mixed strategies**, these become

$$v_L^M = \max_{\boldsymbol{p}\in P} \min_{\boldsymbol{q}\in Q} e(\boldsymbol{p},\boldsymbol{q}) \qquad v_U^M = \min_{\boldsymbol{q}\in Q} \max_{\boldsymbol{p}\in P} e(\boldsymbol{p},\boldsymbol{q})$$

(Where $P$ and $Q$ are the set of possible mixed strategies).

**Lemma**: $v_L \le v_U$

***Proof***: For $\forall\ \boldsymbol{p}\in P, \boldsymbol{q}\in Q$

$$e(\boldsymbol{p},\boldsymbol{q}) \le \max_{\boldsymbol{p}'\in P} e(\boldsymbol{p}',\boldsymbol{q})$$
$$\min_{\boldsymbol{q}\in Q} e(\boldsymbol{p},\boldsymbol{q}) \le \min_{\boldsymbol{q}\in Q} \max_{\boldsymbol{p}'\in P} e(\boldsymbol{p}',\boldsymbol{q})$$

This holds for **all** $\boldsymbol{p}\in P$, so we might as well choose the $\boldsymbol{p}$ that maximises the LHS:

$$\max_{\boldsymbol{p}\in P} \min_{\boldsymbol{q}\in Q} e(\boldsymbol{p},\boldsymbol{q}) \le \min_{\boldsymbol{q}\in Q} \max_{\boldsymbol{p}'\in P} e(\boldsymbol{p}',\boldsymbol{q})$$
$$v_L^M \le v_U^M$$

■

Further, the **minimax theorem** states that

**Theorem (minimax theorem)**: In a two-person zero-sum game where I and II have a finite number of strategies,

$$v_L^M = \max_{\boldsymbol{p}\in P} \min_{\boldsymbol{q}\in Q} e(\boldsymbol{p},\boldsymbol{q}) = \min_{\boldsymbol{q}\in Q} \max_{\boldsymbol{p}\in P} e(\boldsymbol{p},\boldsymbol{q}) = v_U^M = v$$
$$v = e\left(\boldsymbol{p}^*, \boldsymbol{q}^*\right)$$

$v$ is called the **value of the game** and together with the optimal strategies $\boldsymbol{p}^*$ and $\boldsymbol{q}^*$ is called the **solution to the game**.

We also define

**Definition (equilibrium pairs)**: A set of strategies $\boldsymbol{p}^*$ and $\boldsymbol{q}^*$ are **equilibrium pairs** if for any $\boldsymbol{p}\in P$ and $\boldsymbol{q}\in Q$

$$e\left(\boldsymbol{p},\boldsymbol{q}^*\right) \le e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right) \le e\left(\boldsymbol{p}^*,\boldsymbol{q}\right)$$

Two important theorems:

**Lemma**: If $\left(\boldsymbol{p},\boldsymbol{q}\right)$ and $\left(\boldsymbol{p}',\boldsymbol{q}'\right)$ are equilibrium pairs then $e\left(\boldsymbol{p},\boldsymbol{q}\right) = e\left(\boldsymbol{p}',\boldsymbol{q}'\right)$

***Proof***: We have that

$$e\left(\boldsymbol{p}',\boldsymbol{q}\right) \le e\left(\boldsymbol{p},\boldsymbol{q}\right) \le e\left(\boldsymbol{p},\boldsymbol{q}'\right)$$
$$e\left(\boldsymbol{p},\boldsymbol{q}'\right) \le e\left(\boldsymbol{p}',\boldsymbol{q}'\right) \le e\left(\boldsymbol{p}',\boldsymbol{q}\right)$$

The result trivially follows ■

**Theorem**: A pair of stragies $\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)$ in a zero-sum game are an equilibrium pair if and only if $\left(\boldsymbol{p}^*,\boldsymbol{q}^*,e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)\right)$ is a solution to the game.

***Proof***: If $\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)$ is an equilibrium pair, then the boxed statements hold:

$$v_U^M = \min_{\boldsymbol{q}\in Q}\max_{\boldsymbol{p}\in P} e\left(\boldsymbol{p},\boldsymbol{q}\right)$$
$$\min_{\boldsymbol{q}\in Q}\max_{\boldsymbol{p}\in P} e\left(\boldsymbol{p},\boldsymbol{q}\right) \le \max_{\boldsymbol{p}\in P} e\left(\boldsymbol{p},\boldsymbol{q}^*\right)$$

$$\boxed{\begin{aligned}\max_{\boldsymbol{p}\in P} e\left(\boldsymbol{p},\boldsymbol{q}^*\right) &\le e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)\\ e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right) &\le \min_{\boldsymbol{q}\in Q} e\left(\boldsymbol{p}^*,\boldsymbol{q}\right)\end{aligned}}$$

$$\min_{\boldsymbol{q}\in Q} e\left(\boldsymbol{p}^*,\boldsymbol{q}\right) \le \max_{\boldsymbol{q}\in P}\min_{\boldsymbol{q}\in Q} e\left(\boldsymbol{p},\boldsymbol{q}\right)$$
$$\max_{\boldsymbol{q}\in P}\min_{\boldsymbol{q}\in Q} e\left(\boldsymbol{p},\boldsymbol{q}\right) = v_L^M$$

Taking the two extremes, these calculations imply that $v_U^M \le v_L^M$. But we know that $v_L^M \le v_U^M$, and so

$$v_L^M = v_U^M = e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)$$

As such $\left(\boldsymbol{p}^*,\boldsymbol{q}^*,e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)\right)$ is a solution to the game.

Conversely, if $\left(\boldsymbol{p}^*,\boldsymbol{q}^*,e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)\right)$ is a solution to the game, then the boxed statement holds, and for $\forall\ \boldsymbol{p}\in P,\boldsymbol{q}\in Q$:

$$e(\boldsymbol{p},\boldsymbol{q}^*) = \min_{\boldsymbol{q}'\in Q} e(\boldsymbol{p},\boldsymbol{q}')$$
$$\min_{\boldsymbol{q}'\in Q} e(\boldsymbol{p},\boldsymbol{q}') \le \max_{\boldsymbol{p}'\in P}\min_{\boldsymbol{q}'\in Q} e(\boldsymbol{p}',\boldsymbol{q}')$$
$$\max_{\boldsymbol{p}'\in P}\min_{\boldsymbol{q}'\in Q} e(\boldsymbol{p}',\boldsymbol{q}') = e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)$$
$$\boxed{v_L^M = e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right) = v_U^M}$$
$$e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right) = \min_{\boldsymbol{q}'\in Q}\max_{\boldsymbol{p}'\in P} e(\boldsymbol{p}',\boldsymbol{q}')$$
$$\min_{\boldsymbol{q}'\in Q}\max_{\boldsymbol{p}'\in P} e(\boldsymbol{p}',\boldsymbol{q}') \le \max_{\boldsymbol{p}'\in P} e(\boldsymbol{p}',\boldsymbol{q})$$
$$\max_{\boldsymbol{p}'\in P} e(\boldsymbol{p}',\boldsymbol{q}) = e\left(\boldsymbol{p}^*,\boldsymbol{q}\right)$$

Taking the two extremes and the centre, we obtain

$$e\left(\boldsymbol{p},\boldsymbol{q}^*\right) \le e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right) \le e\left(\boldsymbol{p}^*,\boldsymbol{q}\right)$$

So the points are equilibrium points. ■

- **Dominating strategies**
  - If, whatever strategy II chooses, $I_i$ will always result in a lesser or equal payoff than $I_j$, then $I_j$ **dominates** $I_i$. Similarly vice-versa.
  - Dominated strategies can be ignored

> **Lemma**: If a dominated strategy is removed from a game, the solution of the reduced game is a solution of the original game.

***Proof***: Suppose $I_2$ dominates $I_1$ and $\left(\boldsymbol{p}^*,\boldsymbol{q}^*,v\right)$ is a solution to the reduced game when $I_1$ is removed. It is obvious that $e\left(\boldsymbol{p}^*,\boldsymbol{q}\right) \ge v$ for $\forall\ \boldsymbol{q}\in Q$. It is also obvious that

$$e\left(\boldsymbol{p},\boldsymbol{q}^*\right) \le v \qquad \forall\ \boldsymbol{p}\in P' \qquad (*)$$

where $P'$ is the **reduced set** of mixed strategies, with $p_1 = 0$. We simply need to show that this is also true for $\boldsymbol{p}\in P$, where $P$ is the **full set** of mixed strategies.

To do that, let $\boldsymbol{p} = \left(p_1,p_2,p_3,\cdots,p_m\right)$. We then have

$$e\left(\boldsymbol{p},\boldsymbol{q}^*\right) = \boldsymbol{p}^T e\boldsymbol{q}^* = p_1\sum_{j=1}^{m} e_{1j}q_j^* + \sum_{i=2}^{n}\sum_{j=1}^{m} p_i e_{ij} q_j^*$$
$$\le p_1\sum_{j=1}^{m} e_{2j}q_j^* + \sum_{i=2}^{n}\sum_{j=1}^{m} p_i e_{ij} q_j^* = e\left(\boldsymbol{p}',\boldsymbol{q}^*\right)$$

Where $\boldsymbol{p}' = \left(0,p_1+p_2,p_3,\cdots,p_m\right)$ is a strategy in the reduced game. We therefore have that

$$e\left(\boldsymbol{p},\boldsymbol{q}^*\right) \le e\left(\boldsymbol{p}',\boldsymbol{q}^*\right) \le v$$

Where the last inequality follows from (*). Thus

$$e\left(\boldsymbol{p},\boldsymbol{q}^*\right) \le v \qquad \forall\ \boldsymbol{p} \in P$$

As required. ■

- Strategies can also be dominated by *mixed strategies*, but those are harder to find.

- **Solving two-person zero-sum games**
  - For $2 \times m$ games, let player I play $\boldsymbol{p} = \left(p, 1-p\right)$
    - Plot the payoff to I against $p$ for all of II's strategies.
    - At I's optimal strategy, the **lowest value of the payoff** will be **as high as it can be**. This is the **lowest intersection** of two of those lines
    - One of these lines will have **positive gradient**, the other **negative gradient**. A combination of these two lines (=strategies) will have **0 gradient**. This is the **optimal strategy for II**
  - For $m \times n$ games, we want to find vectors $\boldsymbol{p}^*$ and $\boldsymbol{q}^*$ such that $v = e\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)$ and

$$\overbrace{e\left(\boldsymbol{p},\boldsymbol{q}^*\right) \le v}^{\substack{\text{Constraint on player II's}\\ \text{optimal strategy}}} \le \underbrace{v \le e\left(\boldsymbol{p}^*,\boldsymbol{q}\right)}_{\substack{\text{Constraint on player I's}\\ \text{optimal strategy}}}$$

Consider player II's game:

- His constraint is $e\left(\boldsymbol{p},\boldsymbol{q}^*\right) \le v$, which can be re-written as

$$\sum_{j=1}^{m} e_{ij} q_j^* \le v \qquad \forall\ i$$

- His objective is to make $v$ **as small as possible**.

His problem is therefore

$$\min\left\{v : \sum_{j=1}^{m} e_{ij} q_j^* \le v, \qquad q_j^* > 0, \qquad \sum_{j=1}^{m} q_j^* = 1\right\}$$

The problem can be stated more simply by writing $Q_j = q_j^* / v$:

$$\max\left\{\sum_{j=1}^{m} Q_j : \sum_{j=1}^{m} e_{ij} Q_j \le 1, \qquad Q_j \ge 0\right\}$$

We can then find our solution by noting that $\sum_{j=1}^{m} Q_j = \frac{1}{v}$ (this also automatically incorporates the last constraint).

We can ensure that $v > 0$ by adding a constant amount $c$ to every payoff, such that $\tilde{e}_{ij} = e_{ij} + c$. This will not change the equilibrium pairs (it's equivalent to II giving I a present $c$ to play the game) and $c$ can then be removed from the value of the solution to obtain the value of the original game.

Now, consider the dual problem

$$\min\left\{\sum_{j=1}^{m} P_j : \sum_{i=1}^{n} P_i e_{ij} \geq 1, \qquad P_i \geq 0\right\}$$

If we interpret $P_i = p_i^* / v$, we can re-write this as

$$\max\left\{v : \sum_{i=1}^{n} e_{ij} p_i^* \geq v, \qquad p_i^* > 0, \qquad \sum_{i=1}^{n} p_i^* = 1\right\}$$

Thus, the primal LP gives II's solution, whereas the dual LP gives I's solution.

- For $2 \times 2$ games, let $\boldsymbol{p} = (p, 1-p)$ and $\boldsymbol{q} = (q, 1-q)$. The solution of the game can be found directly from the definition by solving the equations

$$\frac{\partial}{\partial p} e(\boldsymbol{p}, \boldsymbol{q})\Big|_{\substack{\boldsymbol{p}=\boldsymbol{p}^* \\ \boldsymbol{q}=\boldsymbol{q}^*}} = 0 \qquad \frac{\partial}{\partial q} e(\boldsymbol{p}, \boldsymbol{q})\Big|_{\substack{\boldsymbol{p}=\boldsymbol{p}^* \\ \boldsymbol{q}=\boldsymbol{q}^*}} = 0$$

## Two-Person, Non-zero-Sum Games

- **Introduction**
  - In non-zero sum games, we write the outcome as **pairs**, and the game is specified by two matrices; $e_1(\boldsymbol{p}, \boldsymbol{q})$ and $e_2(\boldsymbol{p}, \boldsymbol{q})$.
  - The players are no longer totally antagonistic to each other – they might both be happier with one outcome than with another.
- **Solution concepts** – for non-zero-sum games, there is no longer an obvious solution concept. Here are two:
  - **Maximin-Maximin pairs** – each player considers his own game, and find his solution to the game. This results in two values for the

game, $v_{\mathrm{I}}$ and $v_{\mathrm{II}}$, and it is **not** generally the case that the payoff when both players play their maximin strategies against each other is $\left(v_{\mathrm{I}}, v_{\mathrm{II}}\right)$.

The **minimax-minimax** solution concept is generally not considered to be ideal, because it assumes that each player tries to maximise their own profit as well as minimize their opponent's profit; these may be diametrically opposed objectives.

- **Equilibrium pairs** – a pair of strategies $\boldsymbol{p}^* \in P$ and $\boldsymbol{q}^* \in Q$ is an **equilibrium pair** if and only if for any $\boldsymbol{p} \in P$ and $\boldsymbol{q} \in Q$

$$e_1\left(\boldsymbol{p}, \boldsymbol{q}^*\right) \le e_1\left(\boldsymbol{p}^*, \boldsymbol{q}^*\right) \qquad e_1\left(\boldsymbol{p}^*, \boldsymbol{q}\right) \le e_2\left(\boldsymbol{p}^*, \boldsymbol{q}^*\right)$$

There is an important theorem concerning equilibrium pairs:

> **Theorem (Nash's Theorem)**: Any two-person game (zero-sum or non-zero-sum) with a finite set of pure strategies has at least one equilibrium pair.

***Proof***: Let $S = \left\{(\boldsymbol{p}, \boldsymbol{q}) : \boldsymbol{p} \in P, \boldsymbol{q} \in Q\right\}$. $S$ is then closed, convex and bounded. Also define

$$c_i\left(\boldsymbol{p}, \boldsymbol{q}\right) = \max\left\{0, e_1\left(\mathrm{I}_i, \boldsymbol{q}\right) - e_1\left(\boldsymbol{p}, \boldsymbol{q}\right)\right\}$$
$$d_j\left(\boldsymbol{p}, \boldsymbol{q}\right) = \max\left\{0, e_2\left(\boldsymbol{p}, \mathrm{II}_j\right) - e_2\left(\boldsymbol{p}, \boldsymbol{q}\right)\right\}$$

These are the quantities to be gained by playing a given pure strategy instead of the mixed strategy. Define[2]

$$f\left(\boldsymbol{p}, \boldsymbol{q}\right) = \left(\boldsymbol{p}', \boldsymbol{q}'\right) = \left(\frac{\boldsymbol{p} + \boldsymbol{c}(\boldsymbol{p}, \boldsymbol{q})}{1 + \left|\boldsymbol{c}\left(\boldsymbol{p}, \boldsymbol{q}\right)\right|_1}, \frac{\boldsymbol{q} + \boldsymbol{d}(\boldsymbol{p}, \boldsymbol{q})}{1 + \left|\boldsymbol{d}\left(\boldsymbol{p}, \boldsymbol{q}\right)\right|_1}\right)$$

$f$ is continuous, and so by the **Brouwer fixed point theorem**, there exists a fixed point such that

$$f\left(\boldsymbol{p}^*, \boldsymbol{q}^*\right) = \left(\boldsymbol{p}^*, \boldsymbol{q}^*\right)$$

We cannot have $e_1\left(I_i, \boldsymbol{q}^*\right) > e_1\left(\boldsymbol{p}^*, \boldsymbol{q}^*\right)$ for *all* $i$ because that would imply the boxed statement below which leads to a contradiction:

[2] Note that $|\boldsymbol{x}|_1 = x_1 + x_2 + \ldots$ is the $L1$-norm of $\boldsymbol{x}$.

$$e_1\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)=\sum_{i=1}^{n}p_i^*e_1\left(I_i,\boldsymbol{q}^*\right)$$

$$\boxed{\sum_{i=1}^{n}p_i^*e_1\left(I_i,\boldsymbol{q}^*\right)>\sum_{i=1}^{n}p_i^*e_1\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)}$$

$$\sum_{i=1}^{n}p_i^*e_1\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)=e_1\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)\sum_{i=1}^{n}p_i^*=e_1\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)$$

This means that we have at least one $i$ for which $e_1\left(I_i,\boldsymbol{q}^*\right)\le e_1\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)$, and so at least one $i$ for which

$$c_i\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)=0$$

Now, because $\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)$ is a fixed point, we have that $\boldsymbol{p}^*=\boldsymbol{p}^{*\prime}$, and choosing the value of $i$ for which $c_i\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)=0$, we get

$$\sum_{i=1}^{n}c_i\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)=0$$

Since $c_i\ge 0$, this means that $c_i=0$ for all $i$. As such, for all $i$

$$e_1\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)\ge e_1\left(I_i,\boldsymbol{q}^*\right)$$

And so

$$e_1\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)\ge e_1\left(\boldsymbol{p},\boldsymbol{q}^*\right)\qquad\forall\ \boldsymbol{p}\in P$$

Similarly, we can show that $e_2\left(\boldsymbol{p}^*,\boldsymbol{q}^*\right)\ge e_2\left(\boldsymbol{p}^*,\boldsymbol{q}\right)$, and so the fixed point is an equilibrium pair. ■

Finding equilibrium pairs is tricky, since the equations in the proof to Nash's Theorem involve quadratics. For $2\times 2$ games, a graphical method is useful:

- Let $\boldsymbol{p}=\left(p,1-p\right)$ and $\boldsymbol{q}=\left(q,1-q\right)$.
- For a particular $q$, find the $p$ that maximises $e_1\left(\boldsymbol{p},\boldsymbol{q}\right)$. Plot these on a $p$–$q$ graph.
- For a particular $p$, find the $q$ that maximises $e_2\left(\boldsymbol{p},\boldsymbol{q}\right)$, and plot that too.
- The intersections of the two graphs above are the equilibrium points.

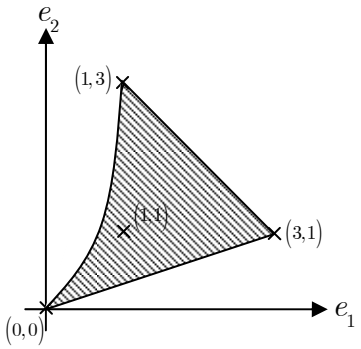
- **Cooperation** – consider the following game

| | $\text{II}_1$ | $\text{II}_2$ |
|---|---|---|
| $\text{I}_1$ | $(0,0)$ | $(1,1)$ |
| $\text{I}_2$ | $(3,1)$ | $(1,3)$ |

If I and II adopt strategies $\boldsymbol{p} = \big(p, 1-p\big)$ and $\boldsymbol{q} = \big(q, 1-q\big)$ respectively,

$$e_1\big(\boldsymbol{p},\boldsymbol{q}\big) = 2q - 3pq + 1 \qquad e_2\big(\boldsymbol{p},\boldsymbol{q}\big) = pq - 2p - 2q + 3$$

The payoff region for this game is as follows:



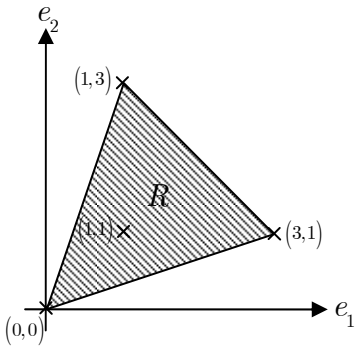

If, however, we allow the players to cooperate, then they can decide to:

- Play a strategy that leads to payoff $\big(u_1, v_1\big)$ with probability $\alpha$.
- Play a strategy that leads to payoff $\big(u_2, v_2\big)$ with probability $1-\alpha$.

The resulting payoff is

$$\beta\big(u_1, v_1\big) + \big(1-\beta\big)\big(u_2, v_2\big)$$

Thus, the payoff region $R$ for the cooperative game is given by the **convex hull** of the region for the non-cooperative game; ie: the smallest convex region that covers the region for the non-cooperative game:




The easiest way of obtaining this region is to draw the convex hull of points equivalent to each player adopting one pure strategy.

Here randomisation is used to average over possible outcomes, each of which might be mixed strategies.

- **Bargaining** – once we move to a cooperative game, the problem of pre-play negotiation comes to the fore. First, some definitions:

> **Definition (Joint domination)**: A pair of payoffs $(u,\ v)$ in a cooperative game is **jointly dominated** by $\left(u',v'\right)$ if $u' \geq u$ and $v' \geq v$ and $\left(u',v'\right) \neq \left(u,v\right)$.

> **Definition (Pareto optimality)**: A pair of payoffs $(u,\ v)$ is **Pareto optimal** if it is not jointly dominated.
>
> It is clear from the diagrams above that a point can only be Pareto optimal if it is on the **edge** of the payoff region.

It is clear that the players of the game would only be interested in Pareto-optimal payoffs. Furthermore, by simply failing to cooperate, each player can guarantee themselves a payoff of at least

$$v_{\text{I}} = \max_{\boldsymbol{p}\in P}\min_{\boldsymbol{q}\in Q} e_1\left(\boldsymbol{p},\boldsymbol{q}\right) \qquad v_{\text{II}} = \max_{\boldsymbol{q}\in Q}\min_{\boldsymbol{p}\in P} e_2\left(\boldsymbol{p},\boldsymbol{q}\right)$$

respectively. Thus, we would expect the solution of the cooperative game to lie within the **bargaining set** (or **negotiation set**) $B$

$$B = \left\{\left(u,v\right) \mid u \geq v_{\text{I}}, v \geq v_{\text{II}}, \left(u,v\right) \text{ Pareto optimal in } R\right\}$$

How do the players choose amongst the members of $B$?[3] Nash suggested that there is a special payoff $\left(u_0,v_0\right) \in R$ called the **status-quo point** which is the outcome if the participants cannot agree on the transaction. An **arbitration procedure** $\psi$ is then a map from that point to another point $\left(u^*,v^*\right) \in R$

$$\psi\left((u_0,v_0),R\right) = \left(u^*,v^*\right)$$

Nash suggested that such a procedure should fulfil a number of properties:

> **Definition (Nash Arbitration Procedure)**: A Nash Arbitration Procedure fulfils the following axioms:
>
> 1. **Feasibility** – $\left(u^*,v^*\right) \in R$

[3] It is important to remember that we must not make inter-player comparison of payoffs, which are measured by their own utilities, not necessarily on the same scales. For example, it is *not* clear that I prefers (4, 1) to (1, 10) less than II prefers (1, 10) to (4, 1).

2. **At least as good as status quo** – $u^* \geq u_0$, $v^* \geq v_0$
3. **Pareto optimality** – If $(u,v) \in R$ and $u \geq u^*$, $v \geq v^*$, then $(u,v) = (u^*, v^*)$.
4. **Symmetry** – if $R$ is symmetric, such that $(u,v) \in R \Rightarrow (v,u) \in R$ and if $u_0 = v_0$, then $u^* = v^*$. This assumes that the two players are roughly of the same size in power and skill in diplomacy.
5. **Invariance under linear transformations** – let $R'$ be obtained from $R$ by the following linear transformation

$$u' = au + b \quad v' = cv + d \quad a, c > 0$$

then if

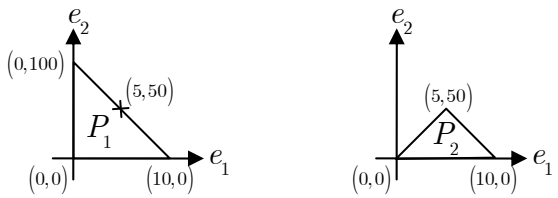
$$\psi\big((u_0, v_0), R\big) = (u^*, v^*)$$

then

$$\psi\big((u_0', v_0'), R'\big) = (u'^*, v'^*)$$

This is simply a statement of the fact utilities are only defined uniquely up to a linear transformations, and we are forbidding interpersonal comparison of utilities
6. **Independence of irrelevant alternatives** – if $R'$ is a subset of $R$, $\psi\big((u_0, v_0), R\big) = (u^*, v^*) \in R'$, then we must also have $\psi\big((u_0, v_0), R'\big) = (u^*, v^*)$.

Axiom 6 is the more controversial. Consider the payoff region $P_1$ with status-quo point (0, 0). Then (5, 50) seems a fair solution. Axiom 6 insists that if we cut the set of feasibly payoffs down to $P_2$, (5, 50) must still be the arbitration solution; this seems very generous to the second player:




Nash found a **unique function** $\psi$ that satisfied these bargaining axioms:

**Theorem (Nash's Arbitration Procedure)**: If there exists $(u,v) \in R$ with $u > u_0, v > v_0$ then consider the function

$$f(u,v) = (u-u_0)(v-v_0)$$

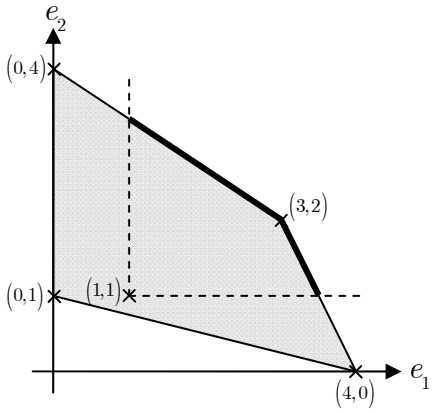
defined over such points. Its maximum occurs at a unique point $(u^*,v^*)$, and define

$$\psi((u_0,v_0),R) = (u^*,v^*)$$

If there are no points with $u > u_0$ *and* $v > v_0$, then simply try to maximise *one of* $u$ and $v$ as much as possible to get $\psi((u_0,v_0),R) = (u^*,v^*)$.

The arbitration function $\psi$ thus defined satisfies all the Nash bargaining axioms, and is the only one to do so.

The **maximin bargaining solution** or **Shapley solution** is obtained by applying the Nash arbitration procedure to the status-quo point $(u_0,v_0) = (v_{\mathrm{I}}, v_{\mathrm{II}})$. Consider the following game, with maximin-maximin solution $(1,1)$:




| | $\mathrm{II}_1$ | $\mathrm{II}_2$ | $\mathrm{II}_3$ |
|---|---|---|---|
| $\mathrm{I}_1$ | (2,1) | (3,2) | (0,4) |
| $\mathrm{I}_{11}$ | (0,1) | (4,0) | (2,1) |

The strategy is:

- Make a diagram of the cooperative payoff region $R$
- Plot the status-quo point $(u_0,v_0) = (v_{\mathrm{I}}, v_{\mathrm{II}})$ in that region
- Find the negotiation set (the edges of $R$ to the right and above the status-quo point that are also Pareto-optimal). In the case above, the negotiation set is shown in bold, and is

$$B = \{(u,v) \mid 2u+3v = 12, 1 \le u \le 3\} \cup \{(u,v) \mid 2u+v = 8, 3 \le u \le \tfrac{7}{2}\}$$

- The maximin bargaining solution must be in $B$, so simply maximise $f(u,v) = (u - u_0)(v - v_0)$ over that region. For each part of $B$, this involves
  - Substituting for $u$ or $v$ in $f$, using the equation of the line that forms that part of $B$
  - Maximising
  - Checking that the result lies in $B$. If not, the result is clearly at the edge of the line

  Finally, find the *global* maximum by comparing the maxima for each part of $B$.

## *N*-Person Games

- The concept of a maximin-maximin pair seems far-fetched for an $N$-person game, because requiring each player to minimize each of his opponent's payoffs can lead to ambiguities. The idea of an **equilibrium pair**, however, can easily be generalised

> **Definition (equilibrium *n*-tuple)**: The $n$-tuple of strategies $\boldsymbol{p}_1^*, \boldsymbol{p}_2^*, \cdots, \boldsymbol{p}_n^*$ where player $i$ plays mixed strategy $\boldsymbol{p}_i^*$ is an **equilibrium *n*-tuple** if for all other strategies $\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n$,
>
> $$e_i\left(\boldsymbol{p}_1^*, \boldsymbol{p}_2^*, \cdots, \boldsymbol{p}_i^*, \cdots, \boldsymbol{p}_n^*\right) \geq e_i\left(\boldsymbol{p}_1^*, \boldsymbol{p}_2^*, \cdots, \boldsymbol{p}_i, \cdots, \boldsymbol{p}_n^*\right)$$

  (An extension of Nash's Theorem states that any finite $n$-person non-cooperative game has at least one equilibrium $n$-tuple).

  $N$-person games without cooperation are hardly interesting; we simply have a large number of equilibrium $n$-tuples, and the problem reduces to deciding which one to adopt. We therefore focus on cooperative games.
- In an $N$-person game, we might get cooperation between some but not all of the players – we need to examine what **coalitions** of players might form. A **coalition** $S$ is a subset of $N = \{1, 2, \cdots, N\}$. The worse thing that could happen for a coalition is for the rest of the players to form a single opposing coalition ***N/S***. We then have a two-person non-cooperative game, and we can define

**Definition (characteristic function)**: The **characteristic function** of an $n$-person game assigns to each subset $S$ of the players the maximum value $v\big(S\big)$ that the coalition $S$ can guarantee itself by coordinating the strategies of its members, no matter what the other players do:

$$v\big(S\big) = \max_{\boldsymbol{p}\in P_S} \min_{\boldsymbol{q}\in Q_{N/S}} \sum_{i\in S} e_i\big(\boldsymbol{p},\boldsymbol{q}\big)$$

By definition, we take $v\big(\varnothing\big) = 0$.

An important property of $v$ is that it is **superadditive**

$$v\big(S \cup T\big) \geq v(S) + v(T) \qquad \text{if } S \cap T = \varnothing$$

(The condition requires that $S$ and $T$ be **disjoint coalitions**).

In some cases, it doesn't pay to form any coalition and $v$ is **additive**; in other words, $S \cap T = \varnothing \Rightarrow v\big(S \cup T\big) = v(S) + v(T)$ . In such a case, $v\big(N\big) = \sum_{i=1}^{n} v\big(\{i\}\big)$, and the game is called **inessential**.

This concept is most appropriate if the game is **constant sum** (ie: every outcome gives the same total payoff), because when $N$–$S$ try to maximise their combined payoffs, they are indeed minimising the payoff to $S$. However, for other games, the characteristic function is often highly pessimistic, because in reality, $N$–$S$ are not "out to get" $S$ as badly as possible – they're just trying to maximise their profit, which might not be the same thing.

For example, consider the "oil game" in which:

- Country 1 has oil it can use for transport, at a profit of $a$ per barrel.
- Country 2 can use it for manufacturing, at a profit of $b$ per barrel.
- Country 3 can use it for food production, at a profit of $c$ per barrel.

The characteristic function here is as follows:

| **Coalition $S$** | **$v(S)$** | **Comment** |
|---|---|---|
| $\varnothing$ | 0 | By definition |
| $\{1\}$ | $a$ | If 2 and 3 form a coalition against 1, they cannot force him to sell the oil. |
| $\{2\},\{3\},\{2,3\}$ | 0 | Because no coalition of buyers can make the seller sell them oil |
| $\{1,2\}$ | $b$ | Because 1 and 2 can use the oil at a profit $b$ |

| | | |
|---|---|---|
| $\{1,3\},\{1,2,3\}$ | $c$ | per barrel (1 sells to 2) and 3 would have to pay at least $b$ to get it<br>Since 1 and 3 can use 1's oil at a profit of $c$ per barrel. |

- Now that we have examined coalitions, we need to consider how to share out rewards amongst members of a coalition (which, in turn, determines which coalitions form). We call "reasonable" share-outs **imputations**

> **Definition (imputation)**: An **imputation** in an $n$-person game with characteristic function $v$ is a vector $\boldsymbol{x} = \left(x_1, x_2, \cdots, x_n\right)$ satisfying
>
> $$\sum_{i=1}^{n} x_i = v\left(N\right) \qquad\qquad x_i \geq v(i)$$
>
> We denote the **set of all imputations** in a game by $E(v)$.

The second condition says everyone must get as much as they could get if they played by themselves. The first condition is a Pareto optimality condition – the RHS is the most the players can get when they work together, so we must have $\sum_{i=1}^{n} x_i \leq v(N)$. However, if the inequality was strict, then by working together they could always share out the rewards so that everyone got more.

In an inessential game, there is only one imputation, but for essential games, there are lots. In the oil market game, for example

$$E\left(v\right) = \left\{(x_1, x_2, x_3) : x_1 + x_2 + x_3 = c, x_1 \geq a, x_2 \geq 0, x_3 \geq 0\right\}$$

Consider two imputations $\boldsymbol{x}$ and $\boldsymbol{y}$. We have $\sum x_i = v(N) = \sum y_i$, and so $\boldsymbol{y}$ cannot be better than $\boldsymbol{x}$ for *everyone*. However it is possible that for a **particular coalition**, $\boldsymbol{x}$ is better than $\boldsymbol{y}$ for all its members. Indeed

> **Definition (dominated imputation)**: For $\boldsymbol{x}, \boldsymbol{y} \in E(v)$, we say $\boldsymbol{x}$ **dominates** $\boldsymbol{y}$ over $S$ (written $\boldsymbol{x} >_S \boldsymbol{y}$) if
>
> $$x_i > y_i \quad \forall\ i \in S \qquad\qquad \sum_{i \in S} x_i \leq v(S)$$

(The second condition requires that $S$ has enough payoff to ensure its members $\boldsymbol{x}$).

- It is reasonable to assume that only imputations in the **core** can persist in pre-game negotiations

> **Definition (The Core)**: The **core** of a game $v$, denoted by $C(v)$, is the set of imputations which are not dominated for any coalition.

The following theorem characterises the core

> **Theorem**: A vector $\boldsymbol{x}$ is in the core if and only if
>
> $$\sum_{i=1}^{n} x_i = v(N) \qquad \sum_{i \in S} x_i \geq v(S) \quad \forall\ S \subset N$$

***Proof***: Putting $S = \{i\}$ for each $\{i\} \in N$, we see that $\boldsymbol{x}$ is indeed an imputation. To show it is not dominated, suppose there is a coalition $S$ for which $y_i > x_i$ for all $i \in S$. We then have

$$\sum_{i \in S} y_i > \sum_{i \in S} x_i \geq v(S) \Rightarrow \sum_{i \in S} y_i > v(S)$$

But this violates the definition of a dominated imputation. So $\boldsymbol{x}$ must be un-dominated.

Conversely, suppose $\boldsymbol{x}$ is in the core. Since it is an imputation, the first condition must hold. Now, imagine the second condition does *not* hold, and define

$$\varepsilon_1 \left|S\right| + \textstyle\sum_{i \in S} x_i = v(S)$$
$$\varepsilon_2 \left|N \setminus S\right| + \textstyle\sum_{i \notin S} v(\{i\}) = v(N) - v(S)$$

Then consider the vector

$$y_i = \begin{cases} x_i + \varepsilon_1 & i \in S \\ v(\{i\}) + \varepsilon_2 & i \notin S \end{cases}$$

Now, it is clear that

- $\sum y_i = v(N)$, from the definition of $\varepsilon_1$ and $\varepsilon_2$
- $y_i \geq v(i)$, because $\boldsymbol{x}$ was itself an imputation, $\varepsilon_1$ is positive since we assume the second equation in the Theorem doesn't hold, and $\varepsilon_2$ is positive by superadditivity

Thus, $\boldsymbol{y}$ is an imputation. However, it is also true that

- $y_i > x_i$ for all $i \in S$
- $\sum_{i \in S} y_i = v(S)$, from the definition of $\varepsilon_1$.

Thus, $\boldsymbol{y}$ dominates $\boldsymbol{x}$, which contradicts $\boldsymbol{x}$ being in the core. Thus, the second equation in the Theorem must hold. ■

Consider the "oil game" as an example. We found that the set of imputations is

$$E(v) = \{(x_1, x_2, x_3) : x_1 + x_2 + x_3 = c, x_1 \geq a, x_2 \geq 0, x_3 \geq 0\}$$

We must apply the second requirement in the definition of the core to each possible coalition (to save space, I haven't bothered to include the coalitions that lead to redundant results):

| **Coalition $S$** | **$v(S)$** | **Requirement** $\sum_{i \in S} x_i \geq v(S)$ | |
|---|---|---|---|
| $\{1,2\}$ | $b$ | $x_1 + x_2 \geq b$ | (1) |
| $\{1,3\}$ | $c$ | $x_1 + x_3 \geq c$ | (2) |
| From the definition of $E(v)$ | | $x_1 + x_2 + x_3 = c$ | (3) |

We first note that (2) and (3) imply $x_2 = 0$, which gives $x_1 + x_3 = c$. (1) then gives $x_1 \geq b$, and so we're left with

$$C(v) = \{(\alpha, 0, c - \alpha) : b \leq \alpha \leq c\}$$

We can interpret this as follows: 1 and 3 form a coalition, and 1 sells oil to 3 at a price that is at least $b$ (or else 1 would be better off selling to 2) and no more than $c$ (otherwise, there's no reason 3 should buy it).

- The main problem with the core is that it often does not exist. The **nucleolus** is an alternative solution concept, that seeks to make the most unhappy coalition under it happier than the most unhappy coalition under any other imputation.

We define $x(S) = \sum_{i \in S} x_i$, which means that $v(S) - x(S)$ is a measure of how unhappy the coalition $S$ is with the imputation $x$. We then define $\boldsymbol{\theta}(\boldsymbol{x})$ to be the vector of $2^n$ values, arranged in decreasing order, of $v(S) - x(S)$ as $S$ varies across possible coalitions (including $\varnothing$ and $N$).

**Definition (The Nucleolus)**: The **nucleolus** of a game $v$, denoted by $N(v)$, is given by

$$N(v) = \left\{ \boldsymbol{x} \in E(v) : \boldsymbol{\theta}(\boldsymbol{x}) < \boldsymbol{\theta}(\boldsymbol{y}) \ \ \forall \boldsymbol{y} \in E(v) \right\}$$

Where by $\boldsymbol{\theta}(\boldsymbol{x}) < \boldsymbol{\theta}(\boldsymbol{y})$, we mean that either $\boldsymbol{\theta}(\boldsymbol{x})_1 < \boldsymbol{\theta}(\boldsymbol{y})_1$ or $\boldsymbol{\theta}(\boldsymbol{x})_k = \boldsymbol{\theta}(\boldsymbol{y})_k$ for $k = 1,2,\cdots,i-1$ and $\boldsymbol{\theta}(\boldsymbol{x})_i < \boldsymbol{\theta}(\boldsymbol{y})_i$.

It can be shown that the core exists, and is unique. Furthermore, the nucleolus lies within the core, provided it is non-empty, because:

- For any $x \in C(v)$, $v(S) - x(S) \leq 0$ for all $S$, so all the entries in $\boldsymbol{\theta}(\boldsymbol{x})$ are 0 or negative.
- This means that this will also be the case for the nucleolus.
- However, all imputations that give $\boldsymbol{\theta}(\boldsymbol{x})$ will all entries 0 or negative must be in the core. Thus, the nucleolus is in the core.

Consider the oil-market game again. In finding the nucleolus, we only need consider imputations in the core $C\left(v\right) = \left\{ \left(\alpha, 0, c-\alpha\right) : b \leq \alpha \leq c \right\}$. We need to compute $v(S) - x(S)$ for each possible coalition (to save space, I haven't bothered to include coalitions that lead to redundant results)

| **Coalition $S$** | $v(S)$ | $v(S) - x(S)$ |
|---|---|---|
| $\{1,2\}$ | $b$ | $b - \alpha$ |
| $\{3\}$ | 0 | $\alpha - c$ |

These two components are candidates for the largest nonzero unhappiness. We need the largest of the two to be as small as possible. This occurs when they are equal; $b - \alpha = \alpha - c \Rightarrow \alpha = \frac{1}{2}\left(c + b\right)$. So the nucleolus is $\left(\frac{1}{2}(c+b), 0, \frac{1}{2}(c-b)\right)$.

- We might also look at what each player could reasonably expect to get.

**Shapley's Axioms**: Player $i$'s expectation of what he could reasonably get from a game with characteristic function $v$, denoted $\phi_i(v)$, should satisfy

1. $\phi_i(v)$ should be independent of the way we label players. If $\pi$ is an operation representing a permutation of the labels, then $\phi_{\pi(i)}\left(\pi v\right) = \phi_i(v)$.
2. The sum of the expectations should equal the maximum available from the game

$$\sum_{i=1}^{n} \phi_i(v) = v(N)$$

3. If $u$ and $v$ are the characteristic functions of two games, $u + v$ is the characteristic function of playing both he game together, and $\phi$ should satisfy:

$$\phi_i\left(u+v\right)=\phi_i\left(u\right)+\phi_i\left(v\right)$$

The third of these axioms, especially, is decidedly odd. Nevertheless:

**Theorem (Shapley)**: The only function that satisfies Shapley's axioms is given by the **Shapley values**

$$\phi_i(v)=\sum_{S:i\in S}\frac{\left(\left|S\right|-1\right)!\left(n-\left|S\right|\right)!}{n!}\left(v\left(S\right)-v\left(S\setminus\{i\}\right)\right)$$

(The summation is over all coalitions that contain the player $i$).

This expression can be interpreted rather neatly by assuming that players arrive in the game randomly. We then have the following:

- $v\left(S\right)-v\left(S\setminus\{i\}\right)$ is the *extra amount* $i$ brings to the coalition $S$ when he arrives.
- $\left(\left|S\right|-1\right)!\left(n-\left|S\right|\right)!$ is the total number of ways $|S|$ – 1 players could have arrived before him, and $n$ – $|S|$ will be able to arrive after.
- $n!$ is the total number of ways players could actually arrive

Thus, we simply find the total amount player $i$ brings to every possible coalition he could join, weighed by the probability he will indeed join that coalition.

This is another solution concept. Note, however, that it results in an imputation that is not necessarily in the core.

## Market Games

- Consider a world with two commodities – A and B, say – and that there are $M$ A-traders and $N$ B-traders. We assume that a trader with $a$ units of A and $b$ units of B has utility $u_i\left(a,b\right)$, and we assume this function is concave, to factor in the fact that ever trader prefers some combination of the two commodities rather than either of the two extremes:

$$u_i\left(\lambda(a_1,b_1)+(1-\lambda)(a_2,b_2)\right)\geq \lambda u_i\left(a_1,b_1\right)+(1-\lambda)u_i\left(a_2,b_2\right)$$

For simplicity, we also assume every trader has the same $u$, so we drop the subscript $i$, and we assume that each trader starts with $a$ or $b$ units of A or B respectively.

Now, consider a coalition of $s_{\mathrm{A}}$ A-traders and $s_{\mathrm{B}}$ B-traders. The worse the other players could do is not trade with them. As such, the amount of A and B they have is constrained by

$$\sum_{i=1}^{s_{\mathrm{A}}+s_{\mathrm{B}}} x_i = s_{\mathrm{A}}a \qquad\qquad \sum_{i=1}^{s_{\mathrm{A}}+s_{\mathrm{B}}} y_i = s_{\mathrm{B}}b$$

They want to maximise their profits within that constraint, so

$$v\left(S\right)=\max_{x_1,\cdots,x_{s_{\mathrm{A}}+s_{\mathrm{B}}},y_1,\cdots,y_{s_{\mathrm{B}}+s_{\mathrm{B}}}}\sum_{i=1}^{s_{\mathrm{A}}+s_{\mathrm{B}}}u\left(x_i,y_i\right)$$

Furthermore, by concavity of $u(\cdot,\cdot)$, we have that

$$u\left(x_1,y_1\right)+u\left(x_2,y_2\right)\leq 2u\left(\frac{x_1+x_2}{2},\frac{y_1+y_2}{2}\right)$$

$$\Rightarrow \sum_{i=1}^{s_{\mathrm{A}}+s_{\mathrm{B}}}u\left(x_i,y_i\right)\leq\left(s_{\mathrm{A}}+s_{\mathrm{B}}\right)u\left(\frac{s_{\mathrm{A}}}{s_{\mathrm{A}}+s_{\mathrm{B}}}a,\frac{s_{\mathrm{B}}}{s_{\mathrm{A}}+s_{\mathrm{B}}}b\right)$$

This implies, however, that the maximum of the LHS is given by the RHS

$$v\left(S\right)=\left(s_{\mathrm{A}}+s_{\mathrm{B}}\right)u\left(\frac{s_{\mathrm{A}}}{s_{\mathrm{A}}+s_{\mathrm{B}}}a,\frac{s_{\mathrm{B}}}{s_{\mathrm{A}}+s_{\mathrm{B}}}b\right)$$

- Consider a $[1,\ N]$ market game, in which trader A is a monopolist. We suspect that he can charge as high a price as he wants, provided it's still worth the other's time to trade. We can prove this by showing that the imputation corresponding to the following description is in the core:
  - Each B-trader having a payoff of at least $u\left(0,b\right)$, which is what they had originally.
  - The A-trader having a payoff of $v\left(S\right)-Nu\left(0,b\right)$, which is the most that can be gained by the coalition, minus what the B-traders get.

  In fact:

**Theorem**: The imputation

$$\boldsymbol{x}^*=\left((N+1)u\left(\frac{a}{N+1},\frac{Nb}{N+1}\right)-Nu\left(0,b\right),u\left(0,b\right),\cdots,u\left(0,b\right)\right)$$

lies in the core.

***Proof***: When $S$ does not include 1, it is trivial that $\sum_{i\in S} x_i^* = v\left(S\right)$. When $S = \{1\} \cup \{K\}$, where $K$ is some $k$ of the B-traders, we need to show that

$$x_1^* + \sum_{k\in K} x_i^* \geq v\left(S\right)$$

$$\left(N+1\right)u\left(\frac{a}{N+1},\frac{Nb}{N+1}\right) - \left(N-k\right)u\left(0,b\right) \geq \left(k+1\right)u\left(\frac{a}{k+1},\frac{kb}{k+1}\right)$$

$$\left(N+1\right)u\left(\frac{a}{N+1},\frac{Nb}{N+1}\right) \geq \left(N-k\right)u\left(0,b\right) + \left(k+1\right)\left(\frac{a}{k+1},\frac{kb}{k+1}\right)$$

the last statement holds by the concavity of $u$, which states that the best way to maximise the utility of $N+1$ people is to ensure that they all get the same. Thus, $\boldsymbol{x}^*$ is indeed in the core. ■

- Thus far, we've looked at something akin to bartering – the two types of traders swap goods. Often, however, we need to consider competition among a few firms who are producing the same or closely related products. These are akin to $\left[M,\infty\right]$ games; the first type of trader is a producing firm, and the second is the consumers, who have money they wish to exchange for the product (when $M = 2$, we call the situation **duopoly**. Otherwise, we call it **oligopoly**).

  There are so many consumers that we no longer consider them as individuals, and represent their requirements by one utility function $u\left(p_1,p_2,\cdots,p_M,q_1,q_2,\cdots,q_M\right)$; the idea is that they are given the prices $p_i$ and choose quantities $q_i$. This reduces to a set of price-demand equations which connect the demand $q_i$ for firm $i$'s product with the announced prices

  $$q_i = f_i\left(p_1,\cdots,p_M\right)$$

  Firm $i$'s utility is given by its profit

  $$e_i\left(p_1,\cdots,p_M\right) = p_i q_i - c_i\left(q_i\right)$$

  Where $c_i$ is the production cost function for firm $i$.

  As an example, consider a duopoly with $c_1\left(q_1\right) = c_2\left(q_2\right) = 0$ in which the price-demand functions are

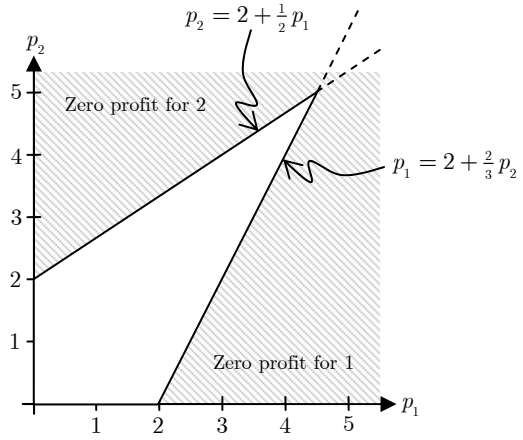
  $$q_1 = f_1\left(p_1,p_2\right) = \max\left\{1 + \tfrac{1}{3}p_2 - \tfrac{1}{2}p_1, 0\right\} \Rightarrow 0 \leq p_1 \leq 2 + \tfrac{2}{3}p_2$$
  $$q_2 = f_2\left(p_1,p_2\right) = \max\left\{1 + \tfrac{1}{4}p_1 - \tfrac{1}{2}p_2, 0\right\} \Rightarrow 0 \leq p_2 \leq 2 + \tfrac{1}{2}p_1$$

Where the constraints above are necessary to ensure that we can write

$$e_1\left(p_1,p_2\right)=p_1q_1-0=p_1+\tfrac{1}{3}p_1p_2-\tfrac{1}{2}p_1^2$$
$$e_2\left(p_1,p_2\right)=p_2q_2-0=p_2+\tfrac{1}{4}p_1p_2-\tfrac{1}{2}p_2^2$$

Once we have these equations, we simply need to solve $\mathrm{d}e_i/\mathrm{d}p_i=0$.

Thus far, we can illustrate our feasible region as follows:




- We define the following

> **Definition (Cournot Equilibrium)**: A **Cournot equilibrium** is a vector of prices $\boldsymbol{p}^c$ so that for every firm $i=1,\cdots,M$:
>
> $$e_i\left(p_1^c,\cdots,p_M^c\right)=\max_{p_i}e_i\left(p_1^c,\cdots,p_i,\cdots,p_M^c\right)$$

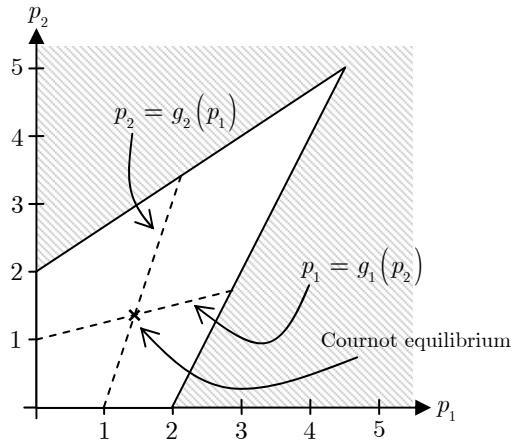
Effectively, $\boldsymbol{p}^c$ is an equilibrium $n$-tuple in an $n$-person non-coorperative game of price competition. Since there is an infinite number of pure strategies, we cannot appeal to Nash's Theorem, but it can be shown that under reasonable conditions, a Cournot equilibrium always exists.

Each of the $e$ must be maximised with every *other* variable held fixed. So effectively, we need $\mathrm{d}e_i/\mathrm{d}p_i=0$.

In our particular example, solving $\partial e_1/\partial p_1=\partial e_2/\partial p_2=0$ gives

$$p_1=g_1\left(p_2\right)=1+\tfrac{1}{2}p_2 \qquad\qquad p_2=g_2\left(p_1\right)=1+\tfrac{1}{4}p_1$$

The **intersection of those two curves** gives the **Cournot equilibrium**:



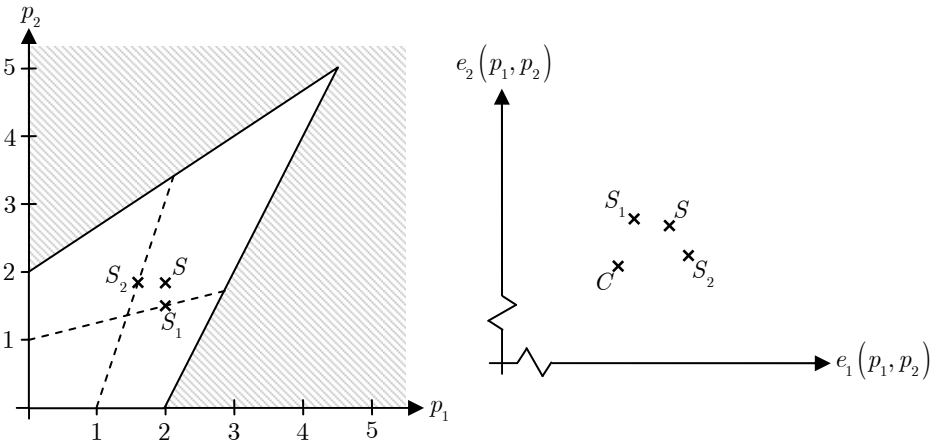

- However, we note that if firm 1 were to announce its price $p_1$ first, firm 2 would simply choose $p_2 = g_2(p_1)$. Realising this, firm 1 would simply choose a price $p_1$ to maximise $e_1\left(p_1, g_2\left(p_1\right)\right)$.

  **Stackleberg** introduced this kind of strategy, where firm 1 is the **leader** and firm 2 is the **follower**.

  There are three kinds of Stackleberg strategies

  - If 1 was the leader, it would choose $p_1$ to maximize
    $$e_1\left(p_1, g_2\left(p_1\right)\right) = p_1 + \tfrac{1}{2}p_1\left(1 + \tfrac{1}{4}p_1\right) - \tfrac{1}{2}p_1^2$$
    We call this solution $S_1$.
  - If 2 was the leader, we would obtain a similar expression and a solution $S_2$.
  - The last obtain is for *both* plays to play as if they were leaders. So 1 would play $p_1$ to maximize $e_1\left(p_1, g_2\left(p_1\right)\right)$ and 2 would play $p_2$ to maximize $e_2\left(g_1\left(p_2\right), p_2\right)$. The would then get a solution $S$

  It is informative to plot these strategies together with the associated profits (points *not* accurately plotted, to highlight differences):




It turns out that the leader does improve over its Cournot equilibrium using this technique, but not as much as the follower. If, however, both firms simply choose their "leader price", then they each achieve an "intermediate" profit.

- We can, instead, think of duopoly as a **cooperative game**. In that case, we might wonder what the negotiation set is. This can be done in two steps:
  - Find the maximin values:

    $$M_1 = \max_{p_1} \min_{p_2} e_1(p_1, p_2) \qquad M_2 = \max_{p_2} \min_{p_1} e_2(p_1, p_2)$$

    In this case, these are somewhat obvious.
  - Find all Pareto optimal prices; this is, price vectors $(\boldsymbol{p}_1^*, \boldsymbol{p}_2^*)$ such that there are no other $(\boldsymbol{p}_1, \boldsymbol{p}_2)$ such that

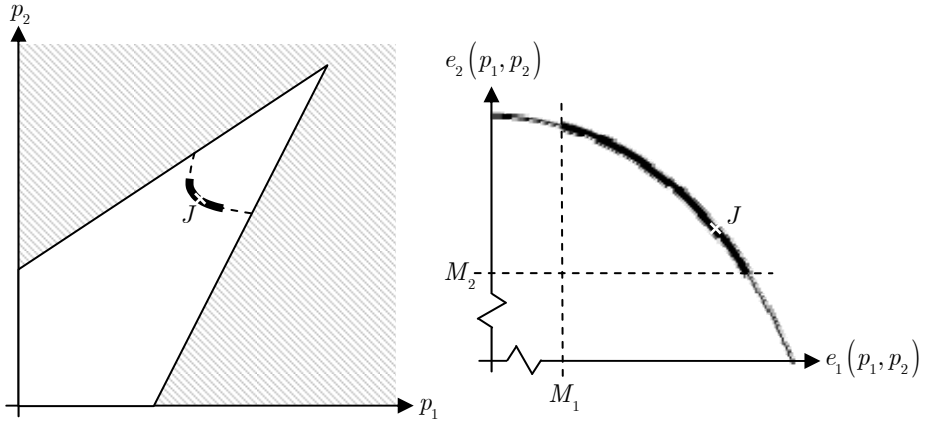
    $$e_i(p_1, p_2) \geq e_i(\boldsymbol{p}_1^*, \boldsymbol{p}_2^*) \qquad i = 1,2$$

    This can be done by solving the non-linear program

    $$\max\{e_1(p_1, p_2)\} \qquad \text{s.t. } e_2(p_1, p_2) \geq c \text{ and } (p_1, p_2) \text{ feasible}$$

    for every value of $c$.

The negotiation set consists of these bold lines:



One point of the negotiation set that if oft quoted in economic theory is the **joint maximisation of profit**, used as a benchmark to measure the possible collusion between the firms. The two firms choose their prices so as to maximise

$$e_1\left(p_1,p_2\right)+e_1\left(p_1,p_2\right)$$

This is, in fact, none other than the characteristic function $v\left(\{1,2\}\right)$.

## Evolutionary Games

- Suppose than some individual uses a (behavioural) strategy $\boldsymbol{x}$ from some set of possible strategies, and that on meeting an individual that uses $\boldsymbol{y}$, the payoff is $e\left(\boldsymbol{x},\boldsymbol{y}\right)$ to the first individual. No rational thought is involved in choose $\boldsymbol{x}$ – instead, the individual whose gene make it use strategy $\boldsymbol{x}$ will have offspring with the same gene. If payoffs from strategy $\boldsymbol{x}$ are high, then it will produce more offspring using strategy $\boldsymbol{x}$.
- Suppose that changes to a strategy arise through a mutation. We look for equilibrium point – strategies whose **fitness is greater than that of any mutant strategy that could arise**:

> **Definition (Evolutionary stable strategy)**: Let $X$ be the set of strategies. A strategy $\boldsymbol{x}^*\in X$ is an **evolutionary stable strategy** (**ESS**) if for every $\boldsymbol{y}\in X,\boldsymbol{y}\neq\boldsymbol{x}^*$
>
> $$e\left(\boldsymbol{x}^*,\overline{\boldsymbol{x}}\right)>e\left(\boldsymbol{y},\overline{\boldsymbol{x}}\right) \qquad (*)$$
>
> where $\overline{\boldsymbol{x}}=\left(1-\varepsilon\right)\boldsymbol{x}^*+\varepsilon\boldsymbol{y}$ for sufficiently small $\varepsilon>0$.

Let's work this definition into a more user-friendly form. First, expand (*)

$$\left(1-\varepsilon\right)e\left(\boldsymbol{x}^*,\boldsymbol{x}^*\right)+\varepsilon e\left(\boldsymbol{x}^*,\boldsymbol{y}\right)>\left(1-\varepsilon\right)e\left(\boldsymbol{y},\boldsymbol{x}^*\right)+\varepsilon e\left(\boldsymbol{y},\boldsymbol{y}\right)$$

For (*) to hold, we therefore certainly need $e\left(\boldsymbol{x}^*,\boldsymbol{x}^*\right)\geq e\left(\boldsymbol{y},\boldsymbol{x}^*\right)$. If the inequality is strict, then (*) will hold for sufficiently small $\varepsilon$. If not, we also need $e\left(\boldsymbol{x}^*,\boldsymbol{y}\right)>e\left(\boldsymbol{y},\boldsymbol{y}\right)$. This leads to an alternative definition:

> **Definition (Evolutionary stable strategy)**: A strategy $\boldsymbol{x}^*\in X$ is an ESS if for every $\boldsymbol{y}\in X,\boldsymbol{y}\neq\boldsymbol{x}^*$
>
> $$e\left(\boldsymbol{x}^*,\boldsymbol{x}^*\right)>e\left(\boldsymbol{y},\boldsymbol{x}^*\right)$$
>
> or

$$e\left(\boldsymbol{x}^*,\boldsymbol{x}^*\right)=e\left(\boldsymbol{y},\boldsymbol{x}^*\right) \quad \text{and} \quad e\left(\boldsymbol{x}^*,\boldsymbol{y}\right)>e\left(\boldsymbol{y},\boldsymbol{y}\right)$$

An ESS is like a two-person non-zero-sum game. If $\boldsymbol{x}^*$ is an ESS, then $\left(\boldsymbol{x}^*,\boldsymbol{x}^*\right)$ is an equilibrium pair for the game. But not every equilibrium pair is an ESS.

- Finding pure ESSs is simple.

In terms of finding mixed ESS, consider a $2 \times 2$ game with non-trivial payoff matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Suppose $\boldsymbol{x}=\left(x,1-x\right)$ is a mixed ESS strategy.

**Lemma**: For any mixed ESS $\boldsymbol{x}$

$$e\left(I_i,\boldsymbol{x}\right)=e\left(\boldsymbol{x},\boldsymbol{x}\right) \qquad \forall i \in \left\{i : x_i > 0\right\}$$

***Proof***: We have

$$e\left(\boldsymbol{x},\boldsymbol{x}\right)=\sum x_i e\left(I_i,\boldsymbol{x}\right)$$

But since $\boldsymbol{x}$ is an ESS, we also have

$$e\left(\boldsymbol{x},\boldsymbol{x}\right)\geq e\left(I_i,\boldsymbol{x}\right)$$

But since $\sum x_i = 1$, then for every $x_i > 0$ we must have

$$e\left(\boldsymbol{x},\boldsymbol{x}\right)=e\left(I_i,\boldsymbol{x}\right)$$

This proves the lemma. ■

*Remark*: this implies that there can be no pure ESS strategies, or else we would require $e\left(I_i,\boldsymbol{x}\right)>e\left(\boldsymbol{x},\boldsymbol{x}\right)$.

Thus, in the case above, we have $e\left(I_1,\boldsymbol{x}\right)=e\left(I_2,\boldsymbol{x}\right)$, and so we simply need to set

$$ax+b\left(1-x\right)=cx+d\left(1-x\right)$$

$$x=\frac{b-d}{b+c-a-d}$$

Since $\boldsymbol{x}$ involves all the pure strategies, we have that for this $\boldsymbol{x}$, $e\left(\boldsymbol{x},\boldsymbol{x}\right)=e\left(\boldsymbol{p},\boldsymbol{x}\right)$ for all $\boldsymbol{p}$. Thus, we need $e\left(\boldsymbol{x},\boldsymbol{q}\right)>e\left(\boldsymbol{q},\boldsymbol{q}\right)$ for all $\boldsymbol{q}$. To show this is the case, note that

$$e\left(\boldsymbol{p},\boldsymbol{q}\right)=d+\left(c-d\right)q+p\left[b-d+(a+d-b-c)q\right]$$

Consider:

- If $q > x$, the bit in square brackets is negative and $e$ increases as $p$ decreases. Thus, $e\left(\boldsymbol{x},\boldsymbol{q}\right) > e\left(\boldsymbol{q},\boldsymbol{q}\right)$ because $x < q$.
- If $q < x$, the exact opposite is true

Thus, all $2 \times 2$ games that are non-trivial (ie: $(a,b) \neq (a,c)$), there is at least one ESS strategy.

# Auctions

- Introduction
  - In a **private value model**, each bidder knows the value he places on the item, but doesn't know the value placed by other bidders. In a **common value model**, the item's actual value is the same to all bidders, but they have different *a priori* information about that value (eg: a jar of coins).
  - Auctions can be **oral** or **written** (bidders submit closed sealed bids in writing)
  - The **symmetric independent private values model (SPIV)** concerns the auction of a **single item** with
    - **Risk neutral** seller and bidders
    - Each bidder knowing his own valuation of the item, which he keeps secret.
    - The valuations modelled as IID random variables.
  - Types of auctions:
    - **English auction** – bids increase in small increments until one bidder remains. Equivalent to the **second price sealed-bid** (or **Vickrey**) **auction** – winner pays the second highest bid.
    - **Dutch auction** – price decreases continuously until some bidder calls stop. Equivalent to **first price sealed bid** – winner pays the bid
    - **All pay sealed-bid auction** – highest bidder wins, but all pay their bids.
- **The Revenue Equivalence Theorem**

## Regret Minimization

- This section is concerned with a situation in which we repeatedly have to make decisions in an uncertain environment. We would like to develop algorithms that allow us to play the game with the guarantee that against any opponent, we will perform nearly as well as the best fixed action in hindsight.
- Model
  - $N$ available actions $X = \{1, \cdots, N\}$
  - At each time step $t$, an algorithm $H$ selects a distribution
    $$\boldsymbol{p}^t = \{p_i^t : i = 1, \cdots, N\}$$
    over the $n$ actions and receives information
    $$\ell^t = \{\ell_i^t : i = 1, \cdots, N\}$$
    and experiences a loss
    $$\ell_H^t = \sum_{i=1}^{N} p_i^t \ell_i^t$$
    For the sake of simplicity, we will assume that $\ell_i^t \in \{0,1\}$
  - The **loss** of action $i$ by time $T$ is
    $$L_i^T = \sum_{i=1}^{T} \ell_i^T$$
    and the loss of the algorithm by that time is
    $$L_H^T = \sum_{t=1}^{T} \ell_H^t$$
  - Now, insight, the best fixed action will be the one that gives us the smallest $L$. In other words
    $$L_{\min}^T = \min_i L_i^T$$
    And defined the **external regret** of the algorithm $H$ by[4]
    $$R_T = L_H^T - L_{\min}^T$$
- We first consider a **greedy algorithm** which proceeds as follows

[4] More generally, the minimum in $L^T$ can be taken over a set that does not simply consist of all the individual actions; it could consist of various mixed strategies.

**Greedy Algorithm**: The greedy algorithm chooses the action which, thus far, has incurred minimum loss:

1. Originally, play $x^1 = 1$
2. At time $t$, play $x^t \in \underset{i}{\operatorname{argmin}} L_i^{t-1} = S^{t-1}$. If $S^{t-1}$ contains more than one item, choose the one with the lowest index

**Theorem**: The greedy algorithm always has loses which satisfy

$$L_{\text{greedy}}^T \leq NL_{\text{min}}^T + \Big(N-1\Big)$$

***Proof***: Consider the set $S$. At step $t$, the algorithm picks one member of $S$. A number of things can happen:

- The algorithm chooses a **winning action**. The action stays in the set, and $L_{\text{greedy}}^t - L_{\text{min}}^t$ doesn't increase.
- The algorithm chooses a **losing action from a choice of several**. The action leaves the set $S$, and $L_{\text{greedy}}^t - L_{\text{min}}^t$ increases by 1. This can occur a **maximum of $N-1$ times**, because the **maximum size of $S$ is $N$**, and the set needs to be populated by more than 1 item for there to have been a choice.
- The algorithm chooses a **losing action**, but it is the only one left in $S$. $L_{\text{min}}^t$ then increases, and $L_{\text{greedy}}^t - L_{\text{min}}^t$ doesn't increase.

Clearly, therefore, every time $L_{\text{min}}^t$ increases by 1, $L_{\text{greedy}}^t - L_{\text{min}}^t$ might have increased by as much as $N-1$ (or less, if the size of $S$ did not start at $N$). Thus

$$L_{\text{greedy}}^t - L_{\text{min}}^t \leq \Big(N-1\Big)\Big(L_{\text{min}}^t + 1\Big)$$

(we use $L_{\text{min}}^t + 1$ because the algorithm may be "in between updates" of $L_{\text{min}}^t$). The result follows. ■

This theorem only imposes a weak limit.

- We next consider a **randomized weighed majority algorithm** which proceeds as follows

**Randomized weighed majority algorithm**:

1. Originally, set $w_i^1 = 1, w^1 = N$ and play action $i$ with probability $p_i^t = w_i^t \,/\, w^t = 1 \,/\, N$.
2. At time $t$, let

$$w_i^t = \begin{cases} w_i^{t-1}\left(1-\eta\right) & \ell_i^{t-1} = 1 \\ w_i^{t-1} & \ell_i^t = 0 \end{cases} \qquad \eta \ll 1$$

and

$$w^t = \sum w_i^t$$

Then play $i$ with probability

$$p_i^t = w_i^t \,/\, w^t$$

Effectively, we are giving each action a weight $w_i^T = \left(1-\eta\right)^{L_i^T}$ and weighing probabilities proportional to the weights.

**Theorem**: For $\eta \le \frac{1}{2}$, the RWM algorithm has losses which satisfy

$$L_{\text{RMW}}^T \le \left(1+\eta\right) L_{\min}^T + \frac{\ln N}{\eta}$$

With $\eta = \min\left\{\sqrt{\frac{\ln N}{T}}, \frac{1}{2}\right\}$ yields

$$L_{\text{RMW}}^T \le L_{\min}^T + 2\sqrt{T \ln N}$$

***Proof***: Our strategy will first be to show that any time the algorithm has a large loss, $w$ must drop substantially. We then use the fact that

$$w^{T+1} \ge \max_i w_i^{T+1} = \max_i \left(1-\eta\right)^{L_i^T} = \left(1-\eta\right)^{L_{\min}^T}$$

To achieve our bound on $L_{\min}$

So, let's begin by noting that the expected loss of the RWM at time $t$ is

$$\ell_{\text{RMW}}^t = \sum_{i=1}^N \frac{w_i^t}{w^t} \ell_i^t = \sum_{\substack{i \text{ is losing} \\ \text{action}}} \frac{w_i^t}{w^t}$$

We also note that each losing action will have its weight multiplied by $\left(1-\eta\right)$. We find, therefore, that

$$\begin{aligned} w^{t+1} &= w^t - \sum_{\substack{i \text{ was losing} \\ \text{action}}} \eta w_i^t \\ &= w^t - \eta w^t \sum_{\substack{i \text{ was losing} \\ \text{action}}} \frac{w_i^t}{w^t} \\ &= w^t - \eta \ell_{\text{RMW}}^t w^t \\ &= \left(1-\eta \ell_{\text{RMW}}^t\right) w^t \end{aligned}$$

We have thus shown that when the algorithm incurs a large loss, $w$ drops significantly.

Now, since $w^1 = N$, we can write

$$w^{T+1} = N\prod_{t=1}^{T}\left(1-\eta\ell_{\text{RMW}}^t\right)$$

but we know that $\left(1-\eta\right)^{L_{\min}^T} \leq w^{T+1}$. And so

$$\left(1-\eta\right)^{L_{\min}^T} \leq N\prod_{t=1}^{T}\left(1-\eta\ell_{\text{RMW}}^t\right)$$

Taking logarithms

$$L_{\min}^T \ln\left(1-\eta\right) \leq \ln N + \sum_{t=1}^{T}\ln\left(1-\eta\ell_{\text{RMW}}^t\right)$$

We can now use the inequality $\ln\left(1-z\right) \leq -z$

$$\begin{aligned} L_{\min}^T \ln\left(1-\eta\right) &\leq \ln N - \sum_{t=1}^{T}\eta\ell_{\text{RMW}}^t \\ &\leq \ln N - \eta L_{\text{RMW}}^T \end{aligned}$$

As such

$$L_{\text{RMW}}^T \leq \frac{-\ln\left(1-\eta\right)}{\eta} L_{\min}^T + \frac{\ln N}{\eta}$$

We can now use the fact that $-\ln\left(1-z\right) \leq z + z^2$ for $z \in \left[0, \frac{1}{2}\right]$

$$L_{\text{RMW}}^T \leq \left(\eta+1\right) L_{\min}^T + \frac{\ln N}{\eta}$$

As expected.

Now, let $\eta = \min\left\{\sqrt{\frac{\ln N}{T}}, \frac{1}{2}\right\}$. Imagine $\eta = \sqrt{\left(\ln N\right)/T}$

$$L_{\text{RMW}}^T \leq L_{\min}^T + L_{\min}^T\sqrt{\frac{\ln N}{T}} + \sqrt{T\ln N}$$

But $L_{\min}^T \leq T$ (since the maximum an action can lose at each time step is 1), and so

$$L_{\text{RMW}}^T \leq L_{\min}^T + 2\sqrt{T \ln N}$$

We need to ensure that $\eta \in [0, \frac{1}{2}]$ for the inequality $-\log(1-z) \leq z^2 + z$ to hold; thus, if $\sqrt{\frac{\ln N}{T}} \geq \frac{1}{2}$, we simply set $\eta = \frac{1}{2}$. ■

As such, we find that for the right choice of $\eta$

$$\frac{R_{\text{RMW}}^T}{T} = \frac{L_{\text{RMW}}^T - L_{\min}^T}{T} \xrightarrow[T\to\infty]{} 0$$

It turns out that there is an analogous algorithm for losses $\ell_i^t \in [0,1]$.

- Consider, now, the relationship this has to game theory. Consider a two-person zero-sum game in which player I has strategies $1,\cdots,N_{\text{I}}$ and player II has strategies $1,\cdots,N_{\text{II}}$. We assume the game has a **loss matrix** for player I of $S_{ij}$ (this is equal to $-e_{ij}$ in our previous notation). We then have

$$S(\boldsymbol{p},\boldsymbol{q}) = \sum_{i=1}^{N_{\text{I}}}\sum_{i=1}^{N_{\text{II}}} p_i S_{ij} q_j = \boldsymbol{p}^T S \boldsymbol{q}$$

Now, let

$$v_{\text{I}} = \min_{\boldsymbol{p}} \max_{\boldsymbol{q}} S(\boldsymbol{p},\boldsymbol{q})$$
$$-v_{\text{II}} = \max_{\boldsymbol{q}} \min_{\boldsymbol{p}} S(\boldsymbol{p},\boldsymbol{q})$$

We know that $-v_{\text{II}} \leq v_{\text{I}}$ from before

- Now, imagine $\{\text{I},\text{II}\}$ plays algorithm $H_{\{\text{I,II}\}}$ which plays $\{\boldsymbol{p}^t,\boldsymbol{q}^t\}$ at time $t$. We clearly have that

$$\ell_{H_{\text{I}}}^t = S(\boldsymbol{p}^t,\boldsymbol{q}^t)$$
$$\ell_{H_{\text{II}}}^t = -S(\boldsymbol{p}^t,\boldsymbol{q}^t)$$

We show that

**Theorem**:

$$L_{H_{\text{I}}}^T \leq -v_{\text{II}} T + R_{\text{I}}^T$$
$$L_{H_{\text{II}}}^T \leq -v_{\text{I}} T + R_{\text{II}}^T$$

***Proof***: Let

$$q_j = \sum_{t=1}^{T} \frac{q_j^t}{T}$$

There exists an $i^*$ such that

$$S_{i^*\bullet}\boldsymbol{q} \le \boldsymbol{p}^T S_{ij}\boldsymbol{q} \qquad \forall \boldsymbol{p}$$

(This just means that we can choose the $i$ for which $S_{i^*\bullet}\boldsymbol{q}$ is smallest; clearly, the result will be smaller than a linear combination of this $S_{i^*\bullet}\boldsymbol{q}$ and other $S_{?\bullet}\boldsymbol{q}$).

We further note that

$$\frac{L_i^T}{T} = S_{i\bullet}\boldsymbol{q} \Rightarrow \frac{L_{i^*}^T}{T} \le \boldsymbol{p}S\boldsymbol{q} \qquad \forall \boldsymbol{p}$$

Now

$$L_{H_{\mathrm{I}}}^T \le L_{\min}^T + R_I \le L_{i^*}^T + R_{\mathrm{I}}$$

And so

$$\begin{aligned}\frac{L_{H_{\mathrm{I}}}^T}{T} &\le \frac{L_{i^*}^T}{T} + \frac{R_{\mathrm{I}}}{T} \\ &= \min_{\boldsymbol{p}} \boldsymbol{p}S\boldsymbol{q} + \frac{R_{\mathrm{I}}}{T} \\ &\le \max_{\boldsymbol{q}} \min_{\boldsymbol{p}} \boldsymbol{p}S\boldsymbol{q} + \frac{R_{\mathrm{I}}}{T} \\ &= -v_{\mathrm{II}} + \frac{R_I}{T}\end{aligned}$$

As required. ■

It remains to show that a corrolary of this theorem is the minimax theorem:

**Theorem (minimax)**: $-v_{\mathrm{II}} = v_{\mathrm{I}}$

***Proof***: Take $H_{\mathrm{I}}$ and $H_{\mathrm{II}}$ such that

$$\frac{R_{\mathrm{I}}^T}{T}, \frac{R_{\mathrm{II}}^T}{T} \xrightarrow[T\to\infty]{} 0$$

(for example, use RWM with $\eta = \sqrt{(\log N)/T}$). Since the game is a zero-sum game, we have

$$-\frac{L_{\mathrm{II}}^T}{T} = \frac{L_{\mathrm{I}}^T}{T}$$

And so

$$v_{\mathrm{I}} - \frac{R_{\mathrm{II}}^T}{T} \le -\frac{L_{\mathrm{II}}^T}{T} = \frac{L_{\mathrm{I}}^T}{T} \le -v_{\mathrm{II}} + \frac{R_{\mathrm{I}}^T}{T}$$

As $T \to \infty$, we obtain

$$v_{\mathrm{I}} \le -v_{\mathrm{II}}$$

But we know that $-v_{\text{II}} \leq v_{\text{I}}$. We conclude that

$$-v_{\text{II}} = v_{\text{I}}$$

which proves the minimax theorem. ■