OPTIMIZATION

Contents

1 Convexity

1.1 Generic optimization problem

Sometimes our aim is to optimize a system. But even if optimization is not the primary aim, the process of optimization can be a useful tool for testing a model's validity. We find an 'optimal solution' and then appraise it against common-sense, data, and sensitivity to assumptions – to refine and improve our model.

All problems in this course are of the following form.

$$\begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & g(x) = b \\ & x \in X \end{array}$$

where

$$\begin{array}{rcll} x & \in & \mathbb{R}^n & (n \text{ decision variables}) \\ f : \mathbb{R}^n & \to & \mathbb{R} & (\text{objective function}) \\ X & \subseteq & \mathbb{R}^n & (\text{regional constraints}) \\ g : \mathbb{R}^n & \to & \mathbb{R}^m & (m \text{ functional equations}) \\ b & \in & \mathbb{R}^m & \end{array}$$

Maximizing $f(x)$ is the same as minimizing $-f(x)$.

'Obvious' constraints like $x \geq 0$ are expressed by **regional constraints**, defining X appropriately; more complicated constraints, that may change differ between instances of the problem, are expressed by **functional constraints**. Sometimes the choice is for mathematical convenience. Solution methods typically treat regional and functional constraints differently.

The constraints define the **feasible set** for x, denoted

$$X(b) = \{x : g(x) = b, x \in X\}.$$

The problem is **feasible** if $X(b)$ is nonempty and **bounded** if $f(x)$ is bounded from below on $X(b)$. If x^* minimizes f over $X(b)$ then it is an **optimal solution**.

A constraint of the form $g(x) \leq b$ can be turned into an equality constraint by the addition of a **slack variable** z. Write

$$g(x) + z = b, \quad z \geq 0.$$

It is frequently mathematically convenient to turn all our constraints into equalities.

Things you already know: *Taylor's theorem, Cauchy-Schwarz inequality, the gradient, the Hessian, convexity, matrix algebra.*

1.2 Gradient descent

Consider a problem whose only constraint is $x \in \mathbb{R}^n$;

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\, f(x).$$

An intuitive idea is start at some initial x_0 and make a sequence of small steps, each "downhill". The steepest downhill direction from x_0 is $-\nabla f(x_0)$. This is because for a unit vector u such that $||u|| = 1$, and $t > 0$, we have the multi-dimensional Taylor's theorem that

$$f(x_0 + tu) = f(x_0) + t\,\nabla f(x_k)^\top u + o(t).$$

Recall the definition of gradient:

$$\nabla f(x)^\top = \left(\frac{df(x)}{dx_1}, \ldots, \frac{df(x)}{dx_n} \right).$$

By Cauchy-Schwarz we see that $\nabla f(x_k)^\top u$ is minimized by the unit vector $u = -\nabla f(x_k)/||\nabla f(x_k)||$.

Gradient descent algorithm

1. Start with an initial guess $x_0 \in \mathbb{R}^n$.
2. Pick a step size $t > 0$.
3. For every $k = 0, 1, \ldots$, let $x_{k+1} = x_k - t\,\nabla f(x_k)$.

When does this work? Consider $f(x) = x^2$. We find $x_{k+1} = x_k - t(2x_k) = (1 - 2t)x_k$ and so convergence to $x^* = 0$ requires $t < 1$. Note that the sequence $x_0, x_1, \ldots$ alternates either side 0 when $t \in (1/2, 1)$.

1.3 Convexity

Definition 1.1. *A set $S \subseteq \mathbb{R}^n$ is a* **convex set** *if for all $x, y \in S$ and $0 \leq \lambda \leq 1$ the point $\lambda x + (1 - \lambda)y \in S$.*
In other words, the line segment joining x and y lies entirely in S.

For functions defined on convex sets we make the following further definitions.

Definition 1.2. *$f : S \to \mathbb{R}$ is a* **convex function** *on convex set S if*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y), \quad \text{for all } 0 < \lambda < 1.$$

f is **strictly convex** *if the inequality is always strict;*

f is **strongly convex** *if there exists $\alpha > 0$ such that $f(x) - (\alpha/2)||x||^2$ is convex;*

f is a **concave function** *if $-f$ is convex.*

For example, $f(x) = x^2$ is strictly convex and $f(x) = \log(x)$ is strictly concave. Linear functions are both convex and concave.

Equivalently, f is convex if the set above its graph (called its **epigraph**) is convex, $\text{epi}(f) = \{(x,t) : f(x) \leq t\}$.

Convexity assumptions rule out behaviours that would prevent the steepest descent algorithm from working (see Theorem 1.3). We begin with an important characterization of convexity.

Theorem 1.1 (**Supporting hyperplane theorem**). *Let $f : S \to \mathbb{R}$, with S a convex set. Then f is convex if and only if for every $x \in S$ there exists $\lambda(x) \in \mathbb{R}^n$ such that for all y,*

$$f(y) \geq f(x) + \lambda(x)^\top (y - x). \tag{1}$$

If f is differentiable, then $\lambda(x) = \nabla f(x)$.

Proof. "$\Leftarrow$" Suppose $y, z, \in S$, $x = py + qz$, where $0 < p = 1 - q < 1$. If $\lambda(x)$ exists then

$$\begin{aligned} pf(y) + qf(z) &\geq p[f(x) + \lambda(x)^\top (y - x)] + q[f(x) + \lambda(x)^\top (z - x)] \\ &= f(x). \end{aligned}$$

"$\Rightarrow$" is true in general, but we prove only for f differentiable. Since f is convex:

$$\frac{f(x + p(y - x)) - f(x)}{p} \leq f(y) - f(x).$$

Let $p \searrow 0$ and simplify the left hand side with vector calculus. The vector $\lambda(x) = \nabla f(x)$ satisfies the desired inequality. □

The following corollary provides a **first-order condition** for a minimum.

Theorem 1.2. *If f is convex and differentiable, x^* is feasible, and $\nabla f(x^*) = 0$ then x^* is a global minimizer of f.*

Theorem 1.3. *Let $f : S \to \mathbb{R}$, with S a convex set. Then*
(a) f is convex $\implies$ every local minimum is a global minimum;
(b) f strictly convex $\implies$ the global minimum is unique;
(c) f is continuous and strongly convex, and $S \subseteq \mathbb{R}^n$ closed, $\implies$ there exists an optimal solution to the problem of minimizing f over S.

Proof. (a) Consider a local minimum x^*, let y be any other point in S, and $z = (1 - \lambda)x^* + \lambda y$. Since $f(x^*) \leq f(z)$ for all sufficiently small λ, we have $f(x^*) \leq f(z) \leq (1 - \lambda)f(x^*) + \lambda f(y)$, which implies $f(x^*) \leq f(y)$.

(b) Suppose x^*, y^* are both optimal solution. Since S is convex $z = \frac{1}{2}x^* + \frac{1}{2}y^*$ is feasible, so

$$f(z) = f\left(\tfrac{1}{2}x^* + \tfrac{1}{2}y^*\right) < \tfrac{1}{2}f(x^*) + \tfrac{1}{2}f(y^*)$$

which contradicts the optimality of x^* and y^*.

(c) As $f(x) - (\alpha/2)||x||^2$ is convex, the supporting hyperplane theorem gives

$$f(x) - (\alpha/2)||x||^2 \geq f(0) - (\alpha/2)||0||^2 + \lambda(0)^\top(x - 0).$$

Let $\lambda = \lambda(0)$. By Cauchy-Schwarz $\lambda^\top x \geq -||\lambda||\,||x||$, so if $||x|| > R = 2||\lambda||/\alpha$,

$$f(x) \geq f(0) - ||\lambda||\,||x|| + (\alpha/2)||x||^2 > f(0).$$

So we may restrict to minimizing over the region where $||x|| \leq R$. We know from analysis that a continuous function attains its minimum on a compact set. □

Strong-convexity puts a bound the sub-optimality of x, as follows.

Theorem 1.4 (**Lower bound on the gradient**). *Suppose $f : S \to \mathbb{R}$ is differentiable and strongly convex with constant $\alpha > 0$. Then for any $x, y \in S$*

$$||\nabla f(x)||^2 \geq 2\alpha\big(f(x) - f(y)\big).$$

Note that if $y = x^*$ is the minimizer of f then

$$f(x^*) \geq f(x) - \frac{1}{2\alpha}||\nabla f(x)||^2.$$

Proof. By the supporting hyperplane theorem applied to $f(x) - (\alpha/2)||x||^2$,

$$\begin{aligned}
&f(y) - (\alpha/2)||y||^2 \geq f(x) - (\alpha/2)||x||^2 + \left(\nabla f(x) - \alpha x\right)^\top (y - x) \\
\iff\ &f(y) - f(x) \geq \nabla f(x)^\top (y - x) + (\alpha/2)||y - x||^2 \qquad (2) \\
&\qquad = \frac{\alpha}{2}\left\|(y - x) + \frac{1}{\alpha}\nabla f(x)\right\|^2 - \frac{1}{2\alpha}||\nabla f(x)||^2 \\
&\qquad \geq -\frac{1}{2\alpha}||\nabla f(x)||^2,
\end{aligned}$$

minimizing the penultimate line at $y = x - \nabla f(x)/\alpha$. □

2 Gradient descent and Newton's method

2.1 Second-order conditions

Definition 2.1. *A $n \times n$ symmetric matrix is* **non-negative definite** *if $x^\top Ax \geq 0$ for every $x \in \mathbb{R}^n$.*

The **Hessian**, $\nabla^2 f(x)$, is the $n \times n$ matrix whose i, jth element is $d^2 f(x)/dx_i dx_j$.

Theorem 2.1. *If $\nabla^2 f(x)$ is non-negative definite for all x then f is convex.*

Proof. For any $x, y \in S$ we have by the multivariate version of Taylor's theorem

$$f(y) = f(x) + \nabla f(x)^\top (y - x) + \tfrac{1}{2}(y - x)^\top \nabla^2 f(\xi)(y - x)$$

where $\xi = px + (1 - p)y$ for some $0 < p < 1$. Hence, taking $\lambda(x) = \nabla(x)$,

$$f(y) \geq f(x) + \lambda(x)^\top (y - x)$$

for all y. By the supporting hyperplane theorem f is convex. □

The converse of this result also holds under mild assumptions.[1]

Recall that f is **strongly convex** if $f(x) - (\alpha/2)||x||^2$ is convex for some $\alpha > 0$. If f is differentiable then (as shown in proving Theorem 1.4) the supporting hyperplane theorem implies this is equivalent to

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\alpha}{2}||x - y||^2 \tag{3}$$

and if f is twice differentiable then it is equivalent to $\nabla^2 f(x) - \alpha I$ being non-negative definite (Theorem 2.1). We write this as $\alpha I \preceq \nabla^2 f(x)$.

Definition 2.2. *For symmetric matrices A and B we write $A \preceq B$ to mean that $B - A$ is non-negative definite.*

2.2 Convergence of gradient descent

Thinking again about steepest descent, what should t be and how fast does $f(x_k)$ approach the minimum? (In machine-learning t is called the **learning-rate**.) We assume the following smoothness condition.

[1] If f is convex and differentiable, then $f(y) \geq f(x) + \nabla f(x)^\top (y - x)$, and so from the Taylor expansion above, $\frac{1}{2}(y - x)^\top \nabla^2 f(\xi)(y - x) \geq 0$. Consider what this implies for $y = x + tx$ and $t \to 0$.

Definition 2.3. *f is said to be β-smooth if*

$$||\nabla f(x) - \nabla f(y)|| \leq \beta ||x - y||.$$

If f is twice differentiable β-smoothness is equivalent to $\nabla^2 f(x) \preceq \beta I$ for all x.[2]

Theorem 2.2. *If f is β-smooth then*

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{\beta}{2}||x - y||^2, \tag{4}$$

Compare (3) and (4).

Proof.

$$\begin{aligned} f(y) - f(x) - \nabla f(x)^\top (y - x) &= \int_0^1 [\nabla f(x + t(y - x)) - \nabla f(x)]^\top (y - x)\, dt \\ &\leq \int_0^1 ||\nabla f(x + t(y - x)) - \nabla f(x)|| \cdot ||y - x||\, dt \\ &\leq \beta ||x - y||^2 \int_0^1 t\, dt \\ &= \frac{\beta}{2}||y - x||^2. \end{aligned}$$

□

This suggests a good learning rate. The right hand side of (4) is quadratic in $(x - y)$ and minimized by $y = x - (1/\beta)\nabla f(x)$, so if we take $t = 1/\beta$ then

$$f\left(x_{k+1}\right) - f(x_k)) \leq -\frac{1}{2\beta}||\nabla f(x_k)||^2.$$

Now we can give conditions under which steepest descent converges nicely.

Theorem 2.3. *Suppose f is twice differentiable and there are positive constants α, β such that*

$$\alpha I \preceq \nabla^2 f(x) \preceq \beta I$$

for all x. Then applying the gradient descent algorithm with $t = 1/\beta$ we have

$$f(x_k) - f(x^*) \leq \left(1 - \frac{\alpha}{\beta}\right)^k (f(x_0) - f(x^*)).$$

[2]We omit the proof, but essentially: β-smoothness needs $||\nabla f(x + h) - \nabla f(x)|| \approx ||\nabla^2 f(x)^\top h|| \leq \beta ||h||$. This holds iff the greatest eigenvalue of $\nabla^2 f(x)$ is no more than β, which is iff $\nabla^2 f(x) \preceq \beta I$.

Proof. Let $x_{k+1} = x_k - (1/\beta)\nabla f(x_k)$. Using Taylor expansion and the lower bound on the gradient (Theorem 1.4), that $||\nabla f(x_k)||^2 \geq 2\alpha(f(x_k) - f(x^*)$,

$$\begin{aligned} f(x_{k+1}) - f(x_k) &= \nabla f(x_k)^\top (x_{k+1} - x_k) + \tfrac{1}{2}(x_{k+1} - x_k)^\top \nabla^2 f(\xi)(x_{k+1} - x_k) \quad \text{Taylor} \\ &\leq \nabla f(x_k)^\top (x_{k+1} - x_k) + \frac{\beta}{2}||x_{k+1} - x_k||^2 \quad \text{Beta Smooth} \\ &= -\frac{1}{2\beta}||\nabla f(x_k)||^2 \\ &\leq -\frac{\alpha}{\beta}(f(x_k) - f(x^*)). \end{aligned}$$

So $f(x_{k+1}) - f(x^*) \leq (1 - \frac{\alpha}{\beta})(f(x_k) - f(x^*))$ and the result follows by induction. □

Note that the theorem holds without the assumption of a second derivative. We need only that f is α-strongly convex and β-smooth. We reach the second line of the above proof by using (4) of Theorem 2.2 instead of using Taylor expansion.

The ratio β/α is called the **condition number**.

Example 2.1. Consider $f(x) = \frac{1}{2}(x_1^2 + 100x_2^2)$. The Hessian is

$$\nabla^2 f(x) = \begin{pmatrix} 1 & 0 \\ 0 & 100 \end{pmatrix}.$$

So $\alpha = 1$, $\beta = 100$. Condition number is 100. Gradient descent with $t = 1/\beta$ is

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{k+1} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_k - \tfrac{1}{100}\begin{pmatrix} x_1 \\ 100x_2 \end{pmatrix}_k = \begin{pmatrix} 0.99x_1 \\ 0 \end{pmatrix}_k$$

So $x_1(k)$ hardly changes and it will take many iterations to converge.

But if we make a simple rescaling, setting $y = 10x_2$, then the condition number of $f(x_1, y) = (1/2)(x_1^2 + y^2)$ is 1 and convergence needs just a single step.

2.3 Newton's method

Suppose f is twice differentiable. Taylor's theorem gives

$$f(x) \approx f(x_0) + (x - x_0)^\top \nabla f(x_0) + \tfrac{1}{2}(x - x_0)^\top \nabla^2 f(x_0)(x - x_0).$$

The right hand side is minimized by $x = x_0 - (\nabla^2 f(x_0))^{-1}\nabla f(x_0)$.

Newton's method

1. Start with an initial guess $x_0 \in \mathbb{R}^n$.
2. For every $k = 0, 1, \ldots$, let $x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1}\nabla f(x_k)$.

Definition 2.4. *Suppose A is $n \times n$. Let $||A||$ be the smallest constant, a, such that $||Az|| \leq a||z||$ for all $z \in \mathbb{R}^n$.* If A is non-negative definite then $||A||$ is the largest eigenvalue *of A.*

Theorem 2.4. *Suppose f is twice differentiable and there are constants $\alpha, L > 0$ such that*

$$\alpha I \preceq \nabla^2 f(x)$$
$$||\nabla^2 f(x) - \nabla^2 f(y)|| \leq L||x - y||$$

for all $x, y \in \mathbb{R}^n$. Then with Newton's method

$$f(x_k) - f(x^*) \leq \frac{2\alpha^3}{L^2} \left(\frac{L}{2\alpha^2} ||\nabla f(x_0)|| \right)^{2^{k+1}}.$$

The exponent of 2^k means the rate of convergence is much faster than with gradient descent. Note that we need to start at an x_0 such that $||\nabla f(x_0)|| < 2\alpha^2/L$. We might find such an x_0 by starting with a few steps of gradient descent.

Example 2.2. With $f(x) = \frac{1}{2}(x_1^2 + 100x_2^2)$ Newton's method converges to the minimum in just 1 step.

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_1 = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_0 - \begin{pmatrix} 1 & 0 \\ 0 & 100 \end{pmatrix}^{-1} \begin{pmatrix} x_1 \\ 100x_2 \end{pmatrix}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

■

Proof. *Non-examinable*. Let $\Delta_k = x_{k+1} - x_k$. Since $\nabla f(x_k) = \nabla^2 f(x_k)\Delta_k$,

$$\begin{aligned} \nabla f(x_{k+1}) &= \nabla f(x_{k+1}) - \nabla f(x_k) - \nabla^2 f(x_k)\Delta_k \\ &= \int_{t=0}^{1} \left[\nabla^2 f(x_k + t\Delta_k) - \nabla^2 f(x_k)\right] \Delta_k \, dt \end{aligned}$$

This implies that

$$||\nabla f(x_{k+1})|| \leq L||\Delta_k||^2 \int_{t=0}^{1} t \, dt = \tfrac{1}{2}L||x_{k+1} - x_k||^2 \leq \frac{L}{2\alpha^2}||\nabla f(x_k)||^2$$

where we have used the fact that $\alpha I \preceq \nabla^2 f(x)$ implies $||(\nabla^2 f(x))^{-1}|| \leq 1/\alpha$. Hence

$$||\nabla f(x_k)|| \leq \frac{2\alpha^2}{L} \left(\frac{L}{2\alpha^2} ||\nabla f(x_0)|| \right)^{2^k}.$$

The lower bound on the gradient (Theorem 1.4) gives

$$f(x_k) - f(x^*) \leq \frac{1}{2\alpha}||\nabla f(x_k)||^2 \leq \frac{2\alpha^3}{L^2} \left(\frac{L}{2\alpha^2} ||\nabla f(x_0)|| \right)^{2^k+1}. \qquad \square$$

2.4 *Neural networks*

Non-examinable Consider the following task of **machine learning**. We have a set of n data records (y_i, x_i) in $\mathbb{R} \times \mathbb{R}^d$, $i = 1, \ldots, n$. We wish to construct a function $f : \mathbb{R}^d \to \mathbb{R}$ which gives close to y_i when the input is $x_{i1}, \ldots, x_{id}$.

Consider an input $x[0] \in \mathbb{R}^{d[0]}$, $d[0] = d$. We construct a **neural network** as follows. First, compute $y[0] \in \mathbb{R}^{d[1]}$ as the vector whose ith component is $w_i[0]^\top x[0] - b_i[0]$, i.e. the inner product of $x[0]$ and a vector of **weights** $w_i[0] = (w_{i1}[0], \ldots, w_{id[0]}[0])$, minus a **bias** $b_i[0]$. Now set $x[1] \in \mathbb{R}^{d[1]}$ as the vector whose ith component is $\phi(y_i[0])$, where ϕ is a function to be specified below. Continuing thus gives

$$\begin{aligned} y_i[0] &= w_i[0]^\top x[0] - b_i[0], \quad x_i[1] = \phi(y_i[0]), \\ y_i[1] &= w_i[1]^\top x[1] - b_i[1], \quad x_i[2] = \phi(y_i[1]), \quad \text{etc.} \end{aligned}$$

With a single-layer of $d[1]$ hidden nodes, we would take $d[2] = 1$, $b_j[1] = 0$, using as our final output

$$y_1[1](x[0]) = \sum_{j=1}^{d[1]} w_j[1]\phi(w_j[0]^\top x[0] - b_j[0])$$

and seek to minimize over weights and biases,

$$\text{Cost} = \sum_{i=1}^{m} (y_i - y_1[1](x_i)))^2.$$

Cost is computed over a **training-set** of m ($< n$) data points. We check that the fitted model works well on the data not used in training. If $\phi(z)$ is linear this amounts to no more than linear regression. But in fact we take σ as a non-linear function, such as the **sigmoid** $\phi(z) = e^x/(1 + e^x)$, or $\phi(z) = \max\{z, 0\}$; this is suggested by the way neurons are thought to operate in the brain. In the above example there is only a single hidden layer, but we might have used 2 hidden layers to fit $y_i[2](x_i)$ to y_i, say with $d[0] = d$, $d[1] = d[2] = 10$, $d[3] = 1$. The method has been shown to work well with multiple hidden layers. The search for a minimum over weights and biases relies on the gradient descent algorithm. May researchers have worked with the MNIST database, which contains $n = 60,000$ images of hand-drawn digits, $0, 1, 2, \ldots, 9$, each of which is digitized to a grid of $28 \times 28 = 784$ ($= d$) pixels. The training set is $m = 10,000$. Error rates of 0.21% can be obtained.

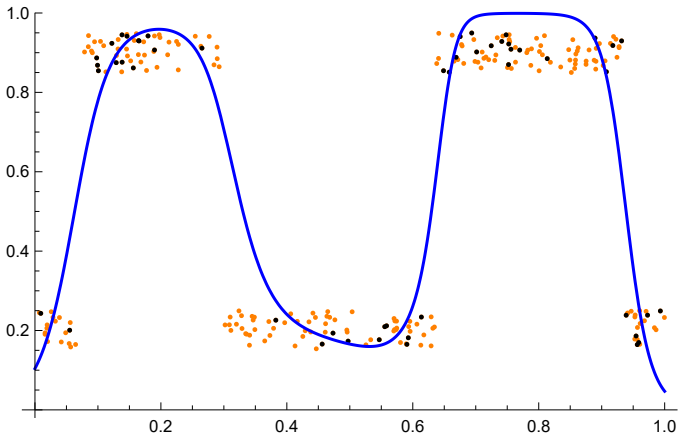


Figure 1: In this example $f(x)$ takes values close to 0.2 or 0.9 within certain intervals. A neural network with one hidden layer of five nodes was trained on 50 points. The blue curve is the resulting model. Observe that if our aim is to discriminate whether or not $f(x) > 0.55$ then the fitted model perfectly discriminates for almost all the points not used in training (those shown orange).

3 Lagrangian Methods

3.1 The Lagrangian sufficiency theorem

Consider again

$$\text{P}: \quad \text{minimize } f(x) \quad \text{s.t. } g(x) = b, \quad x \in X,$$

$X \subseteq \mathbb{R}^n$, $b \in \mathbb{R}^m$ (n variables and m constraints). The **Lagrangian** is defined as

$$L(x, \lambda) = f(x) + \lambda^\top (b - g(x)),$$

with $\lambda \in \mathbb{R}^m$ (one component for each constraint). Each component of λ is called a **Lagrange multiplier**. The following theorem is extremely useful in practice.

Theorem 3.1 (**Lagrangian sufficiency theorem**). *Let x^* be feasible. Suppose there exists $\lambda^* \in \mathbb{R}^m$ such that*

$$L(x^*, \lambda^*) \leq L(x, \lambda^*) \quad \textit{for all } x \in X,$$

then x^* is optimal for P.

Proof. For any feasible x and any λ we have

$$L(x, \lambda) = f(x) + \lambda^\top (b - g(x)) = f(x)$$

since $g(x) = b$. So if x^* is feasible

$$\begin{aligned} f(x^*) &= L(x^*, \lambda^*) \\ &\leq L(x, \lambda^*) \text{ for all } x \in X \text{ by assumption} \\ &= f(x) \text{ for all feasible } x \end{aligned}$$

□

Remarks.

1. There is no guarantee that we can find a λ^* satisfying the conditions of the theorem. However, there is a large class of problems for which λ^* do exist.

2. At first sight the theorem offers us a method for testing that a solution x^* is optimal for P without helping us to find x^* if we don't already know it. Certainly we will sometimes use the theorem this way. But as in the following example, we can find $\lambda*$ by adjusting λ until some $x^*(\lambda)$ is feasible.

3.2 Example: use of the Lagrangian sufficiency theorem

Example 3.1.

$$\begin{gathered}\text{minimize } x_1^2 + 3x_2^2 \\ \text{s.t. } 2x_1 + 3x_2 = b \\ x \in X = \mathbb{R}^2.\end{gathered}$$

Solution.

$$L(x, \lambda) = x_1^2 + 3x_2^2 + \lambda(b - 2x_1 - 3x_2).$$

By calculating $dL/dx_i = 0$, we find this is minimized with at $x(\lambda) = (\lambda, \lambda/2)$. Note that $\nabla^2 L(x, \lambda)|_{x=x(\lambda)}$ is positive definite, so indeed it is a minimum that has been found. This solution satisfies the constraint if

$$2x_1(\lambda) + 3x_2(\lambda) = b, \quad \text{requiring } 2\lambda + 3(\lambda/2) = b$$

which we can make happen by choosing $\lambda = \lambda^* = (2/7)b$. Application of the theorem shows that the optimal solution is $x^* = (2/7, 1/7)b$ and the optimal value is $\phi(b) = ((2/7)^2 + 3(1/7)^2)b^2 = (1/7)b^2$. Observe that $\phi'(b) = (2/7)b = \lambda^*$, an important point that we will discuss later. ■

3.3 Strategy to solve problems with the Lagrangian sufficiency theorem

A strategy to find x^*, λ^* satisfying the conditions of the theorem is as follows.

1. Consider the problem

   $$\text{minimize } L(x, \lambda) \text{ subject to } x \in X.$$

   The only constraints are $x \in X$ so this should be an easier problem to solve than P. Identify the set of λ for which a finite optimum will be achieved:

   $$\Lambda = \{\lambda : \min_{x \in X} L(x, \lambda) > -\infty\}.$$

2. For $\lambda \in \Lambda$, the minimum will be obtained at some $x(\lambda)$ (that depends on λ in general). Typically, $x(\lambda)$ will not be feasible for P.

3. Find $\lambda^* \in \Lambda$ such that $x^* = x(\lambda^*)$ is feasible. Then x^* is optimal for P by the theorem. (Think of λ as being a knob which you turn until $x(\lambda)$ is feasible.)

3.4 Example: further use of the Lagrangian sufficiency theorem

Example 3.2.

$$\text{minimize } \frac{1}{1+x_1}+\frac{1}{2+x_2} \quad \text{s.t. } x_1+x_2\le b, \quad x_1,x_2\ge 0.$$

Solution. To handle the inequality constraint write $x_1+x_2+x_3=b$ where $x_3\ge 0$ is a **slack variable**. We define $X=\{x : x\ge 0\}$ and the Lagrangian is

$$\begin{aligned} L(x,\lambda) &= \frac{1}{1+x_1}+\frac{1}{2+x_2}+\lambda(b-x_1-x_2-x_3) \\ &= \left(\frac{1}{1+x_1}-\lambda x_1\right)+\left(\frac{1}{2+x_2}-\lambda x_2\right)-\lambda x_3+\lambda b. \end{aligned}$$

Note that the term $-\lambda x_3$ has a finite minimum only if $\lambda\le 0$. If $\lambda=0$ the minimum would be as $x_1,x_2\to\infty$, so in fact we must have $\lambda<0$ and hence x_3 should be 0. Consider

$$\left(\frac{1}{1+x_1}-\lambda x_1\right) \text{ and } \left(\frac{1}{2+x_2}-\lambda x_2\right).$$

In the range $x\ge 0$ a function of the form $\left(\frac{1}{a+x}-\lambda x\right)$ has its minimum at $x=0$, if this function is increasing at 0, or at the stationary point of the function, occurring where $x>0$, if the function is decreasing at 0. So the minimum occurs at

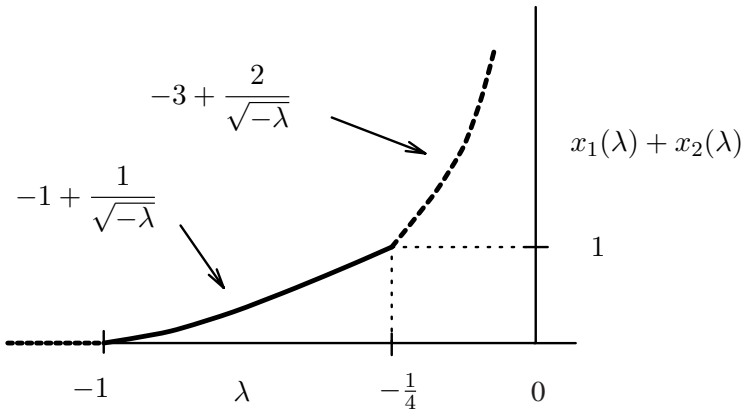
$$x(\lambda)=\left(-a+\sqrt{-1/\lambda}\right)^+.$$

where $c^+=\max(0,c)$.

Now notice that $x_1(\lambda)+x_2(\lambda)$ satisfies

$$\begin{aligned} x_1(\lambda)+x_2(\lambda) &= \left(-1+\sqrt{-1/\lambda}\right)^+ + \left(-2+\sqrt{-1/\lambda}\right)^+ \\ &= \begin{cases} 0 & \le -1 \\ -1+1/\sqrt{-\lambda} & \text{as } \lambda \in [-1,-1/4] \\ -3+2/\sqrt{-\lambda} & \in [-1/4,0] \end{cases} \end{aligned}$$

So we can see that $x_1(\lambda)+x_2(\lambda)$ is an increasing and continuous function (although it is not differentiable at $\lambda=-1$ and at $\lambda=-1/4$).



Thus (by the Intermediate Value Theorem) for any $b > 0$ there will be a unique value of λ, say λ^*, for which $x_1(\lambda^*) + x_2(\lambda^*) = b$. This λ^* and corresponding x^* will satisfy the conditions of the theorem and so x^* is optimal.

In fact, $x^* = (b, 0)$, $b \leq 1$, and $x^* = \frac{1}{2}(b+1, b-1)$, $b \geq 1$. ■

Examples of this kind are fairly common.

3.5 Inequality constraints and complementary slackness

In Example 3.2 we turned $g(x) \leq b$ into an equality constraint by introducing a slack variable x_3. In general we will be interested in problems such as

$$\text{P:} \quad \text{minimize } f(x) \quad \text{s.t. } g(x) \leq b, \quad x \in \mathbb{R}^n,$$

$b \in \mathbb{R}^m$. Rewrite the constraint as $g(x) + z = b$, where $z \in \mathbb{R}^m$ and $z \geq 0$. The Lagrangian is

$$L(x, z, \lambda) = f(x) + \lambda^\top (b - g(x) - z).$$

Minimizing freely over $z_j \geq 0$ we see that $-\lambda_j z_j$ is finite only if $\lambda_j \leq 0$. Thus

$$\Lambda = \left\{ \lambda : \lambda \leq 0, \inf_{x \in X} [f(x) + \lambda^\top (b - g(x)] > -\infty \right\}.$$

Moreover, if $\lambda_j < 0$ then $-\lambda_j z_j$ is minimized by $z_j^* = 0$. Only if $\lambda_j = 0$ could $z_j^* > 0$. Either way, we must have $\lambda^* z_j^* = 0$. This is called **complementary slackness** (of z_j^* and λ_j^*).

3.6 Lagrangian methods might not work

Example 3.3. Consider for $b > 0$,

$$\text{minimize } f(x) \quad \text{s.t.} \quad x = b, \quad x \geq 0,$$

We form the Lagrangian and minimize freely over all $x \geq 0$.

$$L(x, \lambda) = f(x) + \lambda(b - x).$$

If $f(x) = \sqrt{x}$ then there is no λ for which the minimum is at $x = b$, so the Lagrangian method does not work. But if $f(x) = x^2$ then $L(x, \lambda)$ is minimized at $x^* = b$ by taking $\lambda = 2b$; in this case the Lagrangian method works.

Let $b \in \mathbb{R}^m$ and define the **value function**

$$\phi(b) = \inf_{x \in X,\, g(x) = b} f(x)$$

Replace with inequality if there is an inequality constraint.

for the family of problems indexed by b. In the above examples $\phi(b) = \sqrt{b}$ and $\phi(b) = b^2$. It is the convexity of $\phi(b) = b^2$ that makes Lagrangian methods work.

3.7 *Large deviations*

Suppose we roll a die n times. The sum of the rolls is expected to be $3.5n$. Suppose it is $5n$. What is the most likely way this large deviation from a typical result could occur? This poses the problem

$$\underset{n_1,\ldots,n_6}{\text{maximize}} \frac{n!}{n_1! \cdots n_6!}(1/6)^n, \quad \text{s.t.} \sum_{i=1}^{6} i n_i = 5n, \quad \sum_{i=1}^{6} n_i = n.$$

By use of Stirling's approximation, setting $p_i = n_i/n$, the problem reduces to

$$\underset{p_1,\ldots,p_6}{\text{maximize}} \sum_{i=1}^{6} -p_i \log p_i \quad s.t \quad \sum_{i=1}^{6} i p_i = 5, \quad \sum_{i=1}^{6} p_i = 1.$$

The Lagrangian is

$$L(p, (\lambda, \mu)) = \sum_{i=1}^{6} -p_i \log p_i + \lambda \left(5 - \sum_{i=1}^{6} i p_i\right) + \mu \left(1 - \sum_{i=1}^{6} p_i\right)$$

and is minimized with $p_i = e^{-(1+\mu+\lambda i)}$. Numerical solution for λ and μ gives $p^* = (0.020, 0.039, 0.072, 0.136, 0.255, 0.478)$. As $n \to \infty$ the probability tends to 1 that the dice rolls occur in close to these proportions, rather than, say, as n rolls of 5.

3.8 Example: use of the Lagrangian sufficiency theorem

Here is an additional example, not done in lectures.

Example 3.4.

$$\begin{aligned} &\text{minimize } x_1 - x_2 - 2x_3 \\ &\text{s.t. } x_1 + x_2 + x_3 = 5 \\ &\quad x_1^2 + x_2^2 = 10 \\ &\quad x \in X = \mathbb{R}^3. \end{aligned}$$

Solution. Since we have two constraints we take $\lambda \in \mathbb{R}^2$ and write the Lagrangian

$$\begin{aligned} L(x,\lambda) &= f(x) + \lambda^\top (b - g(x)) \\ &= x_1 - x_2 - 2x_3 + \lambda_1(5 - x_1 - x_2 - x_3) + \lambda_2(10 - x_1^2 - x_2^2) \\ &= \Big[x_1(1-\lambda_1) - \lambda_2 x_1^2\Big] + \Big[x_2(-1-\lambda_1) - \lambda_2 x_2^2\Big] \\ &+ \Big[- x_3(2+\lambda_1)\Big] + 5\lambda_1 + 10\lambda_2. \end{aligned}$$

We first try to minimize $L(x,\lambda)$ for fixed λ in $x \in \mathbb{R}^3$. Notice that we can minimize each square bracket separately.

First notice that $-x_3(2+\lambda_1)$ has minimum $-\infty$ unless $\lambda_1 = -2$. So we only want to consider $\lambda_1 = -2$.

Observe that the terms in x_1, x_2 have a finite minimum only if $\lambda_2 < 0$, in which case the minimum occurs at a stationary point where,

$$\begin{aligned} \partial L/\partial x_1 = \quad 1 - \lambda_1 - 2\lambda_2 x_1 = 0 \quad &\Longrightarrow \; x_1 = 3/(2\lambda_2) \\ \partial L/\partial x_2 = -1 - \lambda_1 - 2\lambda_2 x_2 = 0 \quad &\Longrightarrow \; x_2 = 1/(2\lambda_2). \end{aligned}$$

Let Λ be the set of (λ_1, λ_2) such that $L(x,\lambda)$ has a finite minimum. So

$$\Lambda = \{\lambda : \lambda_1 = -2,\ \lambda_2 < 0\},$$

and for $\lambda \in \Lambda$ the minimum of $L(x,\lambda)$ occurs at $x(\lambda) = (3/(2\lambda_2), 1/(2\lambda_2), x_3)^\top$.

For a feasible $x(\lambda)$ we need

$$x_1^2 + x_2^2 = 10 \implies \frac{9}{4\lambda_2^2} + \frac{1}{4\lambda_2^2} = 10 \implies \lambda_2 = 1/2.$$

So $x_1 = 3$, $x_2 = 1$ and $x_3 = 5 - x_1 - x_2 = 1$.

The conditions of the Lagrangian sufficiency theorem are satisfied by

$$x^* = (3, 1, 1)^\top \text{ and } \lambda^* = (-2, -1/2)^\top .$$

So x^* is optimal. Note that $\nabla^2 L$ is positive definite so we have indeed minimized L. The minimum value is $f(x^*) = 0$ ■

4 The Lagrangian Dual

4.1 Lagrangian necessity

Recall that for $b \in \mathbb{R}^m$ ~~and~~ the **value function** is

$$\phi(b) = \inf_{x \in X,\, g(x)=b} f(x).$$

Replace with inequality if there is an inequality constraint.

The following theorem explains when Lagrangian methods are guaranteed to work.

Theorem 4.1 (**Lagrangian necessity**). *If the value function ϕ is convex and finite then there exists λ (depending on b) such that*

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

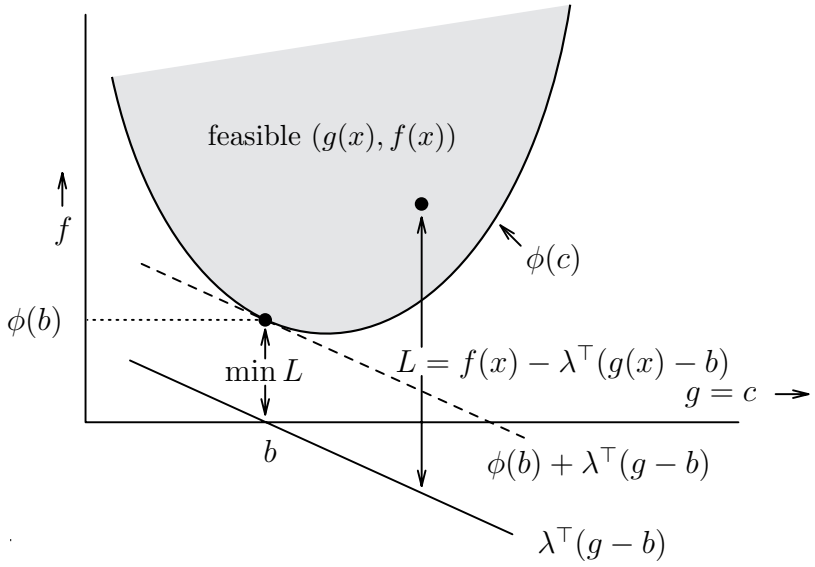
Furthermore, if ϕ is differentiable then $\lambda = \nabla\phi(b)$.

Proof. If ϕ is convex then by the supporting hyperplane theorem there exists λ such that $\phi(c) \geq \phi(b) + \lambda^\top(c - b)$ for all c. So

$$\begin{aligned}
\phi(b) &= \inf_c \big[\phi(c) + \lambda^\top(b - c)\big] \\
&= \inf_c \inf_{x \in X,\, g(x)=c} \big[f(x) + \lambda^\top(b - c)\big] \\
&= \inf_c \inf_{x \in X,\, g(x)=c} \big[f(x) + \lambda^\top(b - g(x))\big] \\
&= \inf_{x \in X} L(x, \lambda).
\end{aligned}$$

When ϕ is differentiable then λ is the gradient vector $\nabla\phi(b)$. □

The following diagram illustrates the idea.



Theorem 4.2 (**Sufficient conditions for convexity of the value function**). *Suppose*

1. *X is convex.*
2. *the objective function f is convex.*
3. *the functional constraint is of type $g(x) \leq b$.*
4. *g_i is convex for all $1 \leq j \leq m$.*

Then ϕ is convex.

The proof is on the examples sheet.

4.2 Shadow prices

Theorem 4.1 stated that if ϕ is differentiable then $\lambda = \nabla\phi(b)$. So $\phi(b+\epsilon) - \phi(b) \approx \lambda^\top \epsilon$. We have seen that if constraints are of the form $g(x) \leq b$, or equivalently $g(x) + z = b$, $z \geq 0$, then $\lambda \leq 0$. If b increases to $b + \epsilon$, $\epsilon > 0$ then the weaker constraint should mean the minimum can decrease. In fact, an increase of b_j to $b_j + \epsilon_j$ will permit the minimized value to change by $\lambda_j \epsilon_j$, which is a (weak) decrease since $\lambda_j \leq 0$ (no change if $\lambda_j = 0$). For this reason the Lagrange multipliers are also called **shadow prices**. An increase in b_j by ϵ_j alters the cost by $\lambda_j \epsilon_j$. So we should be willing to pay a price of $-\lambda_j$ (> 0) per unit increase in b_j.

4.3 The Lagrangian dual problem

We have defined the set

$$\Lambda = \{\lambda : \min_{x \in X} L(x, \lambda) > -\infty\}.$$

For $\lambda \in \Lambda$ define

$$L(\lambda) = \min_{x \in X} L(x, \lambda).$$

Let $X_b = \{x : x \in X, g(x) \leq b\}$.

Theorem 4.3 (**weak duality theorem**). *For any feasible $x \in X_b$ and any $\lambda \in \Lambda$*

$$f(x) \geq L(\lambda).$$

Proof. For $x \in X_b$, $\lambda \in \Lambda$,

$$f(x) = L(x, \lambda) \geq \min_{x \in X_b} L(x, \lambda) \geq \min_{x \in X} L(x, \lambda) = L(\lambda). \qquad \square$$

Thus, provided the set Λ is non-empty, we can pick any $\lambda \in \lambda$, and observe that $L(\lambda)$ is a lower bound for the minimum value of the objective function $f(x)$.

It is natural to seek the greatest lower bound, i.e., consider the problem

$$\text{D: maximize } L(\lambda) \text{ subject to } \lambda \in \Lambda,$$

equivalently,

$$\text{D: } \underset{\lambda \in \Lambda}{\text{maximize}} \left\{ \min_{x \in X} L(x, \lambda) \right\}.$$

This is known as the **Lagrangian dual problem**; the original problem is called the **primal problem**. The optimal value of the dual is no more than the optimal value of the primal. If they are equal we say there is **strong duality**. This happens in many problems, including linear programs.

Economic interpretation Agent A can produce n products in any non-negative amounts $x_1, \ldots, x_n$, and sell them in the market for $f(x_1, \ldots, x_n)$. Production of x requires $g_j(x)$ of resource j. If she has b_j and $g(x) - b_j > 0$ (of < 0) then she can purchase (or sell) the deficit (or surplus). Suppose market prices for these resources are $\lambda_1, \ldots, \lambda_m$ (buying or selling). Her profit is thus

$$f(x) + \lambda^\top (b - g(x)).$$

She maximizes this at $x(\lambda)$. In a competitive market prices will adjust to the point such that the maximum profit which the agent can extract is minimized. Thus the market solves the dual problem

$$\underset{\lambda}{\text{minimize}} \max_{x \geq 0} \left[f(x) + \lambda^\top (b - g(x)) \right].$$

Example 4.1. In Example 3.1, L was minimized at $x^*(\lambda) = (\lambda, \frac{1}{2}\lambda)$ and

$$\begin{aligned} L(\lambda) = L(x^*(\lambda), \lambda) &= \lambda^2 + 3\left((\tfrac{1}{2}\lambda)^2 + \lambda(b - 2\lambda - 3\tfrac{1}{2}\lambda \right) \\ &= \lambda b - \tfrac{7}{4}\lambda^2. \end{aligned}$$

This is maximized over λ at $\lambda^* = \frac{2}{7}b$. Note that $L(\lambda^*) = \frac{1}{7}b^2 = \phi(b)$. So indeed strong duality holds in this example. ■

Example 4.2. In Example 3.2, $Y = \{\lambda : \lambda \leq 0\}$. By substituting the optimal value of x into $L(x, \lambda)$ we obtain

$$L(\lambda) = \begin{cases} 3/2 + \lambda b & \leq -1 \\ 1/2 + 2\sqrt{-\lambda} + (b+1)\lambda & \text{as} \quad \lambda \in [-1, -1/4] \\ 4\sqrt{-\lambda} + (b+3)\lambda & \in [-1/4, 0] \end{cases}$$

We can solve the dual problem, which is to maximize $L(\lambda)$ s.t. $\lambda \leq 0$. The solution lies in $-1 \leq \lambda \leq -1/4$ if $0 \leq b \leq 1$ and in $-1/4 \leq \lambda \leq 0$ if $1 \leq b$. You can confirm that for all b the primal and dual here have the same optimal values. ■

4.4 Barrier methods

Consider

$$P: \quad \text{minimize } f(x) \quad \text{s.t. } g(x) \leq b, \quad x \in \mathbb{R}^n,$$

where f and g are convex and differentiable. Now consider the family of unconstrained minimization problems, $\epsilon > 0$,

$$P_\epsilon: \quad \text{minimize } f(x) - \epsilon \sum_{j=1}^{m} \log(b_j - g_j(x)).$$

Since $\log(x) \to -\infty$ as $x \to 0$ this ensures P_ϵ will be solved away from the boundary, with $g_j(x) < b_j$. The nice thing about having the minimum away from the boundary is that it will be at a stationary point.

Theorem 4.4. *Suppose x^* and x_ϵ are optimal for P and P_ϵ respectively. Then*

$$0 \leq f(x_\epsilon) - f(x^*) \leq m\epsilon.$$

Proof. The objective function for P_ϵ occurs away from the boundaries, at a stationary point, where

$$\nabla f(x_\epsilon) + \epsilon \sum_{j=1}^{m} \frac{\nabla g_j(x_\epsilon)}{b_j - g_j(x_\epsilon)} = 0.$$

Suppose we take λ such that

$$\bar{\lambda}_i = -\frac{\epsilon}{b_i - g_i(x_\epsilon)}.$$

Then the Lagrangian for P is stationary where

$$\nabla[f(x) + \bar{\lambda}^\top (b - g(x))] = \nabla f(x) - \bar{\lambda}^\top \nabla g(x) = 0.$$

This occurs at $x = x_\epsilon$. So by weak duality

$$\begin{aligned} f(x^*) &\geq L(\bar{\lambda}) \\ &\geq \inf_x \ f(x) + \bar{\lambda}^\top (b - g(x)) \\ &= f(x_\epsilon) + \bar{\lambda}^\top (b - g(x_\epsilon)) \\ &= f(x_\epsilon) - m\epsilon. \end{aligned}$$

□

A potential method is to start at x_0, ϵ_0, $k = 0$. Solve P_{ϵ_k} by gradient descent or Newton's method, starting from x_k, to obtain a better solution x_{k+1}; set $\epsilon_{k+1} = \frac{1}{2}\epsilon_k$. Repeat with $k :\to k + 1$.

Here is one further example of a dual problem (not done in lectures.)

Example 4.3. In Example 3.4 we had $Y = \{\lambda : \lambda_1 = -2, \lambda_2 < 0\}$ and that $\min_{x \in X} L(x, \lambda)$ occurred at $x(\lambda) = (3/(2\lambda_2), 1/(2\lambda_2), x_3)$. Thus

$$L(\lambda) = L(x(\lambda), \lambda) = \frac{5}{2\lambda_2} + 10 + 10\lambda_2 \,.$$

The dual problem is thus

$$\underset{\lambda_2 < 0}{\text{maximize}} \left\{ \frac{5}{2\lambda_2} + 10 + 10\lambda_2 \right\}.$$

The max is at $\lambda_2 = -1/2$, and the primal and dual have the same optimal value, namely 0. Again, strong duality holds. ■

5 Linear Programming

5.1 Extreme points and optimality

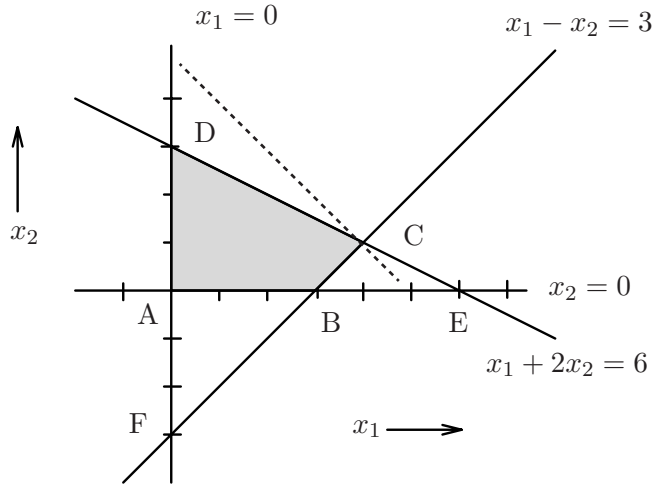
Consider the problem

$$\begin{array}{lll} \text{P: maximize} & x_1 + x_2 & \\ \text{subject to} & x_1 + 2x_2 & \leq 6 \\ & x_1 - x_2 & \leq 3 \\ & x_1, x_2 & \geq 0. \end{array}$$

This is instance of the **linear programming problem**:

$$\text{maximize } c^\top x \,:\, Ax \leq b,\ x \geq 0.$$

It is obvious that $c^\top x$ is minimized at a 'corner' of the feasible set, for any linear objective function. In the instance $c^\top = (1, 1)$ and the maximum is at corner C, where $x = (4, 1)$, $c^\top x = 5$.



If the objective function had been $x_1 + 2x_3$ then it would lie parallel to the edge from C to D, and all points on this edge would be optimal with $c^\top x = 5$, but there is always an optimum at a corner. This motivates the following definition.

Definition 5.1. *We say that x is an* **extreme point** *of a convex set S if whenever $x = \theta y + (1 - \theta)z$, for $y, z \in S$, $0 < \theta < 1$, then $x = y = z$.*

In other words, x is not in the interior of any line segment within S.

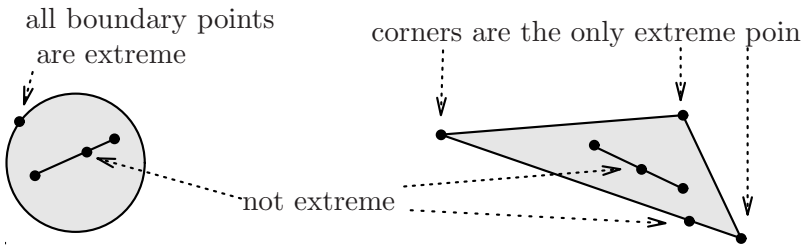


Figure 2: Examples of extreme points of two convex sets

Theorem 5.1 (**Fundamental Theorem of LP**). *If an LP is feasible and bounded then it has an optimum at an extreme point of the feasible set.*

In LP the feasible set will always have a finite number of extreme points (vertices). The feasible set is 'polyhedral', though it may be bounded or unbounded. This suggests the following algorithm for solving LPs.

1. Find all the vertices of the feasible set.
2. Pick the best one.

This will work, but is inefficient, becasue there may be very many vertices. $X(b) = \{x : Ax \leq b,\ x \geq 0\}$ can have $\binom{n+m}{m}$ vertices. So if $m = n$, say, then the number of vertices increases exponentially in n.

5.2 Basic solutions

We develop now a more algebraic (less geometric) characterisation of the extreme points. We write P with equality constraints, using slack variables.

$$\begin{array}{lrcl} \text{P: maximize} & x_1 + x_2 & & \\ \text{subject to} & x_1 + 2x_2 + z_1 & = & 6 \\ & x_1 - x_2 + z_2 & = & 3 \\ & x_1, x_2, z_1, z_2 & \geq & 0 \end{array}$$

Let us rewrite the constraint as $Ax = b$, by relabeling z_1 and z_2 as x_3 and x_4. So

$$\begin{pmatrix} 1 & 2 & 1 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 3 \end{pmatrix}.$$

We can calculate the value of the variables at each of the $\binom{4}{2} = 6$ points marked A–F in our picture of the feasible set for P. For example, to calculate E we solve

$$\begin{pmatrix} A_1 & A_4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_4 \end{pmatrix} = b, \quad \text{i.e.} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 3 \end{pmatrix}$$

where A_i denotes the ith column of A. The values are:

	x_1	x_2	z_1	z_2	f
A	0	0	6	3	0
B	3	0	3	0	3
C	4	1	0	0	5
D	0	3	0	6	3
E	6	0	0	−3	6
F	0	−3	12	0	−3

As expected, at each point two variables are non-zero and the other two are zero.

Geometrically: The 4 lines defining the feasible set can be written $x_1 = 0$; $x_2 = 0$; $z_1 = 0$; $z_2 = 0$. At the intersection of each pair, two variables are zero.

Algebraically: $Ax = b$ is composed of 2 equations in 4 unknowns. If we choose 2 variables (which can be done in $\binom{4}{2} = 6$ ways) and set them equal to zero we will be left with two equations in the other two variables. So (provided A and b are 'nice') there will be a unique solution for the two non-zero variables.

Definition 5.2.

- *The* **support** *of a vector x is the set of indices $S(x) = \{i : x_i \neq 0\}$.*
- *A* **basic solution** *to $Ax = b$ is a solution whose support is no more than m. That is, there exists $B \subset \{1, \ldots, n\}$ with $|B| = m$, such that $x_i = 0$ if $i \notin B$.*
- *Set B is called the* **basis***; x_i is called* **basic** *if $i \in B$ and* **non-basic** *if $i \notin B$.*
- *A basic solution x is* **non-degenerate** *if exactly $n - m$ variables are zero, i.e. $|S(x)|$ is exactly m.*
- *If a basic solution satisfies $x \geq 0$ then it is called a* **basic feasible solution**.

So A–F are basic solutions (and non-degenerate) and A–D are basic feasible solutions. From now on we make some assumptions that eliminate some (but not all) causes of degeneracy.

Assumption 5.1. The $m \times n$ matrix A, with $m \leq n$, has the property that

- The m rows of A are linearly independent, i.e. A has rank m.
- Any m columns of A are linearly independent.

Given a basis, B, we let A_B denote the $m \times m$ matrix whose columns are the m columns of A whose indices are in B. Assumption 5.1 ensures that A_B is invertible

and so $x_B = A_B^{-1}b$ is a basic solution. However, some components of x_B may be 0, giving degeneracy.

Some degeneracy cannot be removed. As an example, consider the simplex looking like a pyramid in $\mathbb{R}^3$, with square base and four sloping sides, defined by $x_3 - x_1 \leq 0, x_3 - x_2 \leq 0, x_1 + x_3 \leq 2, x_2 + x_3 \leq 2$, $x_1, x_2, x_3 \geq 0$. After adding 4 slack variables we have $Ax = b$ with A being 4×7 and satisfying Assumption 5.1. The vertex $(x, z) = (1, 1, 1, 0, 0, 0, 0)$ is at the intersection of 4 planes and the number of non-zero variables is 3, not 4. In theory, degeneracy can cause the simplex algorithm to cycle and so fail to converge, but in practice this is not seen to happen.

Theorem 5.2. *A vector is a basic feasible solution of $Ax = b$ if and only if it is an extreme point of the set $X(b) = \{x : Ax = b, x \geq 0\}$.*

Proof. Suppose x is not a b.f.s. So $|S(x)| > m$. Then there exists y such that $S(y) \subseteq S(x)$ and $Ay = 0$. But then x is the midpoint of $x + \epsilon y$ and $x - \epsilon y$, both of which lie in $X(b)$ for small enough non-zero ϵ. So x is not an extreme point.

Suppose x is not an extreme point, so that $x = \delta y + (1 - \delta)z$ for some distinct $y, z \in X(b)$, $0 < \delta < 1$. Then $Az = Ax = b = Az + \delta A(y - z)$. Now $A(y - z) = 0$ implies $|S(y - z)| > m$, but $S(y - z) \subseteq S(x)$. So $|S(x)| > m$ and x is not a b.f.s. □

Theorem 5.3. *If a linear program is feasible and bounded, then it has an optimal solution that is a basic feasible solution.*

Proof. Suppose x is an optimal solution, but not basic. Then there exists nonzero y s.t. $S(y) \subseteq S(x)$ and $Ay = 0$. Consider $x(\epsilon) = x + \epsilon y$. Suppose we are wishing to maximize $c^\top x$. Choose y such that $c^\top y \geq 0$. Clearly there exist some $\epsilon > 0$ such that $c^\top x(\epsilon) \geq c^\top x$, $x(\epsilon) \geq 0$, and $Ax(\epsilon) = Ax = b$, but $|S(x(\epsilon)| < |S(x)|$. □

Taking Theorems 5.2 and 5.3 together, we have proved Theorem 5.1. So we can do algebra instead of drawing a picture (which is good for a computer, and good for us. A simple (and foolish) algorithm is:

1. Find all the basic solutions.
2. Test to see which are feasible.
3. Choose the best basic feasible solution.

In Step 1, to find a basic solution we choose a basis set $B \subset \{1, \ldots, n\}$ with $|B| = m$. This can be done in $\binom{n}{m}$ ways. Let $N = \{1, \ldots, n\} \setminus B$. If $B = \{i_1, \ldots, i_m\}$ we let A_B be the $m \times m$ matrix

$$A_B = (A_{i_1} \cdots A_{i_m})$$

where A_j is the jth column of A. In our example above, the basic feasible solution at C is with $B = \{1, 2\}$, $N = \{3, 4\}$.

Similarly A_N is assembled from the columns of A indexed by N. The constraint

$$A_B x_B + A_N x_N = b \quad \Longrightarrow \; x_B = A_B^{-1} b - A_B^{-1} A_N x_N$$

Since we are taking $x_N = 0$ this is $x_B = A_B^{-1} b$ which is feasible if non-negative. Unfortunately it is not usually easy to know which basic solutions will turn out to be feasible before calculating them. Hence, even though there are often considerably fewer basic feasible solutions we will still need to calculate all $\binom{n}{m}$ basic solutions.

5.3 Preview of the simplex method

Simplex algorithm:

1. Start with a basic feasible solution.
2. Test — is it optimal?
3. If YES — stop.
4. If NO, move to 'adjacent' and better b.f.s. Return to 2.

The name comes from the fact that the convex hull of a finite set of extreme point is called a **simplex**.

After substituting $x_B = A_B^{-1} b - A_B^{-1} A_N x_N$ the objective function is written

$$\begin{aligned} c^\top x &= c_B^\top x_B + c_N^\top x_N \\ &= c_B^\top A_B^{-1} b + (c_N^\top - c_B^\top A_B^{-1} A_N) x_N \end{aligned}$$

This helps with Step 2. If

$$y^\top = (c_N^\top - c_B^\top A_B^{-1} A_N) \leq 0$$

then for any choice of $x_N \geq 0$ we have $c^\top x \leq c_B^\top A_B^{-1} b$ and hence $(x_B, x_N) = (A_B^{-1} b, 0)$ is optimal.

But if some component of y, is positive, then we can increase the value of the objective function by increasing the value of the corresponding component of x_N from its current value of 0. As we increase that component we must maintain $Ax = b$ and so other variables will need to change appropriately. Most importantly, $x_B = A_B^{-1} b - A_B^{-1} A_N x_N$ must stay non-negative. So we can increase that component until some component of x_B reduces to zero. At that point we have a new basic feasible solution, whose basis has had one deletion and one addition. The new basic feasible solution has a strictly greater value of $c^\top x$.

6 The Simplex Method

6.1 The simplex algorithm

Let us look closely at problem P and apply the simplex algorithm.

Simplex algorithm:

1. Start with a basic feasible solution.
2. Test — is it optimal?
3. If YES — stop.
4. If NO, move to 'adjacent' and better b.f.s. Return to 2.

We express information about a basic feasible solution in a table called the **simplex tableau**. At each step we update the tableau as we move to a better basic feasible solution. To illustrate, we start with the tableau for P in last lecture:

	x_1	x_2	z_1	z_2	a_{i0}
z_1	1	2	1	0	6
z_2	1	-1	0	1	3
a_{0j}	1	1	0	0	0

The initial basis is $B = \{3, 4\}$, corresponding to point A where $(x_1, x_2) = (0, 0)$.
The first two rows express $Ax = b$. (Think of z_1, z_2 as also being x_3, x_4.)
The third row contains $(c^\top, 0) = (c_N^\top, c_B^\top)$.
In the lower right corner we have $-c_B^\top A_B^{-1} b$, the negative of the current value of the objective function. For this B, $c_B^\top = (0, 0)$ and $c_N^\top = (1, 1)$.

In general the tableau contains this information:[3]

	m: x_B	$n-m$: x_N	1
m	$A_B^{-1}A_B = I$	$A_B^{-1}A_N$	$A_B^{-1}b$
1	$c_B^T - c_B^T A_B^{-1} A_B = 0$	$c_N^T - c_B^T A_B^{-1} A_N$	$-c_B^T A_B^{-1} b$

[3]The columns of the tableau have been permuted so that columns corresponding to the basis appear on the left. This has been done just for convenience: in practice we will always be able to identify the columns corresponding to the basis by the columns corresponding to the embedded identity matrix.

The first m rows contain $A = (A_B\ A_N)$ and the column vector b, multiplied by A_B^{-1}. Notice that for any basis B, a linear program with constraints $A_B^{-1}Ax = A_B^{-1}b$ is equivalent to one with constraints $Ax = b$. The first n columns of the last row are equal to $c^T - \lambda^T A$ for $\lambda^T = c_B^T A_B^{-1}$. We will see next lecture that λ can be interpreted as a solution, not necessarily feasible, to the dual problem. In the last column of the last row we finally have the value $-c^\top x$, where x is the b.f.s. with $x_B = A_B^{-1}b$ and $x_N = 0$.

All calculations of the algorithm will correspond to adding multiples of rows of the tableau to other rows or multiplying rows by constants. The tableau will always look like this:

x_1	x_i	x_n	
a_{ij}			a_{i0}
a_{0j}			a_{00}

Thinking of the vertical rule as "=", the rows in the table simply express equations

$$a_{i1}x_1 + \cdots + a_{in}x_n = a_{i0}$$

The last row can be read as

$$a_{01}x_1 + \cdots + a_{0n}x_n = a_{00} + f$$

where f denotes the value of the objective function $c^\top x$, but is not explicitly shown. You can imagine there is an invisible column at the far right for variable f consisting of $(0, \ldots, 0, 1)^\top$.

The algorithm is

1. **Choose a pivot column**. Choose a variable that is currently not in the basis which will now enter the basis. Suppose we are trying to maximize $c^\top x$. Examine the last row of the tableau and pick a j such that $a_{0j} > 0$. In other words, we find a component of $(c_N^\top - c_B^\top A_B^{-1} A_N)$ that is positive. This identifies the **pivot column**. (The variable corresponding to column j will enter the basis.) If all $a_{0j} \leq 0$, $j \geq 1$, then the current solution is optimal.

   Suppose we pick $j = 1$, so x_1 is entering the basis.

2. **Find the pivot row.** Suppose $a_{ij} > 0$. As x_j increases, a variable x_i which is in the basis, must decrease in order to maintain the equation in row i of

$a_{ij}x_j + x_i = a_{i0}$. As x_j increases to a_{i0}/a_{ij}, x_i decreases to 0. Choose i to minimize a_{i0}/a_{ij} from the set $\{i : a_{ij} > 0\}$ and hence discover which basic variable must leave the basis. This identifies i as the **pivot row**. If $a_{ij} \leq 0$ for all i then the problem is unbounded (see examples sheet) and the objective function can be increased without limit. If there is more than one i minimizing a_{i0}/a_{ij} the problem has a degenerate basic feasible solution (see example sheet.) For small problems you will be fine if you just choose any one of them and carry on regardless.

The **pivot** is at the intersection of the pivot row and pivot column.

In the example, row 2 is the pivot row since $3/1 < 6/1$, so the variable corresponding to equation 2 leaves the basis, i.e. z_2.

	x_1	x_2	z_1	z_2	a_{i0}
z_1	1	2	1	0	6
z_2	(1)	−1	0	1	3
a_{0j}	1	1	0	0	0

$\longrightarrow$

	x_1	x_2	z_1	z_2	a_{i0}
z_1	0	3	1	−1	3
x_1	1	−1	0	1	3
a_{0j}	0	2	0	−1	−3

3. **Perform a pivot at a_{ij}.** (i.e., rewrite the equations into the appropriate form for the new basic solution.)
   (a) multiply pivot row i by $1/a_{ij}$.
   (b) add $-(a_{kj}/a_{ij})\times$(pivot row i) to each row $k \neq i$, including the objective function row.

   In our example, the pivot row is the second row and the algebra is simple since $a_{21} = 1$. We just need to subtract row 2 from the first and the last rows, after which the tableau looks as at the right above.

   The second row now corresponds to variable x_1, which has replaced z_2 in the basis. This is the tableau for vertex B, where the basis is $B = \{1, 3\}$.

   We repeat these instructions on the new tableau, with pivot a_{12}, producing the tableau for vertex C. The new second row is the old second row plus 1/3 of the first row.

	x_1	x_2	z_1	z_2	a_{i0}
z_1	0	(3)	1	−1	3
x_1	1	−1	0	1	3
a_{0j}	0	2	0	−1	−3

$\longrightarrow$

	x_1	x_2	z_1	z_2	a_{i0}
x_2	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	1
x_1	1	0	$\frac{1}{3}$	$\frac{2}{3}$	4
a_{0j}	0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	−5

This is the tableau for vertex C, where the basis is $B = \{1, 2\}$, $x_1 = 4$, $x_2 = 1$, and $z_1 = z_2 = 0$, with an objective function value of 5.

Note that the columns of the tableau corresponding to the basic variables always comprise the columns of an identity matrix.

Since the bottom row is now all ≤ 0 the algorithm stops.
We have $f = 5 - \frac{2}{3}z_1 - \frac{1}{3}z_2$. Since $z_1, z_2 \geq 0$, the maximum is where $z_1 = z_2 = 0$.

6.2 Choice of initial basic feasible solution

To run the algorithm we need to pick an initial b.f.s. In the example above we started at vertex A where

$$x_1 = x_2 = 0;\ z_1 = 6,\ z_2 = 3.$$

Even for very large problems it is easy to pick a b.f.s. provided the original constraints are m constraints in n variables, $Ax \leq b$ with $b \geq 0$. Once we add slack variables $Ax + z = b$ we have $n + m$ variables and m constraints. If we pick $x = 0$, $z = b$ this is a b.f.s. But if our problem involved a constraints like $\sum_j a_{1j}x_j \geq b_1$, we cannot write $\sum_j a_{1j}x_j - z_1 = b_1 = 10$, say, and then take $z_1 = -10$, since we require $z_1 \geq 0$. The trick is to introduce another variable, writing $\sum_j a_{1j}x_j - z_1 + t_1 = 10$, start with a b.f.s. in which $t = 10$, and then run a preliminary stage of the algorithm that minimizes $\sum_i t_i$ to 0.

6.3 Choice of pivot column

We might have chosen the first pivot as a_{12} which would have resulted in

	x_1	x_2	z_1	z_2	a_{i0}
z_1	1	(2)	1	0	6
z_2	1	-1	0	1	3
a_{0j}	1	1	0	0	0

$\longrightarrow$

	x_1	x_2	z_1	z_2	a_{i0}
z_1	$\frac{1}{2}$	1	$\frac{1}{2}$	0	3
x_1	(3/2)	0	$\frac{1}{2}$	1	6
a_{0j}	$\frac{1}{2}$	0	$-\frac{1}{2}$	0	-3

This is the tableau for vertex D. A further iteration, with pivot a_{21} takes us to the optimal solution at vertex C. Therefore both choices of initial pivot column need two iterations of the algorithm to reach the optimum.

Remarks.

1. At each stage of the simplex algorithm we have two things in mind.

   (a) a particular choice of basis and basic solution.

   (b) a rewriting of the problem in a convenient form.

2. In general, there is no way to tell in advance which choice of pivot column will result in the smallest number of iterations. We may choose any column where $a_{0j} > 0$. A common rule-of-thumb is to choose the column for which a_{0j} is greatest, since the objective function increases by the greatest amount per unit increase in the variable corresponding to that column. However, it is known that for many rules, there exist examples on which the rule will perform badly.

   Suppose the simplex is a unit cube with 6 faces and 8 vertices. Starting from $(0, 0, 0)$ we might pivot 3 times to reach the optimum at $(1, 1, 1)$. But we also might take 7 steps, visiting all vertices on the way to the optimum.

   In practice, the simplex algorithm seems to most times take a number of steps that is linear in m and n. This has been proved for various probability models in which A and b are randomly chosen.

3. The tableau obviously contains some redundant information. For example, provided we keep track of which equation corresponds to a basic variable, we could omit the columns corresponding to the identity matrix (and zeros in the objective row). This is good for computer programs, but it is probably better to keep the whole thing for hand calculation.

7 The Dual Linear Program

7.1 The dual problem for LP

To find the dual to problem P:

$$\begin{gathered}\text{maximize } c^\top x \\ \text{subject to } Ax \le b,\ x \ge 0 \\ \text{equivalently } Ax + z = b,\ x, z \ge 0.\end{gathered}$$

we write the Lagrangian

$$L(x, z, \lambda) = c^\top x - \lambda^\top (Ax + z - b) = (c^\top - \lambda^\top A)x - \lambda^\top z + \lambda^\top b.$$

As in the general case, we consider the set Λ such that if $\lambda \in \Lambda$ then $\max_{x,z\ge 0} L(x, z, \lambda)$ is finite, and for $\lambda \in \Lambda$ we compute the minimum of $L(\lambda)$.

Consider the linear term $-\lambda^\top z$. If any coordinate $\lambda_i < 0$ we can make $-\lambda_i z_i$ as large as we like, by taking z_i large. So a finite maximum requires $\lambda_i \ge 0$ for all i.

Similarly, considering the term $(c^\top - \lambda^\top A)x$, this can be made as large as we like unless $(c^\top - \lambda^\top A)_i \le 0$ for all i. Thus

$$\Lambda = \{\lambda : \lambda \ge 0,\ \lambda^\top A - c^\top \ge 0\}.$$

If we pick a $\lambda \in \Lambda$ then $\max_{z\ge 0} -\lambda^\top z = 0$ (by choosing $z_i = 0$ if $\lambda_i > 0$ and any z_i if $\lambda_i = 0$) and also $\max_{x\ge 0}(c^\top - \lambda^\top A)x = 0$ similarly. Thus for $\lambda \in \Lambda$, $L(\lambda) = \lambda^\top b$. So a pair of primal P, and dual D is,

$$\begin{aligned}&\text{P:} \quad \text{maximize} \quad c^\top x \quad \text{s.t.} \quad Ax \le b, \qquad x \ge 0 \\ &\text{D:} \quad \text{minimize} \quad \lambda^\top b \quad \text{s.t.} \quad \lambda^\top A \ge c^\top,\ \ \lambda \ge 0.\end{aligned}$$

Notice that D is itself a linear program. For example,

$$\begin{aligned}&\text{P:} \quad \text{maximize} \quad x_1 + x_2 \\ &\qquad \text{subject to} \quad x_1 + 2x_2 \le 6 \\ &\qquad\qquad\qquad\quad x_1 - x_2 \le 3 \\ &\qquad\qquad\qquad\quad x_1, x_2 \ge 0\end{aligned} \qquad\qquad \begin{aligned}&\text{D:} \quad \text{minimize} \quad 6\lambda_1 + 3\lambda_2 \\ &\qquad \text{subject to} \quad \lambda_1 + \lambda_2 \ge 1 \\ &\qquad\qquad\qquad\quad 2\lambda_1 - \lambda_2 \ge 1 \\ &\qquad\qquad\qquad\quad \lambda_1, \lambda_2 \ge 0\end{aligned}$$

Furthermore, we might write D as

$$\text{D:} \quad \text{maximize} \quad (-b)^\top \lambda \quad \text{s.t.} \quad (-A)^\top \lambda \le (-c),\ \ \lambda \ge 0.$$

So D is of the same form as P, but with $c \to -b$, $b \to -c$, and $A \to -A^\top$. This shows that the dual of D is P, and so we have proved the following lemma.

Lemma 7.1. *In linear programming, the dual of the dual is the primal.*

7.2 The weak duality theorem in the case of LP

Theorem 4.3 applied directly to P and D gives the following.

Theorem 7.2 (**weak duality theorem for LP**). *If x is feasible for P (so $Ax \leq b$, $x \geq 0$) and λ is feasible for D (so $\lambda \geq 0$, $A^\top \lambda \geq c$) then $c^\top x \leq \lambda^\top b$.*

It is worth knowing a proof for this particular case which does not appeal to the general Theorem 4.3. Naturally enough, the proof is very similar.

Proof. Write

$$L(x, z, \lambda) = c^\top x - \lambda^\top (Ax + z - b)$$

where $Ax + z = b$, $z \geq 0$. Now for x and λ satisfying the conditions of the theorem,

$$c^\top x \;=\; L(x, z, \lambda) \;=\; (c^\top - \lambda^\top A)x - \lambda^\top z + \lambda^\top b \;\leq\; \lambda^\top b. \qquad \square$$

7.3 Sufficient conditions for optimality

Theorem 7.2 provides sufficient conditions for optimality of x^*, z^*, λ^* in P and D.

Theorem 7.3 (**sufficient conditions for optimality in LP**). If x^*, z^* is feasible for P and λ^* is feasible for D and $(c^\top - \lambda^{*\top} A)x^* = \lambda^{*\top} z^* = 0$ (complementary slackness) then x^* and λ^* are optimal for P and D. Furthermore $c^\top x^* = \lambda^{*\top} b$.

Proof. Write $L(x^*, z^*, \lambda^*) = c^\top x^* - \lambda^{*\top}(Ax^* + z^* - b)$. Now

$$\begin{aligned} c^\top x^* &= L(x^*, z^*, \lambda^*) \\ &= (c^\top - \lambda^{*\top} A)x^* - \lambda^{*\top} z^* + \lambda^{*\top} b \\ &= \lambda^{*\top} b \end{aligned}$$

But for all x feasible for P we have $c^\top x \leq \lambda^{*\top} b$ (by Theorem 7.2) and this implies that for all feasible x, $c^\top x \leq c^\top x^*$. So x^* is optimal for P. Similarly, λ^* is optimal for D (and the problems have the same optimums). $\square$

Theorem 7.4 (**strong duality in LP**). *If both P and D are feasible (each has at least one feasible solution), then there exists x, λ satisfying the conditions of Theorem 7.3 above.*

This is because both P and D can be solved by Lagrangian methods.

7.4 The utility of primal-dual theory

Why do we care about D instead of just getting on with the solution of P?

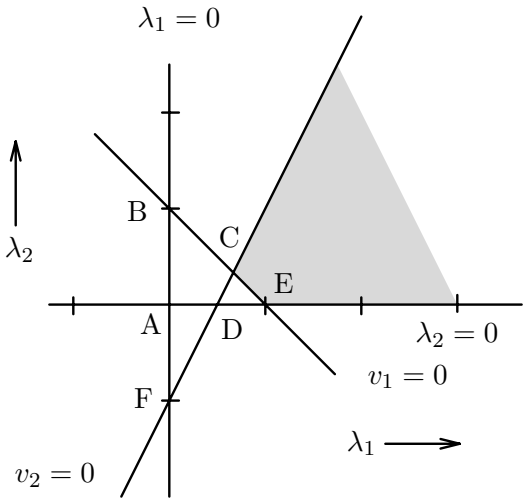
1. D might be easier to solve than P (and they have the same optimal values).
2. For some problems it is natural to consider both P and D together (e.g., two person zero-sum games, see Lecture 9).
3. Theorem 7.3 says that for optimality we need three things: primal feasibility, dual feasibility and complementary slackness.

   Some algorithms start with solutions that are primal feasible and work towards dual feasibility. Others start with dual feasibility. Yet others alternately look at the primal and dual variables and move towards feasibility for both at once.

7.5 Primal-dual relationships

Look at problem D which, after introducing slack variables v_1 and v_2, is written as

$$\begin{array}{lrcl} \text{D: minimize} & 6\lambda_1 + 3\lambda_2 & & \\ \text{subject to} & \lambda_1 + \lambda_2 - v_1 & = & 1 \\ & 2\lambda_1 - \lambda_2 - v_2 & = & 1 \\ & \lambda_1, \lambda_2, v_1, v_2 & \geq & 0 \end{array}$$



The value of the variables, etc., at the points A–F in P (as above) and D are:

P:

	x_1	x_2	z_1	z_2	f
A	0	0	6	3	0
B	3	0	3	0	3
C	4	1	0	0	5
D	0	3	0	6	3
E	6	0	0	−3	6
F	0	−3	12	0	−3

D:

	v_1	v_2	λ_1	λ_2	f
A	−1	−1	0	0	0
B	0	−2	0	1	3
C	0	0	$\frac{2}{3}$	$\frac{1}{3}$	5
D	$-\frac{1}{2}$	0	$\frac{1}{2}$	0	3
E	0	1	1	0	6
F	−2	0	0	−1	−3

Observe, that for D, as for P above, there are two zero and two non-zero variables at each intersection (basic solution). C and E are feasible for D. The optimum is at C with optimum value 5 (assuming we are minimizing and the other basic solutions are not feasible.)

We observe the following by comparing lists of basic solutions for P and D.

1. For each basic solution for P there is a corresponding basic solution for D. [Labels A–F have been chosen so that corresponding solutions have the same labels.] Each pair

   (a) has the same value of the objective function.

   (b) satisfies **complementary slackness**, i.e., $x_i v_i = 0$, $\lambda_i z_i = 0$,

   so for each corresponding pair,

P		D
variables x		constraints
x_i basic ($x_i \neq 0$)	$\iff$	constraint: tight ($v_i = 0$)
x_i non-basic ($x_i = 0$)	$\iff$	constraint: slack ($v_i \neq 0$)
constraints		variables λ
constraint: tight ($z_i = 0$)	$\iff$	λ_i basic ($\lambda_i \neq 0$)
constraint: slack ($z_i \neq 0$)	$\iff$	λ_i non-basic ($\lambda_i = 0$)

   (These conditions determine which basic solutions in P and D are paired; the implications go both ways because in this example all basic solutions are non-degenerate.)

2. There is only one pair that is feasible for both P and D, and that solution is C, which is optimal, with value 5, for both.

3. For any x feasible for P and λ feasible for D we have $c^\top x \leq b^\top \lambda$ with equality if and only if x, λ are optima for P and D.

4. P and D are related thus:

   - P has a finite optimum $\iff$ D has a finite optimum
   - P feasible $\implies$ D is bounded.
   - P is infeasible $\implies$ D is infeasible or unbounded.

$\max c^T x \leq \min b^T \lambda$ so if $c^T x$ bounded then $b^T \lambda$.

for any x, $c^T x \leq \min b^T \lambda$.

?

8 Shadow prices

8.1 Dual problem and the final tableau

Recall the final tableau in our example, of the form $Ax + z = b$,

x_1	x_2	z_1	z_2	a_{i0}
0	1	$\frac{1}{3}$	$-\frac{1}{3}$	1
1	0	$\frac{1}{3}$	$\frac{2}{3}$	4
0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	-5

Notice that in the bottom row of the tableau we have in the z_1 and z_2 columns $-\frac{2}{3}$ and $-\frac{1}{3}$, which are the values of $-\lambda_1$ and $-\lambda_2$ in the optimal solution to the dual. The explanation is that once we have been through the simplex algorithm iterations, we have effectively subtracted some multiples of the rows $1, \ldots, m$ from the last row. Suppose we have subtracted λ_i times row i. The final row looks like

$$(c_N^\top x_N \quad c_B^\top x_B \mid f) - \lambda^\top (A_N x_N \quad A_B x_B \mid b)$$

which expresses the equation

$$(c_N - \lambda^\top A_N)x_N + (c_B^\top - \lambda^\top A_B)x_B = f - \lambda^\top b$$

Since our initial solution was $x = 0, z = b$, $A_B = I$, $A_N = A$, $c_B^\top = 0$ this can be read as

$$(c^\top - \lambda^\top A)x - \lambda^\top z = f - \lambda^\top b.$$

We stop when coefficients of x and z are all non-positive. That is $\lambda \geq 0$ and $\lambda^\top A \geq c^\top$. These are precisely the conditions for λ to be feasible for the dual. It is optimal for the dual because $\lambda^\top b$ takes the same value as the maximum of $c^\top x$.

8.2 Shadow prices and sensitivity analysis

As we have seen, each row of each tableau merely consists of sums of multiples of rows of the original tableau. Objective last row = original last row + scalar multiples of other rows.

Consider the initial and final tableau for problem P.

initial

1	2	1	0	6
1	-1	0	1	3
1	1	0	0	0

final

0	1	$\frac{1}{3}$	$-\frac{1}{3}$	1
1	0	$\frac{1}{3}$	$\frac{2}{3}$	4
0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	-5

Look at the columns 3 and 4, corresponding to variables z_1 and z_2. Note that

Final row (1) = $\frac{1}{3}$ initial row (1) $-\frac{1}{3}$ initial row (2)
Final row (2) = $\frac{1}{3}$ initial row (1) $+\frac{2}{3}$ initial row (2)
Final objective row = initial objective row $-\frac{2}{3}$ initial row (1) $-\frac{1}{3}$ initial row (2).

Suppose we want to make a small change in b, so we replace $\begin{pmatrix} 6 \\ 3 \end{pmatrix}$ by $\begin{pmatrix} 6+\epsilon_1 \\ 3+\epsilon_2 \end{pmatrix}$. Solve for a solution with the same basis the tableau we find

0	1	$\frac{1}{3}$	$-\frac{1}{3}$	$1+\frac{1}{3}\epsilon_1-\frac{1}{3}\epsilon_2$
1	0	$\frac{1}{3}$	$\frac{2}{3}$	$4+\frac{1}{3}\epsilon_1+\frac{2}{3}\epsilon_2$
0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	$-5-\frac{2}{3}\epsilon_1-\frac{1}{3}\epsilon_2$

with corresponding solution $x_1 = 4+\frac{1}{3}\epsilon_1-\frac{1}{3}\epsilon_2$ and $x_2 = 1+\frac{1}{3}\epsilon_1+\frac{2}{3}\epsilon_2$ and objective function value $5+\frac{2}{3}\epsilon_1+\frac{1}{3}\epsilon_2$. If ϵ_1, ϵ_2 are such that we have $x_1 < 0$ or $x_2 < 0$ then vertex C is no longer optimal. But if ϵ_1, ϵ_2 are small then x_1, x_2 are still positive and the optimal solution still has the same basis.

The bottom row of the final tableau shows the **sensitivity** of the optimal solution to changes in b and how the optimum value varies with small changes in b. The values lieing at the bottoms of columns 3 and 4 are $-\lambda_1, -\lambda_2$. Again, Lagrange multipliers are **shadow prices**. We would be willing to a pay price of $\frac{2}{3}\epsilon_1$ for relaxation of the right hand side of the first constraint from 6 to $6+\epsilon_1$.

Notice also that being able to see how the final tableau is related to the initial one without looking at the intermediate steps provides a useful way of checking your arithmetic if you suspect you have got something wrong!

Notice that for problem P the objective rows at the vertices A, B, C and D are:

A	1	1	0	0	0
B	0	2	0	-1	-3
C	0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	-5
D	$\frac{1}{2}$	0	$-\frac{1}{2}$	0	-3

Compare these values with the basic solutions of the dual problem (on page 35). You will see that the objective row of the simplex tableau corresponding to each b.f.s. of problem P contains the values of the variables for a complementary slack basic solution to problem D (after a sign change).

The simplex algorithm can (and should) be viewed as searching amongst basic feasible solutions of P, for a complementary-slack basic solution of D which is also

feasible. At the optimum of P the shadow prices (which we can read off in the bottom row) are also the dual variables for the optimal solution of D.

8.3 Shadow prices and the diet problem

Lagrangian multipliers, dual variables and **shadow prices** are the same things. Let us say a bit more about the latter in the context of the **diet problem**.

Suppose you require an amounts $b_1, \ldots, b_m$ of m different vitamins. There are n foodstuffs available. Let

$$a_{ij} = \text{amounts of vitamin } i \text{ in one unit of foodstuff } j,$$

and suppose foodstuff j costs c_j per unit. Your problem is therefore to choose the amount x_j of foodstuff j you buy to solve the LP

$$\begin{gathered} \text{minimize} \sum_j c_j x_j \\ \text{subject to } \sum_j a_{ij} x_j \geq b_i, \text{ each } i \\ \text{and } x_j \geq 0 \text{ each } j. \end{gathered}$$

Now suppose that a vitamin company decides to market m different vitamin pills (one for each vitamin) and sell them at price p_i per unit for vitamin i. Assuming you are prepared to switch entirely to a diet of vitamin pills, but that you are not prepared to pay more for an artificial carrot (vitamin equivalent) than a real one, the company has to maximize profit by choosing prices p_i to

$$\begin{gathered} \max \sum_i b_i p_i \\ \text{subject to } \sum_i a_{ij} p_i \leq c_j, \text{ each } j \\ \text{and } p_i \geq 0 \text{ each } i. \end{gathered}$$

Note that this is the LP which is dual to your problem. The dual variable p_i is the price you are prepared to pay for a unit of vitamin i and is called a shadow price. By extension, dual variables are sometimes called shadow prices in problems where their interpretation as prices is very hard (or impossible) to see.

The dual variables tell us how the optimum value of our problem changes with changes in the right-hand side (b) of our functional constraints. This makes sense in the example given above. If you require an amount $b_i + \epsilon$ of vitamin i instead of an amount b_i you would expect the total cost of your foodstuff to change by an

amount ϵp_i, where p_i is the value to you of a unit of vitamin i, even though in your problem you cannot buy vitamin i separately from the others.

The above makes clear the relationship between dual variables/Langrange multipliers and shadow prices in the case of linear programming.

More generally, in linear problems we can use what we know about the optimal solutions to see how this works. Let us assume the primal problem

$$P(b): \quad \text{minimize } c^\top x \quad \text{s.t. } Ax - z = b,\ x, z \geq 0.$$

has the optimal solution $\phi(b)$, depending on b. Consider two close together values of b, say b' and b'', and suppose that optimal solutions have the same basic variables (so optimums occur with the same variables being non-zero, though the values of the variables will change slightly). The optimum still occurs at the same vertex of the feasible region though it moves slightly. Now consider the dual problem:

$$\text{maximize } \lambda^\top b \quad \text{s.t. } A^\top \lambda \leq c,\ \lambda \geq 0.$$

In the dual problem the feasible set does not depend on b, so the optimum of the dual will occur with the same basic variables and the same values of the dual variables λ. But the value of the optimum dual objective function is $\lambda^\top b'$ in one case and $\lambda^\top b''$ in the other and we have seen that the primal and dual have the same solutions. Hence

$$\phi(b') = \lambda^\top b' \text{ and } \phi(b'') = \lambda^\top b''$$

and the values of the dual variables λ give the rate of change of the objective value with b. The change is linear in this case.

We saw in Section 4.3 that the same idea works in nonlinear problems.

9 Two Person Zero-Sum Games

9.1 Games with a saddle-point

A **zero-sum** game is one which what one player wins what the other loses. The players make moves simultaneously. Each has a choice of moves (not necessarily the same). If player I makes move i and player II makes move j then player I wins (and player II loses) a_{ij}. Both players know the $m \times n$ **pay-off matrix** $A = (a_{ij})$.

		II plays j				
		1	2	3	4	
I plays i	1	-5	3	1	20	
	2	5	5	4	6	$\leftarrow (a_{ij})$
	3	-4	6	0	-5	

What is the best that player I can do if player II plays move j?

II's move: $j =$	1	2	3	4	
I's best response: $i =$	2	3	2	1	
I wins	5	6	**4**	20	$\leftarrow$ column maximums

Similarly, what is the best that player II can do if I plays move i?

I's move: $i =$	1	2	3	
II's best response: $j =$	1	3	4	
I wins	-5	**4**	-5	$\leftarrow$ row minimums

Here the minimal column maximum $= \min_j \max_i a_{ij} = \max_i \min_j a_{ij} =$ maximal row minimum $= 4$, when player I plays 2 and player II plays 3. In this case we say that A has a **saddle-point** $(2, 3)$ and the game is solved.

Definition 9.1. *A **saddle point** of a payoff matrix A is a pair of strategies (i^*, j^*) such that*

$$a_{i^*j^*} = \min_j \max_i a_{ij} = \max_i \min_j a_{ij}$$

Remarks. The game is solved by 'I plays 2' and 'II plays 3' in the sense that

1. Each player maximizes his minimum gain.

2. If either player announces any strategy (in advance) other than 'I plays 2' and 'II plays 3', he will do worse.

3. If either player announces that he will play the saddle-point move in advance, the other player cannot improve on the saddle-point.

9.2 Example: Two-finger Morra, a game without a saddle-point

Morra is a game dating from Roman and Greek times. Each player displays either one or two fingers and simultaneously guesses how many fingers his opponent will show. If both players guess correctly or both guess incorrectly then the game is a tie. If only one player guesses correctly, then that player wins from the other player an amount equal to the total number of fingers shown. A strategy for a player is (a, b) ='show a, guess b'. The pay-off matrix is

	(1,1)	(1,2)	(2,1)	(2,2)
(1,1)	0	2	−3	0
(1,2)	−2	0	0	3
(2,1)	3	0	0	−4
(2,2)	0	−3	4	0

$= (a_{ij})$

Column maximums are all positive and row minimums are all negative. So there is no saddle point (even though the game is symmetric and fair). If either player announces a fixed strategy (in advance), the other player will win.

We must look for a solution to the game in terms of mixed strategies.

9.3 Determination of an optimal strategy

Each player must use a **mixed strategy**. Player I plays move i with probability p_i, $i = 1, \ldots, m$ and player II plays moves j with probability q_j, $j = 1, \ldots, n$. Player I's expected payoff if player II plays move j is

$$\sum_i p_i a_{ij}.$$

So player I attempts to

$$\text{maximize } \left\{ \min_j \sum_i p_i a_{ij} \right\} \text{ s.t. } \sum_i p_i = 1,\ p_i \geq 0.$$

Note that this is equivalent to

$$\text{P: maximize } v \quad \text{s.t. } \sum_i a_{ij} p_i \geq v, \text{ each } j, \text{ and } \sum_i p_i = 1,\ p_i \geq 0,$$

since v on being maximized will increase until it equals the minimum of the $\sum_i a_{ij} p_i$. By similar arguments, player II's problem is

$$\text{D: minimize } v \quad \text{s.t. } \sum_j a_{ij} q_j \leq v, \text{ each } i, \text{ and } \sum_j q_j = 1,\ q_j \geq 0.$$

In fact, P and D are a pair of dual linear programs (as can be shown by the standard technique of finding the dual of P). Consequently, the general theory gives sufficient conditions for strategies p and q to be optimal.

Let e denote a vector of 1s, the number of components determined by context.

Theorem 9.1. *Suppose $p \in \mathbb{R}^m$, $q \in \mathbb{R}^n$, and $v \in \mathbb{R}$, such that*

(a) $p \geq 0$, $e^\top p = 1$, $p^\top A \geq v e^\top$ (primal feasibility);

(b) $q \geq 0$, $e^\top q = 1$, $Aq \leq ve$ (dual feasibility);

(c) $v = p^\top A q$ (complementary slackness).

Then p is optimal for P and q is optimal for D with common optimum (the **value of the game***) v.*

Proof. The fact that p and q are optimal solutions to linear programs P and D follows from Theorem 7.3. Alternatively, note that Player I can guarantee to get at least

$$\min_q p^\top A q \geq \min_q (v e^\top) q = v,$$

and Player II can guarantee that Player I gets no more than

$$\max_p p^\top A q \leq \max_p p^\top (ve) = v.$$

In fact, (c) is redundant; it is implied by (a) and (b). □

Remarks.

1. Notice that this gives the right answer for a game with a saddle-point (i^*, j^*), (i.e., $v = a_{i^*j^*}$, with $p_{i^*} = q_{j^*} = 1$ and other $p_i, q_j = 0$).

2. Two-finger Morra has an optimal solution $p = q = (0, \frac{3}{5}, \frac{2}{5}, 0)$, $v = 0$, as can be easily checked. E.g. $p^\top A = (0, 0, 0, 1/5) \geq 0 \times e^\top$. It is obvious that we expect to have $p = q$ and $v = 0$ since the game is symmetric between the players ($A = -A^\top$). A is called an **anti-symmetric matrix**.

   The optimal strategy is not unique. Another optimal solution is $p = q = (0, \frac{4}{7}, \frac{3}{7}, 0)$. Player I can play any mixed strategy of the form $(0, \theta, 1-\theta, 0)$ provided $\frac{4}{7} \leq \theta \leq \frac{3}{5}$.

3. These conditions allow us to check optimality. For small problems one can often use them to find the optimal strategies, but for larger problems it will be best to use some other technique to find the optimum (e.g., simplex algorithm). Note, however, that the problems P and D are not in a form where we can apply the simplex algorithm directly;v does not have a positivity constraint. Also the constraints are $\sum_i a_{ij} p_i - v = 0$ with r.h.s.$= 0$. It is possible, however, to transform the problem into a form amenable to the simplex algorithm.

   (a) Add a constant k to each a_{ij} so that $a_{ij} > 0$ each i, j. This doesn't change anything, except the value of the game which is now guaranteed to be positive ($v > 0$).

   (b) Change variables to $x_i = p_i/v$. We now have that P is

   $$\text{maximize } v \quad \text{s.t. } \sum_i a_{ij} x_i \geq 1, \ \sum_i x_i = 1/v, \ x_i \geq 0,$$

   which is equivalent to

   $$\text{minimize} \sum_i x_i \quad \text{s.t. } \sum_i a_{ij} x_i \geq 1, \ x_i \geq 0$$

   and this is the type of LP that we are used to.

9.4 Example: Colonel Blotto

Colonel Blotto has three regiments and his enemy has two regiments. Both commanders are to divide their regiments between two posts. At each post the commander with the greater number of regiments wins one for each conquered regiment, plus one for the post. If the commanders allocate equal numbers of regiments to a post there is a stand-off. This gives the pay-off matrix

		Enemy commander		
		(2,0)	(1,1)	(0,2)
Colonel Blotto	(3,0)	3	1	0
	(2,1)	1	2	−1
	(1,2)	−1	2	1
	(0,3)	0	1	3

Clearly it is optimal for Colonel Blotto to divide his regiments (i, j) and (j, i) with equal probability. So the game reduces to one with the payoff matrix

	(2,0)	(1,1)	(0,2)
(3,0) or (0,3)	$\frac{3}{2}$	1	$\frac{3}{2}$
(2,1) or (1,2)	0	2	0

To derive the optimal solution we can

(a) look at player Colonel Blotto's original problem: maximize $\{\min_j \sum_i p_i a_{ij}\}$, i.e., maximize$_p \min\{\frac{3}{2}p, p + 2(1 - p)\}$, easily solved by a graph.

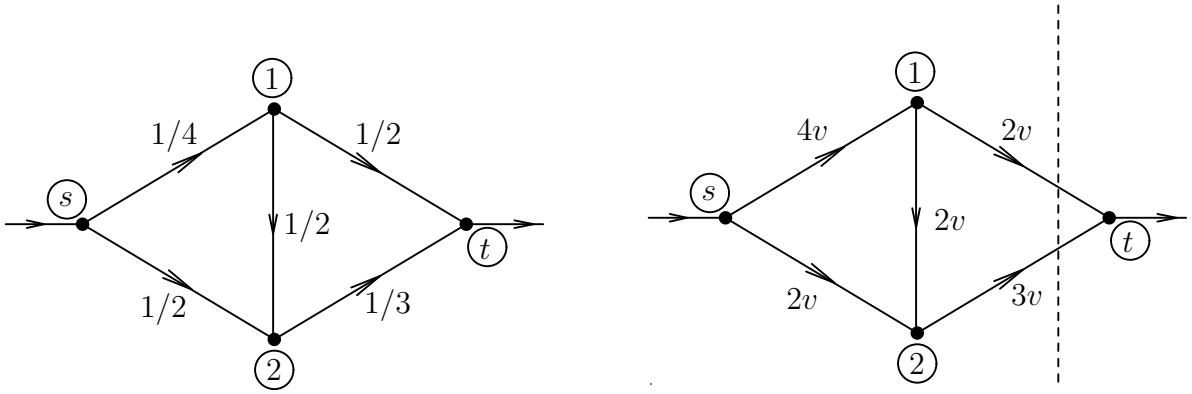
(b) attempt to derive p, q, v from the conditions of Theorem 9.1, or

(c) convert the problem as explained above and use the simplex method.

For this game, $p = (\frac{4}{5}, \frac{1}{5})$, $q = (\frac{1}{5}, \frac{3}{5}, \frac{1}{5})$ and $v = \frac{6}{5}$ is optimal. In the original problem, this means that Colonel Blotto should distribute his regiments as (3,0), (2,1), (1,2), (0,3) with probabilities $\frac{2}{5}, \frac{1}{10}, \frac{1}{10}, \frac{2}{5}$ respectively, and his enemy should distribute hers as (2,0), (1,1), (0,2) with probabilities $\frac{1}{5}, \frac{3}{5}, \frac{1}{5}$ respectively.

9.5 Example: Enforcer-evader game

A smuggler wishes to traverse a path in a network from s to t. The enforcer can set up an inspection point on any one arc. If he inspects on arc i and the smuggler's path uses that arc, then the smuggler is caught with probability α_i. The enforcer/smuggler wish to maximize/minimize the probability of capture. Both will need to use randomized strategies. Suppose the optimal strategy of the smuggler ends up using arc i with probability x_i. He will be seeking to minimize v subject to $\alpha_i x_i \leq v$ for all i, equivalently $x_i \leq v/\alpha_i$. Additionally the x_is must be consistent with flows along paths; i.e. the total flow of probability into and out of each node must be equal (except at s and t). This recasts the problem as a network flow problem in which arcs have capacities v/α_i and the aim is to reduce v to the point where passing 1 unit of flow through the nextwork from s to t is just feasible. The maximum flow can be found using the Ford-Fulkerson algorithm (Lecture 10). Consider the example shown. The min-cut is as shown by the dotted line.



The maximum flow is $5v$, achieved by $2v$, v and $2v$ along paths $s-1-t$, $s-1-2-t$ and $s-2-t$ respectively. Hence the solution is $v = 1/5$. The smuggler choses paths $s-1-t$, $s-1-2-t$, $s-2-t$ with probabilities 2/5, 1/5, 2/5, respectively, and the enforcer inspects arcs $1-t$, $2-t$ with probabilities 3/5, 2/5 respectively. The smuggler could alternatively choose paths $s-1-t$, $s-1-2-t$, $s-2-t$ with probabilities 2/5, 2/5, 1/5. This also puts flow of 2/5 on $1-t$ and 3/5 on $2-t$.

9.6 Example: Hider-searcher game

Player I, the hider, wishes to hide an object in one of n locations so that Player II, the searcher, has the greatest expected cost to find it. To search location i costs c_i. If the searcher searches locations in order $1, 2, \ldots, n$ and the object has been hidden in j then his cost is $c_1 + \cdots + c_j$. In the case $n = 3$ this leads to a game with matrix $A^\top$

	1	2	3
123	c_1	$c_1 + c_2$	$c_1 + c_2 + c_3$
132	c_1	$c_1 + c_2 + c_3$	$c_1 + c_3$
213	$c_1 + c_2$	c_2	$c_1 + c_2 + c_3$
231	$c_1 + c_2 + c_3$	c_2	$c_2 + c_3$
312	$c_1 + c_3$	$c_1 + c_2 + c_3$	c_3
321	$c_1 + c_2 + c_3$	$c_2 + c_3$	c_3

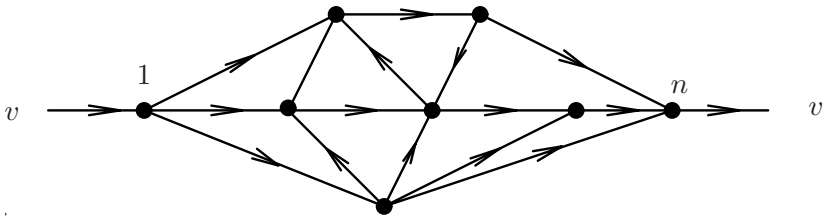
The optimal strategies are that II hides the object in location i with probability proportional to c_i and I searches in order i, j, k with probability $c_i/2$. The value of the game is $v = (c_1 + c_2 + c_3)^2 - (c_1c_2 + c_2c_3 + c_3c_1)$. These facts can be verified by showing the $(c_1, c_2, c_3)A = v1^\top$, and $(1/2)(c_1, c_1, c_2, c_2, c_3, c_3)A^\top = v1^\top$.

In general, if I is hiding k objects he should hide them in locations $i_1, \ldots, i_k$ with a probability proportional to $c_{i_1} \cdots c_{i_k}$. Player II should, with a probability proportional to c_i, search first in i and thereafter search other locations in random order.

10 Maximal Flow in a Network

10.1 Max-flow/min-cut theory

Consider a network consisting of n nodes labeled $1, \ldots, n$ and directed edges between them with capacities c_{ij} on the arc from node i to node j. Let x_{ij} denote the flow in the arc $i \to j$, where $0 \leq x_{ij} \leq c_{ij}$.



Problem: Find maximal flow from node 1 (the **source**) to node n (the **sink**) subject to the conservation of flow at nodes, i.e.,

$$\text{maximize } v \quad \text{s.t. } 0 \leq x_{ij} \leq c_{ij}, \text{ for all } i, j$$

and

$$\boxed{\begin{array}{c}\text{flow out of} \\ \text{node } i\end{array}} - \boxed{\begin{array}{c}\text{flow into} \\ \text{node } i\end{array}} = \sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = \begin{cases} v & \text{if } i = 1 \\ 0 & \text{if } i = 2, \ldots, n-1 \\ -v & \text{if } i = n \end{cases}$$

where the summations are understood to be over existing arcs only. v is known as the **value of the flow**.

This is obviously an LP problem, with lots of variables and constraints. We can solve it more quickly (taking advantage of the special network structure) as follows.

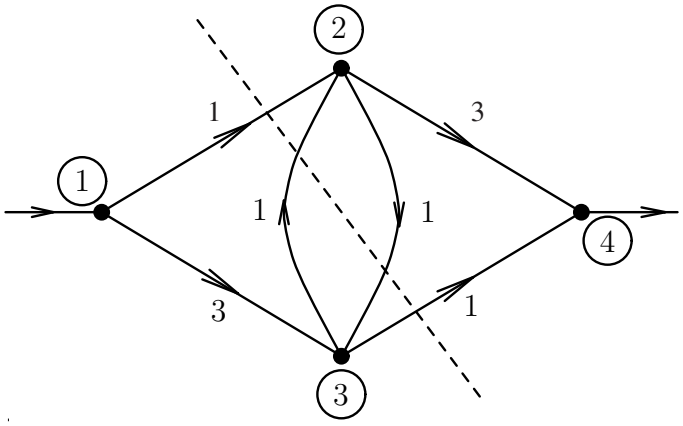
Definition 10.1. *A* **cut** $(S, \bar{S})$ *is a partition of the nodes into two disjoint subsets* S *and* $\bar{S}$ *with* $1 \in S$ *and* $n \in \bar{S}$.

Definition 10.2. *The* **capacity** *of a cut*

$$C(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}} c_{ij}.$$

Thus given a cut $(S, \bar{S})$ the capacity of the cut is the maximal flow from nodes in S to nodes in $\bar{S}$. It is intuitively clear that any flow from node 1 to node n must cross the cut $(S, \bar{S})$, since in getting from 1 to n at some stage it must cross from S to $\bar{S}$. This holds for any flow and any cut.

Example 10.1.



Cut $S = \{1, 3\}$, $\bar{S} = \{2, 4\}$, $C(S, \bar{S}) = 3$. Check that the maximal flow is 3.

In fact, we have:

Theorem 10.1 (Max flow/min cut Theorem)**.** *The maximal flow value through the network is equal to the minimal cut capacity.*

Proof. Summing the feasibility constraint

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = \begin{cases} v & \text{if } i = 1 \\ 0 & \text{if } i = 2, \ldots, n-1 \\ -v & \text{if } i = n \end{cases}$$

over $i \in S$, yields

$$\begin{aligned} v &= \sum_{i \in S, j \in N} x_{ij} - \sum_{j \in N, i \in S} x_{ji} \\ &= \sum_{i \in S, j \in \bar{S}} x_{ij} - \sum_{j \in \bar{S}, i \in S} x_{ji} \quad \text{, if } j \in S \text{ then cancels out.} \\ &\leq C(S, \bar{S}) \end{aligned}$$

since for all i, j we have $0 \leq x_{ij} \leq c_{ij}$. Hence the value of any feasible flow is less than or equal to the capacity of any cut.

So any flow $\leq$ any cut capacity, (and in particular max flow $\leq$ min cut).

Now let f be a maximal flow, and define $S \subseteq N$ recursively as follows:

(1) $1 \in S$.

(2) If $i \in S$ and $x_{ij} < c_{ij}$, then $j \in S$.

(3) If $i \in S$ and $x_{ji} > 0$, then $j \in S$.

Keep applying (2) and (3) until no more can be added to S.

So S is the set of nodes to which we can increase flow. Now if $n \in S$ we can increase flow along a path in S and f is not maximal. Thus when we stop, $n \in \bar{S} = N \setminus S$ and $(S, \bar{S})$ is a cut. From the definition of S we know that for $i \in S$ and $j \in \bar{S}$, $x_{ij} = c_{ij}$ and $x_{ji} = 0$, so in the formula above we get

$$v = \sum_{i \in S, j \in \bar{S}} x_{ij} - \sum_{j \in \bar{S}, i \in S} x_{ji} = C(S, \bar{S}).$$

So max flow = min cut capacity. □

Corollary 10.2. *If a flow value v = cut capacity C then v is maximal and C minimal.*

The proof suggests an algorithm for finding the maximal flow.

10.2 Ford-Fulkerson algorithm

1. Start with a feasible flow (e.g., $x_{ij} = 0$).
2. Construct S recursively by the algorithm defined in the box above.
3. If $n \in S$ then there is a path from 1 to n along which we can increase flow by

$$\epsilon = \min_{(ij)} \max[x_{ji}, c_{ij} - x_{ij}] > 0.$$

   where the minimum is taken with respect to all arcs $i \to j$ on the path.

   Replace the flow by this increased flow. Return to 2.

   If $n \notin S$ then the flow is optimal.

The algorithm is crude and simple; we just push flow through where we can, until we can't do so anymore. There is no guarantee that it will be very efficient. With hand examples it is usually easy to 'see' the maximal flow. You just demonstrate that it is maximal by giving a cut with the same capacity as the flow and appeal to the min cut = max flow theorem.

The algorithm can be made not to converge if the capacities are not rational multiples of one another. However,

Theorem 10.3. *If capacities and initial flows are rational then the algorithm terminates at a maximal flow in a finite number of steps. (Capacities are assumed to be finite.)*

Proof. Multiply by a constant so that all capacities and initial flows are integers. The algorithm increases flow by at least 1 on each step. □

If capacities are not rational then the algorithm is not guaranteed to stop.

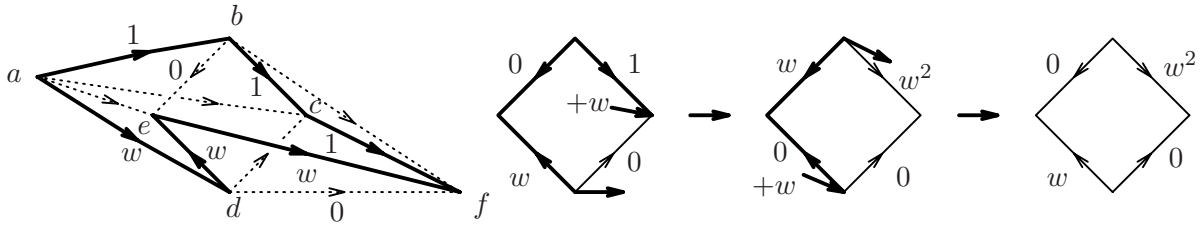


Figure 3: Example: Failure to stop when capacities and initial flows are not rational.

The network in Figure 3 consists of a square $b\,c\,d\,e$ of directed arcs of capacity 1. The corners of the square are connected to a source at a and a sink at f by arcs of capacity 10. The initial flow of $1 + w$ is shown in the first picture, where $w = (\sqrt{5} - 1)/2$, so $1 - w = w^2$. The first iteration is to increase flow by w along $a \to c \to b \to e \to d \to f$. The second increases it by w along $a \to d \to e \to b \to f$. The flow has increased by $2w$ and the resulting flow in the square is the same as at the start, but multiplied by w and rotated by 180^o. The algorithm can continue in this manner forever without stopping and never reach the optimal flow of 40.

10.3 Hall's matching theorem

The max-flow min-cut theorem has many applications. Here is one.

Consider a bipartite graph whose vertices are in two sets L (for left) and R (for right), and $E \subseteq L \times R$. Suppose $|L| = |R| = n$. The bipartite graph is said to have a **perfect matching** if by using a subset of n edges from E, pairs of $i \in L$ and $j \in R$ can be matched as the endpoints of these edges. For $i \in L$. define $N(i) = \{j : (i, j) \in E\}$ and for $X \subseteq L$, $N(X) = \cup_{i \in X} N(i)$.

Theorem 10.4 (Hall's theorem)**.** *Consider a bipartite graph $G = (L \cup R, E)$ with $|L| = |R|$. It has a perfect matching if and only if $|N(X)| \geq |X|$ for every $X \subseteq L$.*

Proof. If a perfect matching exists then clearly $|N(X)| \geq |X|$ for any $X \subseteq L$.

To prove the converse we add a 'start' s, and a 'terminus' t, and all edges of form (s, i), $i \in L$, and (j, t), $j \in R$. Let the original edges in E have capacity ∞ and the new edges capacity 1. Find $(S, V \setminus S)$, the cut of minimum capacity.

By ~~by~~ use of the assumption with $X = L \cap S$ we have $|N(L \cap S)| \geq |L \cap S|$. But $R \cap S = N(L \cap S)$. So

$$C(S) = |L \cap \bar{S}| + |R \cap S| \geq |L \cap \bar{S}| + |L \cap S| = |L|. \qquad \square$$

11 Minimum Cost Circulation Problems

11.1 Minimal cost circulations

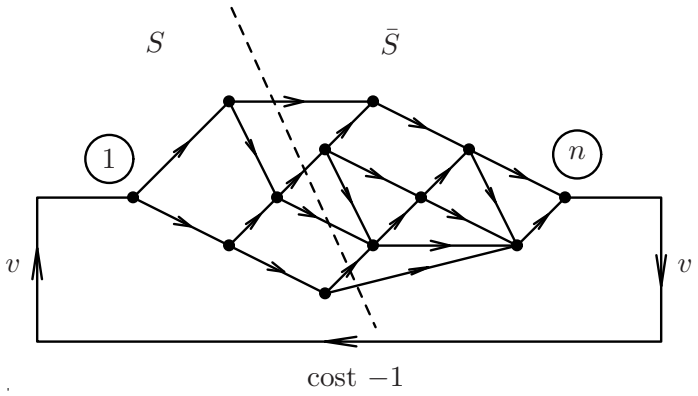
The problem of maximizing flow in a network is a special case of the **minimum cost circulation problem**. A network is a **closed network** if there is no flow into or out of the network. A flow in a closed network is a **circulation** if $\sum_j x_{ij} - \sum_j x_{ji} = 0$ for each node i. Consider the problem

$$\text{minimize } \sum_{ij} d_{ij}x_{ij}$$

$$\text{subject to } \sum_j x_{ij} - \sum_j x_{ji} = 0, \text{ each } i, \text{ and } c_{ij}^- \le x_{ij} \le c_{ij}^+.$$

That is, we are to minimize cost when on each arc (i, j) there is cost per unit flow of d_{ij} and capacity constraint $c_{ij}^- \le x_{ij} \le c_{ij}^+$. A circulation which satisfies the capacity constraints is called a **feasible circulation**.[4]

The maximal flow problem of Lecture 10 can be cast as a minimal cost circulation problem. For each arc we assign a capacity constraint $0 \le x_{ij} \le c_{ij}^+$ and all cost $d_{ij} = 0$. Add an arc from node n to 1 with no capacity constraint and cost -1.



Minimizing the cost of the circulation, $-v$, is the same as maximizing v.

11.2 Sufficient conditions for a minimal cost circulation

Consider the Lagrangian for the minimum cost circulation problem. We shall treat the capacity constraints as the region constraints, so

$$X = \{x_{ij} : c_{ij}^- \le x_{ij} \le c_{ij}^+\}.$$

[4]There is a beautiful algorithm called the out-of-kilter algorithm which will solve general problems of this kind. It does not even require a feasible solution with which to start.

We introduce Lagrange multipliers λ_i (one for each node) and write

$$L(x,\lambda) = \sum_{ij} d_{ij}x_{ij} + \sum_i \lambda_i \left(0 - \sum_j x_{ij} + \sum_j x_{ji}\right)$$

Rearranging we obtain

$$L(x,\lambda) = \sum_{ij}(d_{ij} - \lambda_i + \lambda_j)x_{ij}.$$

We attempt to minimize $L(x,\lambda)$ in X. At the minimum the following must hold:

$$x_{ij} = \begin{cases} c_{ij}^- & \text{if} \quad d_{ij} - \lambda_i + \lambda_j > 0 \\ c_{ij}^+ & \text{if} \quad d_{ij} - \lambda_i + \lambda_j < 0 \end{cases} \tag{1}$$

$$c_{ij}^- \le x_{ij} \le c_{ij}^+ \quad \text{if} \quad d_{ij} - \lambda_i + \lambda_j = 0. \tag{2}$$

Note that (1) and (2) imply that if $c_{ij}^- < x_{ij} < c_{ij}^+$ then we must have $d_{ij} - \lambda_i + \lambda_j = 0$.

Theorem 11.1. *If (x_{ij}) is a feasible circulation and there exists λ such that $(x_{ij}), \lambda$ satisfy conditions* (1) *and* (2) *above, then (x_{ij}) is a minimal cost circulation.*

Proof. This follows from the Lagrangian sufficiency theorem. □

We know that linear programs can be solved by Lagrangian methods, so provided a finite minimum exists, there will indeed exist x_{ij}, λ_i satisfying (1) and (2).

The Lagrange multipliers λ_i are known as **node numbers** or **potentials**. The difference $\lambda_i - \lambda_j$ is known as the **tension** in the arc (i,j).

Specialized to the maximum flow problem, the solution has the character that $\lambda_n = 0$, $\lambda_1 = 1$, $\lambda_i = 1$ for $i \in S$ and $\lambda_i = 0$ for $i \in \bar{S}$.

11.3 The transportation problem

Another important minimum cost flow problem is the **transportation problem**. Consider a supplier who has n supply depots from which goods must be shipped to m destinations. We assume there are quantities $s_1, \ldots, s_n$ of the goods at depots $\{S_1, \ldots, S_n\}$ and that the demands at destinations $\{D_1, \ldots, D_m\}$ are given by $d_1, \ldots, d_m$. We also assume that $\sum_i s_i = \sum_j d_j$ so that total supply = total demand. Any amount of goods may be taken directly from source i to destination j at a cost of d_{ij} $(i = 1, \ldots, n; j = 1, \ldots, m)$ per unit. One formulation of the

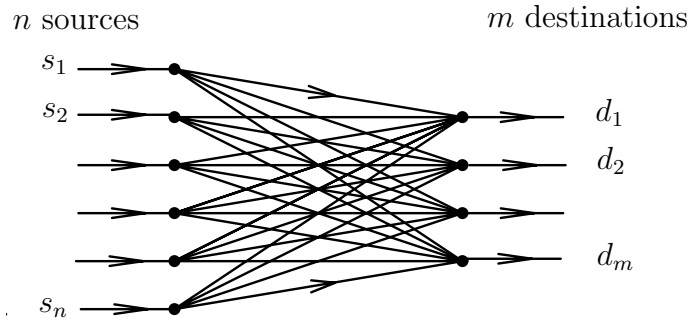
problem is

$$\text{minimize } \sum_{ij} d_{ij}x_{ij}$$

$$\text{subject to } \sum_j x_{ij} = s_i \text{ each } i, \quad \sum_i x_{ij} = d_j \text{ each } j$$

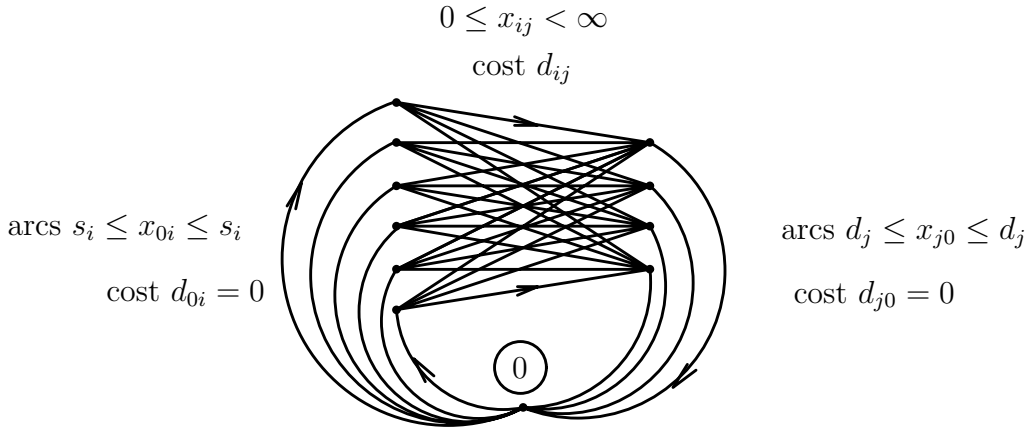
$$\text{with } x_{ij} \geq 0 \text{ each } i, j.$$

Here x_{ij} is the flow from S_i to D_j. The network looks like:



with arcs (i, j), $0 \leq x_{ij} < \infty$, and cost d_{ij} per unit flow.

If we augment the transportation network by connecting all sources and all destinations to a common 'artificial node' by arcs where the flow is constrained to be exactly that which is required (and zero cost) we obtain the same problem as in minimal cost circulation form.



The Lagrangian for the problem can be written

$$L(x, \lambda, \mu) = \sum_{ij} d_{ij}x_{ij} + \sum_i \lambda_i \left(s_i - \sum_j x_{ij} \right) + \sum_j \mu_j \left(d_j - \sum_i x_{ij} \right),$$

where we label Lagrange multipliers (node numbers) λ_i for sources and μ_j for destinations. Rearranging,

$$L(x, \lambda, \mu) = \sum_{ij} (d_{ij} - \lambda_i - \mu_j) x_{ij} + \sum_i \lambda_i s_i + \sum_j \mu_j d_j.$$

There is a finite minimum for $x_{ij} \geq 0$ if $d_{ij} - \lambda_i - \mu_j \geq 0$, for all i, j, and which occurs with complementary slackness of $(d_{ij} - \lambda_i - \mu_j) x_{ij} = 0$ on each arc.

Theorem 11.2. *A flow x_{ij} is optimal for the transportation problem if there exists λ_i, μ_j such that $d_{ij} \geq \lambda_i + \mu_j$ each (i, j) and $(d_{ij} - \lambda_i - \mu_j) x_{ij} = 0$.*

Proof. This follows from the Lagrangian sufficiency theorem. □

We noted above that the transportation problem is an example of a minimum cost circulation problem. In fact the reverse is also true: it can be shown that any minimum cost circulation problem can be reformulated as a transportation problem. This is one reason why transportation problems have been a focus of much theoretical research.

11.4 Example: optimal power generation and distribution

In Lecture 12 we learn an algorithm for solving the transportation problem. The idea is to compute node numbers using the fact that we wish $c_{ij} = \lambda_i + \mu_j$ wherever $x_{ij} > 0$. Then we examine places where $x_{ij} = 0$. If at any such place dual feasibility is violated, because $c_{ij} < \lambda_i + \mu_j$, then we can move to a better basic feasible solution by increasing the value of such x_{ij}.

The following real-life problem can be solved by the 'simplex-on-a-graph algorithm', which is similar to the transportation algorithm of the Lecture 12, and briefly described in Section 12.3. Demand for electricity at node i is d_i. Node i has k_i generators, that can generate electricity at costs of $a_{i1}, \ldots, a_{ik_i}$, up to amounts $b_{i1}, \ldots, b_{ik_i}$. There are $n = 12$ nodes and 351 generators. Transmission capacity from node i to j is c_{ij} $(= c_{ji})$.

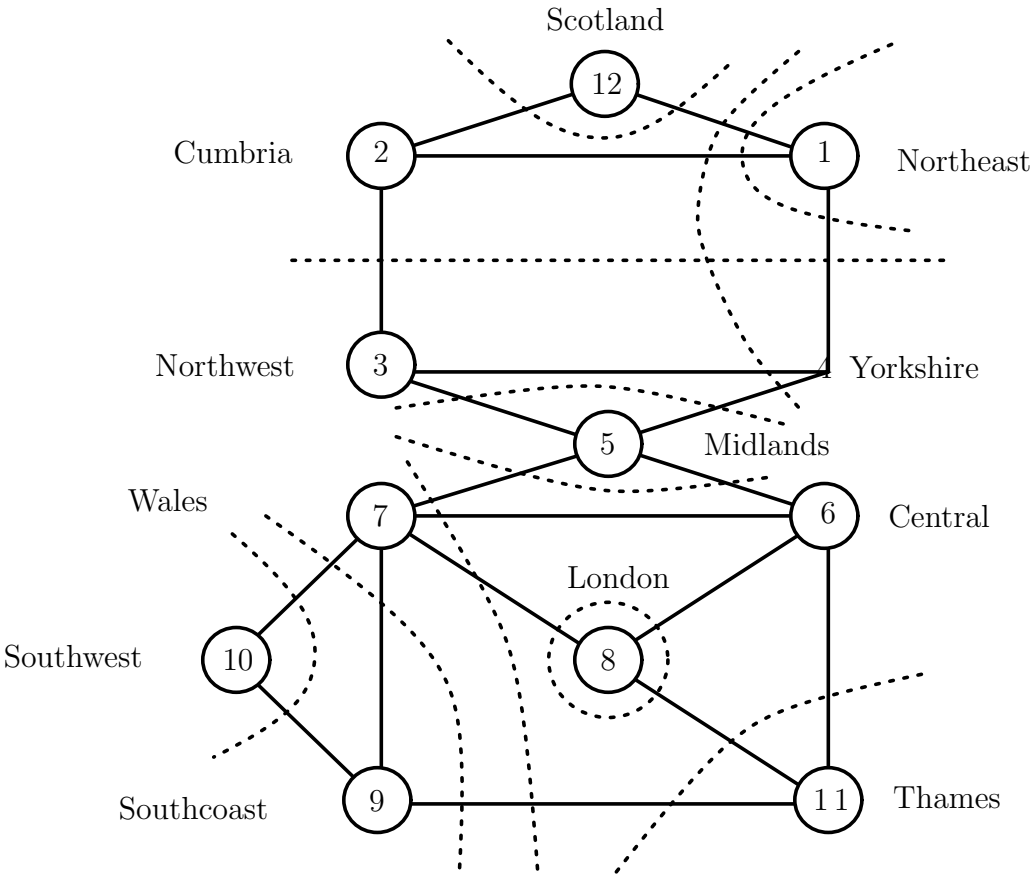
Let x_{ij} = amount of electricity carried $i \to j$ and let y_{ij} = amount of electricity generated by generator j at node i. The LP is

$$\text{minimize} \sum_{ij} a_{ij} y_{ij}$$

$$\text{subject to} \sum_j y_{ij} - \sum_j x_{ij} + \sum_j x_{ji} = d_i, \quad i = 1, \ldots, 12,$$

$$0 \leq x_{ij} \leq c_{ij}, \; 0 \leq y_{ij} \leq b_{ij}.$$

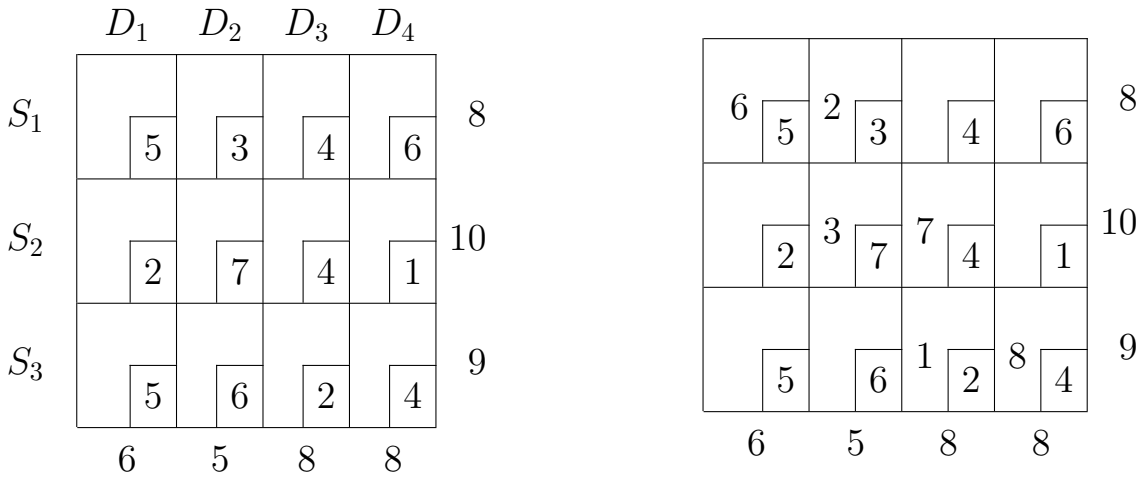
In addition, there are constraints on the maximum amount of power that may be shipped across the cuts shown by the dotted lines in the diagram.


Scotland
12
Cumbria
2
1
Northeast
Northwest
3
4
Yorkshire
5
Midlands
Wales
7
6
Central
London
Southwest
10
8
Southcoast
9
11
Thames

12 Transportation and Assignment Problems

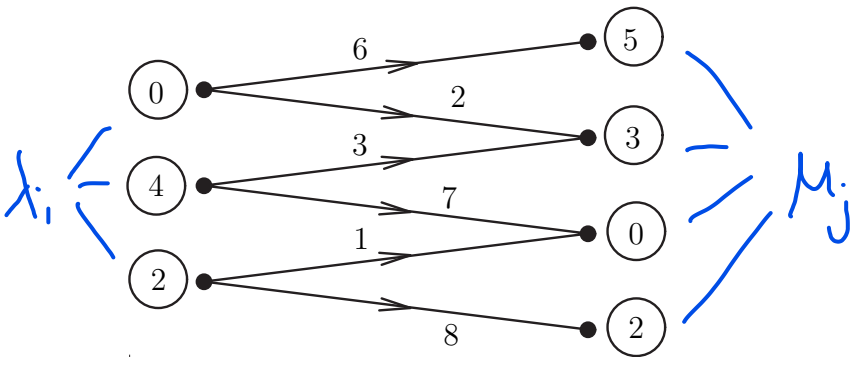
12.1 The transportation algorithm

The **transportation algorithm** is based on the sufficient conditions for optimality in Theorem 11.2. At all steps primal feasibility is maintained, a complementary slack solution to the dual is calculated, and then this is checked for dual feasibility.
1. Set out the supplies and costs in a table, as at the left below



2. Allocate an initial feasible flow (by North-West corner rule or any other sensible method). NW corner rule says start at top left corner and dispose of supplies and fulfill demands in order i, j increasing. In our case we get as at the right above.

In the absence of degeneracy (which we assume) there are not less than $(m+n-1)$ non-zero entries, which appear in a 'stair-case' arrangement.
Remark. In our network picture we have constructed a feasible flow on a **spanning tree** of $m+n-1$ arcs connecting n sources and m destinations.



A set of undirected arcs is spanning if it connects all nodes. It is a **tree** if it contains no circuits. A spanning tree is the equivalent of a basic feasible solution for this problem.

3. For optimality we require $d_{ij} = \lambda_i + \mu_j$ on any arc with strictly positive flow. Set $\lambda_1 = 0$ (arbitrarily) and then compute the remaining λ_i, μ_j by using $d_{ij} = \lambda_i + \mu_j = 0$ on arcs for which $x_{ij} > 0$. On the table we have

$\lambda_i \setminus \mu_j$	5	3	0	2
0	6 [5]	2 [3]	[4]	[6]
4	[2]	3 [7]	7 [4]	[1]
2	[5]	[6]	1 [2]	8 [4]

The node numbers are also shown on the network version above. With non-zero flows forming a spanning tree we can always compute uniquely all node numbers given one of them, such as $\lambda_1 = 0$. Notice that it is only $\lambda_i + \mu_j$ that matters so we could subtract some constant from all λ_i and add the same constant to all μ_j and that would do as well. That is why we can arbitrarily take $\lambda_1 = 0$.

4. We now compute $\lambda_i + \mu_j$ for all the remaining boxes (arcs) and write these elsewhere in the boxes. E.g.,

$\lambda_i \setminus \mu_j$	5	3	0	2
0	6 [5]	2 [3]	0 [4]	2 [6]
4	9 [2]	3 [7]	7 [4]	6 [1]
2	7 [5]	5 [6]	1 [2]	8 [4]

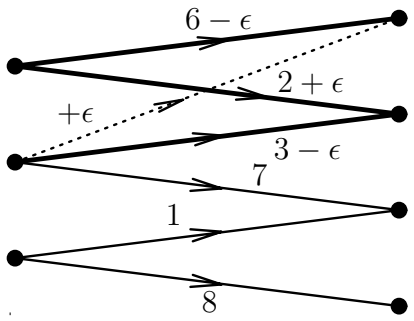
5. If all $d_{ij} \geq \lambda_i + \mu_j$, then the flow is optimal. Stop.

6. If not, (e.g., $(i, j) = (2, 1)$, where $\lambda_2 + \mu_1 = 9 > d_{21} = 2$) we attempt to increase the flow in arc (i, j) for some (i, j) such that $\lambda_i + \mu_j > d_{ij}$. We seek an adjustment of $+\epsilon$ to the flow in arc (i, j) which keeps the solution feasible (and therefore preserves total supplies and demands). In our case we do this by

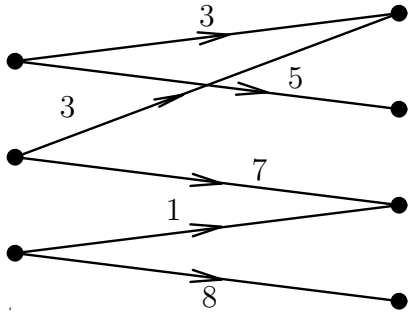
$6 - \epsilon$	$2 + \epsilon$	0	0
$+\epsilon$	$3 - \epsilon$	7	0
0	0	1	8

and pick ϵ as large as possible (without any flow going negative) to obtain a new flow (for $\epsilon = 3$).

In a non-degenerate problem there will be only one way to do this. The operation is perhaps clearer in the network picture.



We attempt to increase flow in the dotted arc. Adding an arc to a spanning tree creates a circuit. Increase flow around the circuit until an arc in the spanning tree drops out, leaving a new spanning tree. The new solution is



6. Now return to step 3 and recompute node numbers:

$\lambda_i \setminus \mu_j$	5	3	7	9
0	3 (5)	5 (3)	7 (4)	9 (6)
-3	3 (2)	0 (7)	7 (4)	6 (1)
-5	0 (5)	-2 (6)	1 (2)	8 (4)

In our example we obtain $\lambda_i = 0, -3, -5$ and $\mu_j = 5, 3, 7, 9$ at the next stage. We find $d_{ij} < \lambda_i + \mu_j$ for $(i, j) = (1, 3)$, $(1, 4)$ and $(2, 4)$. Increase the flow in $(2, 4)$ by 7 to obtain the new flow below. This is now optimal, as we can check from the final

node numbers that $d_{ij} \geq \lambda_i + \mu_j$ everywhere.

$\lambda_i \setminus \mu_j$	5	3	2	4
0	3 | 5	5 | 3	2 | 4	4 | 6
-3	3 | 2	0 | 7	-1 | 4	7 | 1
0	5 | 5	3 | 6	8 | 2	1 | 4

Remark. The route around which you may need to alter flows can be quite complicated but it is always clear how you should do it. For example, had we tried to increase the flow in arc $(3, 1)$ instead of $(2, 1)$ at step 5 we would have obtained

$6 - \epsilon$	$2 + \epsilon$	0	0
0	$3 - \epsilon$	$7 + \epsilon$	0
$+\epsilon$	0	$1 - \epsilon$	8

To summarise:

1. Pick initial feasible solution with $m + n - 1$ non-zero flows (NW corner rule).
2. Set $\lambda_1 = 0$ and compute λ_i, μ_j using $d_{ij} = \lambda_i + \mu_j$ on arcs with non-zero flows.
3. If $d_{ij} \geq \lambda_i + \mu_j$ for all (i, j) then flow is optimal.
4. If not, pick (i, j) for which $d_{ij} < \lambda_i + \mu_j$.
5. Increase flow in arc (i, j) by as much as possible without making the flow in any other arc negative. Return to 2.

12.2 The assignment problem

The **assignment problem** concerns minimizing the cost of assigning n agents to n jobs, where c_{ij} is the cost of assigning job j to agent i. The aim is to

$$\begin{aligned} &\underset{x_{ij}\in\{0,1\}}{\text{minimize}} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ &\text{subject to } \sum_{j=1}^{n} x_{ij} = 1, \quad \sum_{i=1}^{n} x_{ij} = 1 \quad \text{for all } i,j \in \{1,\dots,n\}. \end{aligned} \tag{3}$$

Lemma 12.1. *A feasible solution $\{x_{ij}\}$ to (3) is optimal if there exist $\{\lambda_i\}$, $\{\mu_j\}$ such that $\lambda_i + \mu_j \le c_{ij}$ for all i, j, and $\lambda_i + \mu_j = c_{ij}$ if $x_{ij} = 1$.*

Unfortunately, this cannot be solved using the transportation algorithm. This is because of the high amount of degeneracy. The size of the basis (number of $x_{ij} > 0$) in a general transportation problem is '#rows + #columns -1. We rely on this to calculate the λ_i and μ_j. But a solution to an assignment problem has only '#rows' non-zero variables.

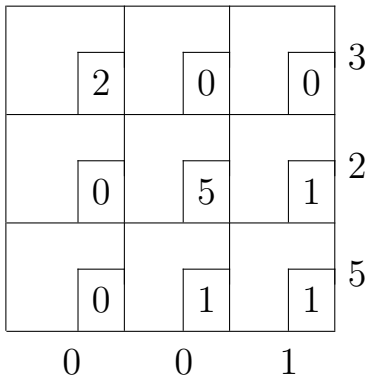
We can search for node numbers in another way. Consider the cost matrix

5	3	4
2	7	4
5	6	7

We notice that however the 3 agents are assigned to the 3 tasks the costs in the rows 1, 2, 3 will be at least 3, 2, and 5, respectively. Subtracting these row minimums from each row we see the problem is equivalent to one with costs

2	0	1	3
0	5	2	2
0	1	2	5

Similarly, the cost incurred in the third column must be at least 1, so subtracting 1 from each entry in that column we arrive at an equivalent problem

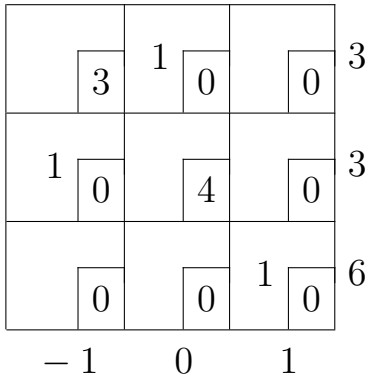


On the example sheet there is an example in which after operations like these we are done. But in this example there does not yet exist a set of 3 'independent 0s', i.e. 3 0s which share no common rows or columns.

We proceed with Step 2 of the so-called **Hungarian algorithm** (non-examinable).

Step 1 is to create at least one 0 into each row and column, as above.

Step 2 is to 'cover' all the 0s with the minimum number of 'lines' (horizontal through rows and vertical through columns). In the cost matrix above, the 0s are covered by a minimum of two lines: one down column 1 and one across row 1. Now we add a constant to every cost in covered columns, and subtract it from every cost in an uncovered rows (where the constant is the minimum of costs in cells not covered); in our example that is $1 = \min\{5, 1, 1, 1\}$; so we add 1 to costs in the first column and subtract 1 from costs in rows 2 and 3, arriving at a equivalent problem:



Now a solution is easily spotted, as shown. Agents 1, 2, 3 are assigned to tasks 2, 1, 3, respectively. The minimum cost is 12 (which is in fact the sum of all the node numbers, since the dual objective function is $\sum_i \lambda_i + \sum_j \mu_j$). The node numbers $\lambda = (3, 3, 6)$ and $\mu = (-1, 0, 1)$ are exactly what we need to apply Lemma 12.1. In general we might need to repeat step 2 at this point.

It can be reasoned that Step 2 always increases by at least 1 the number of lines required to cover the 0s. Once n lines are required to cover we have n independent 0s and the optimal assignment is then clear. The problem of finding the minimum number of lines that 'cover' the 0s can be cast as a minimum-cut problem and solved by using the Ford-Fulkerson algorithm to finds the maximum number of independent 0s. This is **König's theorem**: that states that the minimum number of lines that cover the 0s equals the maximum number of independent 0s. It can be proved by applying min cut = max flow to a suitable network (see if you can think what network provides the proof). Unlike the transportation algorithm, the running time of the Hungarian algorithm is polynomial, as $O(n^4)$.

12.3 *Simplex-on-a-graph*

The transportation algorithm can easily be generalised to a problem of minimizing costs in a general network in which there is a constraint $0 \leq x_{ij} < \infty$ on each directed arc (i, j), and a flow b_i enters the network at each node i (though it is hard to keep track of all the numbers by hand). This has been discussed in Section 11.4.

Here we don't label sources anddestinations separately, but do allow $b_i \geq 0$ and $b_i \leq 0$. Clearly, $\sum_i b_i = 0$ for conservation of flow. The **simplex-on-a-graph algorithm** solves this problem in an identical fashion to the transportation algorithm. Once again a basic solution is a spanning tree of non-zero flow arcs. Suppose there are n nodes.

1. Pick an initial basic feasible solution. Obtain $n - 1$ non-zero flow arcs.
2. Set $\lambda_1 = 0$ and compute λ_i on other nodes using $d_{ij} - \lambda_i + \lambda_j = 0$ on arcs of the spanning tree.
3. Compute $d_{ij} - \lambda_i + \lambda_j$ for other arcs. If all these are ≥ 0 then optimal. If not, pick (i, j) such that $d_{ij} - \lambda_i + \lambda_j < 0$.
4. Add arc (i, j) to the tree. This creates a circuit. Increase flow around the circuit (in direction of arc (i, j)) until one non-zero flow drops to zero and a new basic solution is created. Return to 2.

Appendix

Non-examinable This is not done in lectures, but provides further insight about why $\nabla\phi(b) = \lambda$. Let $P(b)$ be the problem: maximize $f(x) : g(x) \leq b,\ x \in \mathbb{R}^n$. Let $\phi(b)$ be its optimal value.

Theorem 4.5. *Suppose f and g are continuously differentiable on $X = \mathbb{R}^n$, and that for each b there exist unique*

- *$x^*(b)$ optimal for $P(b)$, and*
- *$\lambda^*(b) \in \mathbb{R}^m$, $\lambda^*(b) \geq 0$ such that $\phi(b) = \sup_{x\in X}\{f(x) + \lambda^*(b)^\top(b - g(x))\}$.*

If x^* and λ^* are continuously differentiable, then

$$\frac{\partial\phi(b)}{\partial b_i} = \lambda_i^*(b). \tag{4}$$

Proof.

$$\phi(b) = L(x^*, \lambda^*) = f(x^*) + \lambda^*(b)^\top(b - g(x^*))$$

Since $L(x^*, \lambda^*)$ is stationary with respect to x_j^*, we have for each j,

$$\frac{\partial L(x^*, \lambda^*)}{\partial x_j^*} = 0\,.$$

For each k we have either $g_k(x^*) = b_k$, or $g_k(x^*) < b_k$. Since $\lambda^*(b)^\top(b - g(x^*)) = 0$ we have in the later case, $\lambda_k^* = 0$, and so $\partial\lambda_k^*/\partial b_i = 0$. So

$$\frac{\partial\phi(b)}{\partial b_i} = \frac{\partial L(x^*, \lambda^*)}{\partial b_i} + \sum_{j=1}^{n} \frac{\partial L(x^*, \lambda^*)}{\partial x_j^*}\frac{\partial x_j^*}{\partial b_i}.$$

On the r.h.s. above, the second term is 0 and the first term is

$$\frac{\partial L(x^*, \lambda^*)}{\partial b_i} = \lambda_i^*(b) + \sum_{k=1}^{m} \frac{\partial\lambda_k^*(b)}{\partial b_i}\Big[b_k - g_k(x^*(b))\Big].$$

Now the second term on the r.h.s. above is 0, and so we have (4). □

Index