OR2 Notes

1 Dynamic Programming

Dynamic programing is an approach to solving problems rather than a technique for solving a particular problem. The approach can be applied to a wide range of problems, although in many cases it leads to impractical algorithms.

The problem to be tackled is formulated as making a sequence of decisions. Having made one decision, the problem of choosing the remaining decisions is often a similar but 'smaller' version of the original problem. This can lead to a 'functional equation' for finding the best initial decision and each subsequent decision.

1.1 Simple Production Problem

As a simple example we consider the following problem: a company estimates the demand d_j for one of its products over the next n periods. It costs the company $c(x)$ to manufacture x units in any one period. All demand must be met in the period in which. it occurs but stocks. may be built up to provide for demand in future periods. The maximum stock that can be held at any time is H. How much should be produced in each period to minimise the total cost of production. To make the problem self-contained we have to say something about initial and final stocks. Suppose then that there is an initial stock of i_0 and that any stock left over at the end of period n is worthless.

The problem then is to decide how much to produce in period 1, how much to produce in period 2 etc. Suppose we decide to produce an amount x_1, in period 1, then at the beginning of period 2 we will have a stock level of $i_0 + x_1 - d - 1$ and the problem of minimising the production cost over the next $n - 1$ periods. We can write this down mathematically. Define the quantity $f_r(i)$ to be the minimum cost of meeting demand in periods $r, r + 1 \ldots, n$ given that one has i units in stock at the beginning of period r.

Focussing temporarily on period 1, we can ask the question, if we decide'to produce an amount x, in period 1, what is the minimum production cost obtainable over the whole n periods? This minimum cost is clearly

$$c_1(x_1) + f_2(i_0 + x_1 - d_1). \tag{1}$$

The first term is the cost of period:1 and the second term is the minimum cost over periods $2, 3, ...n$ given that we produced x_1.

The next question is what is the best value of x, to take. The answer must be, the value of x that minimises (1), This will give us the minimum production cost for periods $1, 2, \ldots, n$, starting with a stocki_0, i.e. $f_1(i_0)$. We have thus proved thet

$$f_1(i_0) = \min_{x_1} \left\{ c(x_1) + f_2(i_0 + x_1 - d_1) \right\}. \tag{2}$$

A similar argument about the decision to be taken at the beginning of period r given that the stock level is currently i shows that in general

$$f_r(i) = \min_{x_r} \{c(x_r) + f_{r+1}(i + x_r - d_r)\}. \tag{3}$$

The range over which the 'decision variable' x_r is to be minimised depends on our assumptions about the problem. Firstly, we must have $x_2 \geq 0$ and since we must produce enough to meet the demand d_r, we must have $i + x_r \geq d_r$. The maximum stock level is H and consequently we must havé $i + x_r - d_r \leq H$. Thus x_r, is to be chosen in the range

$$\max\{0, d_r - i\} \leq x_r \leq H + d_r - i. \tag{4}$$

Now the argument that produced (3) only read holds true for $r \leq n-1$, basically because we have not defined $f_{n+1}(i)$. Examining our assumption about final stocks we can see that this is equivalent to

$$f_n(i) = \min_{x_n} c(x_n). \tag{5}$$

This can be put into the framework of (3) by defining $f_{n+1}(i) = 0$. Equations (3) and (5) give us a means of solving our problem. We first calculate $f_n(i)$ for $i = 0, 1, 2, \ldots, H$. We then use (3) to calculate $f_{n-1}(i)$ for $i = 0, 1, 2, \ldots, H$, and then $f_{n-2}(i)$ and so on until we reach $f_1(i)$. If the production quantities x need not be integral then we have to approximate by dividing the range [0,H] into a suitable number of points - depending on the accuracy required and computer storage and time available.

Let us solve the above problem when $n = 4, d_j = 3$ in all periods, the maxinum stock level $H = 4$ and $c(x) = 16x - x^2$.

So that we can keep track of the optimal production policy we make a note of the value of x minimising the R.H.S of (3). Denote this value by $x_r(i)$.

Calculation of f_4 By definition $f_4(i) = \min\{18x - x^2 : \max\{0, 3 - i\} \leq x \leq 7 - i\}$.

$$\begin{array}{ll} f_4(0) = 45 & x_4(0) = 3 \\ f_4(1) = 32 & x_4(1) = 2 \\ f_4(2) = 17 & x_4(2) = 1 \\ f_4(3) = 0 & x_4(3) = 0 \\ f_4(4) = 0 & x_4(4) = 0 \end{array}$$

paragraphCalculation of f_3 By definition $f_3(i) = \min\{18x - x^2 + f_4(i + x - 3) : \max\{0, 3 - i\} \leq x \leq 7 - i\}$.

$$f_3(0) = \min\{45 + f_4(0), 56 + f_4(1), 65 + f_4(2), 72 + f_4(3), 77 + f_4(4)\} = 72 \text{ and } x_3(0) = 6.$$

Continuing this we build up the table

i	$f_4(i)$	$x_4(i)$	$f_3(i)$	$x_3(i)$	$f_2(i)$	$x_2(i)$	$f_1(i)$	$x_1(i)$
0	45	3	72	6	109	7	142	7
1	32	2	65	5	104	2/6	135	5/6
2	17	1	56	4	89	1	126	1
3	0	0	45	0/3	72	0	109	0
4	0	0	32	0/2	65	0	104	0/2

Suppose for example that the initial stock level in period 1 is 0. We see from the table that the minimum total production cost is 142. The optimal productien policy is found as follows: $x_1(0) = 7$ .e. given a stock level of 0 at the beginning of period 1 the optimum production for period 1 is 7. Producing 7 in period 1 means we start period 2 with a stock level 4. From the table $x_2(4) = 0$ i.e. given a stock level of at the beginning of period 2 the optimm production for period 2 is 0. This means we start period 3 with stock level 1. Now $x_3(1) = 5$, so we produce 5 units in period 3 and therefore start period 4 with initial stock 3. As $x_4(3) = 0$ we produce nothing in this period. Thus the optimal policy starting period 1 with zero stock is

x_1	x_2	x_3	x_4
7	0	5	0

We may in a similar manner use the table to find the optimum policy for all possible initial stock levels. In the method above we have worked backwards from period n in calculating the optimum policy. This is called the backward formulation of the problem.

In the backward formulation model we nad to be explicit on what happened to the final stock, in the forward formulation we have to fix the initial stock at some value. For simplicity assume the initial stock is zero.

Now let us define the quantity $g_r(i)$ to be the minimum cost of meeting demand in periods $1, 2, \ldots, r$ given that the stock level at the end of period r is i. Then arguing in a similar manner to the backward formulation we get

$$g_1(i) = c(i + d_1) \tag{6}$$
$$g_r(i) = \min_{x_r} \{c(x_r) + g_{r-1}(i + d_r - x_r)\} \tag{7}$$

where x_3 in (7) ranges over

$$\max\{0, i + d_r - H\} \leq x_r \leq i + d_r.$$

Starting with g_1 as defined in (6) we use (7) iteratively to calculate g_n and we can thus calculate an optimum for any value of the final stock.

Holding Cost We now add a cost of holding non-zero inventory. Suppose that this cost is $h(i_1, i_2)$, where i_1 is the inventory at the start of the period and i_2 is the inventory at the end of the period. Then we replace (3) by

$$f_r(i) = \min_{x_r} \{c(x_r) + h(i, i + x_r - d_r) + f_{r+1}(i + x_r - d_r)\}.$$

Backordering Suppose we are allowed to be B behind in filling the demand, then the changes to (3) are that we allow $i \in [-B, H]$ and so (4) is replaced by

$$\max\{0, d_r - i - B\} \leq x_r \leq H + d_r - i.$$

Smoothing Production levels In the example we computed, the production changed significantly from preiod to period. Suppose we do like that and that we had a cost $s(x_{r-1}, x_r)$. We replace (3) by

$$f_r(x, i) = \min_{x_r} \{c(x_r) + s(x_{r-1}, x_r) + f_{r+1}(x_r, i + x_r - d_r)\}.$$

Here x represents the production level for the previos period.

1.2 Knapsack Problem

$w_1, w_2, \ldots, w_n, W, c_1, c_2, \ldots, c_n$ are positive integers.
Problem

$$\begin{aligned} \text{Maximize} & \sum_{j=1}^{n} c_j x_j \\ \text{Subject to} & \sum_{j=1}^{n} w_j x_j \leq W \\ & x_j \geq 0 \text{ and integer } j = 1, 2, \ldots, n \end{aligned}$$

A scout is going on a trip and has a knpasack that can hold at most W pounds. They have to pack items of type $1, 2, \ldots, n$. Each item of type j weighs w_j pounds and has value c_j. They can take more than one of each item and they want to maximise the value of the items packed.

Let now $f_r(w)$ denote the maximum to the above integer program when W is replaced by w and n is replaced by r.
First recurrence:

$$f_r(w) = \max_{0 \leq x \leq \lfloor w/w_r \rfloor} \{c_r x + f_{r-1}(w - w_r x)\}.$$

For any choice of $x = x_r$ we gain $c_r x$ plus the optimum available from the remining variables $x_1, x_2, \ldots, x_{r-1}$.

Second recurrence:

$$f_r(w) = \max \begin{cases} f_{r-1}(w) & x_r = 0 \text{ in optimum.} \\ c_r + f_r(w - w_r) & x_r \geq 1 \text{ in optimum.} \end{cases}$$

Example:

$$\begin{aligned} \text{Maximize } & 2x_1 + 3x_2 + 5x_3 + 7x_4 \\ \text{Subject to } & 2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 12 \\ & x_1, \ldots, x_4 \geq 0 \text{ and integer} \end{aligned}$$

w	f_1	δ_1	f_2	δ_2	f_3	δ_3	f_4	δ_4
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
2	2	1	2	0	2	0	2	0
3	2	1	3	1	3	0	3	0
4	4	1	4	0	5	1	5	0
5	4	1	5	1	5	0/1	7	1
6	6	1	6	0/1	7	1	7	0/1
7	6	1	7	1	8	1	9	1
8	8	1	8	0/1	10	1	10	0/1
9	8	1	9	1	10	1	12	1
10	10	1	10	0/1	12	1	14	1
11	10	1	11	1	13	1	14	1
12	12	1	12	0/1	15	1	16	1

Solution Let $x_r(w)$ denote the value of x_r in the optimum solution for ω. $\delta_r(\omega) = 1$ iff $x_r(w) \geq 1$.

$\delta_4(12) = 1$ and so $x_4(12) \geq 1$. $\delta_4(7) = 1$ and so $x_4(7) \geq 1$. $\delta_4(2) = 0$ and so $x_4(12) = 2$.
$\delta_3(2) = 0$ and so $x_3(2) = 0$.
$\delta_2(2) = 0$ and so $x_2(2) = 0$.
$\delta_1(2) = 1$ and so $x_1(2) = 1$.

Thus the solution is $x_1 = 1, x_2 = x_3 = 0, x_4 = 2$.

Execution time The above algorithm requires $O(nW)$ arithmetic operations. $O(1)$ for each choice of r, ω.

Another recurrence: let

$$f(w) = \text{Max.} \sum_{j=1}^{n} c_j x_j$$
$$\text{Subject to} \sum_{j=1}^{n} w_j x_j \leq w$$
$$x_j \geq 0 \text{ and integer } j = 1, 2, \ldots, n$$

Then if $\mu = \min_j w_j$,

$$f(w) = \begin{cases} \max_{j=1,2,\ldots,n}(c_j + f(w - w_j)) & w \geq \mu. \\ 0 & w < \mu. \end{cases}$$

1.3 Replacement of a machine

A company uses a machine to manufacture a single product over the next N periods. The demand in period n is known to be d_n and the maximum amount of stock that can be held at one time is H. The cost of producing an amount x depends on the current age of the machine. It costs $c(x, t)$ to produce an amount x using a machine of age t. A machine of age T has to be scrapped. Assume that we start in period 0 with a new machine. A new machine costs A to buy. Here is how we formulate the problem: Let $f_n(t, h)$ denote the minimum cost of meeting demand in periods $n, n+1, \ldots, N$ if we start period n with a machine of age t and h units in stock. Then

$$f_n(t,h) = \min \begin{cases} \min\limits_{\substack{0 \leq x \leq H-h+d_n \\ x \geq d_n - h}} \{c(x,t) + f_{n+1}(t+1, x+h-d_n)\} & \text{Keep old machine} \\ \min\limits_{\substack{0 \leq x \leq H-h+d_n \\ x \geq d_n - h}} \{A + c(x,0) + f_{n+1}(1, x+h-d_n)\} & \text{Replace machine} \end{cases}$$

The above recurrence is computed for $n = N, N-1, \ldots, 1$, $t = 0, 1, \ldots, T-1$ and $h = 0, 1, \ldots, H$. If $t = T$ then we let

$$f_n(T, h) = A + f_n(0, h).$$

1.4 Probabilistic production problem

A company needs to meet demand for its single product over the next N periods. The cost of producing an amount x is $c(x)$ in any period. The demand is a random variable and let us assume that

$$\mathbb{P}(d_n = d) = p_{n,d} \qquad d \geq 0.$$

The company can store up to amount H at any time. The company will try to meet the demand, but if it is too large then there is a penalty cost of π for any demand left unsatisfied. The company wishes to minimises the expected cost of production. Assume first that the company has to make its period n production decision *before* it knows d_n. Let $f_n(h)$ denote the minimum expected cost of production in periods $n, n+1, \ldots, N$ if we start period n with h units in stock. Then, if $\xi^+ = \max\{0, \xi\}$,

$$f_n(h) = \min_{x \geq 0} \left(c(x) + \sum_{d \geq 0} p_{n,d}(f_{n+1}(\min\{(x+h-d)^+, H\}) + \pi \max\{0, d-(h+x)\}) \right).$$

As an alternative criterion, suppose one has to minimise expected cost subject to having at least a 90% chance of meeting demand in every period. Then we let $f_n(h)$ be the minimum cost of operating under these criteria for a given n and h.

$$f_n(h) = \min_{x \geq \alpha_h} \{c(x) + \sum_{d \geq 0} p_{n,d}(f_{n+1}(\min\{(x+h-d)^+, H\}) + \pi(d-(h+x))^+)\}$$

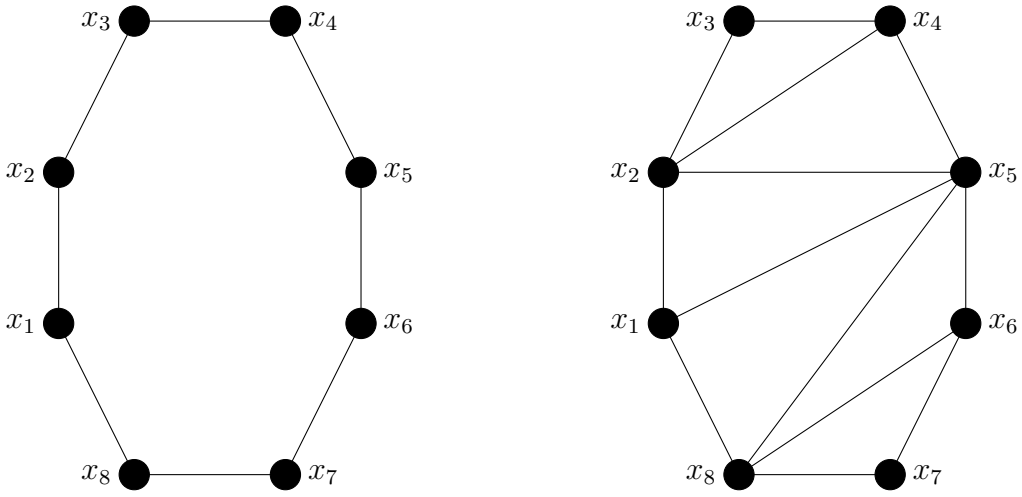
where $\alpha_h = \min_\alpha : \sum_{d > \alpha + h} p_{n,d} \leq .1$.

If the company can make its period n production decision *after* it knows d_n then we have

$$f_n(h) = \sum_{d \geq 0} p_{n,d} \min_{\substack{x \geq (d-h)^+ \\ x \leq H+d-h}} \{c(x) + f_{n+1}(h+x-d)\}.$$

1.5 Minimal triangulation of a convex polygon

Let P be a convex polgon with vertices $X_1, X_2, \ldots, X_n$. We want to triangulate it in such a way as to minimise the sum of the lengths of the chords used.



Let $m^*_{k,l}$ be the length of the minimum length triangulation of the polygon defined by $X_k, X_{k+1}, \ldots, X_l, X_k$. Then

$$m^*_{k,l} = \min_{k<j<l} \{m^*_{k,j} + m^*_{j,l} + |X_k - X_j| + |X_j - X_l|\} \tag{8}$$

where $|X_k - X_j|$ is the length of the edge X_k, X_j etc.

Here $m^*_{k,l} = 0$ if $l = k + 1$ and we use the recurrence (8) to compute what we want i.e. $m^*_{1,n}$.

1.6 Breaking up a stick

A stick of length L is to be broken into pieces of integer length. Let $v_{i,j}$ be the *value* of a piece $[i, i+1, \ldots, j]$. How should the stick be broken in order to maximise the total value.

let $f(r), r = 0, 1, \ldots, L$ be the maximum value obtainable from breaking up $[0, r]$. Then

$$f(0) = 0 \text{ and } f(r) = \max_{0 \le i < r} \{f(i) + v_{i,r}\}, \qquad 0 < r \le L.$$

So $f(L)$ can be computed in $O(L^2)$ operations.

Suppose now that the stick must be broken into k pieces. Now use $f(j, r)$ to be the maximum obtainable from breaking $[0, r]$ into j pieces. Then,

$$f(j, r) = \begin{cases} 0 & j > r. \\ \max_{0 \le i < r} \{f(j-1, i) + v_{i,r}\} & j \le r. \end{cases}$$

So $f(k, L)$ can be computed in $O(kL^2)$ operations.

1.7 A problem with an infinite time horizon

A *system* can be in one of a set V of possible states. For each $v \in V$ one can choose any $w \in V$ and move to w at a cost of $c(v, w)$. The system is to run *forever* and it is required to minimise the *discounted cost* of running the system, assuming that the discount factor is α. A *policy* is a function $\pi : V \to V$. So if $|V| = n$ then there are n^n distinct policies to choose from.

Example

$$\text{Costs} \begin{bmatrix} 2 & 1 & 3 \\ 4 & 3 & 2 \\ 1 & 3 & 2 \end{bmatrix} \quad \alpha = 1/2.$$

Let π be a policy and let y_v be the discounted cost of this policy, starting at $v \in V$. Then

$$y_v = c(v, \pi(w)) + \alpha y_{\pi(v)} \qquad v \in V. \tag{9}$$

Example Let $\pi(1) = \pi(2) = \pi(3) = 1$. Then

$$\begin{aligned} y_1 &= 2 + \frac{1}{2} y_1 \\ y_2 &= 4 + \frac{1}{2} y_1 \\ y_3 &= 1 + \frac{1}{2} y_1. \end{aligned}$$

So

$$y_1 = 4,\ y_2 = 6,\ y_3 = 3.$$

Problem: Find the policy π^* which minimises y_v simultaneously for all $v \in V$.

Theorem 1. Optimality Criterion
π^* is optimal iff its values y_v^* satisfy

$$y_v^* = \min_{w \in V}\{c(v,w) + \alpha y_w^*\} \qquad \forall v \in V. \tag{10}$$

Proof Suppose that (10) does not hold for some π.

$$\begin{aligned} y_u &> c(u,\lambda(u)) + \alpha y_{\lambda(u)} \qquad u \in U \\ y_v &= \min_{w \in V}\{c(v,w) + \alpha y_w\} \qquad u \notin U \end{aligned}$$

Define $\tilde{p}$ by $\tilde{p}(u) = \lambda(u)$ for $u \in U$ and $\tilde{p}(v) = \pi(v)$ for $v \notin U$. Then for $u \in U$,

$$\begin{aligned} y_u &> c(u,\lambda(u)) + \alpha y_{\lambda(u)} \\ \tilde{y}_u &= c(u,\lambda(u)) + \alpha \tilde{y}_{\lambda(u)} \end{aligned}$$

So if $\xi_v = y_v - \tilde{y}_v$ for $v \in V$ then

$$\xi_u > \alpha \xi_{\tilde{p}(u)} \qquad u \in U. \tag{11}$$

Also, for $v \notin U$

$$\begin{aligned} y_v &= c(v,\pi(v)) + \alpha y_{\pi(v)} \\ \tilde{y}_v &= c(v,\pi(v)) + \alpha \tilde{y}_{\pi(v)} \end{aligned}$$

and so

$$\xi_v = \alpha \xi_{\tilde{p}(v)} \qquad v \notin U. \tag{12}$$

It follows from (11), (12) that

$$\begin{aligned} \xi_v &\geq \alpha^t \xi_{\tilde{p}^t(v)} \qquad \forall v \notin U, t \geq 1 \\ \xi_u &> \alpha^t \xi_{\tilde{p}^t(u)} \qquad \forall u \in U, t \geq 1 \end{aligned}$$

Letting $t \to \infty$ we see that

$$\xi_v \geq 0\ \forall v \text{ and } \xi_u > 0\ \forall u \in U.$$

Thus $\tilde{p}$ is *strictly better* than π i.e. if (10) does not hald, then we can improve the current policy.

Conversely, if (10) holds and $\widehat{\pi}$ is any other policy and $\eta_v = \widehat{y}_v - y_v^*$ then

$$\begin{aligned} \widehat{y}_v &= c(v,\widehat{\pi}(v)) + \alpha \widehat{y}_{\widehat{\pi}(v)} \\ y_v^* &\leq c(v,\widehat{\pi}(v)) + \alpha y_{\widehat{\pi}(v)}^* \end{aligned}$$

and so

$$\eta_v \geq \alpha \eta_{\widehat{\pi}(v)} \geq \cdots \geq \alpha^t \eta_{\widehat{\pi}^t(v)} \qquad \text{for } t \geq 1$$

which implies that $\eta_v \geq 0$ for $v \in V$.

Policy Improvement Algorithm

1. Choose arbitrary initial policy π.
2. Compute y as in (9).
3. If (10) holds – current π is optimal, stop.
4. If (10) doesn't hold then
5. compute λ by
   $y_{\lambda(v)} = \min_w\{c(v,w) + \alpha y_w\}$.
6. $\pi \leftarrow \lambda$.
7. goto 2.

In our example with $\pi = (1,1,1)$. First compute $\lambda = (1,3,1)$. Re-compute $y = (\frac{39}{28}, \frac{11}{14}, \frac{95}{56})$. Now $\lambda = \pi$ i.e. (1) holds and we are done.

1.7.1 Probabilistic version

Let us introduce some probability: Suppose now that for each $i \in V$ there is a set X_i of possible decisions. Suppose that if the system is in state i and decision $x \in X_i$ is taken then

- The expected cost of the immediate step is $c(x,i)$.
- The next state is j with probability $P(x,i,j)$

A policy π specifies a decision $\pi(i) \in X_i$ for each $i \in V$.

First let us evaluate this policy. Let y_i denote the expected discounted cost of pursuing policy π indefinitely, starting from $i \in V$. Then

$$y_i = c(\pi(i), i) + \alpha \sum_{j \in V} P(\pi(i), i, j) y_j$$

or

$$y = c_\pi + \alpha P_\pi y \text{ or } y = (I - \alpha P_\pi)^{-1} c_\pi = \sum_{t=0}^{\infty} (\alpha P_\pi)^t c_\pi$$

where $P_\pi(i,j) = P(\pi(i), i, j)$ and $c_\pi(i) = c(\pi(i), i)$.

So policy π can be evaluated.

Theorem 2. *Optimality criterion:*

$$c(\pi(i), i) + \alpha \sum_{j \in V} P(\pi(i); i, j) y_j = \min_{x \in X_i} \left\{ c(x,i) + \alpha \sum_{j \in V} P(x,i,j) y_j \right\} \tag{13}$$

π is optimal iff (13) holds.

Proof Suppose first that (13) does not hold. Define a new policy $\widehat{\pi}$ by

$$c(\widehat{\pi}(i), i) + \alpha \sum_{j \in V} P(\widehat{\pi}(i), i, j) y_j = \min_{x \in X_i} \left\{ c(x,i) + \alpha \sum_{j \in V} P(x,i,j) y_j \right\}$$

We have

$$\begin{aligned} y_i &\geq c(\widehat{\pi}(i), i) + \alpha \sum_{j \in V} P(\widehat{\pi}(i), i, j) y_j \\ \widehat{y}_i &= c(\widehat{\pi}(i), i) + \alpha \sum_{j \in V} P(\widehat{\pi}(i), i, j) \widehat{y}_j \end{aligned} \tag{14}$$

and so

$$(I - \alpha P_{\widehat{\pi}})(y - \widehat{y}) \geq 0$$

and then since $(I - \alpha P_{\widehat{\pi}})^{-1}$ has only non-negative entries:

$$(I - \alpha P_{\widehat{\pi}})^{-1}(I - \alpha P_{\widehat{\pi}})(y - \widehat{y}) \geq 0 \text{ or } y - \widehat{y} \geq 0$$

But $\widehat{y} \neq y$ since there is strict inequality in (14) for at least one i and $\widehat{\pi}$ is strictly better than π.

Conversely, if (13) holds and $\widehat{\pi}$ is any other policy, we get that

$$\begin{aligned} y_i &\leq c(\widehat{\pi}(i), i) + \alpha \sum_{j \in V} P(\widehat{\pi}(i), i, j) y_j \\ \widehat{y}_i &= c(\widehat{\pi}(i), i) + \alpha \sum_{j \in V} P(\widehat{\pi}(i), i, j) \widehat{y}_j \end{aligned}$$

and so

$$(I - \alpha P_{\widehat{\pi}})(y - \widehat{y}) \leq 0$$

and then since $(I - \alpha P_{\widehat{\pi}})^{-1}$ has only non-negative entries:

$$(I - \alpha P_{\widehat{\pi}})^{-1}(I - \alpha P_{\widehat{\pi}})(y - \widehat{y}) \leq 0 \text{ or } y - \widehat{y} \leq 0$$

□

A taxi driver's territory comprises 3 towns A,B,C. If he is in town A he has 3 altrenatives:

1. He can cruise in the hope of picking up a passenger by being hailed.
2. He can drive to the nearest cab stand and wait in line.
3. He can pull over and wait for a radio call.

In town C he has the same 3 alternatives, but in town B he only has alternatives 1 and 2.

The transition probabilities and the rewards for being in the various states and making the various transitions are as follows:

A:

$$P = \begin{bmatrix} .5 & .25 & .25 \\ .0625 & .75 & .1875 \\ .25 & .125 & .625 \end{bmatrix} \quad R = \begin{bmatrix} 10 & 4 & 8 \\ 8 & 2 & 4 \\ 4 & 6 & 4 \end{bmatrix}$$

B:

$$P = \begin{bmatrix} .5 & 0 & .5 \\ .0625 & .875 & .0625 \end{bmatrix} \quad R = \begin{bmatrix} 14 & 0 & 18 \\ 8 & 16 & 8 \end{bmatrix}$$

C:

$$P = \begin{bmatrix} .25 & .25 & .5 \\ .125 & .75 & .125 \\ .75 & .0625 & .1875 \end{bmatrix} \quad R = \begin{bmatrix} 10 & 2 & 8 \\ 6 & 4 & 2 \\ 4 & 0 & 8 \end{bmatrix}$$

He wishes to find the policy which maximises his long run average gain per period.

1.8 Traveling Salesperson problem

We are given a matrix of costs $c(i,j), 1 \leq i,j \leq n$. The problem is to find a permutation π of $[n] = \{1,2,\ldots,n\}$ that minimises

$$TSP(\pi) = c_{1,\pi(1)} + c(\pi(1),\pi^2(1)) + \cdot + c(\pi^n)(1),1).$$

This represents the total cost of a "tour through $[n]$ in the order $1,\pi(1),\pi^2(1),\ldots,\pi^n(1),1$.

There are $(n-1)!$ distinct tours (each tour, as a set of directed edges of $\vec{K}_n$, arises from n distinct permutations.)

With DP we can solve the problem in $O(n^2 2^n)$ time. For $1 \in S \subseteq [n]$ and $x \in S$, let $f(x,S)$ denote the minimum cost of a path that begins at 1, ends at x and visits each vertex in S exactly once. Then, $f(x,S) = 0$ for $S = \{1\}$ and

$$f(x,S) = \min\{f(z, S \setminus \{x\}) + c(z,x) : z \in S \setminus \{x\}\}.$$

There are $\binom{n-1}{k-1}$ choices for $|S| = k$ and given S there are $k-1$ choices for x and then $k-2$ choices for z. So, to compute $f(x,[n])$ for all $1 \neq x \in [n]$ takes time

$$\sum_{k=2}^{n}(k-1)(k-2)\binom{n-1}{k-1} = \sum_{k=3}^{n}(k-1)(k-2)\binom{n-1}{k-1} =$$

$$(n-1)(n-2)\sum_{k=3}^{n}\binom{n-3}{k-3} = (n-1)(n-2)2^{n-3}.$$

To finish we compute $\min\{f(x,[n]) + c(x,1) : x \neq 1)$.

2 Integer Programming

This is the name given to Linear Programming problems which have an extra constraint in that some or all of the variables have to be integer.

2.1 Examples

Capital budgeting A firm has n projects that it would like to undertake but because of budget limitations not all can be selected. In particular project j is expected to produce a revenue of c_j but requires an investment of $a_{i,j}$ in time period i for$i = 1, , ...m$. The capital available in time period i is b_i. The problem of maximising revenue subject to the budget constraints can be formulated as follows: let $x_j = 0/1$ correspond to not proceeding or respectively proceeding with project j then we have to

$$\begin{aligned} \text{Maximise } & \sum_{j=1}^{n} c_j x_j \\ \text{Subject to } & \sum_{j=1}^{n} a_{i,j} x_j \leq b_i, \qquad i = 1, \ldots, m. \\ & 0 \leq x_j \leq 1, \ x_j \text{ integer for } j = 1, 2, \ldots, n. \end{aligned}$$

Depot location We consider here a simple problem of this type: a company has selected m possible sites for distribution of its products in a certain area. There are n customers in the area and the transportation cost of supplying the whole of customer j's requirements over the given planning period from potential site i is $c_{i,j}$. Should site i be developed it will cost f_i to construct a depot there. Which sites should be selected to minimise the total construction plus transport cost?

To do this we introduce variables $y_1, \ldots, y_m$ which can only take values 0 or 1 and correspond to a particular site being not developed or developed respectively. We next define $x_{i,j}$ to be the fraction of customer j's requirements supplied from depot i in a given solution. The problem can then be expressed,

$$\begin{aligned} \text{Minimise } & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} x_{i,j} + \sum_{i=1}^{m} f_i y_i \\ \text{Subject to } & \sum_{i=1}^{m} x_{i,j} = 1, \qquad j = 1, 2, \ldots, n \\ & x_{i,j} \leq y_i, \quad i = 1, 2, \ldots, m, \ j = 1, 2, \ldots, n \\ & x_{i,j} \geq 0, \ 0 \leq y_i \leq 1, y_i \text{ integer}, \qquad i = 1, 2, \ldots, m, j = 1, 2, \ldots, n. \end{aligned}$$

Note that if $y_i = 0$ then $f_i y_i = 0$ and there is no contribution to the total cost. Also, $x_{i,j} \leq y_i$ implies $x_{i,j} = 0$ and no goods are distributed from site i. This corresponds exactly to there not being a depot at location i.

On the other hand, if $y_i = 1$, then $f_i y_i = f_i$ which is the cost of constructing depot i. Also, $x_{i,j} \leq y_i$ becomes $x_{i,j} \leq 1$ which holds anyway from the first constraint.

Set Covering Let $S_1, S_2, \ldots, S_n$ be a family of subsets of a set $S = \{1, 2, \ldots, m\}$. A covering of S is a subfamily S_j for $j \in I$ such that $S = \bigcup_{j \in I} S_i$. Assume that each subset S_j has a cost $c_j > 0$ associated with

it. We define the cost of a cover to be the sum of the costs of the subsets included in the cover.

The problem of finding a cover of minimum cost is of particular practical significance. As an integer program it can be specifiifed as follows: define the $m \times n$ matrix $A = [a_{i,j}]$ by

$$a_{i,j} = \begin{cases} 1 & i \in S_j. \\ 0 & i \notin S_j. \end{cases}$$

Let $x_j, j = 1, 2, \ldots, n$ be 0 /1 variables with $x_j = 1(0)$ to mean set S_j is included (respectively not included) in the cover. The problem is to

$$\begin{aligned} \text{Minimize} & \sum_{j=1}^{n} c_j x_j \\ \text{Subject to} & \sum_{j=1}^{n} a_{i,j} x_j \geq 1, \qquad i = 1, 2, \ldots, m. \\ & x_j = 0 \text{ or } 1, j = 1, 2, \ldots, n. \end{aligned} \tag{15}$$

The m inequality constraints have the following significance: since $x_j = 0$ or 1 and the coefficients $_{i,j}$y are also 0 or 1 we see that $\sum_{j=1}^{n} a_{i,j} x_j$ can be zero only if $x_j = 0$ for all j such that $a_{i,j} = 1$. In other words only if no set S_j, is chosen such that $i \in S_j$. The inequalities are put in to avoid this.

As an example consider the following simplified airline crew scheduling preblem. An airline has m scheduled flight-iegs per week in its current service. A flight-leg being a single flight flown by a single crew e.g. London - Paris leaving Heathrow at 10.30 am. Let $S_j, j = 1, 2, \ldots, n$ be the collection of all possible weekly sets of flight-legs that can be flown by a single crew. Such a subset must take account of restrictions like a crew arriving in Paris at 11.30 am, cannot take a flight out of New York at 12.00 pm. and so if c_j is the cost of set S_j of flight-legs then the problem of minimising cost subject to covering all flight-legs is a set coverirg problem. Note that if crews are not allowed to be passengers on a Flight e.g. so that they can be flown to their next flight, then we have to make (15) an equality – the set partitioning problem.

General terminology The most general problem called the *mixed integer programming problem* can be specified as

$$\begin{aligned} \text{Minimise } & x_0 = \mathbf{c^T x} \\ \text{Subject to } & A\mathbf{x} = \mathbf{b} \\ & x_j \geq 0, \qquad j = 1, 2, \ldots, n. \\ & x_j \text{ integer for } j \in I \end{aligned}$$

where $I \subseteq [n]$.

When $I = [n]$ and all the quantities $c_j, a_{i,j}, b_i$ are integer then we have a *pure integer programming problem.*

Further uses of integer variables
(i) If a variable x can only take a finite number of values $p_1, p_2, \ldots, p_m$, then we can replace x by the expression

$$x = p_1 w_1 + p_2 w_2 + \cdots + p_m w_m, \ w_1 + w_2 + \cdots + w_m = 1, \ w_i = 0 \text{ or } 1 \text{ for } i = 1, 2, \ldots, m.$$

For example X might be the output of a plant which can be small p_1, medium p_2 or large p_3. The cost $c(x)$ of the plant could be represented by $c_1w_1 + e_2w_2 + c_3w_3$ where $c_1$in the cdost of a small plant etc.

(ii) In L.P. one generally consider all constraints to be holding simultaneously. It is possible that the variable might have to satisfy one or other of a set of constraints.

$$0 \leq x \leq M \text{ and } (0 \leq x \leq 1 \text{ OR } x \geq 2).$$

We replace this by

$$x \leq 1 + M(1 - \delta) \text{ and } x \geq 2 - M\delta \text{ and } x \geq 0, \delta = 0/1.$$

Hardness Integer programming problems generally take much longer to solve then the corresponding linear program obtained by ignoring integrality. It is wise therefore to consider the possibility of solving as a straight forward L.P. and then rounding e.g, in the trim-loss problem. This is not always possible for example if x is a $0/1$ variable such that $x = 0$ means do build a plant and $x = 1$ means build a plant then rounding $x1/2$ is not very satisfactory.

2.2 A cutting plane algorithm for the pure problem

The rationale behind this approach is:-

Step 1 Solve the continuous problem as an L.P. i.e. ignore integrality.

Step 2 If by chance the optimal basic variables are all integer then the optimum solution has been found. Otherwise,

Step 3 Generate a cut i.e. a constraint which is satisfied by all integer solutions to the problem but not by the current L.P. solution.

Step 4 Add this new constraint and go to Step 1.

It is straight forward to show that if at any stage the current L.P. Solution $\mathbf{x}$ is integer it is the optimal integer solution. This is because $\mathbf{x}$ is optimal over a region containing all feasible integer solutions. The problem is to define cuts that ensure the convergence of the algorithm in a finite number of steps. The first finite algorithm was devised by R.E, Gomory. It is based on the following construction: let

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

be an equation which is to be satisfied by non-negative integers $x_1, x_2, \ldots, x_n$ and let S be the set of possible integer solutions.

For a real number ξ we define $\lfloor \xi \rfloor$ to be the largest integer which is less than or equal to ξ. Thus $\xi = \lfloor \xi \rfloor + \varepsilon$ where $0 \leq \varepsilon < 1$.

$$\lfloor 6.5 \rfloor = 6, \ \lfloor 3 \rfloor = 3, \ \lfloor -4.5 \rfloor = -5.$$

Now let $a_j = \lfloor a_j \rfloor + f_j$ and $b = \lfloor b \rfloor + f$. Then we have

$$\sum_{j=1}^{n} (\lfloor a_j \rfloor + f_j) x_j = \lfloor b \rfloor + f$$

and hence

$$\sum_{j=1}^{n} f_j x_j - f = \lfloor b \rfloor - \sum_{j=1}^{n} \lfloor a_j \rfloor x_j. \tag{16}$$

Now for $x \in S$, the RHS of (16) is an integer and the LHS is at least $-f > -1$. This implies that

$$\sum_{j=1}^{n} f_j x_j - f \in \{0, 1, \ldots\}.$$

Suppose now that one has solved the LP relaxation and the solution is not integer. Therefore there is a basic variable x_i with

$$x_i + \sum_{j \notin I} b_{i,j} x_j = b_{i,0}$$

where $b_{i,0}$ is not an integer. (Here I is the set of indices of basic variables and the $b_{i,j}$ are the coefficients of the simplex tableaux.)

Putting $f_i = b_{i,j} - \lfloor b_{i,j} \rfloor$ for $j \notin I$ and $f = b_{i,0} - \lfloor b_{i,0} \rfloor$ we see that

$$\sum_{j \notin I} f_j x_j > f \tag{17}$$

for all integer solutions to our problem.

Now $f > 0$ since $b_{i,0}$ is not an integer and so (17) is not satisfied by the current L.P. solution since $x_j = 0$ for $j \notin I$ and so (17) is a cut.

The initial continuous problem solved by the algorithm is the L.P. problem obtained by ingoring integrality.

Statement of the Algorithm

Step 1 Solve current continuous problem.

Step 2 If the solution is integral it is the optimal integer solution, otherwise.

Step 3 Choose a basic variable x_i, which is currently non-integer, construct the corresponding constraint (17) and add it to the problem. Go to step 1.

We note that the tableau obtained after adding the cut is dual feasible and so the dual simplex algorithm can be used to re-optimise.

Example:

$$\begin{aligned}\text{Maximise } & x_1 + 4x_2 \\ \text{Subject to } & 2x_1 + 4x_2 \leq 7. \\ & 10x_1 + 3x_2 \leq 14. \\ & x_1, x_2 \geq 0 \text{ and integer.}\end{aligned}$$

$B.V.$	x_1	x_2	x_3	x_4	ξ_1	ξ_2	RHS	
x_0	-1	-4					0	
x_3	2	**4**	1				7	
x_4	10	3		1			14	
x_0	1		1				7	
x_2	1/2	1	1/4				7/4	cut made from this row
x_4	17/2		$-3/4$	1			35/4	
ξ_1	$\mathbf{-1/2}$		$-1/4$		1		$-3/4$	
x_0			1/2		2		11/2	
x_2		1			1		1	
x_4			$\mathbf{-5}$	1	17		-4	
x_1	1		1/2		-2		3/2	
x_0				1/10	37/10		51/10	cut made from this row
x_2		1			1		1	
x_3			1	$-1/5$	$-17/5$		4/5	
x_1	1			1/10	$-3/10$		11/10	
ξ_2				$\mathbf{-1/10}$	$-7/10$	1	$-1/10$	
x_0					3		5	
x_2		1			1		1	
x_3			1		-1	-2	1	
x_1					-1	1	1	
x_4					7	-10	1	

- One can show that the Gomory cuts $\sum_j f_j > f$ when expressed in terms of the original non-basic variables have the form $\sum_j w_j x_j \leq W$ where the w_j, W are integer and the value of $\sum_j w_j x_j$ after solving the current continuous problem is $W + \varepsilon$ where $0 < \varepsilon < 1$ assuming the current solution is non-integer, Thus the cut is obtained by moving a hyperplane parallel to itself to an extent which cannot exclude an integer solution. It is worth noting that the plane can usually be moved further without excluding integer points thus generating deeper cuts. For a discussion on how this can be done see the reference given for integer programming,

- After adding a cut and carrying out one iteration of the dual simplex algorithm the slack variable corresponding to this cut becomes non-basic, If during a succeeding iteration this slack variable becomes basic then it may be discarded along with its current row without affecting termination. This means that the tableau never has more than $n + 1$ rows or $m + n$ columns.

- A valid cut can be generated from any row containing a non-integral variable, One strategy is to choose the variable with the largest fractional part as this helps' to produce a "large' change in the objective

valve. It is interesting that finitness of the algorithm has not been proved for this strategy although finitness has been proved for the strategy of always choosing the 'topmost' row the tableau with a non-integer variable.

- The behaviour of this algorithm has been erratic. It has for example worked well on set covering problems but in other cases the algorithm has to be terminated because of excessive use of computer time, This raises an important point; if the algorithm is stopped prematurely then one does not have a good sub-optimal solution to use. Thus in some sense the algorithm is unreliable,

3 Branch and bound

We consider the problem P_0:

$$\text{Minimize } f(x) \text{ subject to } x \in S_0.$$

Here S_0 is our set of feasible solutions and $f : S_0 \to \Re$.

As we proceed in Branch-and-Bound we create a set of sub-problems $\mathcal{P}$. A sub-problem $P \in \mathcal{P}$ is defined by *the description of* a subset $S_P \subseteq S_0$. We also keep a *lower bound* b_P where

$$b_P \le \min\{f(x) : \ x \in S_P\}.$$

At all times we act as if we have $x^* \in S_0$, some known feasible solution to P_0 and $v^* = f(x^*)$. If we do not actually have a solution x^* then we let $v^* = -\infty$. We will have a procedure BOUND that computes b_P for a sub-problem P. In many cases, BOUND *sometimes* produces a solution $x_P \in S_0$ and sometimes determines that $S_P = \emptyset$.

We initialize $\mathcal{P} = \{P_0\}$.

Branch and Bound:

Step 1 If $\mathcal{P} = \emptyset$ then x^* solves the problem.

Step 2 Choose $P \in \mathcal{P}$. $\mathcal{P} \leftarrow \mathcal{P} \setminus \{P\}$.

Step 3 **Bound:** Run BOUND(P) to compute b_P.

Step 4 If $S_P = \emptyset$ or $b_P \ge v^*$ then we consider P to be solved and go to Step 1.

Step 5 If BOUND generates $x_P \in S_0$ and $f(x_P) < v^*$ then we update, $x^* \leftarrow x_P, v^* \leftarrow f(x_P)$.

Step 6 **Branch:** Split P into a number of subproblems $Q_i, i = 1, 2, \ldots, \ell$, where $S_P = \bigcup_{i=1}^{\ell} S_{Q_i}$. And $S_{Q_i} \neq S_P$ is a strict subset for $i = 1, 2, \ldots, \ell$.

Step 7 $\mathcal{P} \leftarrow \mathcal{P} \cup \{Q_1, Q_2, \ldots, Q_\ell\}$.

Assuming S_0 is finite, this procedure will eventually terminate with $\mathcal{P} = \emptyset$. This is because the feasible sets S_P are getting smaller and smaller as we branch.

Most often the procedure BOUND has the following form: while it may be difficult to solve P directly, we may be able to find $T_P \supseteq S_P$ such that there is an efficient algorithm that determines whether or not $T_P = \emptyset$ and finds $\xi_P \in T_P$ that minimizes $f(\xi), \xi \in T_P$, if $T_P \neq \emptyset$. In this case, $b_P = f(\xi_P)$ and Step 5 is implemented if $\xi_P \in S_0$. We call the problem of minimizing $f(\xi), \xi \in T_P$, a *relaxed problem.*

Examples:

Ex. 1 Integer Linear Programming. Here S_P is the set of integer solutions and T_P is the set of solutions, if we ignore integrality. The procedure BOUND solves the linear program. If the solution ξ_P is not integral, we choose a variable x, whose value is $\zeta \notin \mathbb{Z}$ and form 2 sub-problems by adding $x \le \lfloor z \rfloor$ to one and $x \ge \lceil z \rceil$ to the other.

Ex. 2 Traveling Salesperson Person Problem (TSP): Here S_P is the set of tours i.e. single directed cycles that cover all the vertices. We can take T_P to be the set of collections of vertex disjoint directed cycles that cover all the vertices. More precisely, to solve the TSP we must minimise $\sum_{i=1}^{n} C(I, \pi(i))$ as π ranges over all cyclic permutations. Our relaxation is to minimise $\sum_{i=1}^{n} C(I, \pi(i))$ as π ranges over all permutations, i.e. the assignment problem. We branch as follows. Suppose that the assignment solution consists of cycles $C_1, C_2, \ldots, C_k, k \geq 2$. Choose a cycle, C_1 say. Suppose that $C_1 = (v_1, v_2, \ldots, v_r)$ as a sequence of vertices. Then in Q_1 we disallow $\pi(v_1) = v_2$, in Q_2 we insist that $\pi(v_1) = v_2$, but that $\pi(v_2) \neq v_3$, in Q_3 we insist that $\pi(v_1) = v_2$, $\pi(v_2) = v_3$, but that $\pi(v_3) \neq v_4$ and so on.

Ex. 3 Implicit Enumeration: Here the problem is

$$\text{Minimize } \sum_{j=1}^{n} c_j x_j \text{ subject to } \sum_{j=1}^{n} a_{i,j} x_j \geq b_i,\ i \in [m], \quad x_j \in \{0,1\}, j \in [n].$$

A sub-problem is assciated with two sets $I, O \subseteq [n]$. This the sub-problem $P_{I,O}$ where we add the constraints $x_j = 1, j \in I, x_j = 0, j \in O$. We also check to see if $x_j = 1, j \in I, x_j = 0, j \notin I$ gives an improved feasible solution. As a bound $b_{I,O}$ we use $\sum_{j \notin O} \max\{c_j, 0\}$. To test feasibility we check that $\sum_{j \notin O} \max\{a_{i,j}, 0\} \geq b_i, i \in [m]$. To branch, we split $P_{I,O}$ into $P_{I \cup \{j\},O}$ and $P_{I,O \cup \{j\}}$ for some $j \notin I \cup O$.

4 Combinatorial Optimization

4.1 Shortest path

4.1.1 Non-negative lengths

We are given a digraph $D = ([n], E)$ with vertex set $[n]$. Let $\mathcal{P}$ denote the set of paths in D and let $\ell : \mathcal{P} \to \Re$. Think initially that there are edge lengths $\ell : E \to \Re_+$ and that

$$\ell(P) = \ell_{reg}(P) = \sum_{e \in P} \ell(e).$$

Dijstra's Algorithm:
begin
for $i = 2, \ldots, n$, $d(i) \leftarrow \ell(1, i)$, $P_i \leftarrow (1, i)$; $S_1 \leftarrow \{1\}$;
    for $k = 2, \ldots, n$ **do**;
    begin
        $d(i) = \min\{d(j) :\notin S_k\}$;
        $S_{k+1} \leftarrow S_k \cup \{i\}$;
        for $j \notin S_{k+1}$ **do**
        if $d(j) > \ell(P_i, j)$ then $d(j) \leftarrow \ell(P_i, j)$, $P_j \leftarrow (P_i, j)$;
    end
end

Lemma 3. *On termination of Dijstra's Algorithm, $d(i) = \ell(P_i)$ is the minimum length of a path from 1 to i, for all i*

Proof. At each stage we can verify by induction on k that for each $i \notin S_k$, $d(i)$ is the minimum length of a path from 1 to i for which all vertices but i are in S_k. If true for k then when we add vertex i we simply update the d's correctly.

Suppose that i is added at Step r. Let $P = (x_0 = 1, x_2, \ldots, x_m = i)$ be a path from 1 to i. Suppose that $x_0, x_1, \ldots, x_{l-1} \in S_{r-1}$ and $x_l \notin S_{r-1}$. Then,

$$\ell(P) \geq \ell(x_0, x_1, \ldots, x_l) \geq d(x_l) \geq d(i) = \ell(P_i).$$

□

Note now that all we have assumed about ℓ is that

$$P = (P_1, P_2) \text{ implies } \ell(P_1) \leq \ell(P). \tag{18}$$

In which case, we can apply the algorithm to solve problems where path length is defined as follows:

Time dependent path lengths: Suppose edge $e = (x, y)$ has two parameters $a_e, b_e \geq 0$ and that if we start a walk at time 0 and arrive at x at time t then the edge length is $a_e + b_e t$. Suppose that $P = (e_0, e_1, \ldots, e_k)$ as a sequence of edges and that $P_i = (e_0, e_1, \ldots, e_i)$. Then we now have $\ell(P_0) = a_{e_0}$ and $\ell(P_i) = a_{e_i} + b_{e_i}\ell(P_{i-1})$.

Visit S in a fixed order: S is a set of vertices and feasible paths must visit S in some fixed order. Individual edge lengths are non-negative. Then

$$\ell(P) = \begin{cases} \ell_{reg}(P) & P \cap S \text{ visited in correct order.} \\ \infty & Otherwise. \end{cases}$$

Avoid S: S is a set of vertices and there is a penalty of $f(k)$ for visiting S, k times. Here $f(k)$ is monotone increasing in k. Individual edge lengths are non-negative. Then $\ell(P) = \ell_{reg}(P) + f(|V(P) \cap S|$.

4.2 No negative cycles

Our algorithms find shortest walks between vertices. When there are no negative length cycles, this amounts to finding shortest paths. When there are negative cycles, there will sometimes be no minimum length walk.

Suppose first that P is a path that begins at vertex 1 and x is an arbitrary vertex. Then we define

$$P * x = \begin{cases} (P, x) & x \notin P. \\ P(1, x) & x \in P. \end{cases}$$

Here $P(1, x)$ is the subpath of P from 1 to x.

Assumption: Suppose that P, Q are paths from vertex 1 to vertex y. Suppose that $x \notin P$ and that $\ell(Q) \le \ell(P)$. Then $\ell(Q * x) \le \ell(P, x)$.

Putting $P = Q$ we see that when $\ell = \ell_{reg}$ this requires $\ell(C) \ge 0$ for a cycle C. Here is an example where $\ell = \ell_{reg}$ and there are no negative cycles.

Electric cars: Suppose when we drive along edge e, $\ell(e)$ the amount of energy used is $\ell(e)$. This is normally positive, but when going down hill it can be negative. In this scenario, there can be no negative cycles under ℓ_{reg}.

Assume that the edges of D are $E = \{e_i = (x_i, y_i), i = 1, 2, \ldots, m\}$. Let $P_i, i = 1, 2, \ldots, n$ be a collection of paths, where $P_1 = (1)$ and P_i goes from 1 to i.

Lemma 4. *The following is a necessary and sufficient condition for $P_1, P_2, \ldots, P_n$ to be a collection of shortest paths with start vertex 1:*

$$\ell(P_y) \le \ell(P_x * y) \text{ for all } (x, y) \in E. \tag{19}$$

Proof. It is clear that (19) is necessary. If it fails then $P_x * y$ is "shorter" than P_y.

Suppose that (4) holds. Let $P = (1 = x_0, x_1, x_2, \ldots, x_k = i)$ be a path from 1 to i. We show by induction on j that

$$\ell(P_{x_j}) \le \ell(P(1, x_1, x_2, \ldots, x_j)). \tag{20}$$

Now when $j = 0$, both sides of (20) are zero. Then if it holds for some $j \ge 0$ then (19) and the inductive assumption imply that

$$\ell(P_{x_{j+1}}) \le \ell(P_{x_j} * x_{j+1}) \le \ell(P(1, x_1, \ldots, x_{j+1})).$$

Thus (19) is sufficient. □

Ford's Algorithm:
begin
for $i = 2, \ldots, n$, $d(i) \leftarrow \ell(1, i)$, $P_i \leftarrow (1, i)$;
    repeat;
    $flag \leftarrow 0$;
    for $i = 1, 2, \ldots, m$;
    begin
        if $\ell(P_{y_i}) > \ell(P_{x_i} * y_i)$ **then**;
        begin;
            $P_{y_i} \leftarrow (P_{x_i} * y_i)$; $flag \leftarrow 1$;
        end;
    end;
    until $flag = 0$;
    end
end

Lemma 5. *Ford's algorithm terminates after at most n rounds with a collection of shortest paths.*

Proof. If the algorithm terminates then because $flag = 0$ at this point, we have that (57) holds. Thus we have shortest paths.

We now argue that if the minimum number of arcs in a shortest path from 1 to i has ν_i edges then P_i is correct after ν_i rounds. We argue by induction. This is true for $i = 1$ and $\nu_i = 0$. Suppose that it is true for all i such that $\nu_i \leq \nu$ and that vertex j satisfies $\nu_j = \nu + 1$. Let $P = (1 = x_0, x_1, \ldots, x_{\nu+1} = j)$ be a shortest path from 1 to j. Then, by induction, after ν rounds P_{x_ν} is a shortest path from 1 to x_ν and then after one more round P_j is correct. □

4.3 Digraphs without circuits

These are important, not least because they occur in Critical Path Analysis. Their application in this area involves computing longest paths.

4.3.1 Topological Ordering

Let the vertices of a digraph $D = ([n], E)$ be ordered $v_1, v_2, \ldots, v_n$. This ordering is *topological* if $(v_i, v_j) \in E$ implies that $i < j$.

Lemma 6. *Digraph D has a topological ordering if and only if D has no directed circuits.*

Proof. Suppose first that $v_1, v_2, \ldots, v_n$ is a topological ordering and that D has a directed cycle $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$. then we have $i_1 < i_2 < \cdots < i_k < i_1$, contradiction.

Conversely, suppose there are no directed circuits. Let $P = (x_1, x_2, \ldots, x_k)$ be a longest path in D. Then x_k is a *sink* i.e. there are no directed edges (x_k, y). (If $y \in X = \{x_1, x_2, \ldots, x_{k-1}\}$ then D contains a circuit. If $y \notin X$ then (P, x) is longer than P.)

To get a topological ordering, we let $v_n = x_k$ and inductively order the subgraph H induced by $[n] \setminus \{v_n\}$. This is a topological ordering. If $(v_i, v_j) \in E(H)$ then $i < j$ because H is toplologically ordered. Any other edge must be of the form (v_i, v_n). □

To solve the longest path problem for paths starting at v_1, we take a topological ordering and then compute $d(v_1) = 0$ and then for $j \geq 2$,

$$d(v_j) = \max \{d(v_i) + \ell(v_i, v_j) : i < j \text{ and } (v_i, v_j) \in E\}. \tag{21}$$

Lemma 7. *Equation* (21) *computes the value of a longest path from* v_1 *to every other vertex.*

Proof. That $d(v_j)$ is correct follows by induction on j. It is trivially true for $j = 0$ and then for $j > 0$ we use the fact if $P = (x_1 = v_1, x_2, \ldots, x_k = v_j)$ is a longest path from v_1 to v_j then (i) $x_{k-1} = v_l$ for some $l < j$ and (ii) $(x_1, x_2, \ldots, x_{k-1})$ is a longest path from v_1 to v_l and (iii) $\ell(P) = d(v_l) + \ell(v_l, v_j)$. □

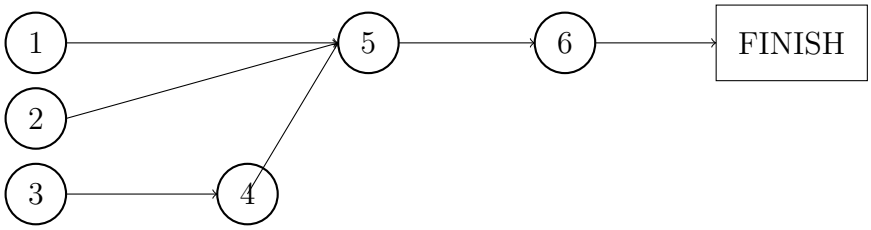
Critical Path Analysis: Imagine that a project consists of n activites.

Making a cup of tea:

1. Get a cup from the cupboard.
2. Get a tea bag.
3. Fill the kettle with water.
4. Boil the water.
5. Pour water into cup.
6. Allow to brew.

We define a digraph with n vertices, one for each activity and an edge (i, j) if (i) activity j cannot start until acivity i has been completed but (ii) only include (i, j) if it is not implied by a path (i, k, j). Each edge (i, j) has a length equal to the estimated duration of the activity i.

Tea Digraph:



Associate a time t_i to start activity i. Then t_i is the length of the longest path to vertex i. The estimated completion time of the project is then the length of the longest path to FINISH.

5 Assignment Problem

A *matching* M in a graph is a set of vertex disjoint edges. A vertex v is covered by M if there exists $e \in M$ such that $v \in e$. A matching M is *perfect* if every vertex of G coverd by M. For the complete bipartite graph $K_{A,B}$ on vertex set $A = \{a_i : i \in [n]\}, B = \{b_i : i \in [n]\}$, perfect matchings can be represented by permutations of n i.e $M = \{(a_i, b_{\pi(i)}) : i \in [n]\}$. Given a cost matrix $(c(i,j)$, the cost of a perfect matching $M = M(\pi)$ be given by

$$c(M) = \sum_{i=1}^{n} c(i, \pi(i)).$$

The assignment problem is that of finding a perfect matching of minimum cost.

5.1 Alternating paths

Given a matching M, a path $P = (e_1, e_2, \ldots, e_k)$ (as a sequence of edges) is *alternating* if the edges alternate between being in M and not in M.

An alternating path is *augmenting* if it begins and ends at uncovered vertices. If P is augmenting with respect to matching M, then $M' = M \oplus P$ is also a matching and $|M'| = |M| + 1$.

5.2 Successive shortest path algorithm

The algorithm produces a sequence $M_1, M_2, \ldots, M_n$ where M_k is a minimum cost matching from $[k]$ to $[k]$. It begins with $M_1 = (1,1)$.

Suppose that $k > 1$ and that we have constructed $M_{k-1} = \{(a_i, b_{\pi(i)}) : i = 1, 2, \ldots, k-1\}$. The graph Γ_k is the complete graph K_{A_k,B_k}. The digraph $\vec{\Gamma}_k$ on vertex set $A_k = \{a_i : i \in [k]\}$, $B_k = \{b_i : i \in [k]\}$ is defined as follows. The directed edges are $X = \{(b_{\pi(i)}, a_i) : i \in [k-1]\}$ and $Y = \{(a_i, b_j) : i \in [k], j \in [k], j \neq \pi(i)\}$. The edge $(b_{\pi(i)}, a_i) \in X$ is given length $-c(i, \pi(i))$ and the edge $(i,j) \in Y$ is given length $c(i,j)$.

We observe the following:

- If M is a perfect matching of Γ_k then $M \oplus M_{k-1}$ consists of a collection $C_1, \ldots, C_p$ of vertex disjoint alternating cycles plus an augmenting path from a_k to b_k.
- $$c(M) - c(M_{k-1}) = \sum_{i=1}^{p} \ell(C_i) + \ell(P)$$
  where length ℓ is defined with respect to $\vec{\Gamma}_k$.
- $\ell(C_i) \geq 0$ for all i. Otherwise $M_{k-1} \oplus C_i$ is a matching of Γ_{k-1} with a cost $c(M_{k-1}) + \ell(C_i) < c(M_{k-1})$.

It follows from the above that to find a minimum cost matching of Γ_k, we should find a shortest path in $\vec{\Gamma}_k$ from a_k to b_k. Second, because $\vec{\Gamma}_k$ has no negative circuits, we can apply Ford's algorithm to find this path.

5.3 Linear Programming Solution – Hungarian Algorithm

Consider the linear program ALP:

$$\begin{aligned} &\text{Minimize} && \textstyle\sum_{i=1}^n \sum_{j=1}^n c_{i,j} x_{i,j} && (22)\\ &\text{Subject to} \\ &&& \textstyle\sum_{j=1}^n x_{i,j} = 1 \qquad \text{for } i = 1, 2, \ldots, n. && (23)\\ &&& \textstyle\sum_{i=1}^n x_{i,j} = 1 \qquad \text{for } j = 1, 2, \ldots, n. && (24)\\ &&& x_{i,j} \geq 0 \qquad \text{for } i, j = 1, 2, \ldots, n. && (25) \end{aligned}$$

The assignment problem is the solution to ALP where we replace (25) by

$$x_{i,j} = 0 \text{ or } 1 \text{ for } i, j = 1, 2, \ldots, n. \tag{26}$$

This is because (23), (24) force the set $\{(i, j) : x_{i,j} = 1\}$ to be a perfect matching and (22) is then the cost of this matching.

In general replacing non-negativity constraints (25) by *integer* contraints (26) makes an LP hard to solve. Not however in this case.

The dual of ALP is the linear program DLP:

$$\begin{aligned} &\text{Maximize} && \textstyle\sum_{i=1}^n u_i + \sum_{j=1}^n v_j && (27)\\ &\text{Subject to} \\ &&& u_i + v_j \leq c(i, j) \qquad \text{for } i, j = 1, 2, \ldots, n. && (28) \end{aligned}$$

The *primal-dual* algorithm that we describe relies on *complimentary slackness* to find a solution.

Complimentary Slackness: If a feasible solution **x** to ALP and a feasible solution **u,v**, to DLP satisfy

$$x_{i,j} > 0 \text{ implies that } u_i + v_j = c(i, j). \tag{29}$$

then **x** solves ALP and **u,v**, solves DLP. For then

$$0 = \sum_{i=1}^n \sum_{j=1}^n (c(i, j) - u_i - v_j) x_{i,j} = \sum_{i=1}^n \sum_{j=1}^n c_{i,j} x_{i,j} - \left(\sum_{i=1}^n u_i + \sum_{j=1}^n v_j \right), \tag{30}$$

and the two solutions have the same objective value.

(We have used $\sum_{i=1}^n u_i \sum_{j=1}^n x_{i,j} = \sum_{i=1}^n u_i$, which follows from (23) etc.)

The steps of the Primal-Dual algorithm are as follows:

Step 1 Choose an initial dual feasible solution. E.g. $v_j = 0, j \in [n]$ and $u_i = \min_j c(i, j)$.

Step 2 Given a dual feasible solution, **u,v**, define the graph $K_{\mathbf{u},\mathbf{v}}$ to be the bipartite graph with vertex set A, B and an edge (i, j) whenever $u_i + v_j = c(i, j)$.

Step 3 Find a maximum size matching M in $K_{\mathbf{u},\mathbf{v}}$.

Step 4 If M is perfect then (29) holds and M provides a solution to the assignment problem.

Step 5 If M is not perfect, update **u,v** and go to Step 3.

To carry out Step 3, we proceed as follows:

Step 3a Begin with an arbitrary matching M of $K_{\mathbf{u},\mathbf{v}}$.

Step 3b Let A_U denote the set of vertices in A not covered by M.

Step 3c Let $\vec{K}_{\mathbf{u},\mathbf{v}}$ be the digraph obtained from $K_{\mathbf{u},\mathbf{v}}$ by orienting matching edges from B to A and other edges from A to B.

Step 3d Let A_M, B_M denote the set of vertices in A, B that are reachable by a path in $\vec{K}_{\mathbf{u},\mathbf{v}}$ from A_U. Such paths are necessarily alternating.

Step 3e If there is a vertex $b \in B_M$ that is not covered by M then there is an augmenting path P from some $a \in A_U$ to v. In this case we use P to construct a matching M' with $|M'| > |M|$. We then go to Step 3b, with M replaced by M'. Otherwise, Step 3 is finished.

To carry out Step 5, we assume that we have finished Step 3 with M, A_M, B_M. We then let

$$\theta = \min\{c_{i,j} - u_i - v_j : a_i \in A_M, b_j \notin B_M\} > 0.$$

We know that $\theta > 0$. Otherwise, if a_i, b_j is the minimising pair, then we should have put $b_j \in B_M$.

We then amend **u,v** to $\mathbf{u}^*, \mathbf{v}^*$ via

$$u_i^* = \begin{cases} u_i + \theta & a_i \in A_M. \\ u_i & \textit{Otherwise.} \end{cases} \text{ and } v_j^* = \begin{cases} v_j - \theta & j \in B_M. \\ v_j & \textit{Otherwise.} \end{cases}$$

Observe the following:

1. $\mathbf{u}^*, \mathbf{v}^*$ is feasible for DLP. $u_i^* + v_j^* \le u_i + v_j$ except for the case where $a_i \in A_M, b_j \notin B_M$ and θ is chosen so that the increase maintains feasiblity.
2. If $b \in B_M$ for the pair **u,v** then it will stay in B_M when we replace **u,v** by $\mathbf{u}^*, \mathbf{v}^*$. This is because there is a path $P = (a_{i_1} \in A_U, b_{i_1}, \ldots, a_{i_k}, b_{i_k} = b)$ such that each edge of P contains one vertex in A_M and one vertex in B_M. Hence the sum $u_i + v_j$ is unchanged for edges along P.
3. A vertex $b \notin B_M$ contained in a pair that defines θ will be in B_M when we replace **u,v** by $\mathbf{u}^*, \mathbf{v}^*$.

In summary: if we reach Step 4 with a perfect matching then we have solved ALP. After at most n changes of **u,v** in Step 5, the size of M increases by at least one. This is because updating **u,v** increases B_M by at least one. Thus the algorithm finishes in $O(n^4)$ time. ($O(n^3)$ time if done carefully.)

6 Matroids and the Greedy Algorithm

Given a *ground set* X, an *independence system* on X is collection of subsets $\mathcal{I} = \{I_1, I_2, \ldots, I_m\}$ such that

$$I \in \mathcal{I} \text{ and } J \subseteq I \text{ implies that } J \in \mathcal{I}. \tag{31}$$

Examples

Ex. 1 The set $\mathcal{M}$ of matchings of a graph $G = (V, X)$.

Ex. 2 The set of (edge-sets of) forests of a graph $G = (V, X)$.

Ex. 3 The set of *stable* sets of a graph $G = (X, E)$. We say that S is stable if it contains no edges.

Ex. 4 The set of solutions to the $\{0, 1\}$-knapsack problem. Here we are given positive integers $w_1, w_2, \ldots, w_n, W$ and $X = [n]$ and $\mathcal{I} = \left\{S \subseteq [n] : \sum_{i \in S} w_i \leq W\right\}$.

Ex. 5 Let $\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_n}$ be the columns of an $m \times n$ matrix $\mathbf{A}$. Then $X = [n]$ and $\mathcal{I} = \{S \subseteq [n] : \{\mathbf{c_i}, \mathbf{i} \in \mathbf{S}\}$ are lin

An independence system is a *matroid* if whenever $I, J \in \mathcal{I}$ with $|J| = |I| + 1$ there exists $e \in J \setminus I$ such that $I \cup \{e\} \in \mathcal{I}$. Only Ex. 2 and 5 above are matroids. To check Ex. 5, let $\mathbf{A}_I$ be the $m \times |I|$ sub-matrix of $\mathbf{A}$ consisting of the columns in I. If there is no $e \in J \setminus I$ such that $I \cup \{e\} \in \mathcal{I}$ then $\mathbf{A}_J = \mathbf{A}_I\mathbf{M}$ for some $|I| \times |J|$ matrix. But then

$$|J| = rank(\mathbf{A}_J) \leq \min\{rank(\mathbf{A}_I), rank(\mathbf{M})\} \leq |I|,$$

contradiction.

To check Ex. 2 we can argue (exercise) that $I \subseteq E$ defines a forest if and only if the columns corresponding to I in the vertex-edge incidence matrix $\mathbf{M}_G$ are linearly independent.
($\mathbf{M}_G$ has a row for each vertex of G and a column for each edge of G. The column $\mathbf{c_e}, \mathbf{e} = \{\mathbf{x}, \mathbf{y}\}$ has a one in row x and a -1 in row y and a zero in all other rows. It doesn't matter which of the two endpoints is viewed as x.)

6.1 Greedy Algorithm

Suppose that each $e \in E$ is given a weight w_e and that the weight $w(I)$ of an independent set I is given by $w(I) = \sum_{e \in I} c_e$. The problem we discuss is

$$\text{Maximize } w(I) \text{ subject to } I \in \mathcal{I}.$$

Greedy Algorithm:
begin
  Sort $E = \{e_1, e_2, \ldots, e_m\}$ so that $w(e_i) \geq w(e_{i+1})$ for $1 \leq i < m$;
  $S \leftarrow \emptyset$;
  for $i = 1, 2, \ldots, m$;
  begin
    if $S \cup \{e_i\} \in \mathcal{I}$ **then**;
    begin;
      $S \leftarrow S \cup \{e_i\}$;
    end;
  end;
end

Theorem 8. *The greedy algorithm finds a maximum weight independent set* **for all choices of** w *if and only if it is a matroid.*

Proof. Suppose first that the Greedy Algorithm always finds a maximum weight independent set. Suppose that $\emptyset \neq I, J \in \mathcal{I}$ with $|J| = |I| + 1$. Define

$$w(e) = \begin{cases} 1 + \frac{1}{2|I|} & e \in I. \\ 1 & e \in J \setminus I. \\ 0 & e \notin I \cup J. \end{cases}$$

If there does not exist $e \in J \setminus I$ such that $I \cup \{e\} \in \mathcal{I}$ then the Greedy Algorithm will choose the elements of I and stop. But I does not have maximum weight. Its weight is $|I| + 1/2 < |J|$. So if Greedy succeeds, then (31) holds.

Conversely, suppose that our independence system is a matroid. We can assume that $w(e) > 0$ for all $e \in E$. Otherwise we can restrict ourselves to the matroid defined by $\mathcal{I}' = \{I \subseteq E^+\}$ where $E^+ = \{e \in E : w(e) > 0\}$.

Suppose now that Greedy chooses $I_G = e_{i_1}, e_{i_2}, \ldots, e_{i_k}$ where $i_t < i_{t+1}$ for $1 \leq t < k$. Let $I = e_{j_1}, e_{j_2}, \ldots, e_{j_\ell}$ be any other independent set and assume that $j_t < j_{t+1}$ for $1 \leq t < \ell$. We can assume that $\ell \geq k$, for otherwise we can add something from I_G to I to give it larger weight. We show next that $k = \ell$ and that $i_t \leq j_t$ for $1 \leq t \leq k$. This implies that $w(I_G) \geq w(I)$.

Suppose then that there exists t such that $i_t > j_t$ and let t be as small as possible for this to be true. Now consider $I = \{e_{i_s} : s = 1, 2, \ldots, t-1\}$ and $J = \{e_{j_s} : s = 1, 2, \ldots, t\}$. Now there exists $e_{j_s} \in J \setminus I$ such that $I \cup \{e_{j_s}\} \in \mathcal{I}$. But $j_s \leq j_t < i_t$ and Greedy should have chosen e_{j_s} before choosing $e_{i_{t+1}}$. Also, $i_k \leq j_k$ implies that $k = \ell$. Otherwise Greedy can find another element from $I \setminus I_G$ to add. □

7 Two person zero-sum games

We discuss here an application of linear programming to the theory of games. This theory is an attempt to provide an analysis of situations involving conflict and competition.

Game 1: There are two players A and B and to play the game they each choose a number 1,2,3 or 4 without the other's knowledge and then they both simultaneously announce their numbers. If A calls i and B calls j then B pays A $a_{i,j}$ – the *payoff* – given in the matrix below. (if $a_{i,j} < 0$, this is equivalent of A paying B $-a_{i,j}$.)

$$\begin{bmatrix} 2 & 4 & 2 & 1 \\ -2 & 5 & 1 & -1 \\ 1 & -5 & 3 & 0 \\ 6 & 2 & -3 & -2 \end{bmatrix}$$

This is a *two person zero-sum game*, zero sum because the algebraic sum of the player's winnings is always zero.

Game 2: (Penalty kicks) Suppose that A and B play the following game of soccer. A plays in goal and B takes penalty kicks. B can kick the ball into the left hand corner, the right hand corner or into the midle. If A guesses correctly where B will kick then A will make a save. The payoff to A is given by the following matrix.

$$\begin{bmatrix} & KR & KL & KM \\ DR & 2 & -1 & -2 \\ DL & -1 & 2 & -2 \\ M & -1 & -1 & -1 \end{bmatrix}$$

We will be considering $m \times n$ generalisations of Game 1 and other games like Game 2 that can be reduced to this form.

Thus there is given some $m \times n$ payoff $\mathbf{A}$. In a *play* of the game, A chooses $i \in M - \{1, 2, \ldots, m\}$ and B chooses $j \in N = \{1, 2, \ldots, n\}$. These choices are made independently without either player knpowing what the other has chosen. They then announce their choices and B pays $a_{i,j}$ to A.

M, N will be referred to as the sets of *tactics* for A,B respectively.

A *match* is an unending sequence of plays. A's objective is to maximize her expected winnings from the match and B's objective is to minimize his expected loss.

A *strategy* for the match is some rule for selecting the tactic for the next play.

Let S_A, S_B be sets of strategies for A, B respectively. We shall initially consider the case where $S_A = \{(1), \ldots, (m)\}$ and $S_B = \{(1), \ldots, (n)\}$ where (t) is the *pure* strategy of using tactic t in each play We Shall subsequently be enlarging S_A and S_B and we therefore introduce new notation to allow for this possibility.

Thus for each $u \in S_A$ and $v \in S_B$ let $PAY(u, v)$ denote the average payment of B to A.

Stable Solutions $u_0, v_0) \in S_A \times S_B$ is a *stable solution* if

$$PAY(u, v_0) \leq PAY(u_0, v_0) \leq PAY(u_0, v) \tag{32}$$

holds for all u, v.

If (32) holds then neither A nor B has any incentive to change strategy if each assumes his opponent is not going to change his or hers.

The subsequent analysis is concerned with finding a stable solution.

Thinking of S_A as the row indiceses and S_B as the column indices of some matrix we define

$$ROWMIN(u) = \min_{v \in S_B} PAY(u,v), \qquad u \in S_A.$$
$$COLMAX(v) = \max_{u \in S_A} PAY(u,v), \qquad v \in S_B.$$

Suppose now that A chooses u. We assume that after some finite time, B will be able to deduce this choice. B will then choose his strategy v to minimize $PAY(u,v)$. Thus if A chooses u then she can expect her average winnings to be $ROWMIN(u)$.

Similarly if B chooses v he can expect his average losses to be $COLMAX(v)$.

Thus if $P_A = ROWMIN(u_0) = \max_{u \in S_A} ROWMIN(u)$ and $P_B = COLMAX(v_0) = \min_{v \in S_B} COLMAX(v)$ then A can by choosing u_0 ensure that her average winnings are at least P_A and B by choosing v_0 can ensure that his losses are at most P_B. If $P_A = P_B$ then this seems to solve the game but is $P_A = P_B$ always?

Theorem 9.

(a) $P_A \le P_B$.

(b) $S_A \times S_B$ *contains a stable solution iff* $P_A = P_B$.

Proof. (a)

$$P_A = ROWMIN(u_0) \le PAY(u_0, v_0) \le COLMAX(v_0) = P_B. \tag{33}$$

(b) Suppose first that (u_0, v_0) is stable. Then, from (32), we have

$$COLMAX(v_0) = PAY(u_0, v_0) = ROWMIN(u_0)$$

and hence

$$P_B \le COLMAX(v_0) = ROWMIN(u_0) \le P_A,$$

which from (a) implies that $P_A = P_B$.

Conversely, if $P_A = P_B$ then from (33) we deduce that

$$ROWMIN(u_0) = PAY(u_0, v_0) = COLMAX(v_0)$$

which implies (32). □

We now consider specifically the case $S_A = \{(1), \ldots, (m)\}$ and $S_B = \{(1), \ldots, (n)\}$.

For Game one we have $P_A = P_B = 1 = a_{1,4}$ and hence A plays 1 and B plays 4 solves the game and A can guarantee to win at least 1 and B can guarantee to lose at most 1 on average.

The matrix of this game is said to have a *saddle point* (i_0, j_0) which means that $(i_0), (j_0)$ satisfies (32).

For a game who's matrix does not have a saddle point things are more complex. Consider for example Game two. $P_A = -1$ and $P_B = 1$. It follows from Theorem 9 that no pair of pure strategies solves the game. A knows she can average at least -1 by playing (3) and B knows he need lose no more than 1 on average by playing (3) but note that if A plays (3) then B has an incentive to play (1) or (2) but if he plays (1) then A will play (1) and so on.

Mixed strategies:
To break this seeming deadlock we allow the players to choose mixed strategies. A mixed strategy for A is a vector of probabilities $\mathbf{p} = (p_1, \ldots, p_m)$ where $p_i \geq 0$ for $i \in M$ and $p_1 + \cdots + p_m = 1$. A then chooses tactic i with probability p_i for $i \in M$ i.e. before each play A carries out a statistical experiment that has an outcome $i \in M$ with probability p_i. A then plays the corresponding tactic. Similarly B's mixed strategies are vectors $\mathbf{q} = (q_1, \ldots, q_n)$ satisfying $q_j \geq 0, j \in N$ and $q_1 + \cdots + q_n = 1$.

Pure strategies can be represented as vectors with a single non-zero component equal to 1. We now enlarge S_A, S_B to

$$\begin{aligned} S_A &= \{p \in \Re^m : \mathbf{p} \geq 0 \text{ and } p_1 + \cdots + p_m = 1\}. \\ S_B &= \{q \in \Re^n : \mathbf{q} \geq 0 \text{ and } q_1 + \cdots + q_n = 1\}. \end{aligned} \tag{34}$$

We now show using the duality theory of linear programming that $S_A \times S_B$ as defined in (34) contains a stable solution.

We shall first show how to compute P_A. Let $c_j(\mathbf{p}) = \sum_{i \in M} a_{i,j} p_i$. Then

$$P_A = \max_{\mathbf{p} \in S_A} \left(\min_{\mathbf{q} \in S_B} \sum_{j=1}^{n} c_j(\mathbf{p}) q_j \right) \tag{35}$$

Lemma 10.

$$\min_{\mathbf{q} \in S_B} \sum_{j=1}^{n} \xi_j q_j = \min \{\xi_1, \ldots, \xi_n\}. \tag{36}$$

Proof. Let $\xi_t = \min \{\xi_1, \ldots, \xi_n\}$ and let L be the LHS of (36). Putting $\widehat{q}_j = 0$ for $j \neq t$ and $\widehat{q}_t = 1$ we have $\widehat{\mathbf{q}} \in S_B$ and $\sum_{j=1}^{n} \xi_j \widehat{q}_j = \xi_t$. Thus $L \leq \xi_t$. However, for any $\mathbf{q} \in S_B$,

$$\sum_{j=1}^{n} \xi_j q_j \geq \sum_{j=1}^{n} \xi_t q_j = \xi_t \sum_{j=1}^{n} q_j = \xi_t.$$

□

It follows from the lemma and (35) that

$$\begin{aligned} P_A &= \max_{\mathbf{p}\in S_A} \min\{c_1(\mathbf{p}),\ldots,c_n(\mathbf{p})\} \\ &= \max \min\{c_1(\mathbf{p}),\ldots,c_n(\mathbf{p})\} \end{aligned}$$

Subject to

$$\begin{aligned} & p_1+\cdots+p_m = 1 \\ & p_1,\ldots,p_m \geq 0 \\ &= \max\ \xi \end{aligned} \tag{37}$$

Subject to

$$\begin{aligned} & \xi \leq \sum_{i=1}^{m} a_{i,j}p_i, \qquad j=1,\ldots,n \\ & p_1+\cdots+p_m = 1 \\ & p_1,\ldots,p_m \geq 0 \end{aligned}$$

Using similar arguments we can show that

$$P_B = \quad \max\ \eta \tag{38}$$

Subject to

$$\begin{aligned} & \eta \geq \sum_{j=1}^{n} a_{i,j}q_j, \qquad i=1,\ldots,m \\ & q_1+\cdots+q_n = 1 \\ & q_1,\ldots,q_n \geq 0 \end{aligned}$$

We note next that (37), (38) are a pair of dual linear programs. They are both feasible and hence $P_A = P_B$ and stable solutions exist. In fact if $\mathbf{p}^0$ solves (37) and $\mathbf{q}^0$ solves (38) $(\mathbf{p}^0, \mathbf{q}^0)$ is stable as $PAY(\mathbf{p}^0, \mathbf{q}^0) = P_A = P_B$.

Random payoff: We note that the above analysis goes through unchanged if A, B having selected tactics I, J, the payoff to A is a random variable who's expected value is $a_{i,j}$.

7.1 Dominance

If $A(i,j) \geq A(i,j')$ for all i then player B will never use strategy j. It is preferable for her/him to use strategy j' instead. So, column j can be removed from the matrix A.

Similarly, if $A(i,j) \leq A(i',j)$ for all j then player A will never use strategy i. It is preferable for her/him to use strategy i' instead. So, row i can be removed from the matrix A.

Repeated use of this idea can reduce a game substantially.

7.2 Latin Square Game

Suupose that every row sum is equal to $R > 0$ and every column sum is equal to $C > 0$ where $mR = nC$. Then both players can choose uniformly. Consider the two LP's that solve the game:

$$A \text{ Minimize } \sum_{i=1}^{m} x_i \text{ subject to } \sum_{i=1}^{m} a_{i,j} x_i \geq 1 \text{ for all } j, \sum_{i=1}^{m} x_i = 1. \tag{39}$$

$$B \text{ Maximize } \sum_{j=1}^{n} y_j \text{ subject to } \sum_{j=1}^{n} a_{i,j} j_j \leq 1 \text{ for all } i, \sum_{j=1}^{n} y_j = 1. \tag{40}$$

Putting $x_i = 1/C$ and $y_j = 1/R$ gives two feasible solutions with the same objective value.

7.3 Non-singular games

Suppose that A is non-singular and that $\mathbf{1^T A^{-1} 1} > \mathbf{0}$. Then the value of the game is $V = \frac{1}{\mathbf{1^T A^{-1} 1}}$. Then, $x^T = \frac{\mathbf{1^T A^{-1}}}{V}$ and $y = \frac{A^{-1}\mathbf{1}}{V}$ solve (39), (40) respectively.

7.4 Symmetric games

A game is symmetric if $A^T = -A$ i.e if A is anti-symmetric. Then the game has value 0. If A and B both use strategy p then because $p^T A p = 0$ for ant-symmetric A, we see that $PAY(p, p) = 0$. This implies that $0 \geq P_A = P_B \geq 0$.

8 Inventory Control

We discuss some simple models that attempt to control some of the costs of keeping inventory/stock. Our analysis involves trying to answer the question (i) when to order goods and (ii) how much to order.

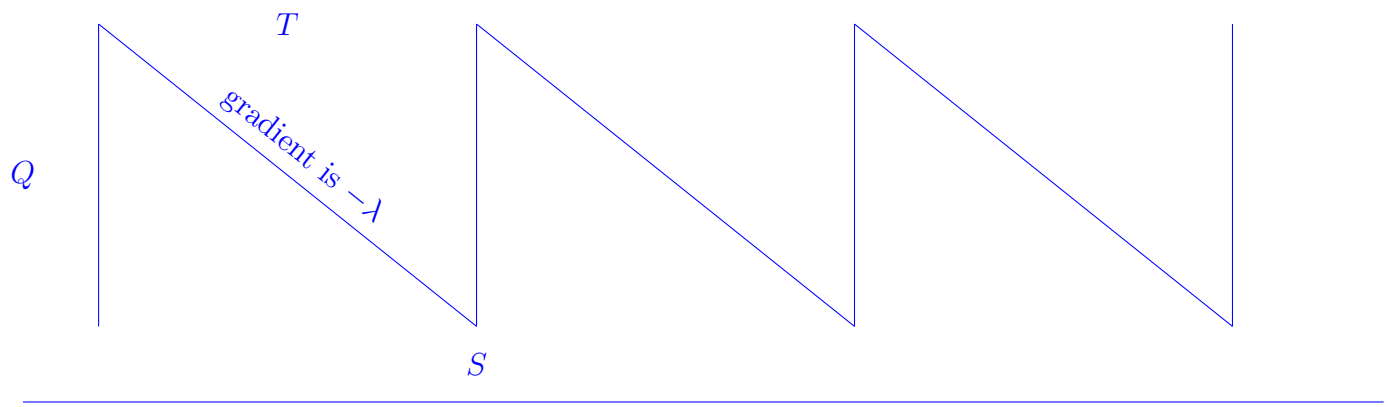
8.1 Model 1

The demand for the product is constant and λ per period there is a zero lead time and no stock outs are allowed. At constant intervals of time T we make an order for Q items of stock. The problem is to choose T and Q so as to minimize the average cost per period of running the system.

Notation

- A is the fixed cost associated with making an order.
- C is the unit cost per item.
- I is the inventory carrying charg. If one unit of stock is kept for t periods then the inventory charge is It.

Let S be the stock level when each order arrives. The stock level will have a pattern as shown in the diagram below:



From the diagram we see that $Q = \lambda T$ and so only one of these variables can be chosen independently.
Ordering cost
The actual cost of the items bought or produced is λC per period on average regardless of the inventory policy and will not be affect by the choice of Q. It is therefore ignored.

The average number of orders per period will be $1/T = \lambda/Q$. Therefore the average fixed cost of making orders is $A\lambda/Q$.

Holding cost
The average stock level is $(S + (S + Q))/2 = S + Q/2$ and therefore the average holding cost per period is $I(S + Q/2$.

Therefore the average total variable cost per period K is given by

$$K = \frac{A\lambda}{Q} + I\left(\frac{Q}{2} + S\right). \tag{41}$$

K is to be minimized for Q and $S \geq 0$. Now for given Q we minimize K by taking $S = 0$. To find the optimal Q we differentiate the right hand side of (41). Now

$$\frac{dK}{dQ} = -\frac{A\lambda}{Q^2} + \frac{I}{2}.$$

Putting $\frac{dK}{dQ} = 0$ gives the optimal Q_W where

$$Q_W = \left(\frac{2\lambda A}{I}\right)^{1/2}. \tag{42}$$

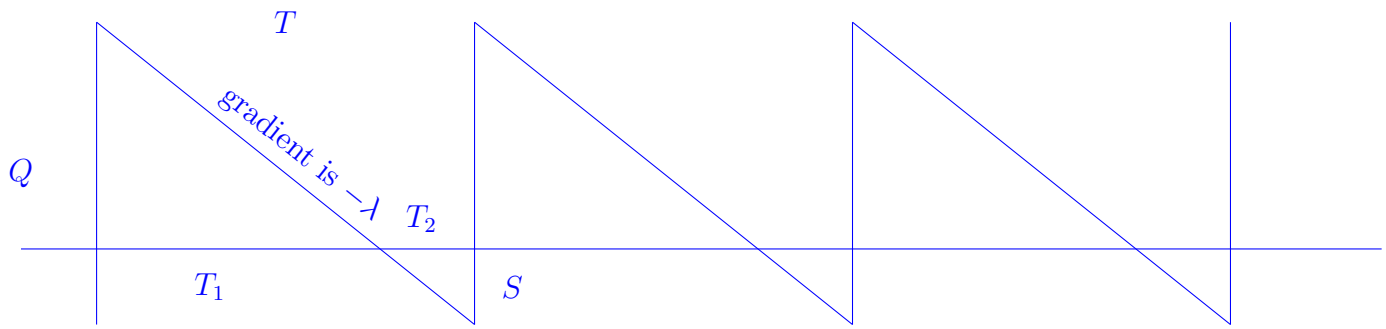
The RHS of (42) is sometimes referred to as the Wilson lot size formula. Using this we see that the optimal time interval T_W and the minimum cost K_W are given by

$$T_W = \left(\frac{2A}{\lambda I}\right)^{1/2} \text{ and } K_W = (2\lambda A I)^{1/2}.$$

Discrete case
Suppose that we can only order discrete quantities Q 2 Q Etc and that is not an exact multiple of Q in this case we simply compare the cost for Q rounded up and Q rounded down and take the smaller

8.2 Model 2

This model is the same as for Model 1 except that items out of stock can be backordered and supplied when goods come into stock. The cost of each item back ordered is πt where t is the length of time the demand remains unfilled. The Infantry level will follow the pattern below:



s is the number of back orders when the order Q arrives. The objective is to choose Q and S so that the average cost per period is minimized. The following relations are apparent from the diagram:

$$Q = \lambda T, \quad S = \lambda T_2, \quad Q - S = \lambda T_1.$$

Ordering cost:
The number of orders per period is again $1/T = \lambda/Q$ and so the variable ordering cost is $\lambda A/Q$.

Holding cost:
The proportion of time that there are goods in stock is T_1/T. During this time the average infantry level is $(Q-Q)/2$ and so the average holding cost per period is $I(Q-S)T_1/2T$. But from the above we have that $T_1/T = (Q-S)/Q$ and so this cost is in fact $I(Q-S)^2/2Q$.

Backorder cost:
The proportion of time that the system is out of stock is T_2/T. Because of the given form of backorder cost we may work out the average back order cost in a similar way to that of the holding cost. During the time the system is out of stock the average amount backordered is $S/2$. Therefore the average back order cost is $\pi S T_2/2t = \pi S^2/2Q$. Thus the total variable annual cost K is then

$$K = \frac{\lambda A}{Q} + \frac{I(Q-S)^2}{2Q} + \frac{\pi S^2}{2Q}.$$

In order to minimize K we solve the equations $\frac{\partial K}{\partial Q} = \frac{\partial K}{\partial S} = 0$, yielding

$$-\frac{\lambda A}{Q^2} - \frac{I(Q-S)^2}{2Q^2} + \frac{I(Q-S)}{Q} - \frac{\pi S^2}{2Q^2} = 0. \tag{43}$$

$$-\frac{I(Q-S)}{Q} + \frac{\pi S}{Q} = 0. \tag{44}$$

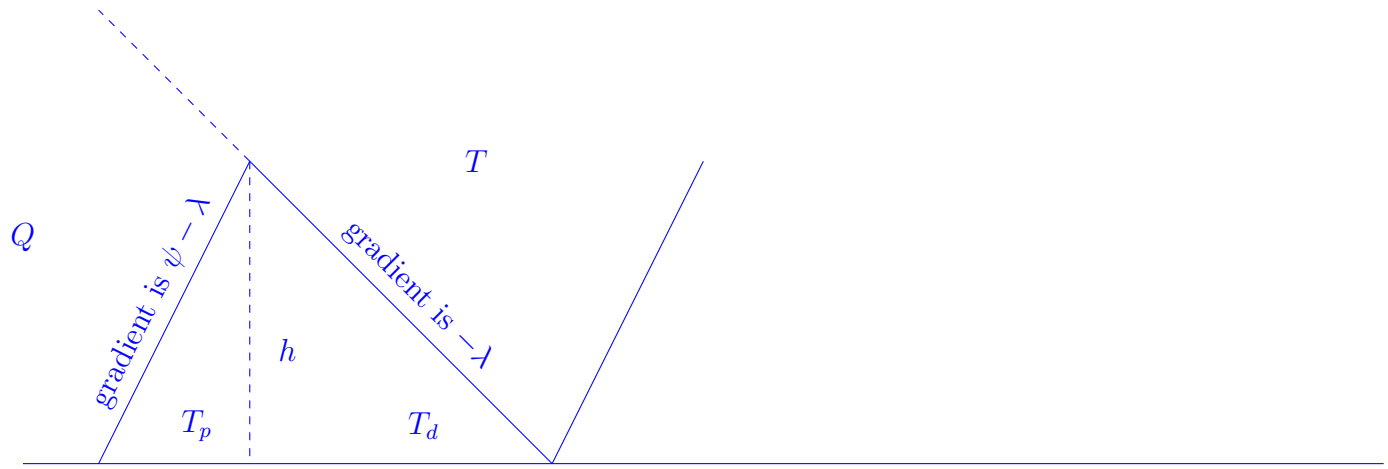
These can be solved by using (44) to express Q in terms of S, substitute in (43) to give an equation in S and then solving, giving

$$S = \left(\frac{2\lambda A I}{\pi(\pi+I)}\right)^{1/2}, \quad Q = Q_W\left(\frac{\pi+I}{\pi}\right)^{1/2}, \quad K = K_W\left(\frac{\pi}{\pi+I}\right)^{1/2}.$$

Comparing with model 1 we see that the extra freedom of allowing back orders enables us to reduce the total cost. We can obtain the results for Model 1 by putting $\pi = \infty$.

8.3 Model 3:

In the previous models we assumed that the orders arrived instantaneously in a single lot size of Q units we now consider a situation in which having made an order the items arrive in a continuous stream at a rate $\psi > \lambda$ per period. The entrance level assuming no back ordering will then have the pattern below during



During period T_p the stock level increases at a rate $\psi - \lambda$. During period T_d the stock level decreases at a rate λ. If the length of the time between orders is T and the total amount Q is produced at a time then we must have $Q = \lambda T$, otherwise stock levels will not be zero again at the end of each cycle.

Ordering cost:
The average number of orders per period is again $1/T = \lambda/Q$ and so the average ordering cost $A\lambda/Q$.

Holding cost:
If h is the maximum stock level during a cycle then the average stock $h/2$. Now h is the amount of stock built up during T_p and so $h = (\psi - \lambda)T_p$. We also have that $Q = \psi T_p$ as the order is produced in time T_p. Substituting in the above gives $h = (1 - \lambda/\psi)Q$. Therefore the average inventory cost per period is $I(1 - \lambda/\psi)Q/2$ total variable cost is

$$\frac{\lambda A}{Q} + \frac{IQ(\psi - \lambda)}{2\psi Q}.$$

Solving the equation $\frac{dK}{dQ} = 0$ gives the optimal batch quantity as

$$Q = Q_W \left(\frac{\psi}{\psi - \lambda}\right)^{1/2} \text{ and } K = K_W \left(\frac{\psi - \lambda}{\psi}\right)^{1/2}.$$

The total cost has decreased again as against Model 1 due to decreasing inventory costs. Note that taking $\psi = \infty$ gives Model 1 again.

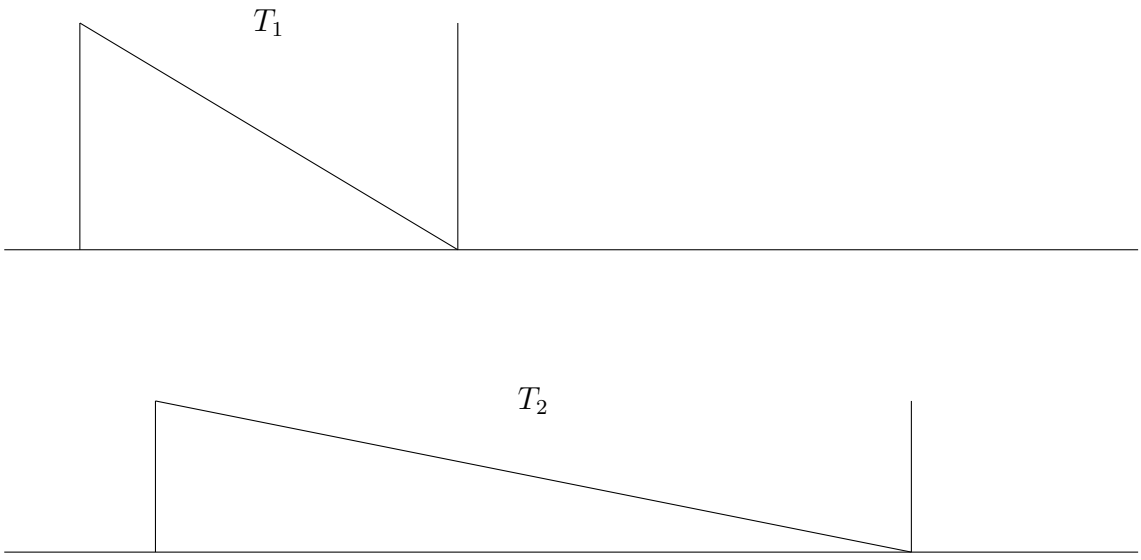
8.4 Deterministic multi-item problems

We consider here some problems where more than one type of item is being stored and when the inventory cost of the items interact. If there is no interaction we can consider each item separately.

Models 4 and 5 of this section are generalizations of Model 1 although we could equally well have generalized Model 2 or Model 3.

8.5 Model 4

Assume there are n distinct types of item who's individual characteristics are that of Model 1. When an order is made it is possible to order more than one type of item but the fixed cost of ordering A is unaffected by the number of different types of goods ordered. Supposedly then that there is a demand λ_j and holding cost I_j for item j. Let the optimal policy be to all the item at period T_j.



We show that in an optimum scheule, $T_1 = T_2 = \cdots = T_n$. For let $T = T_k = \min_j T_j$. Then if $T_j > T$ for some j then by increasing the frequency of ordering of item j to that of item k and ensuring that item J is ordered at the same time as k we
(i) eliminate any extra ordering cost for j over those of k
, (ii) decrease the average stock level of j.

This implies non-optimality for the current schedule and the claim follows. The problem is then to find the optimal value for T.

Ordering cost
The average number of all those per period is and so the average ordering cost per period is A/T.

Holding cost
Let Q_j be the amount of item j ordered at a time. Then since orders are made at intervals of time T we must have $Q_j = \lambda_j T$. Now the average inventory for item j is $Qj/2$ and so the average inventory cost per period for item j is $I_j Q_j/2$. We see therefore that the average total invemtory cost is $T\sum_j I_j Q_j/2$. So the average total cost period is given by

$$K = \frac{A}{T} + T\sum_{j=1}^{n} \frac{I_j Q_j}{2}. \tag{45}$$

The optimal value of T is obtained by solving $\frac{dK}{dT} = 0$ which gives

$$T = \left(\frac{2A}{\sum_{j=1}^{n} \lambda_j T_j} \right)^{1/2}.$$

The optimal lot sizes $Q_j = \lambda_j T$ and the minimal cost K as in (45) can now be calculated.

8.6 Model 5

In the previous model the items were able to share facility (ordering) without an increase in cost. Thus the total cost was smaller than if all the items were ordered separately. In some situations items will make

conflicting demands on resources, for example if there is a shortage of storage space then calculating the optimal lot size for each item separately may lead to too great a demand for storage space. We again to assume there are n items whose individual characteristics are those of Model 1 but that the lots sizes Q_J have to satisfy the constraint

$$\sum_{j=1}^{n} f_j Q_j \leq f. \tag{46}$$

For given batch sizes $Q_1, \ldots, Q_n$ we may evaluate the average total cost by something individual costs as calculated in Model 1. This gives

$$K = \sum_{j=1}^{n} \frac{\lambda_j A_j}{Q_j} + \frac{I_j Q_j}{2}. \tag{47}$$

The problem is therefore to minimize K subject to (46) and we can proceed as follows:

(i) We first find the minimum of K ignoring constraint (47). Now to minimize K overall we can simply minimize each term in the summation separately. This will naturally to the Wilson lot sizes $Q_{jW} - (2\lambda A_j/I_j)^{1/2}$, $j = 1, 2, \ldots, n$. Iif thse values satisfy the constraint then we have solved the problem.

(ii) Otherwise may assume that the constraint is active at the optimum $\mathbf{Q}^* = (Q_1^*, \ldots, Q_n^*)$. This follows because we may express

$$K(\mathbf{Q}) = K(\mathbf{Q}^*) + \sum_{j=1}^{n} \frac{\partial K}{\partial Q_j}(Q_j - Q_j^*) + \cdots$$

and if $\sum_j f_j Q_j^* < f$ then $\left(Q_1^* - \theta\frac{\partial K}{\partial Q_1}, \ldots, Q_n^* - \theta\frac{\partial K}{\partial Q_n}\right)$ is both feasible and better than $\mathbf{Q}^*$ for small enough $\theta > 0$, unless $\frac{\partial K}{\partial Q_j} = 0$ for $j = 1, 2, \ldots, n$, which we have already ruled out.

So now we tackle the problem

$$\text{Minimise } K \text{ subject to } F = \sum_{j=1}^{n} f_j Q_j - f = 0. \tag{48}$$

(We are *lucky* in that we can ignore the constraints $Q_j \geq 0, j = 1, \ldots, n$.)

Problem (48) is in a form suitable for application of the classical Lagrange Multiplier method. This states that there exists θ such that at the optimum to (48)

$$\frac{\partial K}{\partial Q_j} = \theta \frac{\partial F}{\partial Q_j} \text{ for } j = 1, \ldots, n. \tag{49}$$

On differentiation we see that this states

$$-\frac{\lambda_j A_j}{Q_j^2} + \frac{I_j}{2} = \theta f_j \text{ for } j = 1, \ldots, n.$$

or

$$Q_j = \left(\frac{2\lambda_j A_j}{I_j - 2\theta f_j}\right)^{1/2}. \tag{50}$$

The value for θ can be calculatd by solving

$$\sum_{j=1}^{n} f_j \left(\frac{2\lambda_j A_j}{I_j - 2\theta f_j}\right)^{1/2} = f.$$

Given θ we use (50) to find the Q_j's.

9 Job Shop Scheduling

Jobs $J_1, J_2, \ldots, J_n$ have to be processed on machines $M_1, M_2, \ldots, M_m$ in order to complete the production of some item. We study a few associated optimisation problems. Here we will only have $m = 1$ or 2.

9.1 Single machine, minimise weighted completion time

$$C_j = \text{ completion time of job } j.$$
$$p_j = \text{ processing time of job } j.$$

Objective: find the order (π) in which to do the jobs that minimises $S = S(\pi) = \sum_{j=1}^{n} w_j C_j$.

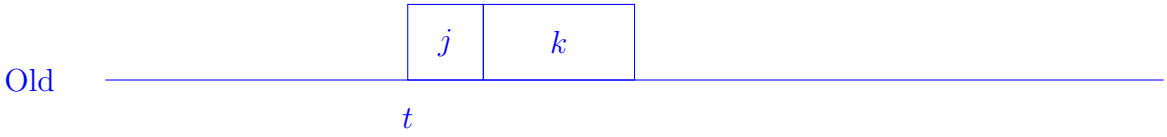
Example: $n = 4$, $p_1 = 3, p_2 = 2, p_3 = 5, p_4 = 6$ and $w_1 = 1, w_2 = 10, w_3 = 3, w_2 = 2$.

$S(1, 2, 3, 4) = 1 \times 3 + 10 \times 5 + 3 \times 10 + 2 \times 16 = 105.$
$S(2, 3, 1, 4) = 10 \times 2 + 3 \times 7 + 1 \times 10 + 2 \times 16 = 83.$

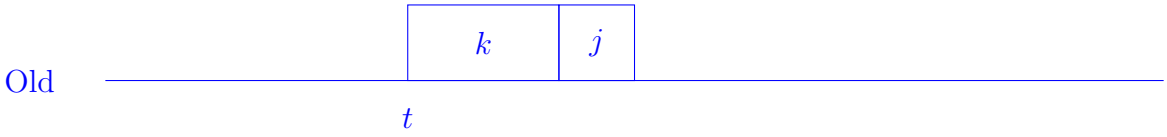
In general the optimum order satisfies

$$\frac{w_{\pi(1)}}{p_{\pi(1)}} \geq \frac{w_{\pi(2)}}{p_{\pi(2)}} \geq \cdots \geq \frac{w_{\pi(n)}}{p_{\pi(n)}}.$$

Suppose the processing order does not satisfy this: j is processed before k.



Here $w_j/p_j < w_k/p_k$. Now interchange j, k in the order. Only the completion times of j, k are affected.



$$Z_{new} - Z_{old} = w_k(t + p_k) + w_j(t + p_k + p_j) - w_j(t + p_j) - w_k(t + p_k + p_j) = w_j p_k - w_k p_j < 0.$$

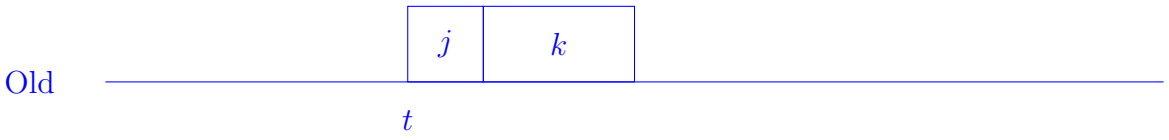
9.2 Single machine, minimise maximum lateness

$$d_j = \text{ due date for job } j.$$
$$L_j = \max\{C_j - d_j, 0\} = (C_j - d_j)^+ = \text{ lateness of job } j.$$
$$L_{\max} = \max_j L_j.$$

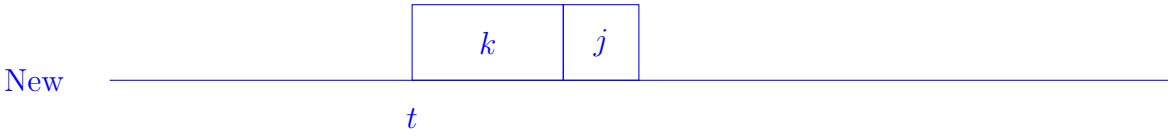
Objective: find the order (π) in which to do the jobs that minimises $L_{\max}$.

In general the optimum order satisfies $d_1 \leq d_2 \leq \cdots \leq d_n$.

Suppose we do not use this order: Suppose the processing order does not satisfy this: j is processed before k.



Here $d_j > d_k$. Now interchange j, k in the order. Only the completion times of j, k are affected.



Contribution of j, k to $L_{\max}$:

$$\text{Old: } \max\left\{(t+p_j-d_j)^+, (t+p_j+p_k-d_k)^+\right\} = (t+p_j+p_k-d_k)^+.$$
$$\text{New: } \max\left\{(t+p_k-d_k)^+, (t+p_j+p_k-d_j)^+\right\}.$$

But clearly

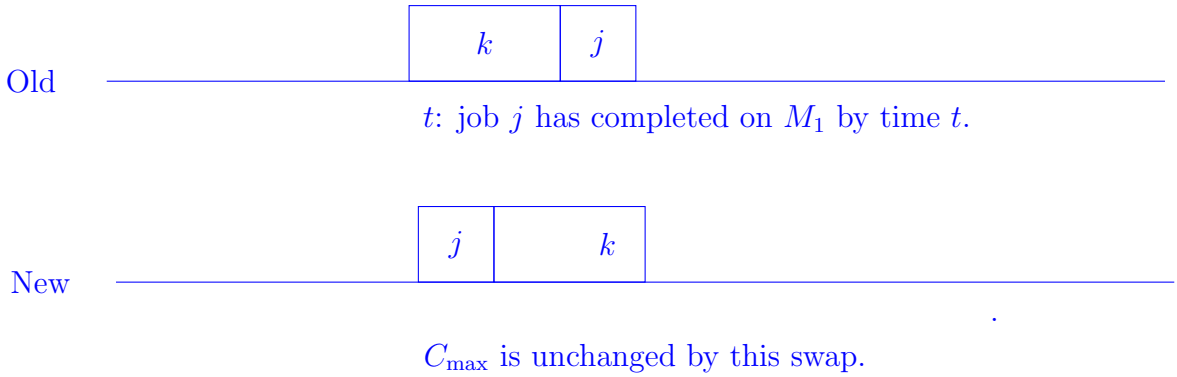
$$\max\left\{(t+p_k-d_k)^+, (t+p_j+p_k-d_j)^+\right\} \leq (t+p_j+p_k-d_k)^+.$$

9.3 Two machine flow shop: Johnson's rule

Each job has to be processed first on machine M_1 and then on machine M_2. The goal is to minimise the completion time of all the jobs. In principle the order of jobs on the two machines π_1, π_2 can be different.

9.3.1 $\pi_1 = \pi_2$: Permutation flow shop:

We first show that it is optimal to fix $\pi_1 = \pi_2$. By re-labelling if needed, we can assume that $\pi_1(i) = i$ for $i \in [n]$. Now suppose there exists $j < k$ such that $\pi_2(j) > \pi_2(k)$. The diagram below refers to M_2.

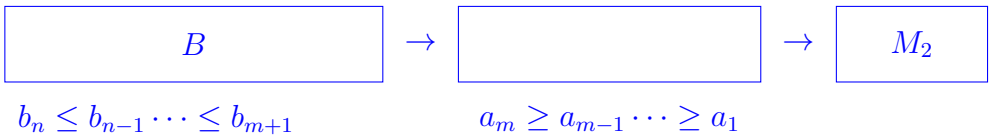


So, we will choose a single π for both machines.

9.3.2 Johnson's Rule

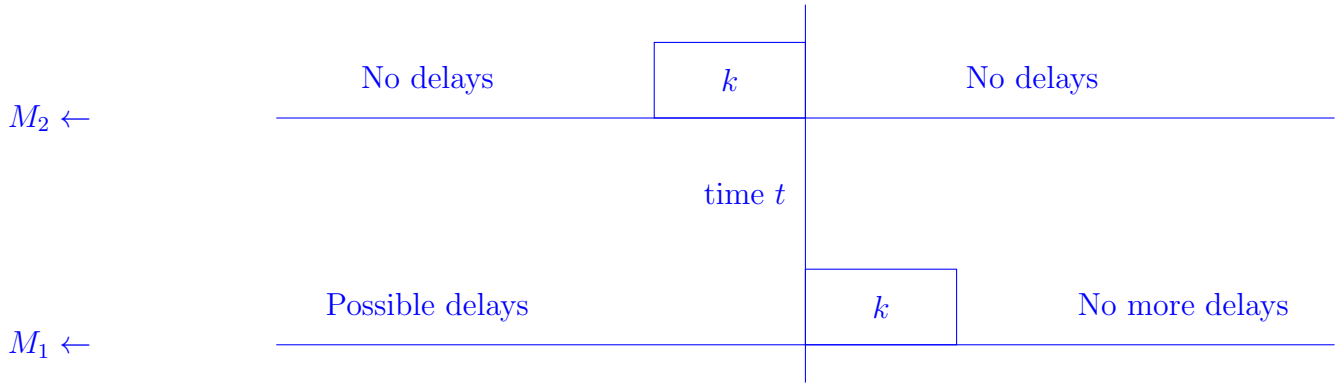
Processing times: $a_1, \ldots, a_n$ on M_1.
$b_1, \ldots, b_n$ on M_2.

Then we let $A = \{j : a_j \leq b_j\}$ and $B = [n] \setminus A$. Suppose that $|A| = m$. Renumber so that



JOHNSON'S RULE

For all permutation schedules there is a time t, after which there are no delays on machine M_2.

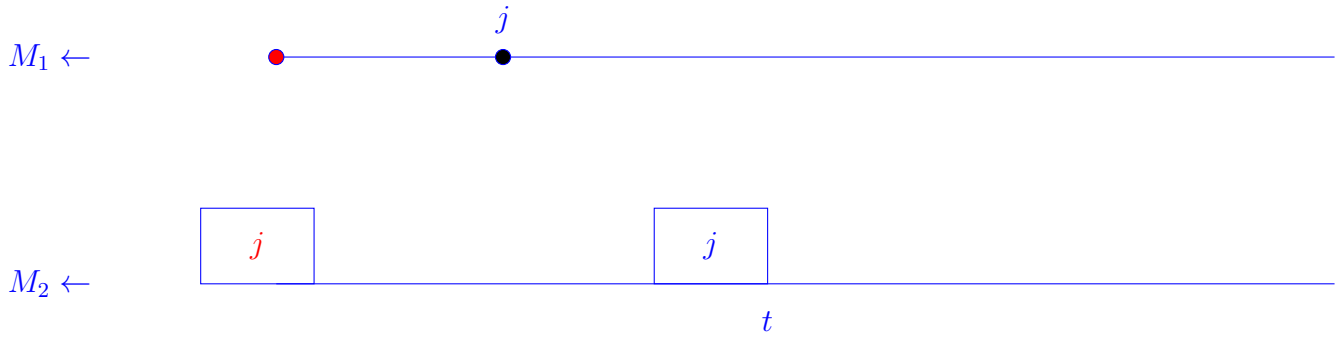


It follows that

(i) $C_{\max}$ is the sum of $n+1$ job processing times.

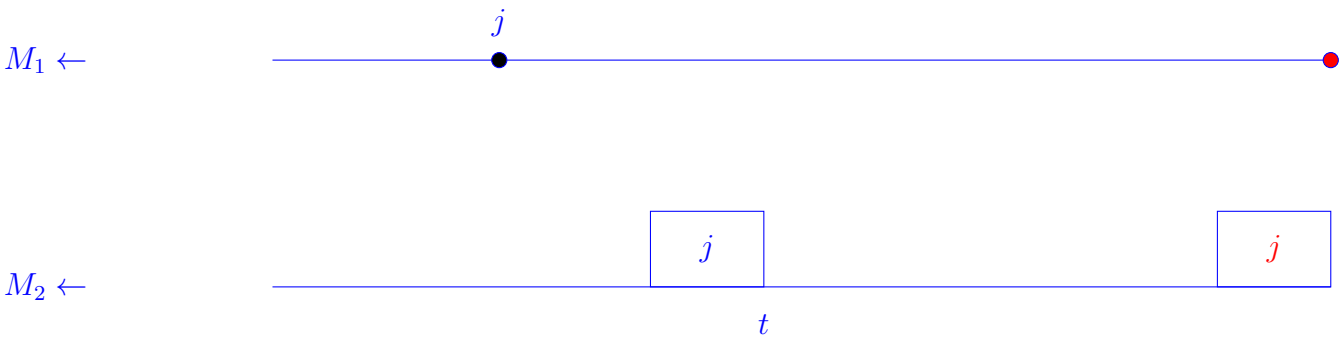
(ii) Reducing the processing time of each item by an amount ρ does not affect the optimal ordering. This is because each ordering schedule is reduced by $(n+1)\rho$.

Now let $\rho = \rho([n]) = \min\{a_1, \ldots, a_n, b_1, \ldots, b_n\}$. There are two cases:
Case (a): $\rho = a_j$ for some j. Now let $a'_i = a_i - \rho, b'_i = b_i - \rho$ for $i = 1, \ldots, n$. Note that $a'_j = 0$. We claim that j can go first onto the machines.



Moving job j to the front does not increase $C_{\max}$. To see this, first move j to be the first job on machine M_1. This has no affect on $C_{\max}$. Now the orders on the two machines are different and we use the argument of Section 9.3.1 that moving job j closer to the beginning of the order will not increase $C_{\max}$.

Case (b): $\rho = b_j$ for some j. Now let $a'_i = a_i - \rho, b'_i = b_i - \rho$ for $i = 1, \ldots, n$. Note that $b'_j = 0$. We claim that j can go last onto the machines.



Moving job j to the end does not increase $C_{\max}$. To see this, first move j to be the last job on machine M_2. This has no affect on $C_{\max}$. Now the orders on the two machines are different and we can modify the argument of Section 9.3.1 that moving job j closer to the end of the order will not increase $C_{\max}$.

Let now S denote the jobs that have not been ordered. The general rule is

1. Suppose that the current state of the ordering is $i_1, \ldots, i_t, S, j_1, \ldots, j_s$ where $s + t = n - |S|$.
2. $\rho = r(S) = \min\{\min_{i \in S} a_i, \min_{i \in S} b_i\}$.
3. If $\rho = a_j$ then job j is placed immediately after job i_t.
4. If $\rho = b_j$ then job j is placed immediately before job j_1.

10 Decision Analysis

Company GFB owns a plot of land. It is considering drilling for oil.

Alternatives	Oil	Dry
Drill	700K	-100K
Sell Land	90K	90K
Probability	0.25	0.75.

Decision strategies:

(i) Choose alternative that maximises minimum payoff: Sell Land.

(ii) Choose alternative that maximises minimum payoff *under the most likely alternative*: Sell land.

(iii) Choose alternative that maximises the expected payoff:
Drill: $0.25 \times 700 - 0.75 \times 100 = 100^*$.
Sell: 90.

Now introduce a third alternative: do a seismic study (SS) and then make a decision. The cost of the survey is 30K.

Outcome: FSS=Favorable, USS=Unfavorable.
Data:

$$\begin{aligned} \mathbb{P}(FSS \mid Oil) &= 0.6 \qquad \mathbb{P}(USS \mid Oil) = 0.4. \\ \mathbb{P}(FSS \mid Dry) &= 0.2 \qquad \mathbb{P}(FSS \mid Oil) = 0.8. \end{aligned}$$

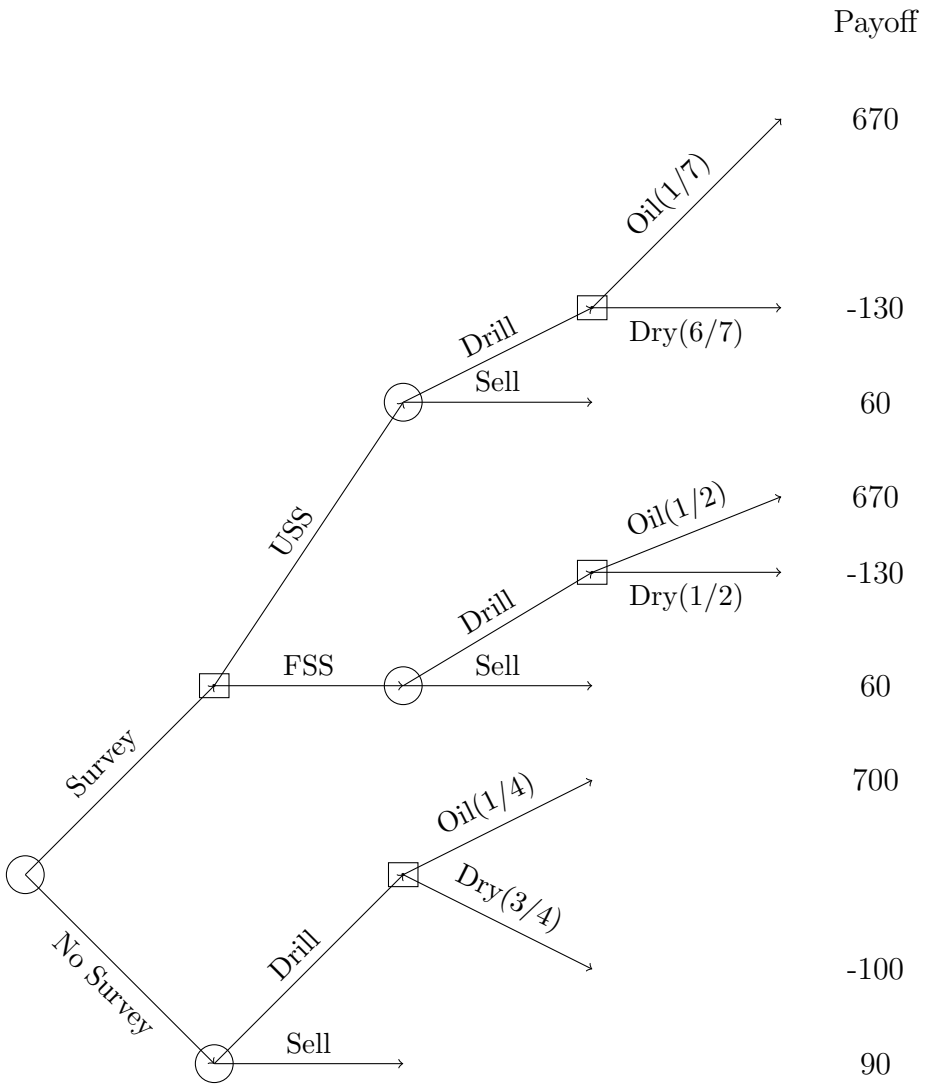
Bayes computation. We need

$$\begin{aligned} \mathbb{P}(Oil \mid FSS) &= \frac{\mathbb{P}(Oil \wedge FSS)}{\mathbb{P}(FSS)} \\ &= \frac{\mathbb{P}(FSS \mid Oil)\mathbb{P}(Oil)}{\mathbb{P}(FSS \mid Oil)\mathbb{P}(Oil) + \mathbb{P}(FSS \mid Dry)\mathbb{P}(Dry)} \\ &= \frac{0.6 \times 0.25}{0.6 \times 0.25 + 0.2 \times 0.75} \\ &= \frac{1}{2}. \end{aligned}$$

Similar calculations give

$$\mathbb{P}(Oil \mid FSS) = \frac{1}{2},\ \mathbb{P}(Dry \mid FSS) = \frac{1}{2},\ \mathbb{P}(Oil \mid USS) = \frac{1}{7},\ \mathbb{P}(Dry \mid USS) = \frac{6}{7}.$$

Decision tree.

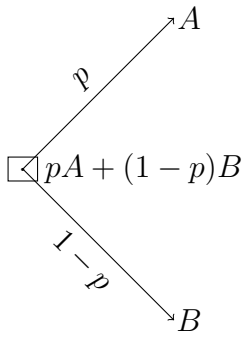


Decision node ◯

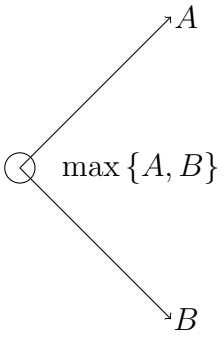
Chance node ⊡

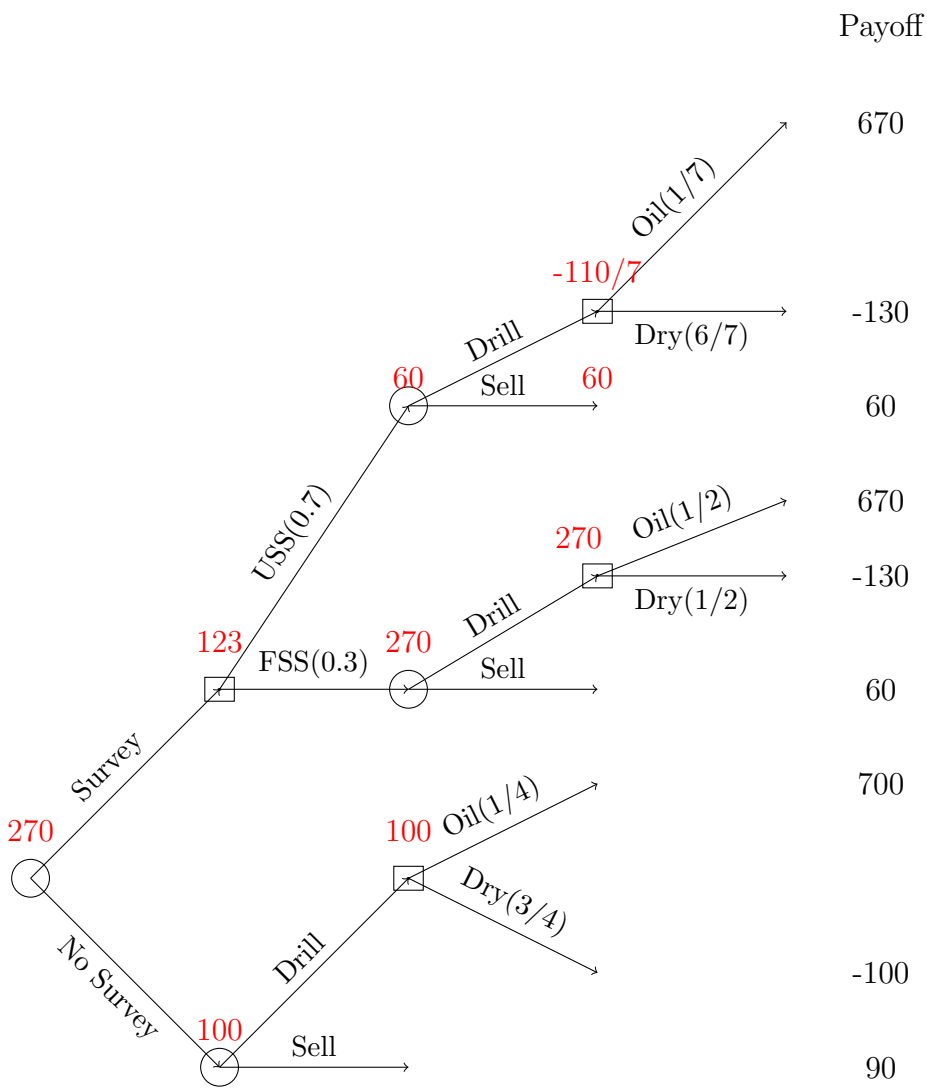
Evaluating nodes of tree. A, B are the expected values at these nodes.

Chance node:



Decision node.




Payoff
670
-130
60
670
-130
60
700
-100
90
Oil(1/7)
-110/7
Drill
Dry(6/7)
60
Sell
60
USS(0.7)
Oil(1/2)
270
Drill
Dry(1/2)
123
FSS(0.3)
270
Sell
Survey
Oil(1/4)
100
Dry(3/4)
270
Drill
No Survey
100
Sell

11 Approximation algorithms

If solving an optimization problem is considered to be hard, then we can sometimes efficiently find approximate solutions with guarantees how far from optimum they are. An α-approximation algorithm for a minimisation problem computes a solution whose value is at most αv^*, where v^* is the minimum value for the problem. (For maximisation we have at least αv^*.)

11.1 Traveling Salesperson Problem – TSP

11.1.1 Unrestricted

There is no polynomial time M-approximation algorithm unless P=NP. Given a graph G, give a cost of 1 to each edge of G and $Mn+1$ to each non-edge. Then, the TSP has a tour of length n iff G is Hamiltonian. Otherwise the cost of the tour is at least $(M+1)n$. An M-approximation algorithm could tell if G is Hamiltonian. If G is Hamiltonian, it produces a tour of length at most $Mn < (M+1)n$, implying that G is Hamiltonian. Otherwise it produces a tour of length at least $(M+1)n$, implying that the minimum length tour is greater than n, so G is not Hamiltonian.

11.1.2 Triangle Inequality

We assume next that the costs $C(i,j), 1 \leq i,j \leq n$ satisfy the *triangular inequality* i.e. $C(i,j) + C(J,k) \geq C(i,k)$.

Tree heuristic

Step 1 Find a minimum cost spanning tree T.

Step 2 Double the edges of T to make an Eulerian multigraph K. (Eulerian because all degrees are even. Such a graph contains a closed walk that goes through each edge exactly once.)

Step 3 Construct an Euler tour $i_1, i_2, \ldots, i_{2n-2}$ through the edges of K.

Step 4 Shortcut the Euler tour until it is a Hamilton cycle H. I.e. go through the vertices $i_1, i_2, \ldots,$ in sequence and skip over any vertex that has already been visited.

Theorem 11. *The tour H found by the tree heuristic satisfies $C(H) \leq 2C^*$, where C^* is the minimum cost of a tour.*

Proof. We observe that

$$C(H) \leq C(K) \leq 2C(T) \leq 2C^*.$$

For $C(H) \leq C(K)$ we use the triangle inequality repeatedly to argue that $C(j_1, j_2) + C(j_j, j_3) + \cdots + C(j_{k-1}, j_k) \geq C(j_1, j_k)$. The other inequalities are obvious. □

Christofides' heuristic A simple idea reduces the 2 to 3/2.

Step 1 Find a minimum cost spanning tree T.

Step 2 Let O be the set of vertices of T of odd degree. $|O|$ is even.

Step 3 Find a minimum cost matching M that covers O.

Step 4 Let $K = M + T$. K is Eulerian.

Step 5 Construct an Euler tour $i_1, i_2, \ldots, i_{2n-2}$ through the edges of K.

Step 6 Shortcut the Euler tour until it is a Hamilton cycle H.

Theorem 12. *The tour H found by the Christofides' heuristic satisfies $C(H) \leq 3C^*/2$.*

Proof. This follows from the fact that $C(M) \leq C^*/2$. Start with the optimal tour. Shortcut the vertices that are not in O. What is left has cost at most C^* and is the union of two disjoint matchings. Each of them have cost at least that of M. □

11.2 Knapsack problem

We consider the problem

$$\begin{aligned} \text{Maximize} \sum_{i=1}^{n} p_i x_i \\ \text{Subject to} \sum_{i=1}^{n} a_i x_i \leq B \\ x_i = 0/1, i = 1, 2, \ldots, n \end{aligned}$$

We will assume that $p_1/a_1 \geq p_2/a_2 \geq \cdots \geq p_n/a_n$.

11.2.1 Greedy Algorithm

Find i such that $a_1 + a_2 + \cdots + a_{i-1} \leq B < a_1 + a_2 + \cdots + a_i$. Then choose the better of $\{1, 2, \ldots, i-1\}$ and $\{i\}$. (We associate the set $\{j : x_j = 1\}$ with a solution.)

Let $S = a_1 + a_2 + \cdots + a_{i-1}$ and $T = p_1 + p_2 + \cdots + p_{i-1}$. We have the maximum value

$$OPT \leq T + \frac{B - S}{a_i} p_i.$$

The RHS is the optimal value allowing fractional values for the x_j. So, either

$$T \geq \frac{OPT}{2} \text{ or } p_i \geq \frac{B - S}{a_i} p_i \geq \frac{OPT}{2}.$$

11.2.2 Profit rounding algorithm

Let $\widehat{p}_i = \lfloor Np_i/p_{\max} \rfloor$ for $i = 1, 2, \ldots, n$. Here $p_{\max} = \max_{i\in[n]} p_i$ and $N = \lceil n/\varepsilon \rceil$. We solve the knapsack problem with profits $\widehat{p}_i$. Because the profits are "small" ($\leq N$), we can solve the new problem in polynomial time via Dynamic programming.

Theorem 13. *The solution produced in this way has value at least* $(1-\varepsilon)OPT$.

Proof. We first discuss the solution of the knapsack problem. Let

$$\begin{aligned} g_r(p) = &\text{minimum } a_1x_1 + \ldots + a_rx_r \\ &\text{Subject to } \widehat{p}_1x_1 + \cdots + \widehat{p}_rx_r \geq p \\ &\qquad x_i = 0/1 \text{ for } i \in [r]. \end{aligned} \tag{51}$$

Note that

$$g_r(p) = \min \begin{cases} g_{r-1}(p) & x_r = 0. \\ a_r + g_r(p - \widehat{p}_r) & x_r \geq 1. \end{cases} \tag{52}$$

We evaluate (52) for $p = 0, 1, \ldots, Nn$ and $r = 1, 2, \ldots, n$. This takes $O(N^2) = O(n^3/\varepsilon)$ time.

If we know $g_n(p), 0 \leq p \leq Nn$, then

$$OPT = g_n(p^*) \text{ where } g_n(p^*) \leq B \text{ and } g_n(p^*+1) > B.$$

We now verify the quality of the solution. Let $\widehat{S}$ define the solution to (51) and let S^* define the solution to the actual knapsack problem. Then

$$\sum_{i\in\widehat{S}} \widehat{p}_i \geq \sum_{i\in S^*} \widehat{p}_i.$$

Therefore

$$\sum_{i\in\widehat{S}} p_i \geq \sum_{i\in\widehat{S}} \left\lfloor \frac{p_iN}{p_{\max}} \right\rfloor \frac{p_{\max}}{N} = \frac{p_{\max}}{N} \sum_{i\in\widehat{S}} \widehat{p}_i \geq \frac{p_{\max}}{N} \sum_{i\in S^*} \widehat{p}_i. \tag{53}$$

But,

$$\frac{p_{\max}}{N} \sum_{i\in S^*} \lfloor Np_i/p_{\max} \rfloor \geq \frac{p_{\max}}{N} \sum_{i\in S^*} \left(\frac{p_iN}{p_{\max}} - 1 \right) \geq \sum_{i\in S^*} p_i - \frac{np_{\max}}{N} \geq$$

$$\sum_{i\in S^*} p_i - \varepsilon p_{\max} \geq \sum_{i\in S^*} p_i - \varepsilon OPT = (1-\varepsilon)OPT.$$

□

11.3 Submodular functions

A function $f : 2^X \to \mathbb{R}$ is *submodular* if for every $A, B \subseteq X$,

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B). \tag{54}$$

Note that if f, g satisfy (54), then so does $f + g$.

Example: Simple plant location problem Here f refers to profit:

$$f(S) = -\sum_{i \in S} f_i + \sum_{j=1}^{n} \max\{p_{i,j} : i \in S\}.$$

The constant function is clearly submodular and this deals with $\sum_i f_i$. Then we observe that for reals $x_1, x_2, \ldots, x_n$,

$$\max\{x_i : i \in A \cap B\} + \max\{x_i : i \in A \cup B\} \leq \max\{x_i : i \in A\} + \max\{x_i : i \in B\}.$$

(Assume that the largest x_i is for $i \in A$. Then, $\max\{x_i : i \in A \cup B\} = \max\{x_i : i \in A\}$ and $\max\{x_i : i \in A \cap B\} \leq \max\{x_i : i \in B\}$.)

f is monotone increasing if

$$f(B) \geq f(A) \text{ whenever } B \supseteq A.$$

Greedy Algorithm:
Step 0: $S_0 = \emptyset$.
Step i**:** $S_i = S_{i-1} \cup \{x_i\}$ where $x_i \notin S_{i-1}$ maximises $f(S_{i-1} \cup \{x\})$.

Theorem 14. *If f is monotone increasing and submdular and if $f(\emptyset) = 0$ then after k steps of Greedy*

$$f(S_k) \geq (1 - e^{-1}) f(S_k^*),$$

where S_k^* maximises f over sets of size k.

Proof. Let $\Delta(v \mid T) = f(T \cup \{v\}) - f(v)$. If $S \supseteq T$ and $v \notin S$ then

$$f(S \cup \{v\}) + f(T) \leq f(T \cup \{v\}) + f(S) \qquad (A = T \cup \{v\}, B = S),$$

which implies that

$$\Delta(v \mid S) \leq \Delta(v \mid T). \tag{55}$$

(The larger the set, the smaller the gain from v.)

Note that (55) is still true if $v \in S$, from monotonicity. Let $S_k^* = \{v_1^*, \ldots, v_k^*\}$.

$$\begin{aligned}
f(S_k^*) &\leq f(S_k^* \cup S_i) \\
&= f(S_i) + \sum_{j=1}^{k} \Delta(v_j^* \mid S_i \cup \{v_1^*, \ldots, v_{j-1}^*\} \\
&\leq f(S_i) + \sum_{j=1}^{k} \Delta(v_j^* \mid S_i) \\
&\leq f(S_i) + \sum_{j=1}^{k} (f(S_{i+1}) - f(S_i)) \\
&= f(S_i) + k(f(S_{i+1}) - f(S_i)).
\end{aligned}$$

We re-write the last line as

$$f(S_k^*) - f(S_{i+1}) \leq \left(1 - \frac{1}{k}\right)(f(S_k^*) - f(S_i)).$$

This implies that

$$f(S_k^*) - f(S_k) \leq \left(1 - \frac{1}{k}\right)^k (f(S_k^*) - f(\emptyset)) \leq e^{-1} f(S_k^*).$$

□

11.4 Local Search

This is a general approach to solving hard problems in Combinatorial Optimisation. Suppose that the problem is to

$$\text{Maximise } f(\mathbf{x}) \text{ Subject to } \mathbf{x} \in X. \tag{56}$$

One proceeds as follows: for each $\mathbf{x} \in X$ we define a *neighborhood* $N_{\mathbf{x}} \subseteq X$ containing $\mathbf{x}$. It is defined so that finding $\max\{f(\mathbf{y}) : \mathbf{y} \in \mathbf{N_x}\}$ can be done efficiently. We can then find a good (not necesarily optimal solution as follows: let $\mathbf{x}_0 \in X$ be chosen in some way. Then define the sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_m$ where $\mathbf{x}_i$ maximises $f(\mathbf{y}) : \mathbf{y} \in \mathbf{N_{x_{i-1}}}$. The value m will the smallest i such that $\mathbf{x}_i = \mathbf{x}_{i-1}$.

11.4.1 MaxCut

We are given a connected graph $G = (V, E)$ and a function $w : E \to \mathbb{Z}_+$. We let $X = 2^V$ and $f(S) = w(S, \bar{S})$ i.e. $f(S)$ is the weight of the cut $S : \bar{S}$. We then let $N_S = \{T : |T \setminus S| + |S \setminus T| = 1\}$. Let $W = \sum_{e \in E} w(e)$.

Theorem 15. *Local search finds a solution $\widehat{S}$ such that $f(\widehat{S}) \geq OPT/2$. It requires at most W iterations.*

Proof.

$$f(\widehat{S}) \geq f(\widehat{S} \cup \{v\}) \text{ for } v \notin \widehat{S} \text{ implies that } w(v, \widehat{S}) \geq w(v, \bar{\widehat{S}}).$$
$$f(\widehat{S}) \geq f(\widehat{S} \setminus \{v\}) \text{ for } v \in \widehat{S} \text{ implies that } w(v, \bar{\widehat{S}}) \geq w(v, \widehat{S}).$$

So,

$$\begin{aligned} 4f(\widehat{S}) &= 2\sum_{v \in \widehat{S}} w(v, \bar{\widehat{S}}) + 2\sum_{v \notin \widehat{S}} w(v, \widehat{S}) \\ &\geq \sum_{v \in \widehat{S}} w(v, \bar{\widehat{S}}) + \sum_{v \notin \widehat{S}} w(v, \widehat{S}) + \sum_{v \in \widehat{S}} w(v, \widehat{S}) + \sum_{v \notin \widehat{S}} w(v, \bar{\widehat{S}}) \\ &= 2W \geq 2OPT. \end{aligned}$$

□

The bound W on the number of iterations might be excessive. We can reduce this to a polynomial at a small degradation in performance. Let $w_{\max} = \max\{w(e) : e \in E\}$ and $N = |E|w_{\max}/(\varepsilon W)$. Then let $w^*(e) = \lceil Nw(e)/w_{\max} \rceil$ for $e \in E$ and $W^* = \sum_{e \in E} w^*(e)$.

Suppose we run the above algorithm, using w^* in place of w. Then we have

$$\sum_{e \in \widehat{S}} \frac{Nw(e)}{w_{\max}} + |E| \geq \frac{W^*}{2} \geq \frac{NW}{2w_{\max}}.$$

So,

$$w(\widehat{S}) \geq \frac{W}{2} - \frac{w_{\max}|E|}{N} \geq W\left(\frac{1}{2} - \varepsilon\right).$$

The running time is at most $nW^* \leq 2n|E|/\varepsilon$.

12 Optimization Problems

We consider the following problem:

$$\text{Minimize } f(\mathbf{x}) \text{ subject to } \mathbf{x} \in S, \tag{57}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $S \subseteq \Re^n$.

Example: $f(\mathbf{x}) = \mathbf{c^T x}$ and $S = \{\mathbf{x} \in \Re^n : A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$ – Linear Programming.

Local versus Global Optima: $\mathbf{x}^*$ is a *global minimum* if it is an actual minimizer in (57).

$\mathbf{x}^*$ is a *local minimum* if there exists $\delta > 0$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in B(\mathbf{x}^*) \cap S$, where $B(\mathbf{x}, \delta) = \{\mathbf{y} : |\mathbf{y} - \mathbf{x}| \leq \delta\}$ is the *ball* of radius δ, centred at $\mathbf{x}$.

See Diagram 1 at the end of these notes.

If $S = \emptyset$ then we say that the problem is *unconstrained*, otherwise it is *constrained*.

13 Convex sets and functions

13.1 Convex Functions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *convex* if

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda\mathbf{f}(\mathbf{x}) + (\mathbf{1}-\lambda)\mathbf{f}(\mathbf{y}).$$

See Diagram 2 at the end of these notes.

Examples of convex functions:

F1 A linear function $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x}$ is convex.

F2 If $n = 1$ then f is convex iff

$$f(y) \geq f(x) + f'(x)(y-x) \text{ for all } x, y. \tag{58}$$

Proof. Suppose first that f is convex. Then for $0 < \lambda \leq 1$,

$$f(x + \lambda(y-x)) \leq (1-\lambda)f(x) + \lambda f(y).$$

Thus, putting $h = \lambda(y-x)$ we have

$$f(y) \geq f(x) + \frac{f((x+h) - f(x))}{h}(y-x).$$

Taking the limit as $\lambda \to 0$ implies (58).

Now suppose that (58) holds. Choose $x \neq y$ and $0 \leq \lambda \leq 1$ and let $z = \lambda x + (1-\lambda)y$. Then we have

$$f(x) \geq f(z) + f'(z)(x-z) \text{ and } f(y) \geq f(z) + f'(z)(y-z).$$

Multiplying the first inequality by λ and the second by $1-\lambda$ and adding proves that

$$\lambda f(x) + (1-\lambda)f(y) \geq f(z).$$

□

F3 If $n \geq 1$ then f is convex iff $f(\mathbf{y}) \geq \mathbf{f}(\mathbf{x}) + (\nabla \mathbf{f}(\mathbf{x}))^{\mathbf{T}}(\mathbf{y} - \mathbf{x})$ for all $\mathbf{x}, \mathbf{y}$.
Apply F2 to the function $h(t) = f(t\mathbf{x} + (1-t)\mathbf{y})$.

F4 A $n = 1$ and f is twice differentiable then f is convex iff $f''(z) \geq 0$ for all $z \in \mathbb{R}$.

Proof. Taylor's theorem implies that

$$f(y) = f(x) + f'(x)(y-x) + \frac{1}{2}f''(z)(y-x)^2 \text{ where } z \in [x, y].$$

We now just apply (58). □

F5 It follows from F4 that e^{ax} is convex for any $a \in \mathbb{R}$.

F6 x^a is convex on $\mathbb{R}_+$ for $a \geq 1$ or $a \leq 0$. x^a is concave for $0 \leq a \leq 1$.
Here f is *concave* iff $-f$ is convex.

F7 Suppose that A is a symmetric $n \times n$ positive semi-definite matrix. Then $Q(\mathbf{x}) = \mathbf{x}^T A\mathbf{x}$ is convex.
By positive semi-definite we mean that $Q(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$.
We have

$$\begin{aligned} & Q(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) - \lambda\mathbf{Q}(\mathbf{x}) - (\mathbf{1}-\lambda)\mathbf{Q}(\mathbf{y}) && (59)\\ =& \lambda^2 Q(\mathbf{x}) + (1-\lambda)^2 Q(\mathbf{y}) + \mathbf{2}\lambda(\mathbf{1}-\lambda)\mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{y} - \lambda\mathbf{Q}(\mathbf{x}) - (\mathbf{1}-\lambda)\mathbf{Q}(\mathbf{y}) && (60)\\ =& -\lambda(1-\lambda)Q(\mathbf{y}-\mathbf{x}) \leq \mathbf{0}. && (61) \end{aligned}$$

F8 If $n \geq 1$ then f is convex iff $\nabla^2 F = \left[\frac{\partial f^2}{dx_i dx_j}\right]$ is positive semi-definite for all $\mathbf{x}$.
Apply F7 to the function $h(t) = f(\mathbf{x} + t\mathbf{d})$ for all $\mathbf{x}, \mathbf{d} \in \mathbb{R}^n$.

Operations on convex functions

E1 If f, g are convex, then $f + g$ is convex.

E2 If $\lambda > 0$ and f is convex, then λf is convex.

E3 If f, g are convex then $h = \max\{f, g\}$ is convex.

Proof.

$$\begin{aligned} h(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) &= \max\{f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}), \mathbf{g}(\lambda\mathbf{x} + (\mathbf{1}-\lambda)\mathbf{y})\} && (62)\\ &\leq \max\{\lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}), \lambda\mathbf{g}(\mathbf{x}) + (\mathbf{1}-\lambda)\mathbf{g}(\mathbf{y})\} && (63)\\ &\leq \lambda \max\{f(\mathbf{x}), g(\mathbf{x})\} + (1-\lambda)\max\{f(\mathbf{y}), \mathbf{g}(\mathbf{y})\} && (64)\\ &= \lambda h(\mathbf{x}) + (1-\lambda)h(\mathbf{y}). && (65) \end{aligned}$$

□

Jensen's Inequality
If f is convex and $\mathbf{a}_i \in \mathbb{R}^n, \lambda_i \in \mathbb{R}_+, 1 \leq i \leq m$ and $\lambda_1 + \lambda_2 + \cdots + \lambda_m = 1$ then

$$f\left(\sum_{i=1}^{m} \lambda_i \mathbf{a}_i\right) \leq \sum_{i=1}^{m} f(\lambda_i \mathbf{a}_i).$$

The proof is by induction on m. $m = 2$ is from the definition of convexity and then we use

$$\sum_{i=1}^{m} \lambda_i \mathbf{a}_i = \lambda_m \mathbf{a}_m + (1 - \lambda_m) \sum_{i=1}^{m-1} \frac{\lambda_i}{1 - \lambda_m} \mathbf{a}_i.$$

Application: Arithmetic versus geometric mean.
Suppose that $a_1, a_2, \ldots, a_m \in \mathbb{R}_+$. Then

$$\frac{a_1 + a_2 + \cdots + a_m}{m} \geq (a_1 a_2 \cdots a_m)^{1/m}. \tag{66}$$

$-\log(x)$ is a convex function for $x \geq 0$. So, applying (66),

$$-\log\left(\sum_{i=1}^{m} \lambda_i \mathbf{a}_i\right) \leq \sum_{i=1}^{m} -\log(\lambda_i \mathbf{a}_i).$$

Now let $\lambda_i = 1/m$ for $i = 1, 2, \ldots, m$.

13.2 Convex Sets

A set $S \subseteq \mathbb{R}^n$ is said to be *convex* if $\mathbf{x}, \mathbf{y} \in \mathbf{S}$ then the *line segment*

$$L(\mathbf{x}, \mathbf{y}) = \{\lambda \mathbf{x} + (\mathbf{1} - \lambda)\mathbf{y} \in \mathbf{S} : \mathbf{0} \leq \lambda \leq \mathbf{1}\}.$$

See Diagram 3 at the end of these notes.

Examples of convex sets:

C1 $S = \left\{\mathbf{x} : \mathbf{a}^T\mathbf{x} = 1\right\}$. $\mathbf{x}, \mathbf{y} \in \mathbf{S}$ implies that

$$\mathbf{a}^T(\lambda \mathbf{x} + (1 - \lambda)\mathbf{y}) = \lambda \mathbf{a^T x} + (\mathbf{1} - \lambda)\mathbf{a^T y} = \lambda + (\mathbf{1} - \lambda) = \mathbf{1}.$$

C2 $S = \left\{\mathbf{x} : \mathbf{a}^T\mathbf{x} \leq 1\right\}$. Proof similar to C1.

C3 $S = B(0, \delta)$: $\mathbf{x}, \mathbf{y} \in \mathbf{S}$ implies that

$$|\lambda \mathbf{x} + (1 - \lambda)\mathbf{y}| \leq |\lambda \mathbf{x}| + |(\mathbf{1} - \lambda)\mathbf{y}| \leq \lambda\delta + (\mathbf{1} - \lambda)\delta = \delta.$$

C4 If f is convex, then the *level set* $\{\mathbf{x} : f(\mathbf{x}) \leq 0\}$ is convex.
$f(\mathbf{x}), f(\mathbf{y}) \leq \mathbf{0}$ implies that $f(\lambda \mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda \mathbf{f}(\mathbf{x}) + (\mathbf{1} - \lambda)\mathbf{f}(\mathbf{y}) \leq \mathbf{0}$.

Operations on convex sets:

O1 S convex and $\mathbf{x} \in \mathbb{R}^n$ implies that $\mathbf{x} + S = \{\mathbf{x} + \mathbf{y} : \mathbf{y} \in \mathbf{S}\}$ is convex.

O2 S, T convex implies that $A = S \cap T$ is convex. $\mathbf{x}, \mathbf{y} \in \mathbf{A}$ implies that $\mathbf{x}, \mathbf{y} \in \mathbf{S}$ and so $L = L(\mathbf{x}, \mathbf{y}) \subseteq \mathbf{S}$. Similarly, $L \subseteq T$ and so $L \subseteq S \cap T$.

O3 Using induction we see that if $S_i, 1 \leq i \leq k$ are convex then so is $\bigcap_{i=1}^{k} S_i$.

O4 If S, T are convex sets and $\alpha, \beta \in \mathbb{R}$ then $\alpha S + \beta T = \{\alpha \mathbf{x} + \beta \mathbf{y}\}$ is convex.
If $\mathbf{z}_i = \alpha \mathbf{x}_i + \beta \mathbf{y_i} \in \mathbf{T}, \mathbf{i} = \mathbf{1}, \mathbf{2}$ then

$$\lambda \mathbf{z}_1 + (1-\lambda)\mathbf{z}_2 = \alpha(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) + \beta(\lambda \mathbf{y_1} + (\mathbf{1}-\lambda)\mathbf{y_2}) \in \mathbf{T}.$$

It follows from C1,C2 and O3 that an affine subspace $\{\mathbf{x} : A\mathbf{x} = \mathbf{b}\}$ and a halfspace $\{\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}$ are convex for any matrix A any vector $\mathbf{b}$.

We now prove something that implies the importance of the above notions. Most optimization algorithms can only find local minima. We do however have the following theorem:

Theorem 16. *Let f, S both be convex in (57). Then if $\mathbf{x}^*$ is a local minimum, it also a global minimum.*

Proof.

See Diagram 4 at the end of these notes.

Let δ be such that $\mathbf{x}^*$ minimises f in $B(\mathbf{x}^*, \delta) \cap S$ and suppose that $\mathbf{x} \in S \setminus B(\mathbf{x}^*, \delta)$. Let $\mathbf{z} = \lambda \mathbf{x}^* + (1-\lambda)\mathbf{y}$ be the point on $L(\mathbf{x}^*, \mathbf{y})$ at distance δ from $\mathbf{x}^*$. Note that $\mathbf{x} \in S$ by convexity of S. Then by the convexity of f we have

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \leq \lambda f(\mathbf{x}^*) + (1-\lambda) f(\mathbf{x})$$

and this implies that $f(\mathbf{x}^*) \leq f(\mathbf{x})$. □

The following shows the relationship between convex sets and functions.

Lemma 17. *let $f_1, f_2, \ldots, f_m$ be convex functions on $\mathbb{R}^n$. Let $\mathbf{b} \in \mathbb{R}^m$ and let*

$$S = \{\mathbf{x} \in \mathbb{R}^n : f_i(\mathbf{x}) \leq b_i, i = 1, 2, \ldots, m\}.$$

Then S is convex.

Proof. It follows from O3 that we can consider the case $m = 1$ only and drop the subscript. Suppose now that $\mathbf{x}, \mathbf{y} \in \mathbf{S}$ i.e. $f(\mathbf{x}), f(\mathbf{y}) \leq \mathbf{b}$. Then for $0 \leq \lambda \leq 1$

$$f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda \mathbf{f}(\mathbf{x}) + (\mathbf{1}-\lambda)\mathbf{f}(\mathbf{y}) \leq \lambda \mathbf{b} + (\mathbf{1}-\lambda)\mathbf{b} = \mathbf{b}.$$

So, $\lambda \mathbf{x} + (1-\lambda)\mathbf{y} \in \mathbf{S}$. □

14 Algorithms

14.1 Line search $-$ $n = 1$

Here we consider the simpler problem of minimising a convex (more generally *unimodal*) function $f : \mathbb{R} \to \mathbb{R}$.

See Diagram 5 at the end of these notes.

We assume that we are given a_0, a_1 such that $a_0 \leq x^* \leq a_1$ where x^* minimises f. This is not a significant assumption. We can start with $a_0 = 0$ and then consider the sequences $\zeta_i = f(2^i), \xi_i = f(-2^i)$ until we find $\zeta_{i-1} \leq \min\{\zeta_0, \zeta_i\}$ (resp. $\xi_{i-1} \leq \min\{\xi_0, \xi_i\}$). Then we know that $x^* \in [\zeta_0, \zeta_i]$ (resp. $x^* \in [\xi_0, \xi_i]$).

Assume then that we have an interval $[a_0, a_1]$ of uncertainty for x^*. Furthermore, we will have evaluated f at two points in this interval, two points inside the interval at $a_2 = a_0 + \alpha^2(a_1 - a_0)$ and $a_3 = a_0 + \alpha(a_1 - a_0)$ respectively. We will determine α shortly. And at each iteration we make one new function evaluation and decrease the interval of uncertainty by a factor α. There are two possibilities:

(i) $f(a_2) \leq f(a_3)$. This implies that $x^* \in [a_0, a_3]$. So, we evaluate $f(a_0 + \alpha^2(a_3 - a_0))$ and make the changes $a_i \to a_i'$:

$$a_0' \leftarrow a_0,\ a_1' \leftarrow a_3,\ a_2' \leftarrow a_0 + \alpha^2(a_3 - a_0),\ a_3' \leftarrow a_2.$$

(ii) $f(a_2) > f(a_3)$. This implies that $x^* \in [a_2, a_1]$. So, we evaluate $f(a_0+)$ and make the changes $a_i \to a_i'$:

$$a_0' \leftarrow a_2,\ a_1' \leftarrow a_1,\ a_2' \leftarrow a_3,\ a_3' \leftarrow a_2 + \alpha^2(a_1 - a_0).$$

In case (i) we see that $a_1' - a_0' = a_3 - a_0 = \alpha(a_1 - a_0)$ and so the interval has shrunk by the required amount. Next we see that $a_2' - a_0' = \alpha^2(a_3 - a_0) = \alpha^2(a_1' - a_0)$. Furthermore, $a_3' - a_0' = a_2 - a_0 = \alpha^2(a_1 - a_0) = \alpha(a_1' - a_0')$.

In case (ii) we see that $a_1' - a_0' = a_1 - a_2 = a_1 - (a_0 + \alpha^2(a_1 - a_0)) = (1 - \alpha^2)(a_1 - a_0)$. So, shrink by α in this case we choose α to satisfy $1 - \alpha^2 = \alpha$. This gives us

$$\alpha = \frac{\sqrt{5} - 1}{2} \text{ – } \textit{the golden ratio.}$$

Next we see that $a_2' - a_0' = a_3 - a_2 = (\alpha - \alpha^2)(a_1 - a_0) = \frac{\alpha - \alpha^2}{\alpha}(a_1' - a_0') = (1 - \alpha)(a_1' - a_0') = \alpha^2(a_1' - a_0')$. Finally, we have $a_3' - a_0' = a_2 + \alpha^2(a_1 - a_0) - a_2 = \alpha^2(a_1 - a_0) = \alpha(a_1' - a_0')$.

Thus to achieve an accuracy within δ of x^* we need to take t steps, where $\alpha^t D \leq \delta$ where D is our initial uncertainty.

14.2 Gradient Descent

See Diagram 6 at the end of these notes.

Here we consider the unconstrained problem. At a point $\mathbf{x} \in \mathbb{R}^n$, if we move a small distance h in direction $\mathbf{d}$ then we have

$$f(\mathbf{x} + h\mathbf{d}/|\mathbf{d}|) = f(\mathbf{x}) + h(\nabla f)^T \frac{\mathbf{d}}{|\mathbf{d}|} + O(h^2) \geq f(\mathbf{x}) - h|\nabla f| + O(h^2).$$

Thus, at least infinitessimally, the best direction is $-\nabla f$. So, for us, the steepest algorithm will follow a sequence of points $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k, \ldots$, where

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k).$$

Then we have

$$|\mathbf{x}_{k+1} - \mathbf{x}^*|^2 = |\mathbf{x}_k - \mathbf{x}^*|^2 - 2\alpha_k \nabla f(\mathbf{x}_k)^T(\mathbf{x}_k - \mathbf{x}^*) + \alpha_k^2 |\nabla f(\mathbf{x}_k)^2| \tag{67}$$

$$\leq |\mathbf{x}_k - \mathbf{x}^*|^2 - 2\alpha_k(f(\mathbf{x}_k) - f(\mathbf{x}^*)) + \alpha_k^2 |\nabla f(\mathbf{x}_k)|^2. \tag{68}$$

The inequality comes from F3.

Applying (68) repeatedly we get

$$|\mathbf{x}_k - \mathbf{x}^*|^2 \leq |\mathbf{x}_0 - \mathbf{x}^*|^2 - 2\sum_{i=1}^{k} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) + \sum_{i=1}^{K} \alpha_i^2 |\nabla f(\mathbf{x}_k)|^2. \tag{69}$$

Putting $R = |\mathbf{x}_0 - \mathbf{x}^*|$, we see from (69) that

$$2\sum_{i=1}^{k} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) \leq R^2 + \sum_{i=1}^{K} \alpha_i^2 |\nabla f(\mathbf{x}_k)|^2. \tag{70}$$

On the other hand,

$$\sum_{i=1}^{k} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) \geq \left(\sum_{i=1}^{k} \alpha_i\right) \min\{f(\mathbf{x}_k) - f(\mathbf{x}^*) : i \in [k]\} = \left(\sum_{i=1}^{k} \alpha_i\right) (f(\mathbf{x}_{min} - f(\mathbf{x}^*)), \tag{71}$$

where $f(\mathbf{x}_{min}) = \min\{f(\mathbf{x}_i) : i \in [k]\}$.

Combining (70) and (71) we get

$$f(\mathbf{x}_{min}) - f(\mathbf{x}^*) \leq \frac{R^2 + G^2 \sum_{i=1}^{k} \alpha_i^2}{2\sum_{i=1}^{k} \alpha_i}, \tag{72}$$

where $G = \max\{|\nabla f(\mathbf{x}_i)| : i \in [\kappa]\}$.

So, if we choose α_k so that $\sum_{i=1}^{\infty} \alpha_i = \infty$ and $\sum_{i=1}^{\infty} \alpha_i^2 = O(1)$ then

$$|f(\mathbf{x}_{min}) - f(\mathbf{x}^*)| \to 0 \text{ as } k \to \infty. \tag{73}$$

As an example, we could let $\alpha_i = 1/i$.

15 Separating Hyperplane

See Diagram 7 at the end of these notes.

Theorem 18. *Let C be a convex set in $\mathbb{R}^n$ and suppose $\mathbf{x} \notin C$. Then there exists $\mathbf{0} \neq \mathbf{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ such that (i) $\mathbf{a}^T\mathbf{x} \geq b$ and (ii) $C \subseteq \{\mathbf{y} \in \mathbb{R}^{\mathbf{n}} : \mathbf{a}^{\mathbf{T}}\mathbf{y} \leq \mathbf{b}\}$.*

Proof.
Case 1: C is closed.
Let $\mathbf{z}$ be the closest point in C to $\mathbf{x}$. Let $\mathbf{a} = \mathbf{x} - \mathbf{z} \neq \mathbf{0}$ and $b = (\mathbf{x} - \mathbf{z})^T\mathbf{z}$. Then

$$\mathbf{a}^T\mathbf{x} - b = (\mathbf{x} - \mathbf{z})^T\mathbf{x} - (\mathbf{x} - \mathbf{z})^T\mathbf{z} = |\mathbf{x} - \mathbf{z}|^2 > 0.$$

This verifies (i). Suppose (ii) fails and there exists $\mathbf{y} \in \mathbf{C}$ such that $\mathbf{a}^T\mathbf{y} > \mathbf{b}$. Let $\mathbf{w} \in C$ be the closest point to $\mathbf{x}$ on the line segment $L(\mathbf{y}, \mathbf{z}) \subseteq \mathbf{C}$. The triangle formed by $\mathbf{x}, \mathbf{w}, \mathbf{z}$ has a right angle at $\mathbf{w}$ and an acute angle at $\mathbf{z}$. This implies that $|\mathbf{x} - \mathbf{w}| < |\mathbf{x} - \mathbf{z}|$, a contradiction.

Case 2: $\mathbf{x} \notin \bar{C}$.
We observe that $\bar{C} \supseteq C$ and is convex (exercise). We can thus apply Case 1, with $\bar{C}$ replacing C.

Case 3: $\mathbf{x} \in \bar{C} \setminus C$**.** Every ball $B(\mathbf{x}, \delta)$ contains a point of $\mathbb{R}^n \setminus \bar{C}$ that is distinct from $\mathbf{x}$. Choose a sequence $\mathbf{x}_n, \notin \bar{C}, n \geq 1$ that tends to $\mathbf{x}$. For each $\mathbf{x}_n$, let $\mathbf{a}_n, b_n = \mathbf{a}_n^T \mathbf{z}_n$ define a hyperplane that separates $\mathbf{x}_n$ from $\bar{C}$, as in Case 2. We can assume that $|\mathbf{a}_n| = 1$ (scaling) and that b_n is in some bounded set and so there must be a convergent subsequence of $(\mathbf{a}_n, b_n), n \geq 1$ that converges to $(\mathbf{a}, b), |\mathbf{a}| = 1$. Assume that we re-label so that this subsequence is $(\mathbf{a}_n), n \geq 1$. Then for $\mathbf{y} \in \bar{\mathbf{C}}$ we have $\mathbf{a}_n^T \mathbf{y} \leq \mathbf{b_n}$ for all n. Taking limits we see that $\mathbf{a}^T \mathbf{y} \leq \mathbf{b}$. Furthermore, for $\mathbf{y} \notin \bar{\mathbf{C}}$ we see that for large enough n, $\mathbf{a}_n^T \mathbf{y} > \mathbf{b_n}$. taking limits we see that $\mathbf{a}^T \mathbf{y} \leq \mathbf{b}$. □

Corollary 19. *Suppose that $S, T \subseteq \mathbb{R}^n$ are convex and that $S \cap T = \emptyset$. Then there exists $\mathbf{a}, b$ such that $\mathbf{a}^T \mathbf{x} \leq b$ for all $\mathbf{x} \in S$ and $\mathbf{a}^T \mathbf{x} \geq b$ for all $\mathbf{x} \in T$.*

Proof. Let $W = S + (-1)T$. Then $\mathbf{0} \notin W$ and applying Theorem 18 we see that there exists $\mathbf{a}$ such that $\mathbf{a}^T \mathbf{z} \leq 0$ for all $\mathbf{z} \in W$. Now put

$$b = \frac{1}{2}\left(\sup_{\mathbf{x} \in S} \mathbf{a}^T \mathbf{x} + \inf_{\mathbf{x} \in T} \mathbf{a}^T \mathbf{x}\right).$$

□

Corollary 20 (Farkas Lemma)**.** *For an $m \times n$ matrix and $\mathbf{b} \in \mathbb{R}^m$, exactly one of the following holds:*

(i) There exists $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x} \geq \mathbf{0}, A\mathbf{x} = \mathbf{b}$.

(ii) There exists $\mathbf{u} \in \mathbb{R}^{\mathbf{m}}$ such that $\mathbf{u^T A} \geq \mathbf{0}$ and $\mathbf{u^T b} < \mathbf{0}$.

Proof. We cannot have both (i), (ii) holding. For then we have

$$0 \leq \mathbf{u^T A x} = \mathbf{u^T b} < \mathbf{0}.$$

Suppose then that (i) fails to hold. Let $S = \{\mathbf{y} : \mathbf{y} = \mathbf{Ax} \text{ for some } \mathbf{x} \geq \mathbf{0}\}$. Then $\mathbf{b} \notin S$ and since S is closed there exists $\boldsymbol{\alpha}, \beta$ such that (a) $\boldsymbol{\alpha}^T \mathbf{b} \leq \beta$ and (b) $\boldsymbol{\alpha}^T A\mathbf{x} \geq \beta$ for all $\mathbf{x} \geq 0$. This implies that $\boldsymbol{\alpha}^T(\mathbf{b} - A\mathbf{x}) \leq 0$ for all $\mathbf{x} \geq 0$. This then implies that $\mathbf{u} = \boldsymbol{\alpha}$ satisfies (ii). □

15.1 Convex Hulls

See Diagram 8 at the end of these notes.

Given a set $S \subseteq \mathbb{R}^n$, we let

$$conv(S) = \left\{\sum_{i \in I} \lambda_i \mathbf{x}_i : (i)\ |I| < \infty,\ (ii)\ \sum_{i \in I} \lambda_i = 1,\ (iii)\ \lambda_i > 0, i \in I,\ (iv)\ \mathbf{x}_i \in S, i \in I\right\}.$$

Clearly $S \subseteq conv(S)$, since we can take $|I| = 1$.

Lemma 21. *$conv(S)$ is a convex set.*

Proof. Let $\mathbf{x} = \sum_{i \in I} \lambda_i \mathbf{x}_i, \mathbf{y} = \sum_{\mathbf{j} \in \mathbf{J}} \mu_{\mathbf{j}} \mathbf{y_j} \in \mathbf{conv(S)}$. Let $K = I \cup J$ and put $\lambda_i = 0, i \in J \setminus I$ and $\mu_j = 0, j \in I \setminus J$. Then for $0 \leq \alpha \leq 1$ we see that

$$\alpha\mathbf{x} + (1-\alpha)\mathbf{y} = \sum_{\mathbf{i} \in \mathbf{K}} (\alpha\lambda_{\mathbf{i}} + (\mathbf{1} - \alpha)\mu_{\mathbf{i}})\mathbf{x_i} \text{ and } \sum_{\mathbf{i} \in \mathbf{K}} (\alpha\lambda_{\mathbf{i}} + (\mathbf{1} - \alpha)\mu_{\mathbf{i}}) = \mathbf{1}$$

implying that $\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \in \mathbf{conv(S)}$ i.e. $conv(S)$ is convex. □

Lemma 22. *If S is convex, then $S = conv(S)$.*

Proof. Exercise. □

Corollary 23. *$conv(conv(S)) = conv(S)$ for all $S \subseteq \mathbb{R}^n$.*

Proof. Exercise. □

15.1.1 Extreme Points

A point $\mathbf{x}$ of a convex set S is said to be an *extreme point* if **THERE DO NOT EXIST** $\mathbf{y}, \mathbf{z} \in \mathbf{S}$ such that $\mathbf{x} \in L(\mathbf{y}, \mathbf{z})$. We let $ext(S)$ denote the set of extreme points of S.

EX1 If $n = 1$ and $S = [a, b]$ then $ext(S) = \{a, b\}$.

EX2 If $S = B(0, 1)$ then $ext(S) = \{\mathbf{x} : |\mathbf{x}| = 1\}$.

EX3 If $S = \{\mathbf{x} : A\mathbf{x} = \mathbf{b}\}$ is the set of solutions to a set of linear equations, then $ext(S) = \emptyset$.

Theorem 24. *Let S be a closed, bounded convex set. Then $S = conv(ext(S))$.*

Proof. We prove this by induction on the dimension n. For $n = 1$ the result is trivial, since then S must be an interval $[a, b]$.

Inductively assume the result for dimensions less than n. Clearly, $S \supseteq T = conv(ext(S))$ and suppose there exists $\mathbf{x} \in S \setminus T$. Let $\mathbf{z}$ be the closest point of T to $\mathbf{x}$ and let $H = \left\{\mathbf{y} : \mathbf{a^T y} = \mathbf{b}\right\}$ be the hyperplane defined in Theorem 18. Let $b^* = \max\left\{\mathbf{a}^T\mathbf{y} : \mathbf{y} \in \mathbf{S}\right\}$. We have $b^* < \infty$ since S is bounded. Let $H^* = \left\{\mathbf{y} : \mathbf{a^T y} = \mathbf{b}^*\right\}$ and let $S^* = S \cap H^*$.

We observe that if $\mathbf{w}$ is a vertex of S^* then it is also a vertex of S. For if $\mathbf{w} = \lambda\mathbf{w}_1 + (1-\lambda)\mathbf{w}_2, \mathbf{w}_1, \mathbf{w}_2 \in S, 0 < \lambda < 1$ then we have

$$b^* = \mathbf{a}^T\mathbf{w} = \lambda\mathbf{a}^T\mathbf{w}_1 + (1-\lambda)\mathbf{a}^T\mathbf{w}_2 \leq \lambda b^* + (1-\lambda)b^* = b^*.$$

This implies that $\mathbf{a}^T\mathbf{w}_1 = \mathbf{a}^T\mathbf{w}_2 = b^*$ and so $\mathbf{w}_1, \mathbf{w}_2 \in S^*$, contradiction.

Now consider the point $\mathbf{w}$ on the half-line from $\mathbf{z}$ through $\mathbf{x}$ that lies in S^* i.e

$$\mathbf{w} = \mathbf{z} + \frac{b^* - b}{\mathbf{a}^T\mathbf{x} - b}(\mathbf{x} - \mathbf{z}).$$

Now by induction, we can write $\mathbf{w} = \sum_{i=1}^k \lambda_i\mathbf{w}_i$ where $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k$ are extreme points of S^* and hence of S. Also, $\mathbf{x} = \mu\mathbf{w} + (1-\mu)\mathbf{z}$ for some $0 < \mu \leq 1$ and so $\mathbf{x} \in ext(S)$. □

The following is sometimes useful.

Lemma 25. *Suppose that S is a closed bounded convex set and that f is a convex function. The f achieves its* maximum *at an extreme point.*

Proof. Suppose the maximum occurs at $\mathbf{x} = \lambda_1\mathbf{x}_1 + \cdots + \lambda_k\mathbf{x}_k$ where $0 \leq \lambda_1, \ldots, \lambda_k \leq 1$ and $\lambda_1 + \cdots + \lambda_k = 1$ and $\mathbf{x}_1, \ldots, \mathbf{x}_k \in ext(S)$. Then by Jensen's inequality we have $f(\mathbf{x}) \leq \lambda_1 f(\mathbf{x}_1) + \cdots + \lambda_k f(\mathbf{x}_k) \leq \max\{f(\mathbf{x}_i) : 1 \leq i \leq k\}$. □

This explains why the solutions to linear programs occur at extreme points.

16 Lagrangean Duality

See Diagram 9 at the end of these notes.

Here we consider the *primal problem*

$$\text{Minimize } f(\mathbf{x}) \text{ subject to } g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, m, \tag{74}$$

where $f, g_1, g_2, \ldots, g_m$ are convex functions on $\mathbb{R}^n$.

The Lagrangean

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g(\mathbf{x}).$$

The *dual problem* is

$$\text{Maximize } \phi(\boldsymbol{\lambda}) \text{ subject to } \boldsymbol{\lambda} \geq 0 \text{ where } \phi(\boldsymbol{\lambda}) = \min_{\mathbf{x}\in\mathbb{R}^n} L(\mathbf{x}, \boldsymbol{\lambda}). \tag{75}$$

We note that ϕ is a concave function. It is the minimum of a collection of convex (actually linear) functions of $\boldsymbol{\lambda}$ – see E3.

D1 :Linear programming. Let $f(\mathbf{x}) = \mathbf{c^T x}$ and $g_i(\mathbf{x}) = -\mathbf{a}_i^T\mathbf{x} + b_i$ for $i = 1, 2, \ldots, m$. Then

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \left(\mathbf{c^T} - \boldsymbol{\lambda^T A}\right)\mathbf{x} + \mathbf{b}^T\boldsymbol{\lambda} \text{ where } A \text{ has rows } \mathbf{a}_1, \ldots, \mathbf{a}_m.$$

It follows that $A\boldsymbol{\lambda} \neq \mathbf{c}$ implies that $\phi(\boldsymbol{\lambda}) = -\infty$. So the dual problem is

$$\text{Minimize } \mathbf{b}^T\boldsymbol{\lambda} \text{ subject to } A^T\boldsymbol{\lambda} = \mathbf{c}.$$

Weak Duality: If $\boldsymbol{\lambda}$ is feasible for (75) and $\mathbf{x}$ is feasible for (74) then $f(\mathbf{x}) \geq \phi(\boldsymbol{\lambda})$.

$$\phi(\boldsymbol{\lambda}) \leq L(\mathbf{x}, \boldsymbol{\lambda}) \leq f(\mathbf{x}) \text{ since } \lambda_i \geq 0, g_i(\mathbf{x}) \leq 0, \ i = 1, 2, \ldots, m. \tag{76}$$

Now note that $\phi(\boldsymbol{\lambda}) = -\infty$, unless $\mathbf{c^T} = \boldsymbol{\lambda^T A}$, since $\mathbf{x}$ is unconstrained in the definition of ϕ. And if $\mathbf{c^T} = \boldsymbol{\lambda^T A}$ then $\phi(\boldsymbol{\lambda}) = \mathbf{b}^T\boldsymbol{\lambda}$. So, the dual problem is to
Maximize $\mathbf{b}^T\boldsymbol{\lambda}$ subject to $\mathbf{c^T} = \boldsymbol{\lambda^T A}$ and $\boldsymbol{\lambda} \geq 0$, i.e. the LP dual.

Strong Duality: We give a sufficient condition *Slater's Constraint Condition* for tightness in (76).

Theorem 26. *Suppose that there exists a point* $\mathbf{x}^*$ *such that* $g_i(\mathbf{x}^*) < 0, i = 1, 2, \ldots, m$. *Then*

$$\max_{\boldsymbol{\lambda}\geq\mathbf{0}} \phi(\lambda) = \min_{\mathbf{x}: g_i(\mathbf{x})\leq 0, i\in[m]} f(\mathbf{x}).$$

Proof. Let

$$\mathcal{A} = \{\mathbf{u}, t) : \exists \mathbf{x} \in \mathbb{R}^n, g_i(\mathbf{x}) \leq u_i, i = 1, 2, \ldots, m \text{ and } f(\mathbf{x}) \leq t\}.$$
$$\mathcal{B} = \left\{(0, s) \in \mathbb{R}^{m+1} : s < f^*\right\} \text{ where } f^* = \min_{\mathbf{x}: g_i(\mathbf{x}) \leq 0, i \in [m]} f(\mathbf{x}).$$

Now $\mathcal{A} \cap \mathcal{B} = \emptyset$ and so from Corollary 19 there exists $\boldsymbol{\lambda}, \gamma, b$ such that $(\boldsymbol{\lambda}, \gamma) \neq \mathbf{0}$ and

$$b \leq \min\left\{\boldsymbol{\lambda}^T \mathbf{u} + \gamma \mathbf{t} : (\mathbf{u}, \mathbf{t}) \in \mathcal{A}\right\}. \tag{77}$$
$$b \geq \max\left\{\boldsymbol{\lambda}^T \mathbf{u} + \gamma \mathbf{t} : (\mathbf{u}, \mathbf{t}) \in \mathcal{B}\right\}. \tag{78}$$

We deduce from (77) that $\boldsymbol{\lambda} \geq 0$ and $\bar{g} \geq 0$. If $\gamma < 0$ or $\lambda_i < 0$ for some i then the minimum in (77) is $-\infty$. We deduce from (78) that $\gamma t < b$ for all $t < f^*$ and so $\gamma f^* \leq b$. And from (77) that

$$\gamma f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) \geq b \geq \gamma f^* \qquad \text{for all } \mathbf{x} \in \mathbb{R}^n. \tag{79}$$

If $\gamma > 0$ then we can divide (79) by γ and see that $L(\mathbf{x}, \boldsymbol{\lambda}) \geq f^*$, and together with weak duality, we see that $L(\mathbf{x}, \boldsymbol{\lambda}) = f^*$.

If $\gamma = 0$ then substituting $\mathbf{x}^*$ into (79) we see that $\sum_{i=1}^m \lambda_i g_i(\mathbf{x}^*) \geq 0$ which then implies that $\boldsymbol{\lambda} = 0$, contradiction. □

17 Conditions for a minimum: First Order Condition

17.1 Unconstrained problem

We discuss necessary conditons for $\mathbf{a}$ to be a (local) minimum. (We are not assuming that f is convex.) We will assume that our functions are differentiable. Then Taylor's Theorem

$$f(\mathbf{a} + \mathbf{h}) = f(a) + (\nabla f(\mathbf{a}))^T \mathbf{h} + o(|\mathbf{h}|)$$

implies that

$$\nabla f(\mathbf{a}) = 0 \tag{80}$$

is a necessary condition for $\mathbf{a}$ to be a local minimum. Otherwise,

$$f\left(\mathbf{a} - t\nabla f(\mathbf{a})\right) \leq f(\mathbf{a}) - t|\nabla f(\mathbf{a})|^2/2$$

for small $t > 0$.

Of course (80) is not sufficient in general, $\mathbf{a}$ could be a local maximum. Generally spealking, one has to look at second order conditions to distinguish between local minima and local maxima.

However,

Lemma 27. *If f is convex then* (80) *is also a sufficient condition.*

Proof. This follows directly from F3. □

17.2 Constrained problem

We will consider Problem (74), but we will not assume convexity, only differentiability. The condition corresponding to (80) is the *Karush-Kuhn-Tucker* or KKT condition. Assume that $f, g_1, g_2, \ldots, g_m$ are differentiable. Then (subject to some *regularity conditions*, a necessary condition for $\mathbf{a}$ to be a local minimum (or maximum) to Problem (74) is that there exists $\boldsymbol{\lambda}$ such that

$$g_i(\mathbf{a}) \leq 0, \qquad 1 \leq i \leq m. \tag{81}$$

$$\lambda_i \geq 0 \qquad 1 \leq i \leq m. \tag{82}$$

$$\nabla f(\mathbf{a}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{a}) = 0. \tag{83}$$

$$\lambda_i g_i(\mathbf{a}) = 0, \qquad 1 \leq i \leq m. \qquad \text{Complementary Slackness} \tag{84}$$

The second condition says that only *active* constraints ($g_i(\mathbf{a}) = 0$) are involved in the first condition.

One deals with $g_i(\mathbf{x}) \geq 0$ via $-g_i(\mathbf{x}) \leq 0$ (and $\lambda_i \leq 0$) and $g_i(\mathbf{x}) = 0$ by $g_i(\mathbf{x}) \geq 0$ and $-g_i(\mathbf{x}) \leq 0$ (and λ_i not constrined to be non-negative or non-positive).

In the convex case, we will see that (83), (82) and (84) are sufficient for a global minimum.

17.2.1 Heuristic Justification of KKT conditions

See Diagram 10 at the end of these notes.

Suppose that $\mathbf{a}$ is a local minimum and assume w.l.o.g. that $g_i(\mathbf{a}) = 0$ for $i = 1, 2, \ldots, m$. Then (heuristically) Taylor's theorem implies that if (i) $\mathbf{h}^T \nabla g_i(\mathbf{a}) \leq 0, i = 1, 2, \ldots, m$ then (ii) we should have $\mathbf{h}^T \nabla f(\mathbf{a}) \geq 0$. (The heuristic argument is that (i) holds then we should have (iii) $\mathbf{a} + \mathbf{h}$ feasible for small $\mathbf{h}$ and then we should have (ii) since we are at a local minimum. You need a regularity condition to ensure that (ii) implies (iii).)

Applying Corollary 20 we see that the KKT conditions hold. We let A have *columns* $\nabla g_i(\mathbf{a}), i = 1, 2, \ldots, m$. Then the KKT conditions are $A\boldsymbol{\lambda} = -\nabla f(\mathbf{a})$.

Convex case: Suppose now that $f, g_1, \ldots, g_m$ are all convex functions and that $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ satisfies the KKT conditions. Now $\boldsymbol{\lambda}^* \geq 0$ implies that $\phi(\mathbf{x}) = L(\mathbf{x}, \boldsymbol{\lambda}^*)$ is a convex function of $\mathbf{x}$. Equation (83) and Lemma 27 implies that $\mathbf{x}^*$ minimises ϕ. But then for any feasible $\mathbf{x}$ we have

$$f(\mathbf{x}^*) = \phi(\mathbf{x}^*) \leq \phi(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i^* g_i(\mathbf{x}) \leq f(\mathbf{x}).$$

For much more on this subject see Convex Optimization, by Boyd and Vendenberghe.

Diagram 1

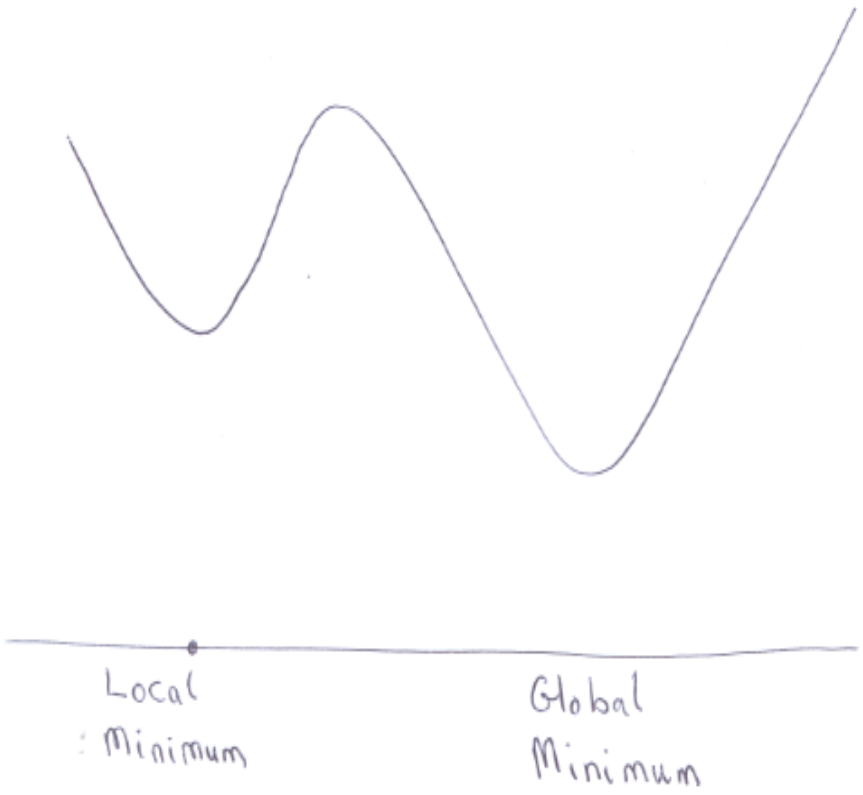
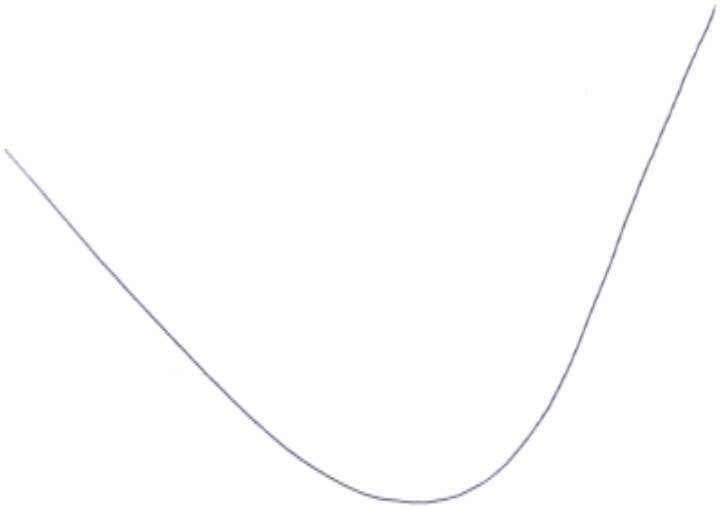
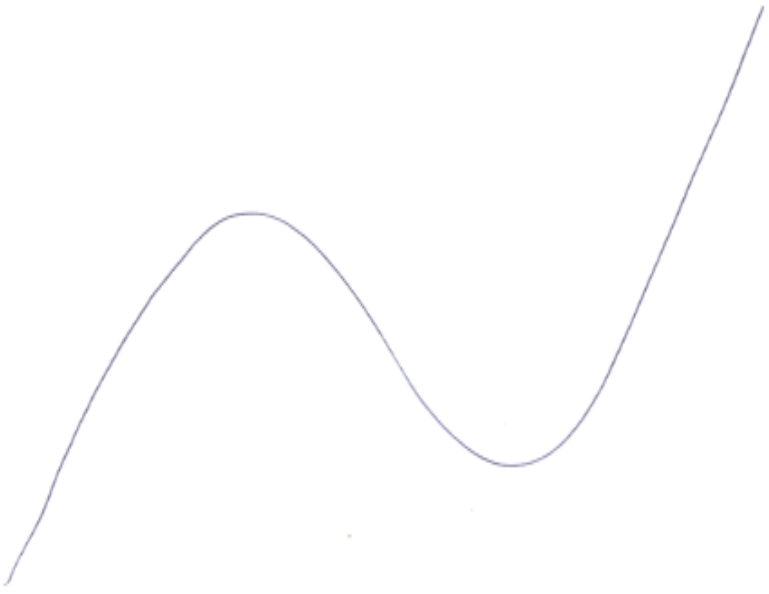


Diagram 2

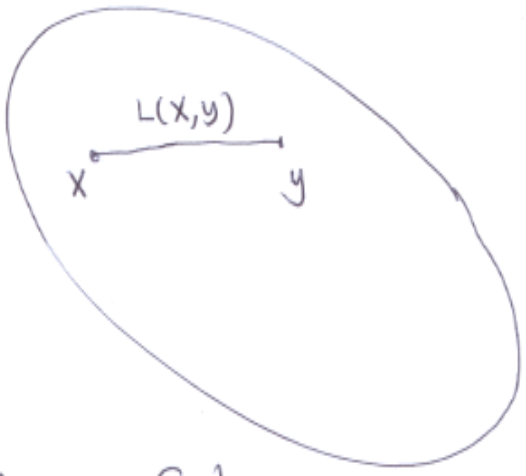


Convex Function

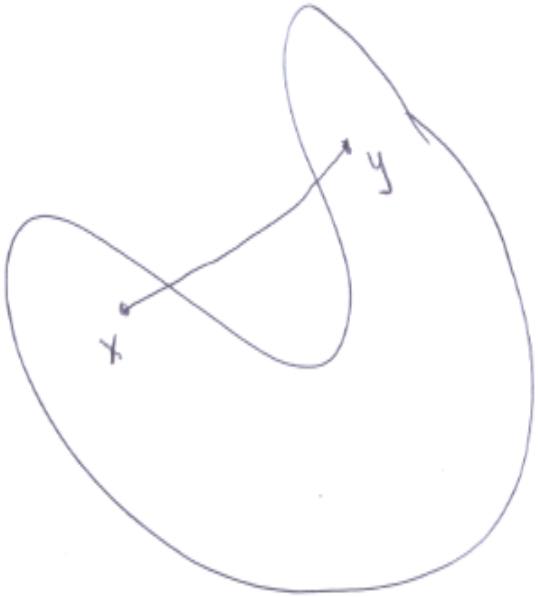


Non-Convex Function

Diagram 3



Convex Set



Non-Convex Set

Diagram 4

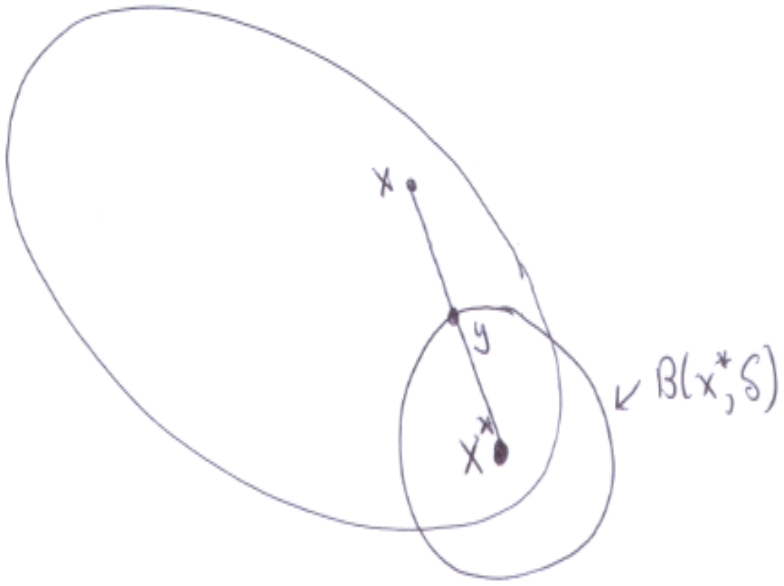


Diagram 5

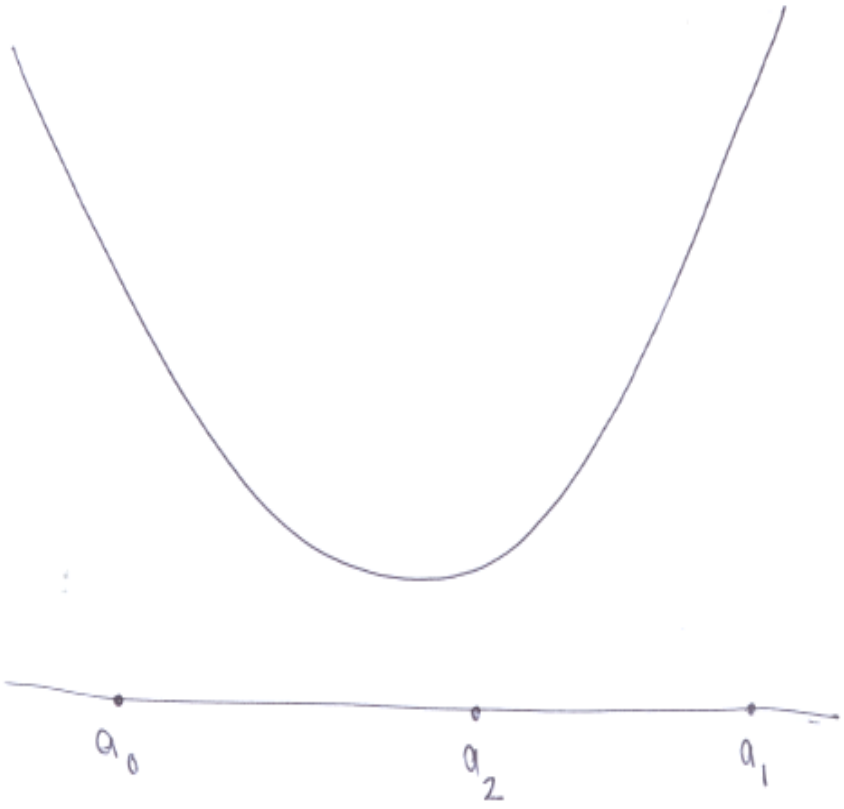


Diagram 6

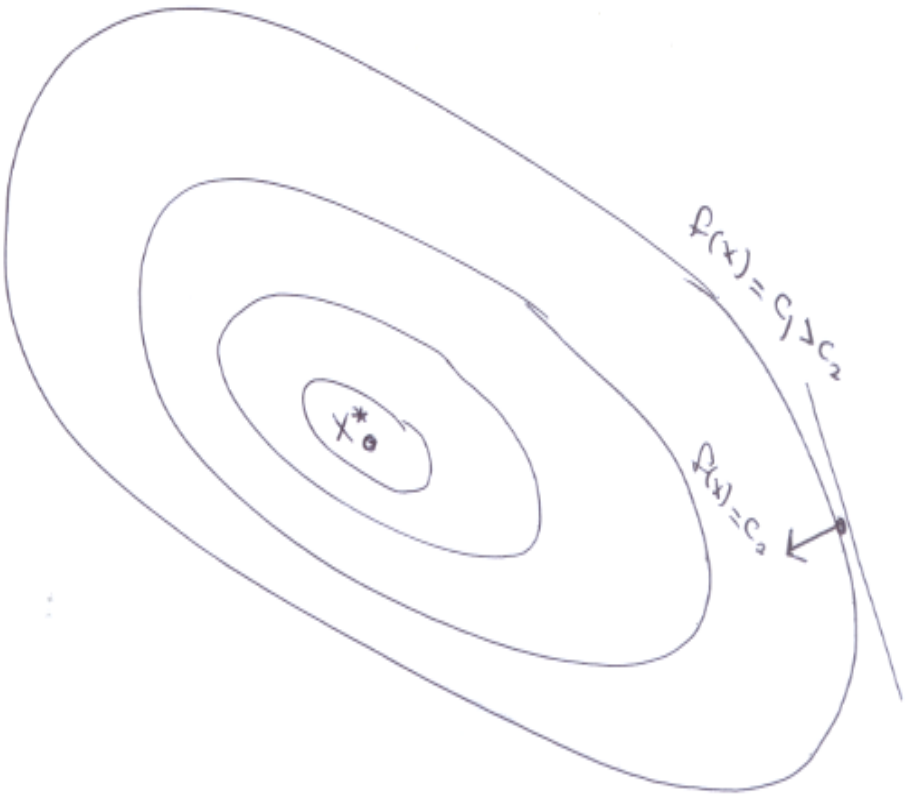


Diagram 7

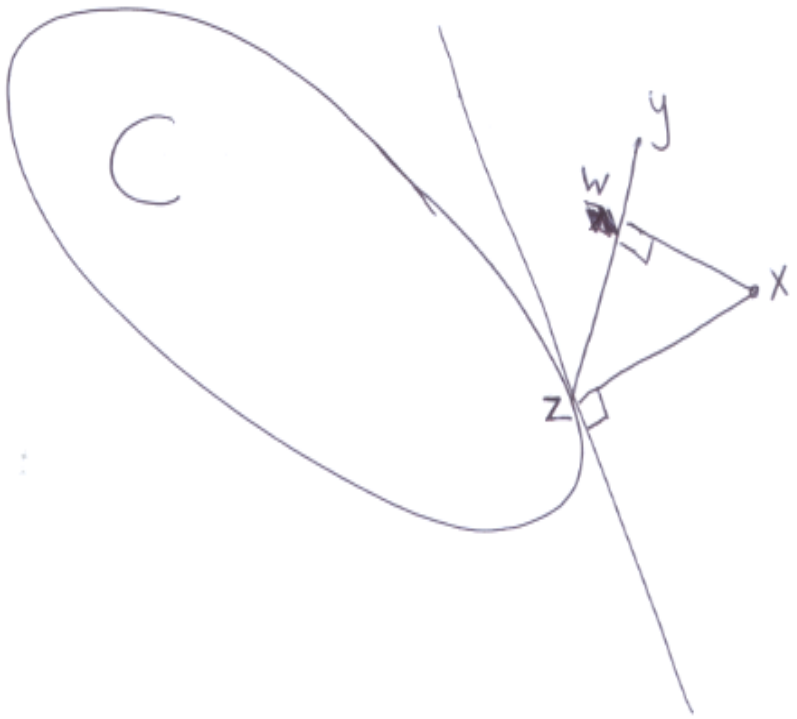


Diagram 8

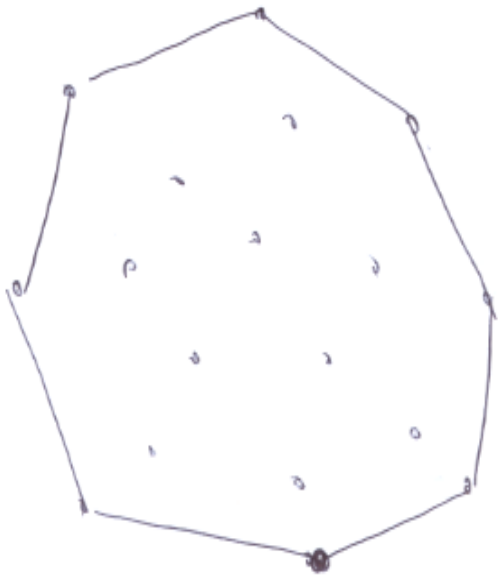


Diagram 9

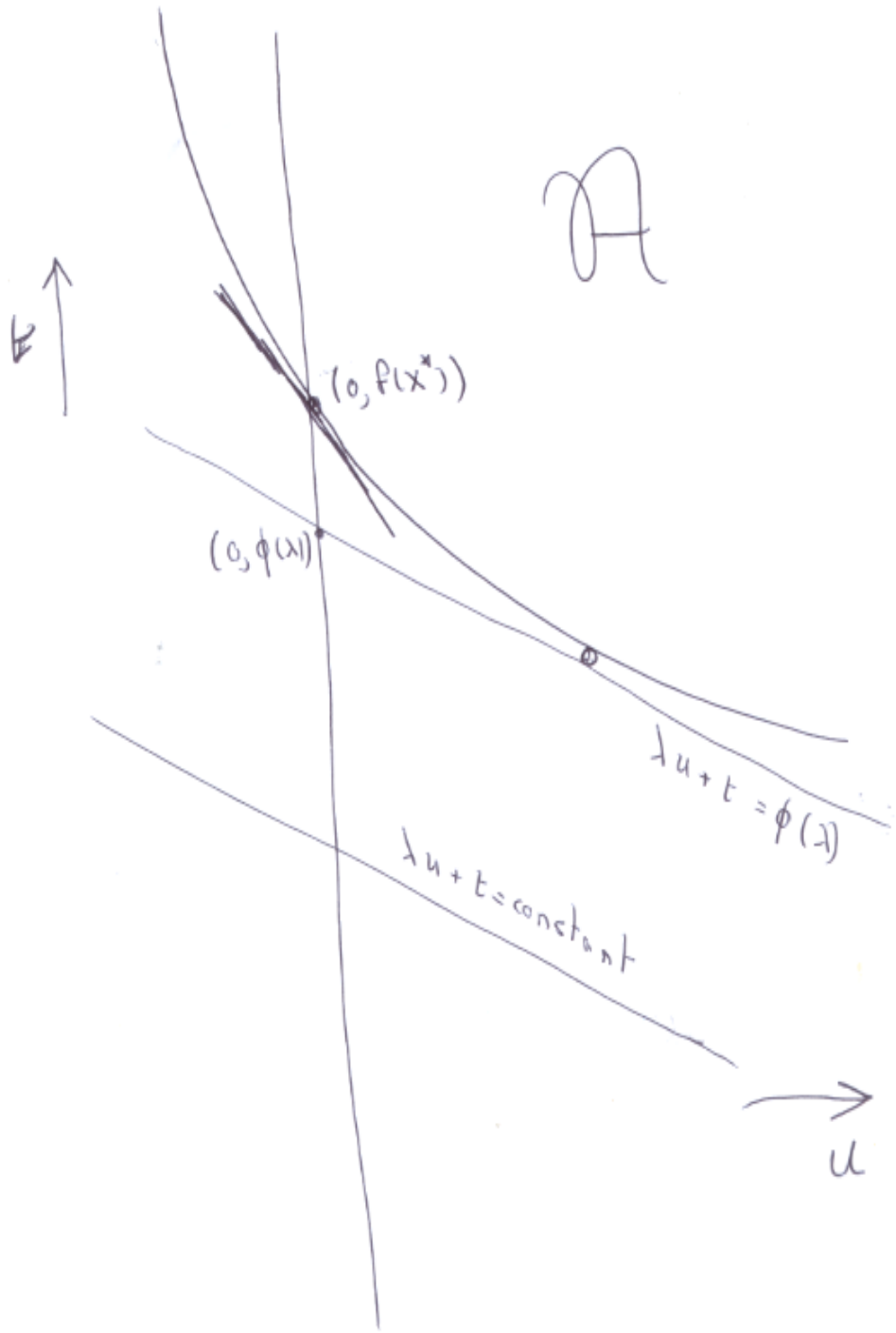


Diagram 10