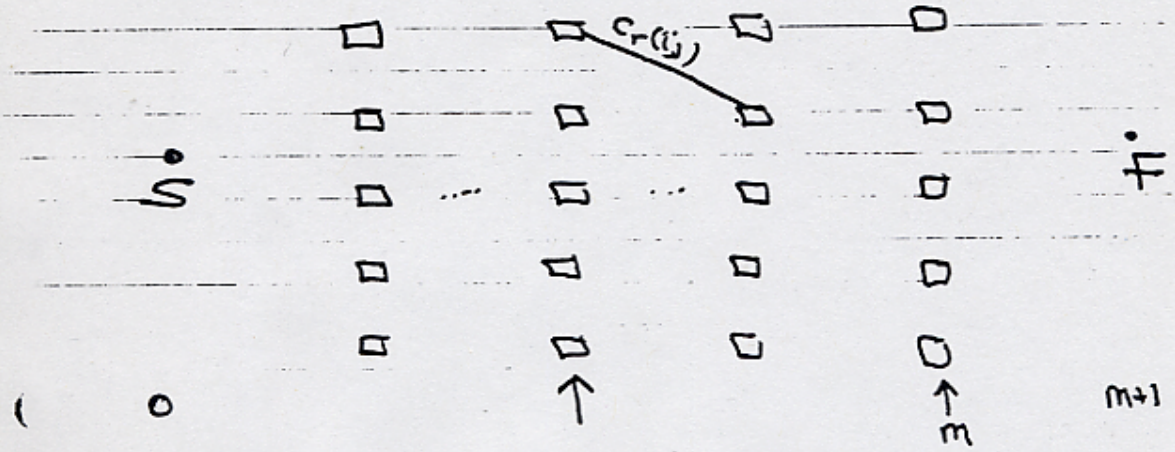
Dynamic Programming and optimum paths

Problem

Find shortest path through mountain ranges



range r

□ represents a pass

$c_r(i,j)$ = length of edge from i^{th} pass in range r to j^{th} pass in range r+1

$f_r(i)$ = shortest distance from i^{th} pass in range r to F

Problem: compute $f_0(S)$

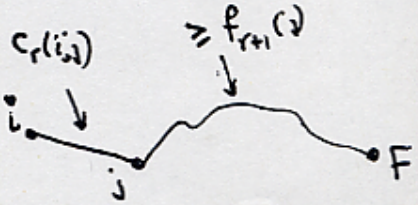
Functional Equation

$$f_r(i) = \min_j \{ c_r(i,j) + f_{r+1}(j) \} \qquad r = 0, 1, \cdots m \quad (*)$$

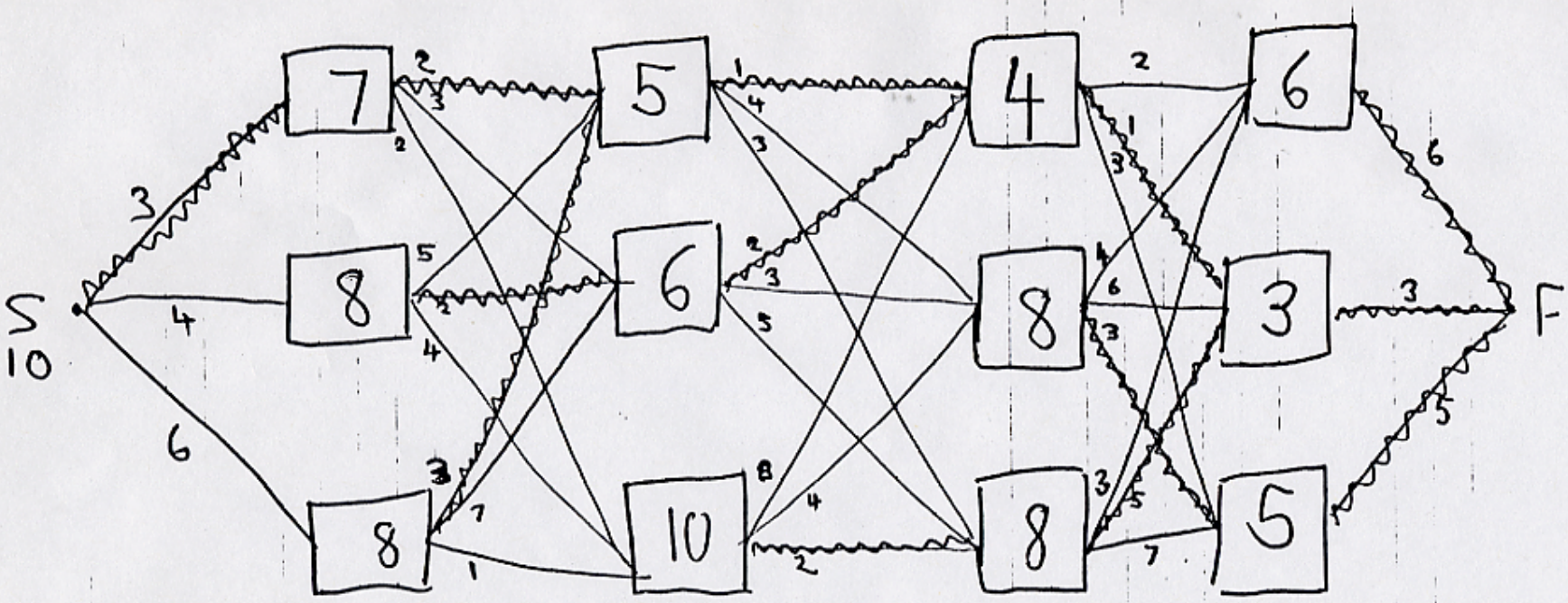
$$f_{m+1}(F) = 0$$

Proof of (*)

$\geq$ RHS : $f_r(i)$ is the length of some path



$\leq$ RHS : can always go from i to F via j which minimizes RHS.



$f_r(i)$

Indicates minimising 'i'

Dynamic Programming

Dynamic programming 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 A 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_1 in period 1, what is the minimum production cost obtainable over the whole n periods? This minimum cost is clearly

$$(1.1) \qquad c_1(x_1) + f_2(i_0 + x_1 - d_1)$$

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

The next question is what is the best value of x_1 to take. The answer must be, the value of x_1 that minimises (1.1). This will give us the minimum production cost for periods $1,2,\ldots n$ starting with a stock i_0 i.e. $f_1(i_0)$. We have thus proved that

$$(1.2) \qquad f_1(i_0) = \min_{x_1} \left(c(x_1) + f_2(i_0 + x_1 - d_1)\right)$$

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

(1.3) $$f_r(i) = \min_{x_r} \left(c(x_r) + f_{r+1}\,(i + x_r - d_r)\right)$$

The range over which the 'decision variable' x_r is to be minimised depends on our assumptions about the problem. Firstly we must have $x_r \geq 0$ 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 have $i + x_r - d_r \leq H$. Thus x_r is to be chosen in the range

(1.4) $$\max (0, d_r - i) \leq x_r \leq H + d_r - i.$$

Now the argument that produced (1.3) 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

(1.5) $$f_n(i) = \min_{x_n} \left(c(x_n)\right)$$

This can be put into the framework of (1.3) by defining $f_{n+1}(i) = 0$. Equations 1.3 and 1.5 give us a means of solving our problem. We first calculate $f_n(i)$ for $i = 0,1,2,\ldots H$. We thsn use (1.3) 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 maximum stock level $H = 4$ and $c(x) = 18x - x^2$.

So that we can keep track of the optimal production policy we make a note of the value of x_r minimising the R.H.S of (1.3) for each i. Denote this value by $x_r(i)$.

Stage 1 - calculation of f_4

By definition $f_4(i) = \min (18x-x^2 | \max(0,3-i) \leq x \leq 7-i)$

$f_4(0) = 45$, $x_4(0) = 3$; $f_4(1) = 32$, $x_4(1) = 2$; $f_4(2) = 17$, $x_4(2) = 1$; $f_4(3) = 0$, $x_4(3) = 0$; $f_4(4) = 0$, $x_4(4) = 0$

Stage 2 - calculation of f_3

In this case 1.3 becomes

$$f_3(i) = \min (18x-x^2 + f_4(i + x - 3) | \max(0,3 - i) \leq x \leq 7 - i)$$

$$f_3(0) = \min(45 + f_4(0), 56 + f_4(1), 65 + f_4(2), 72 + f_4(3), 77 + f_4(4)) = 72$$

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	216	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 production policy is found as follows:

$x_1(0) = 7$ i.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 4 at the beginning of period 2 the optimum 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.

It is also possible to solve the problem working forwards from period 1, giving us a forward formulation.

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,...r given that the stock level at the <u>end</u> of period r is i. Then arguing in a similar manner to

the backward formulation we get

$$(1.6) \qquad g_1(i) = c(i + d_1)$$

$$(1.7) \qquad g_r(i) = \min_{x_r}(c(x_r) + g_{r-1}(i + d_r - x_r))$$

where x_r in 1.7 ranges over

$$\max(0, i + d_r - H) \leq x_r \leq i + d_r$$

Starting with g_1 as defined in (1.6) we use (1.7) iteratively to calculate g_n and we can thus calculate an optimum for any value of the final stock.

(i) Add a holding cost

(ii) ~~Add~~ Allow backordering, up to $-B$

(iii) Add a "smoothing" penalty.

Knapsack Problem

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

Problem

$$\text{maximise} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_j x_j \leq W$$

$$x_j \geq 0 \text{ and integer}, \quad j = 1, 2, \ldots, n.$$

Let now $f_r(w)$ = maximum above when W is replaced by w and n is replaced by r.

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

Ex: maximise $2x_1 + 3x_2 + 5x_3 + 7x_4$

subject to

$$2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 12$$

$$x_1, \ldots, x_4 \geq 0 \text{ and integer.}$$

Knapsack Problem – Simpler Approach

$$f_r(w) = \max \begin{cases} f_{r-1}(w) & x_r = 0 \text{ in optimum} \\ c_r + f_r(w - w_3) & x_r \geq 1 \text{ in optimum} \end{cases}$$

Example

Maximise $2x_1 + 3x_2 + 5x_3 + 7x_4$

subject to $2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 12$

$x_1, \ldots x_4 \geq 0$ and integer

w	f_1	δ_1	f_2	δ_3	f_3	δ_3	f_4	δ_4
0	0	0	0	0	0	0	0	0
1	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 $x_r(w)$ = value of x_r in optimum solution for w

$\delta_r(w) = 1$ if $x_r(w) > 0$, $= 0$ if $x_r(w) = 0$

$x_4(12) > 0$
$x_4(7) > 0$ → $x_4(12) = 2$
$x_4(2) = 0$

$x_3(2) = 0$, $x_2(2) = 0$, $x_1(2) = 1$

Solution $x_1 = 1$, $x_2 = x_3 = 0$, $x_4 = 2$

Complexity of above algorithm $= O(nW)$

② Let $f(w) = \max. \quad c_1x_1 + \cdots + c_nx_n$

subject to $w_1x_1 + \cdots + w_nx_n \leq W$

$x_1, \cdots x_n \geq 0$ & integer

Let $\mu = \min_j w_j$ then

Ⓐ $$f(w) = \max_{j=1,\cdots n} \left(c_j + f(w - w_j) \right) \qquad w = \mu, \mu+1, \cdots W$$

$$= 0 \qquad w = 0, 1, \cdots \mu - 1$$

Proof

If $w \geq \mu$ then in optimum solution for w, $x_t \geq 1$ for at least one t. Then $f(w) = c_t + f(w - w_t)$. But $c_j + f(w - w_j)$ is always the value of some solution and so $f(w)$ is not less than all such values.

Ex: use Ⓐ to solve problem on previous sheet.

Dynamic Programming: 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

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

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

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

§5 Pig Farming Problem

The problem described below shows how the analysis of the dynamic programming functional equation can sometimes simplify the computation.

A farmer is planning his pig production over the next N periods. At the beginning of period n he will have to decide how many of the pigs he has he should sell and how many he should keep. For the sale of x pigs in period n he will receive $R_n(x)$. The cost of keeping y pigs in period n is $C_n(y)$. If he has x pigs left at the end of period n he will have bx at the beginning of period $n+1$ due to breeding. He wishes to maximise his profit from pigs starting with P_0 pigs at the beginning of period 1.

Let $f_n(P)$ = the maximum profit he can make from periods $n, n+1, \ldots N$ starting with P pigs at the beginning of period n. The normal dynamic programming argument gives

$$f_n(P) = \max_{0 \le y \le P} \left(R_n(y) - C_n(P-y) + f_{n+1}(b(P-y)) \right) \quad (5.1)$$

The quantity y to be decided in (5.1) is the number of pigs to be sold.

Defining $f_{N+1}(P) = R_{N+1}(P)$ we can use (5.1) repeatedly to compute $f_N, f_{N-1}, \ldots f_1$ in the normal way. This is all we can do for arbitrary R_n and C_n. If however R_n and C_n are linear functions the problem can be simplified considerably.

Assume then that $R_n(y) = R_n y$ and $C_n(y) = C_n y$ for $y \ge 0$ and $n = 1, \ldots N+1$. We shall show that there exist $a_1, \ldots a_{N+1}$ such that

$$f_n(P) = a_n P \qquad n = 1, \ldots N+1. \quad (5.2)$$

The proof is by backward induction on n.

$n = N+1$

(5.3) $f_{N+1}(P) = R_{N+1}P$

and so $a_{N+1} = R_{N+1}$

Inductive step

Assume inductively that for some n $f_{n+1}(P) = a_{n+1}P$ then (5.1) becomes

(5.4) $f_n(P) = \max_{0 \le y \le P} (R_n y + C_n(P-y) + a_{n+1}b(P-y))$

$= (a_{n+1}b - C_n)P + \max_{0 \le y \le P} ((R_n + C_n - a_{n+1}b)y)$

$= (a_{n+1}b - C_n)P + S$

where

$S = 0$ if $R_n + C_n - a_{n+1}b \le 0$

$= (R_n + C_n - a_{n+1}b)P$ if $R_n + C_n - a_{n+1}b \ge 0$

Thus

(5.5) $f_n(P) = a_n P$

where

$a_n = a_{n+1}b - C_n$ if $R_n + C_n - a_{n+1}b \le 0$

$= R_n$ if $R_n + C_n - a_{n+1}b \ge 0$

Thus by induction (5.2) holds.

The calculation of f_n shows that the maximum y_n^* in (5.4) is given by

(5.6) $y_n^* = 0$ if $a_{n+1}b - C_n \ge R_n$

$= P$ if $a_{n+1}b - C_n \le R_n$

Example - $b = 1.5$

n	1	2	3	4	5	6
R_n	16	13	8	12	10	10
C_n	6	5	4	6	4	-

From (5.3) we get

$a_6 = 10 \qquad y_6^* = P$

and then using (5.5)

$a_5 = 11 \qquad y_5^* = 0$

$a_4 = 12 \qquad y_4^* = P$

$a_3 = 14 \qquad y_3^* = 0$

$a_2 = 16 \qquad y_2^* = 0$

$a_1 = 18 \qquad y_1^* = 0$

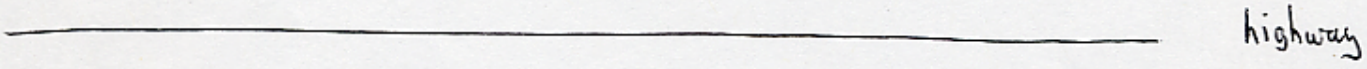
From this we can deduce that with P_0 pigs initially the farmer can earn $18 P_0$ from $a_1 = 18$. His optimum strategy is, from $y_1^* = y_2^* = y_3^* = 0$ and $y_4^* = P$ to sell no pigs until the beginning of the 4th period and then to sell them all.

Problem

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.

Example



$v_{i,j}$ = franchise value of stretch i,j for some enterprise

Solution

Let $f(r)$, $r = 0, 1, 2, \cdots L$ be maximum value obtainable from a stick of length r.

$$f(0) = 0$$

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

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

Suppose next that stick must be broken into k pieces. Now use $f(j,r)$, $j = 1, 2, \cdots k$, $r = 0, 1, \cdots L$.

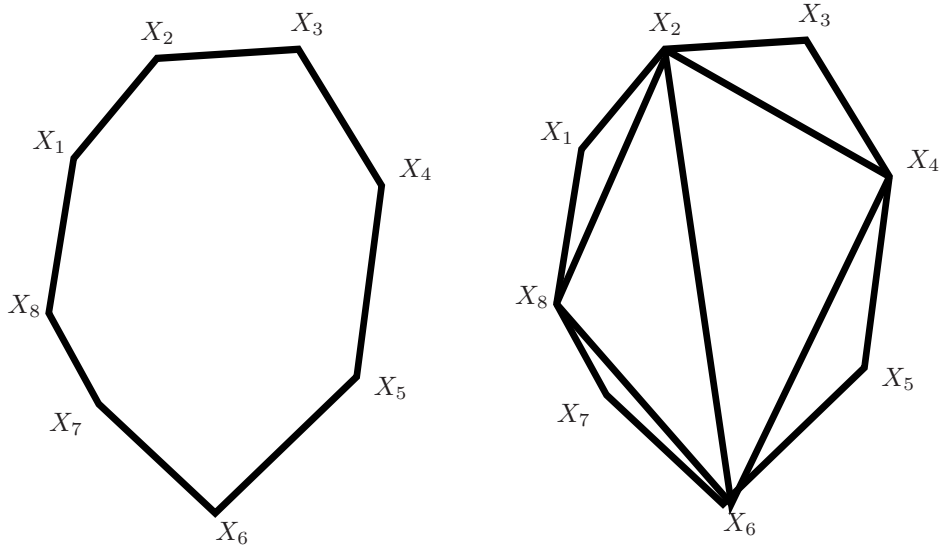
$$f(j,r) = 0 \qquad j > r$$

$$= \max_{0 \le i < r} [f(j-1, i) + v_{i,r}]$$

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

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

$$m^*_{k,l} = \min_{k<j<l} \{m^*_{k,j} + m^*_{j,l} + |X_k - X_j| + |X_j - X_l|\} \tag{1}$$

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 (1) to compute what we want i.e. $m^*_{1,n}$.

Probabilistic shortest path

Now consider the mountain range problem where at pass $i \in P_r, 0 \leq r \leq N$ (P_r denotes the set of passes in range r) you have to choose a decision $d \in D_{i,r}$ and then you have probability $\rho_d(r, i, j, t), j \in P_{r+1}, t \geq o$ of arriving at pass j with the journey taking time t. The first problem is to minimise the expected time to reach the final destination F. So if $f_r(i)$ denotes the minimum expected time to reach F from $i \in P_r$, we have

$$f_r(i) = \min_{d \in D_{i,r}} \left\{ \sum_{\substack{t \geq 0 \\ j \in P_{r+1}}} \rho_d(r, i, j, t)(t + f_{r+1}(j)) \right\}.$$

To guarantee that we reach F we should put $P_{N+1} = \{F\}$.

We can also consider the alternative problem. We can ignore costs and try to maximise the probability that we arrive at F within time T. Then if $g_r(i, t), i \in P_r, 0 \leq t \leq T$ denotes the maximum probability of reaching F by time T,

$$g_r(i, t) = \max_{d \in D_{i,r}} \left\{ \sum_{\tau \geq 0} \rho_d(r, i, j, \tau) g_{r+1}(j, t + \tau) \right\}.$$

As a boundary condition we have

$$g_{N+1}(F, t) = \begin{cases} 1 & t \leq T \\ 0 & t > T \end{cases}$$

Dynamic Programming: 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

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

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 $\xi^+ = \max\{0, \xi\}$,

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

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

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

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 requiredto 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

$$y_v = c(v, \pi(w)) + \alpha y_{\pi(v)} \qquad v \in V. \tag{1}$$

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

$$y_v^* = \min_{w \in V} \{c(v,w) + \alpha y_w^*\} \qquad \forall v \in V. \tag{2}$$

Proof Suppose that (2) 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 $\tilde{\pi}$ by $\tilde{\pi}(u) = \lambda(u)$ for $u \in U$ and $\tilde{\pi}(v) = \pi(v)$ 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

$$\xi_u > \alpha \xi_{\tilde{\pi}(u)} \qquad u \in U. \tag{3}$$

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

$$\xi_v = \alpha \xi_{\tilde{\pi}(v)} \qquad v \notin U. \tag{4}$$

It follows from (3), (4) that

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

Letting $t \to \infty$ we see that

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

Thus $\tilde{\pi}$ is *strictly better* than Π i.e. if (2) does not hald, then we can improve the current policy.
Conversely, if (2) holds and $\hat{\pi}$ is any other policy and $\eta_v = \hat{y}_v - y_v^*$ then

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

and so

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

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 (1).
3. If (2) holds – current π is optimal, stop.
4. If (2) 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.

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:*

$$c(\pi(i), i) + \alpha \sum_{j \in V} P(\pi(i); i, j) y_j = \min_{x \in X_i} \left\{ c(x, i) + \alpha \sum_{j \in V} P(x, i, j) y_j \right\} \tag{5}$$

π is optimal iff (5) holds.

Proof Suppose first that (5) does not hold. Define a new policy $\hat{\pi}$ by

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

We have

$$\begin{aligned} y_i &\geq c(\hat{\pi}(i), i) + \alpha \sum_{j \in V} P(\hat{\pi}(i), i, j) y_j \qquad (6) \\ \hat{y}_i &= c(\hat{\pi}(i), i) + \alpha \sum_{j \in V} P(\hat{\pi}(i), i, j) \hat{y}_j \end{aligned}$$

and so

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

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

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

But $\hat{y} \neq y$ since there is strict inequality in (6) for at least one i and $\hat{\pi}$ is strictly better than π.
Conversely, if (5) holds and $\hat{\pi}$ is any other policy, we get that

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

and so

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

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

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

□

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.

Shortest Path Problems

A digraph $D = (N,A)$ consists of 2 sets:

N = the set of nodes

$A \subseteq N \times N$ is the set of arcs

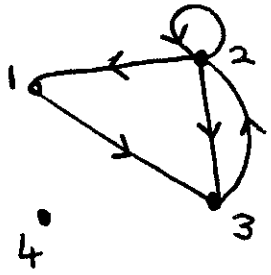


Fig 1

The above is a pictorial representation of the digraph with

$N = \{1,2,3,4\}$

$A = \{(1,3), (2,1), (2,2), (2,3), (3,2)\}$

We will only consider digraphs with N and A finite.

A walk W from a node i_1 to a node i_p is a sequence of arcs $((i_1,i_2), (i_2,i_3),\ldots,(i_{p-1},i_p))$ or equivalently a sequence of nodes $(i_1,\ldots,i_p)$ where $(i_{i-1},i_t) \in A$ for $2 \leq t \leq p$.

Although W is not a set we use the notaiton $u \in W$ or $i \in W$ to say arc u is in W or node i is in W.

Examples: (1,3,2) and (2,2,3,2,1) are walks in the digraph give in fig. 1.

If $W = (i_1,\ldots,i_p)$ and $1 \leq a < b \leq p$ we use the notation $W[a,b] = (i_a,i_{a+1},\ldots,i_b)$ for the sub-walk of W from i_a to i_b.

Next given a walk $W_1 = (i_1,\ldots,i_p)$ and a walk $W_2 = (i_p=j_1,j_2,\ldots,j_p)$ we define the walk

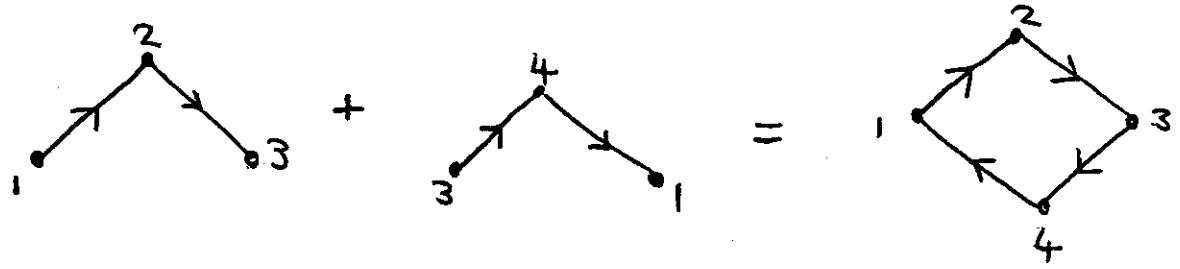


Fig 2

$$W_1 + W_2 = (i_1, \ldots, i_p, j_2, \ldots, j_p).$$

(We can only form $W_1 + W_2$ if the terminal node of W_1 = the inital node of W_2).

A Path P is a walk in which no node is visited more than once.

A circuit is a walk from a node to itself e.g. (1,2,3,4,1) of fig. 2 is a circuit.

Length

We now assume that associated with each arc $u \in A$ is a length $\ell(u)$ i.e.: $\ell: A \to \mathbb{R}$.

The length of a walk W is then defined by

$$\ell(W) = \sum_{u \in W} \ell(u)$$

i.e. the length of a walk is the sum of the lengths of the arcs in the walk.

We shall be concerned here with the following problem: given a node $s \in N$, find for each node $j \neq s$ a minimum length path from s to j.

Because of an assumption we will make about the non-existence of negative circuits we will be able to show that a shortest path from s to j is also a shortest walk from s to j.

Negative Circuits

If some arc lengths are negative it is possible that there is a circuit

C such that $\ell(C) < 0$. We exclude this possibility for the following reason:

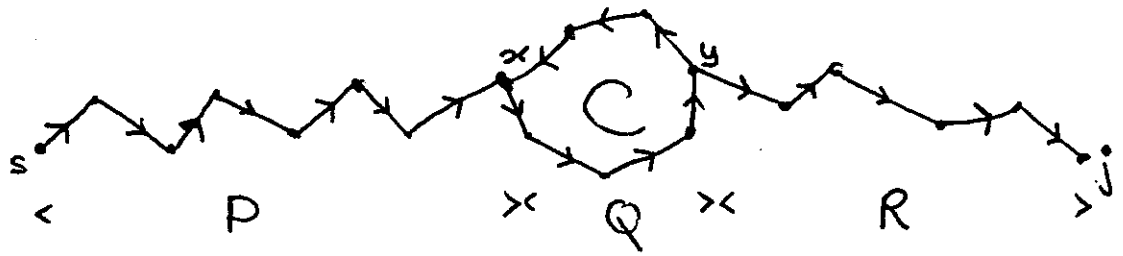


Fig 3

Suppose $\ell(C) < 0$. Define walk W_n to consist of (the path P from s to x) + (the path Q from x to y) + (n times round C) + (the path R from y to j). Then $\ell(W_n) = \ell(P) + \ell(Q) + n\ell(C) + \ell(R)$

$$\to -\infty \quad \text{as} \quad n \to \infty.$$

Thus there is no shortest walk from s to j.

Since the number of paths from s to t is finite ($< 2^{|A|}$) there is always a shortest path from s to t but there are no known polynomial algorithms for finding shortest paths if there are negative circuits. This is essentially because they rely on

Theorem 2.1

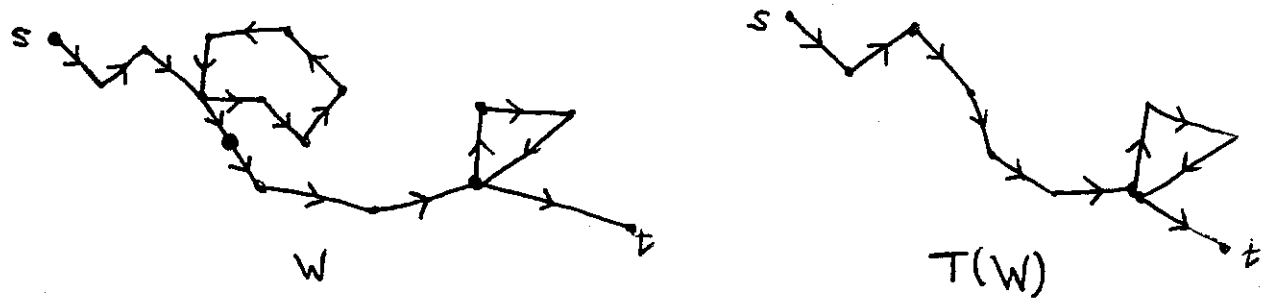
If D has no negative circuits then a shortest path from node s to node $t \neq s$ is a shortest walk from s to t.

Proof

Let W be any walk from s to t and let P^* be a shortest path from s to t. We can construct a path P from s to t such that $\ell(W) \geq \ell(P)$. As $\ell(P) \geq \ell(P^*)$ we have $\ell(W) \geq \ell(P^*)$ and our theorem.

Construction of P

If W is a path let P = W, otherwise suppose $W = (s = i_1, \ldots, i_p = t)$. Let $i_a = i_b$ be the first repeated node. Let T(W) be the walk W[1,a] + W[b,p].



Then T(W) is a walk from s to t with fewer edges than W and

$$\ell(T(W)) = \ell(W) - \ell(W[a,b])$$
$$\leq \ell(W) \qquad \text{as } W[a,b] \text{ is a circuit.}$$

If T(W) is not a path we construct $T^2(W)$ and so on. Thus there exists $k \geq 0$ such that $T^k(W)$ is a path. Let $P = T^k(W)$. Then $\ell(P) \leq \ell(T^{k-1}(W)) \leq \ldots \leq \ell(W)$. □

Properties of Shortest Paths

Theorem 3.1 (Optimality of sub-paths)

Let $P = (i_1 = s, \ldots, i_p = t)$ be a shortest path from s to t. Then for $1 \leq a < b \leq t$ $P[a,b]$ is a shortest path from i_a to i_b.

Proof

Let R be any path from i_a to i_b. Let $W = P[1,a] + R + P[b,t]$. W is a walk from s to t. We know that

$$0 \geq \ell(W) - \ell(P) \qquad [P \text{ is also a shortest walk}$$
$$= \ell(R) - \ell(P[a,b]).$$

□

Suppose next that for each $j \in N$ we have a path P_j from s to j. ($P_s = (s)$) and that $d(j) = \ell(P_j)$ ($d(s) = 0$).

Theorem 3.2

$\{P_j;\ j \rfloor N\}$ is a collection of shortest paths from s to each node of D if and only if $\forall\ j \in N$ we have

$$d(j) \leq d(i) + \ell(i,j) \qquad \forall (i,j) \in A. \tag{3.1}$$

Proof

only if: Suppose 3.1 does not hold and that there exist nodes x,y and arc (x,y) such that

$$d(y) > d(x) + \ell(x,y).$$

Consider $W = P_x + (x,y)$. W is a walk from s to y and $\ell(W) = d(x) + \ell(x,y) < \ell(P_y)$. This contradicts the fact that P_y is a shortest path.

If: Suppose 3.1 holds $\forall\ j \in N$. Let $x \in N$ and $P = (s = i_1, \ldots, i_p = x)$ be a path from s to x. We show that $\ell(P) \geq d(x)$.

From 3.1 we have $d(i_k) - d(i_{k-1}) < \ell(i_{k-1}, i_k)$ $2 \leq k \leq p$. Thus

$$(3.2) \qquad \sum_{k=2}^{P} (d(i_k) - d(i_{k-1})) \leq \sum_{k=2}^{P} \ell(i_{k-1}, i_k).$$

But the LHS of 3.2 "collapses" to $d(i_k) - d(i_1) = d(x) - 0$ and the RHS of 3.2 is $\ell(P)$. □

The aim of all shortest path algorithms is to find a set of paths satisfying 3.1.

Ford's Algorithm

Suppose that D is given as a list of arcs and $A = \{u_1,\ldots,u_m\}$ where $u_i = (x_i,y_i)$.

We will first consider how to compute the lengths of shortest paths and then show how to produce paths.

At a general stage of the algorithm we will have estimates $d(j)$ for the shortest path length to $j \in N$. Initially $d(s) = 0$ and $d(j) = \infty$ for $j \in N\backslash\{s\}$.

The idea behind Ford's algorithm stems from theorem 3.2: if we have an arc (x,y) such that $d(y) > d(x) + \ell(x,y)$ then replace $d(y)$ by $d(x) + \ell(x,y)$.

Ford's Algorithm (ignore statements on first reading)

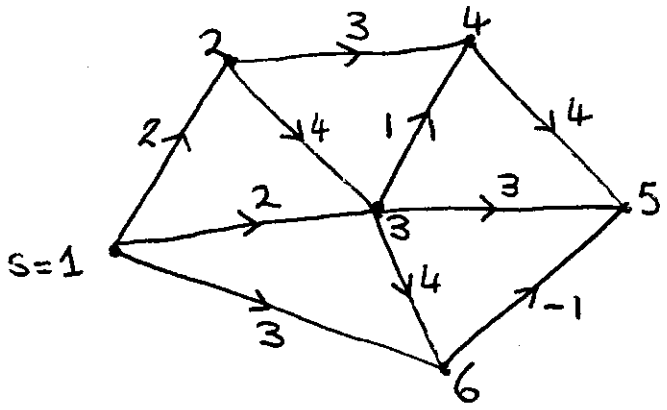
begin

{Initialization: assume s = 1 and |N| = n, |A| = m}

d(1) := 0; for j := 2 to n do [d(i) := ∞; π(j) := s];

```
repeat {main loop}
    flag := false
    for a := 1 to m do
     begin {process arc (x_a,y_a)}
     if d(y_a) > d(y_a) + ℓ(x_a,y_a) then
    begin
      d(y_a) := d(x_a) + ℓ(x_a,y_a); π(y_a) := x_a;
    end
  end
until flag = false
end
```

Example



Arc	d(1)	d(2)	d(3)	d(4)	d(5)	d(6)
-	0	∞	∞	∞	∞	∞
(1,2:2)	0	2	∞	∞	∞	∞
(1,3:2)	0	2	2	∞	∞	∞
(1,6:3)	0	2	2	∞	∞	∞
(2,3:4)	0	2	2	∞	∞	∞
(2,4:3)	0	2	2	5	∞	3
(3,4:1)	0	2	2	3	∞	3
(3,5:3)	0	2	2	3	5	3
(3,6:4)	0	2	2	3	5	3
(4,5:4)	0	2	2	3	5	3
(6,5:1)	0	2	2	3	2	3

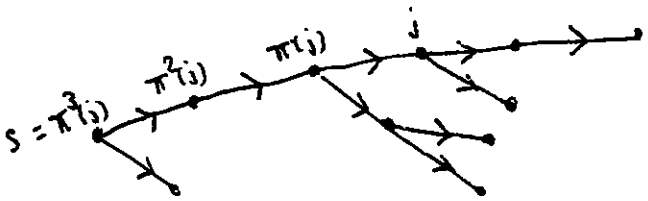
END OF 1ST PASS

There were no changes during second pass

END OF 2ND PASS

Recording the shortest paths

We will show that the set of paths produced by the algorithm forms a directed tree rooted at s.



i.e. for each $j \in N$ we record a predecessor node $\pi(j)$ so that on termination of the algorithm we have the following:

(5.1a) $$\pi(s) = s$$

(5.1b) for $j \neq s$ $\exists\, k > 0$ such that $s = \pi^k(j)$ and
$P_j = (s = \pi^k(j), \pi^{k-1}(j), \ldots, \pi^2(i), \pi(j), j)$
is a shortest path from s to j.

The statements of Ford's algorithm actually find shortest paths.

Lemma 5.1

Throughout the algorithm if $d(j) \neq \infty$ then $d(j)$ is the length of some walk from s to j.

Proof

The statement of true initially. We show that processing an arc does not alter the statements truth. Suppose that immediately prior to processing arc (x,y) that $d(x) = \ell(W_x)$ or ∞ and $d(y) = \ell(W_y)$ or ∞ for walks W_x, W_y. After processing (x,y) only d(y) can be altered and then $d(y) = \ell(W_y)$ or $\ell(W_x + (x,y))$ or ∞. □

During execution of the algorithm let label d(j) be correct if d(j) = the length of a shortest path from s to j.

Note that once d(j) is correct it does not change anymore. If it did change it would be reduced and then lemma 5.1 would imply the existence of a walk shorter than the shorest path which contradicts theorem 2.1.

For $0 \leq k \leq |V| - 1$ let

$H(k) = \{j \in N : \exists$ a shortest path from s to j which has k arcs or less$\}$

Lemma 5.2

After k passes through the main loop all nodes in H(k) have correct d labels.

Proof

By induction on k. For k = 0 the result is true because H(0) = {s}. Suppose this result is true for all $k < K$. Let $j \in H(K) \backslash H(K-1)$ and let $P = (s = i_1, \ldots, i_k = j)$ be a shortest path from s to j. By theorem 3.1 $i_{k-1} \in H(K-1)$ and so its label is correct at the beginning of the K^{th} main loop. During execution of the K^{th} main loop arc (i_{K-1}, j) will be processed and so at the end of the K^{th} main loop $d(j) \leq d(i_{K-1}) + \ell(i_{K-1}, j) = \ell(P)$ and then

lemma 5.1 and theorem 2.1 imply $d(j) = \ell(P)$.

Now $H(|N| - 1) = \{j : \exists$ a path from s to $j\}$. We therefore have

Theorem 5.1

Ford's algorithm terminates after at most $|N| - 1$ iterations, having computed the shortest distance from s to each $j \in N$. □

It is easily seen that the computation time for the main loop is bounded by some multiple of $|A|$. Thus the overall computation time of Ford's algorithm is $O(|N| \times |A|)$.

We have still to verify that on completion π provides shortest paths.

Lemma 5.3

On completion of the algorithm the π labels are correct.

Proof

On completion we have

$$d(j) = d(\pi(i)) + \ell(\pi(j),j) \qquad \forall\ j \in N. \tag{5.2}$$

This is because there will have been no further reduction in $d(\pi(j))$ after the last assignment to $d(j)$.

If we can show that (5.1) holds it will follow from (5.2) that $d(j) = \ell(P_j)$ and we are through.

Fix $j \in N$ and consider the sequence $j, \pi(j), \pi^2(j), \ldots$. We have to show that $\exists\ k$ such that $\pi^k(j) = s$. If this is not true then $\exists\ \ell < m$ such that $\pi^\ell(j) = \pi^m(j) \neq s$.

Now for $i \in N \backslash \{s\}$ let $T(i)$ be the number of arc processings up to and

including the processing of arc $(\pi(i),i)$ that gave $d(i)$ its final value. Let $T(s) = 0$. Now $i \neq s$ implies $T(i) > T(\pi(i))$ because $d(i)$ is not made correct until after $d(\pi(i))$ is made correct.

Thus $T(\pi^{\ell}(j)) > T(\pi^{\ell+1}(j)) > \ldots > T(\pi^{m}(j))$ which contradicts $\pi^{\ell}(j) = \pi^{m}(j)$. □

A Computational Improvement

Consider the following situations we are about to process arc (x,y). This arc has been processed before but $d(x)$ has not been reduced since (x,y) was last processed. Thus we know that prior to processing (x,y) that $d(y) \leq d(x) + \ell(x,y)$ and so in fact there is no point in processing (x,y).

We can speed up execution of the algorithm if we obey the following rule: arc (x,y) is processed only if $d(x)$ has been reduced since arc (x,y) was last processed.

In the main loop arcs are processed in blocks. A block consists of all the arcs leaving a specific node.

The search for nodes whose d labels have been reduced since their blocks were last processed is speeded up by keeping them in a queue Q.

A queue is a linked list of nodes where insertions are made at the "back end" and deletions are from the "front end" only.

Ford's algorithm (final version)

```
procedure   process (node : x); {process all the arcs leaving  x}
  begin
     for (x,y) ∈ A  do
      begin
       if  d(x) + ℓ(x,y) < d(y) then
          begin
            d(y): = d(x) + ℓ(x,y); ℘(y): = x;
           if  y  is not in  Q  then insert  y  into  Q
          end
        end
  end;
{initialization}
begin
     d(1) = 0; π(1) := 1; for  j = 2 to n  do [d(j): = ∞; π(j): = 1]
     Q: = |• |1| | ;
     while  Q ≠ ϕ  do
       begin
        delete  x = front(Q)  from  Q,
        process (x)
       end
end
```

Example (see digraph on p8)

$\alpha(j)$ / $\pi(j)$ — j

1		2		3		4		5		6		Q
0	1	∞	1	∞	1	∞	1	∞	1	∞	1	1
0	1	2	1	2	1	∞	1	∞	1	3	1	2→3→6
0	1	2	1	2	1	5	2	∞	1	3	1	3→6→4
0	1	2	1	2	1	3	3	5	3	3	1	6→4→5
0	1	2	1	2	1	3	3	4	3	3	1	4→5
0	1	2	1	2	1	3	3	4	3	3	1	5
0	1	2	1	2	1	3	3	4	3	3	1	ϕ

To prove that the modification is valid we relate the new algorithm to the old.

Let $p(1),p(2),\ldots$ be the sequence of nodes processed by the new algorithm. Define $k_0 = 0$ and $k_1,\ell_1,\ldots k_t,\ell_t\ldots$ as follows:
$p(1),\ldots,p(k_1-1)$ are all distinct but $p(k_1) = p(\ell_1)$ where $1 \le \ell_1 \le k_1 - 1$
$p(k_1),\ldots p(k_2-1)$ are all distinct but $p(k_2) = p(\ell_2)$ where $k_1 \le \ell_2 \le k_2-1$
and so on.

Let $X_t = \{p(k_{t-1}),\ldots,p(k_t-1)\}$. $Y_t = \{x \notin X_t$ and x is not in Q immediately prior to the ℓ_tth node processing$\}$.

Exercise: show that $N = X_t \cup Y_t$. Suppose $x \in \bar{X}_t \cap Q$ — $p(k_{t-1})\,p(k_{t-1}+1)\,\ell_t\,x\,p(k_t-1)\,p(\ldots$

[Hint: it is a simple direct consequence of Q being a queue]

Suppose now we re-ran the new algorithm but just before we process $p(\ell_t)$ we process all the nodes in Y_t. Nothing will happen to d or π because none of Y_t are in Q. But the exercise shows that now between processing $p(k_{t-1})$ and $p(k_t-1)$ all nodes and hence all arcs are processed, i.e. we have gone through a main loop of the old algorithm (the order in which we process arcs in a main loop is irrelevant). Thus convergence in $O(|N| \times |A|)$ time is assured.

Exercise: show that we do one less main loop in the new method.

The algorithm above is very efficient. We give a table of reported results for some random problems. The run time t is in seconds on a CDC 6600

Problem	|N|	|A|	t	Problem	|N|	|A|	t
1	500	12500	.267	9	1000	4000	.139
2	500	7500	.177	10	1000	3000	.123
3	500	2500	.089	11	1000	2000	.088
4	500	1250	.050	12	2000	16000	.476
5	500	1000	.045	13	2000	12000	.371
6	500	750	.039	14	2000	8000	.315
7	1000	10000	.288	15	2000	4000	.191
8	1000	5000	.165				

Exercise: the initialization step wastes a little time, what should it be?

Department of Mathematics
CARNEGIE MELLON UNIVERSITY

Dijkstra's algorithm

When arc lengths are all non-negative the following algorithm is applicable.

For $k \in N$ let $\Gamma_k = \{v: (k,v) \in A\}$ = set of out-neighbours of k.

Dijkstra's algorithm

begin

A: $d(1) := 0$; $\pi(1) := 1$; for $j := 2$ to n do $[d(j) := \ell(s,j)$; $\pi(j) := s]$

$S := \{1\}$ [S = {nodes that have been processed}]

for $i := 1$ to $n-2$ do

begin

B: let $d(k) = \min (d(j): j \in N \setminus S)$;

$S := S \cup \{k\}$;

{process k}

C: for $v \in \Gamma_k - S$ do

if $d(v) > d(k) + \ell(k,v)$ then

begin

$d(v) := d(k) + \ell(k,v)$,

$\pi(v) := k$

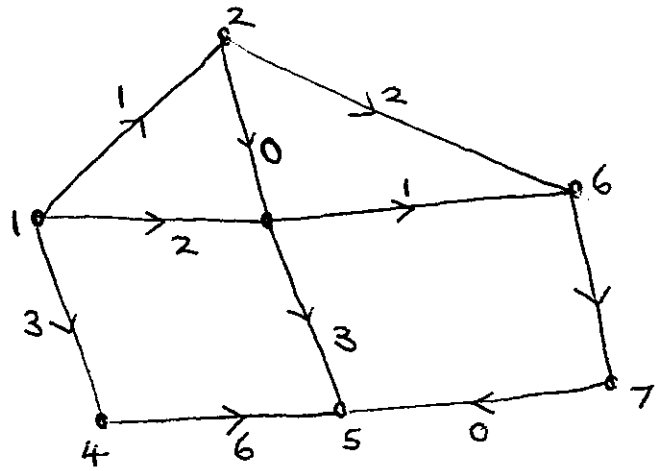
end

end

end

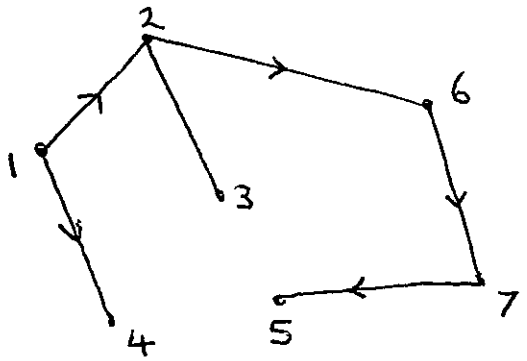
A nice feature of Dijkstra's algorithm is that each node is processed exactly once (except for last node which need not be processed).

Example



1		2		3		4		5		6		7		k
0	1	1	1	2	1	3	1	∞	1	∞	1	∞	1	2
				1	2	3	1	∞	1	3	2	∞	1	3
						3	1	4	3	2	3	∞	1	6
						3	1	4	3			3	6	4
								4	3			3	6	7
								3	7					

shortest path tree



Theorem

Assuming that all arc lengths are non-negative, Dijkstra's algorithm terminates with a shortest path from s to each node of D.

Proof

At each stage the digraph (S,X_S) where $X_S = \{(\pi(j),i): j \in S - \{s\}\}$ is a directed tree rooted at s and for $j \in S$, $d(j)$ is the length of the path from s to j in this tree. Furthermore if $j \notin S$ then $d(j)$ is the minimum length of a path from s to j, which follows a tree path in S and then jumps to j.

We prove by induction on $|S|$ that throughout the algorithm $j \in S$ implies $d(j)$ is the length of a shortest path from s to j.

This is clearly true when $|S| = 1$ and $S = \{s\}$. Suppose it is true for $|S| < q$ and suppose k is the q^{th} node added to S. Let $P = (S = i_1, i_2, \ldots, i_a = k)$ be any path from s to k and let i_b be the first node of P that is not in $S - \{k\}$. Then

$$\begin{aligned} \ell(P) &\geq \ell(P[1,b]) && \text{[arc lengths non-negative} \\ &\geq d(i_{b-1}) + \ell(i_{b-1}, i_b) \\ &\geq d(i_b) && \text{[See first paragraph} \\ &\geq d(k) && \text{[definition of k} \end{aligned}$$

□

Execution Time

A naive implementation is $O(n^2)$.

Line A is O(n) and is executed once

Line B is O(n) and is executed n-1 times

Line C is $O(|\Gamma_k|)$ and total execution time is $O(\sum_k |\Gamma_k|) = O(|A|) = O(n^2)$.

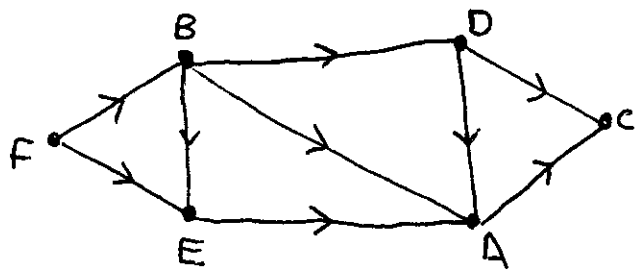
Digraphs without circuits

These are important not least because they occur in critical path analysis. Their application in this area involves computing longest paths. For this problem inequalities are reversed throughout the previous sections but most important the optimality conditions (3.1) becomes

$$d(j) = \max_{j} (d(i) + \ell(i,j)). \tag{7.1}$$

topological Ordering

Let the nodes of digraph $G = (N,A)$ be ordered or numbered $1,2,\ldots,n$. This ordering is topological if $(i,j) \in A \rightarrow i < j$



F B E D A C is a topological ordering for the above digragh.

Theorem 7.1

There exists a topological ordering for a digraph G if and only if the digraph does not contain any circuits.

Proof

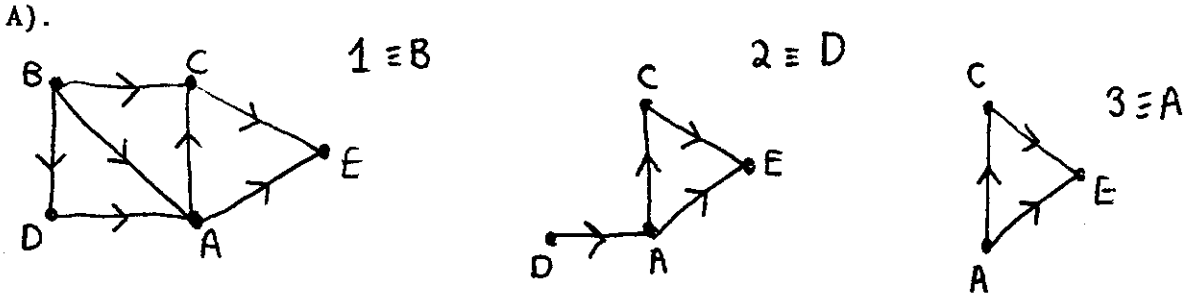
Suppose first that the nodes of G have been topologically ordered $1,2,\ldots,n$. Suppose G has a circuit $(i_1,i_2,\ldots,i_p=i_1)$ then we have

$$i_1 < i_2 < \ldots < i_p = i_1 \qquad \text{contradiction.}$$

Conversely assume G has no circuits. We describe an algorithm for numbering the nodes of G. It's general step is: if nodes $1,2,\ldots,k$ have been chosen define $G_k = (N_k, A_k)$ where $N_k = N - \{1,2,\ldots,k\}$

$$A_k = A \cap N_k \times N_k.$$

Note $G_0 = G$ and A_k consists of these arcs not involving $1,2,\ldots,k$. Now since G has no circuits G_k will not have any for $k = 0,1,\ldots,n-1$. Thus (Lemma 7.1 below) G_k has at least one node without predecessors. Let $k+1$ be such a node (a predecessor of a node j is a node i such that $(i,j) \in A$).



The algorithm above will number the nodes $1,2,\ldots,n$. Now let $(i,j) \in A$. From the way j was chosen from G_{j-1} we see that $(i,j) \notin A_{j-1}$ which implies that $i \in \{1,2,\ldots,j-1\}$ or $i < j$. □

Lemma 7.1

Let $G = (N,A)$ be a digraph without circuits. Then G contains a node without predecessors.

Proof

Let $x_1 \in N$. If x_1 has no predecessors we are finished otherwise there is an arc (x_2,x_1). If x_2 has no predecessors we are finished, otherwise there is an arc (x_3,x_2). Continuing thus we generate a sequence $x_1,x_2,\ldots$ where $(x_{k+1},x_k) \in A$ for $k \geq 1$. This sequence must terminate with a node without predecessors or repeat a node because G is finite. But repeating a node means there is a circuit in G. □

Assume now that we have topologically ordered the nodes of a digraph G. Then (1.9) can be replaced by

$$d(j) = \max_{i<j} (d(i) + \ell(i,j)) \qquad j = 1,\ldots,n \tag{7.2}$$

where it is assumed that $s = 1$ (what if $s \neq 1$?) Given that $d(1) = 0$ we can use (7.2) to compute $d(2),\ldots$ in a manner analogous to back-substitution for solving a lower triangular set of linear equations

```
begin
    d(1): = 0;
    for j: = 2 to n do
     d(j): = max (d(i) + ℓ(i,j): (i,j) ∈ A)                    *
end
```

(Note that because $(i,j) \in A$ implies $i < j$ * can be carried out validly.)

Since (7.1) holds on completion of the algorithm a proof of validity is obvious.

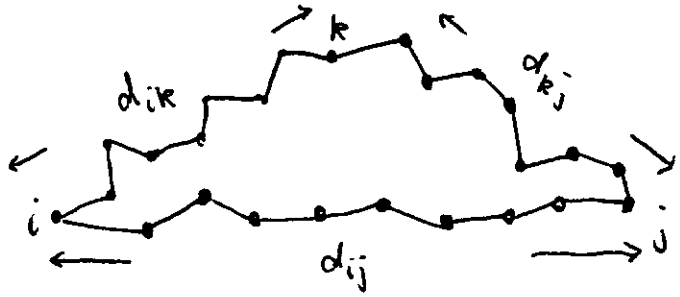
Computaional Considerations

The complexity of the algorithm is $O(|A|)$ and there exists an $O(|A|)$ method for topologically ordering the nodes.

Shortest paths between all pairs of nodes

We complete the material on shortest paths by describing an algorithm for finding shortest paths between all pairs of nodes.

We use the two matrices $D = \|d_{ij}\|$, $N = \|n_{ij}\|$ where d_{ij} is the length of the current best known path from i to j and n_{ij} is the second node after i on this path. the algorithm tries to improve paths as follows.



If in figure 12 $d_{ik} + d_{kj} < d_{ij}$ then the known path from i to j should be replaced by the known path from i to k followed by the one from k to j.

Floyd's Algorithm

(Initialization)

begin

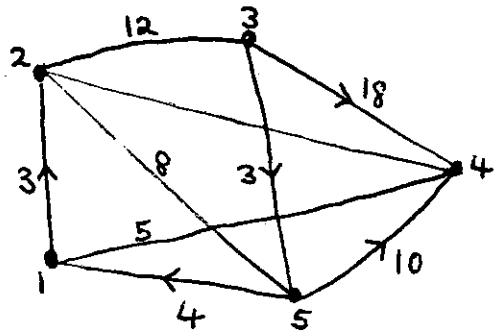
for i := 1 to n do

for j: = to n do

(d(i,j): = ℓ(i,j);

(n(i,j) := j)

```
(main algorithm)

    for  k := 1  to  n  do

    for  i := 1  to  n  do

    for  j := 1  to  n  do

      if  d(i,j) > d(i,k) + d(k,j)  then

          d(i,j) := d(i,k) + d(k,j);  n(i.j) := n(i,k)

end
```



Initially

D =	0	3	∞	5	∞	N =	1	2	3	4	5
	∞	0	12	∞	8		1	2	3	4	5
	∞	12	0	18	3		1	2	3	4	5
	5	4	∞	0	∞		1	2	3	4	5
	4	8	∞	10	0		1	2	3	4	5

Stage 1

D =	0	3	∞	5	∞	N =	1	2	3	4	5
	∞	0	12	∞	8		1	2	3	4	5
	∞	12	0	18	3		1	2	3	4	5
	5	4	∞	0	∞		1	2	3	4	5
	4	7	∞	9	0		1	1	3	1	0

Stage 2

D =	0	3	15	5	11	N =	1	2	2	4	2
	∞	0	12	∞	8		1	2	3	4	5
	∞	12	0	18	3		1	2	3	4	5
	5	4	16	0	12		1	2	2	4	2
	4	7	19	9	0		1	1	1	4	5

Stage 3

$$D = \begin{matrix} 0 & 3 & 15 & 5 & 11 \\ \infty & 0 & 12 & 30 & 8 \\ \infty & 12 & 0 & 18 & 3 \\ 5 & 4 & 16 & 0 & 12 \\ 4 & 7 & 19 & 9 & 0 \end{matrix} \qquad N = \begin{matrix} 1 & 2 & 2 & 4 & 2 \\ 1 & 2 & 3 & 3 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 2 & 4 & 2 \\ 1 & 1 & 1 & 4 & 5 \end{matrix}$$

Stage 4

$$D = \begin{matrix} 0 & 3 & 15 & 5 & 11 \\ 35 & 0 & 12 & 30 & 8 \\ 23 & 12 & 0 & 18 & 3 \\ 5 & 4 & 16 & 0 & 12 \\ 4 & 7 & 19 & 9 & 0 \end{matrix} \qquad N = \begin{matrix} 1 & 2 & 2 & 4 & 2 \\ 3 & 2 & 3 & 3 & 5 \\ 4 & 2 & 2 & 4 & 5 \\ 1 & 2 & 2 & 4 & 2 \\ 1 & 1 & 1 & 4 & 5 \end{matrix}$$

Stage 5

$$D = \begin{matrix} 0 & 3 & 15 & 5 & 11 \\ 12 & 0 & 12 & 17 & 8 \\ 7 & 10 & 0 & 12 & 3 \\ 5 & 4 & 16 & 0 & 12 \\ 4 & 7 & 19 & 9 & 0 \end{matrix} \qquad N = \begin{matrix} 1 & 2 & 2 & 4 & 2 \\ 5 & 2 & 3 & 5 & 5 \\ 5 & 5 & 3 & 5 & 5 \\ 1 & 2 & 2 & 4 & 2 \\ 1 & 1 & 1 & 4 & 5 \end{matrix}$$

Ex

$$d_{32} = 10 \qquad n_{32} = 5,\ n_{52} = 1,\ n_{12} = 2$$

Shortest path from 3 to 2 is (3,5,1,2)

Theorem 7.1

Floyds algorithm finds a shortest path between any pair of nodes.

Proof

We shall prove this by induction, the hypothesis being that at the beginning of the kth stage d_{ij} is the minimum length of a path from i to j with intermediate nodes taken from 1,2,....,k-1. This is true for k = 1 and so assume it to be true for a general k. Now paths from i to j which only use 1,2,....,k as intermediate nodes either use node k or they do not. The minimum length of such paths which do not use k is the value of d_{ij} at the beginning of the k'th stage. A path from i to j which uses k

is the catenation of a path from i to k and a path from k to j and both these paths use only $1,2,\ldots,k-1$ as intermediate nodes. Thus the minimum length of a path from i to j which only uses $1,2,\ldots,k$ as intermediate nodes is $\min \{d_{ij}, d_{ik}+d_{kj}\}$ which is the value of d_{ij} at the beginning of stage $k+1$. □

Department of Mathematics
CARNEGIE MELLON UNIVERSITY

Operations Research II

Notation

$x \oplus y = \min\{x,y\}$, $\bigoplus_{i=1}^{n} x_i = \min\{x_1, x_2, \ldots, x_n\}$

$x \otimes y = x + y$

If $A = \|a_{ij}\|$, $B = \|b_{ij}\|$ are $n \times n$ matrices then $A \oplus B = \|c_{ij}\|$ where

$$c_{ij} = \bigoplus_{k=1}^{n} a_{ik} \oplus b_{kj} = \min \{a_{ik} + b_{kj} = 1 \leq k \leq n\}$$

e.g.

$$\begin{bmatrix} 4 & 2 \\ -1 & 3 \end{bmatrix} \otimes \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 5 \\ 1 & 0 \end{bmatrix}.$$

Now let D be a digraph and $L = \|\ell_{ij}\|$ where ℓ_{ij} = length of arc (i,j).

Claim

$$L^t = L \otimes L \otimes \ldots \otimes L = \|\ell_{ij}^{(t)}\|$$

satisfies

$\ell_{ij}^{(t)}$ = min length of a walk from i to j using t arcs or less.

Proof

By induction on t. Obvious for $t = 1$. □

Let $\lambda_{ij}^{(u)}$ = minimum length of a walk from i to j using u arcs or less. Now

(1) $$\lambda_{ij}^{(u+1)} = \min \{\lambda_{ik}^{(u)} + \ell_{kj}: \quad k = 1,2,\ldots,n\}.$$

To see this note that for any $u \geq 1$

(a) $$\lambda_{ij}^{(u+1)} \leq \min \{\lambda_{ik}^{(u)} + \ell_{kj}: \quad k = 1,2,\ldots,n\}.$$

Since each term in the minimisation is the length of some walk from i to j using $\leq u + 1$ arcs

and

(b) If $W = (i = i_1, i_2, \ldots, i_v = j)$, $v \leq u + 1$ is a shortest walk using $\leq u + 1$ arcs then

$$\lambda_{ij}^{(u+1)} = \ell(i_1, i_2, \ldots, i_{v-1}) + \ell_{i_{v-1}j} \geq \lambda_{ii_{v-1}}^{(u)} + \ell_{i_{v-1}j}$$

$$\geq \min \{\lambda_{ik}^{(u)} + \ell_{kj}: \quad k = 1,2,\ldots,n\}.$$

(a) and (b) imply (1).

We can now use induction on t to prove the claim.

Now

$$\ell_{ij}^{(t+1)} = \bigoplus_{k=1}^{n} (\ell_{ik}^{(t)} \otimes \ell_{kj})$$

$$= \min \{\lambda_{ik}^{(t)} + \ell_{kj} : k = 1,2,\ldots,n\}$$

$$= \min \{\lambda_{ik}^{(t)} + \ell_{kj} : k = 1,2,\ldots,n\} \text{ induction}$$

$$= \lambda_{ij}^{(t+1)} \qquad \text{by (1).}$$

□

Now if D has no negative circuits then $\lambda_{ij}^{(n-1)}$ = minimum length of a <u>path</u> from i to j (why?) and so

D^{n-1} is the matrix of shortest path lengths.

Question: if there are no negative cycles, why is

$$D^r = D^s \quad \text{for all} \quad r,s, \geq n-1 \ ?$$

Operations Research II

Assignment Problems

Suppose that there are m jobs available and m people are to be considered for filling these jobs. Suppose it has been estimated that if person i is given job j then the cost of the job will be c_{ij}. The problem is how to assign people to jobs in such a way as to minimise the total cost.

We show how this problem can be solved as a sequence of shortest path problems.

Notation

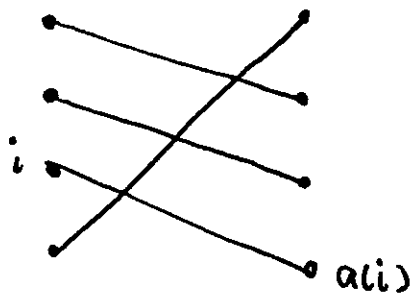
An assignment a is a permutation of $[m] = \{1,\ldots,m\}$.

A solution to the assignment problem can be expressed as person i does job $a(i)$ for some assignment a. Let $A_m = \{\text{assignments}\}$. Then the problem can be expressed

$$\text{minimise} \sum_{i=1}^{m} c_{ia(i)}$$

$$\text{subject to} \quad a \in A_m$$

Another, fruitful, way of viewing this problem is that we seek a minimum weight perfect matching of the complete bipartite graph $K_{m,m}$.



c_{ij} = 'weight' of edge (i,j)

A k-assignment a is a permutation of $[k]$. For $a \in A_k$ we let

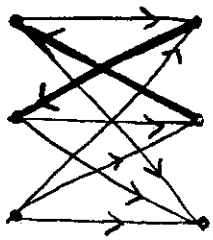
$$c(a) = \sum_{i=1}^{k} c_{ia(i)}.$$

For $a \in A_k$ define the bipartite digraph $G(a)$ with nodes $V_m = \{v_1, v_2, \ldots, v_m\}$, $W_m = \{w_1, w_2, \ldots, w_m\}$ and arcs $B(a) \cup F(a)$ where

$B(a) = \{(w_{a(i)}, v_i) : 1 \leq i \leq k\}$ Backward Arcs

$F(a) = \{(v_i, w_j) : i > k \text{ or } i \leq k \text{ and } i \neq a(i)\}$ Forward Arcs

Example $m = 3$, $k = 2$, $a(1) = 2$, $a(2) = 1$



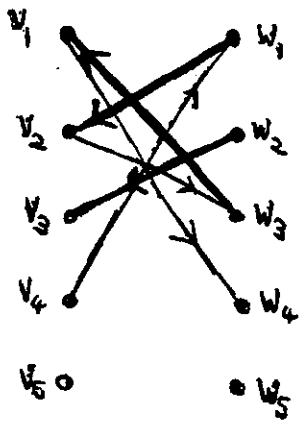
Arc lengths in $G(a)$ are defined as follows:

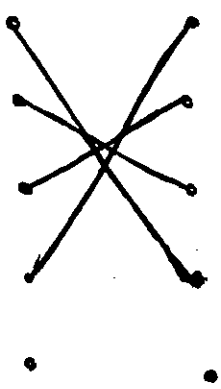
if $(v_i, w_j) \in F(a)$ $\ell(i,j) = c_{ij}$

if $(w_j, v_i) \in B(a)$ $\ell(j,i) = -c_{ij}$

Suppose next that $a \in A_k$, $k < m$ and $P = (v_{k+1} = v_{i_1}, w_{j_1}, \ldots, v_{i_p}, w_{j_p} = w_{k+1})$ is a path in $G(a)$ from $v_{k+1} \in V_m$ to $w_{k+1} \in W_m$

$a(1) = 3$
$a(2) = 1$
$a(3) = 2$





$\hat{a} = a * P$

add $V_4 W_1$
drop $V_2 W_1$
add $V_2 W_3$
drop $V_1 W_3$
add $V_1 W_4$

$P = V_4, W_1, V_2, W_3, V_1, W_4$

We can construct a new assignment $\hat{a}$ (denoted by $a*P$) where

$$\hat{a}(i_t) := j_t \qquad t = 1,\ldots,p$$
$$\hat{a}(i) = a(i) \qquad \text{otherwise}$$

It is not too difficult to see that $a * P \in A_{k+1}$ and further that

$$c(a*P) = c(a) + c_{i_1,j_1} - c_{i_2,j_1} + c_{i_2,j_2} - \ldots + c_{i_p,j_p}$$

$$= c(a) + \ell(P).$$

We have a similar result if $C = (v_{i_1}, w_{j_1}, \ldots, v_{i_p}, w_{j_p}, v_{i_{p+1}} = v_{i_1})$ is a cycle in $G(a)$ - assuming $i_1 \leq k$.

We again define $a*C$ and this time $c(a*C) = c(a) + \ell(C)$ and $a*C \in A_k$

We need the following lemma: for a digraph $G = (V,E)$ we define for each $v \in V$

$$d^+(v) = |\{(v,j) \in E\}| \quad - \text{ outdegree of } v$$
$$d^-(v) = |\{(i,v) \in E\}| \quad - \text{ indegree of } v$$

Lemma 1

Suppose that for digraph $G = (V,E)$ we have $d^+(v), d^-(v) \leq 1$ for all $v \in V$. Then G is a collection of isolated nodes ($d^+(v) = d^-(v) = 0$) plus a collection of node disjoint cycles $C_1,\ldots,C_p$ plus a collection of node disjoint paths $P_1,\ldots,P_q$ where $q = |\{v : d^+(v) = 1, d^-(v) = 0\}|$

$$= |\{v : d^+(v) = 0, d(v) = 1\}|$$

Proof (Outline)

By induction on $|E|$. Trivially true for $|E| = 0$ and so assuming it is true for all digraphs with $|E| < m$ and suppose we have a digraph with $|E| = m > 0$. Delete an arc (x,y), apply the induction hypothesis and then put (x,y) back - which either turns a path into a cycle or lengthens a path or joins 2 isolated nodes. All cases maintain the result. □

A k-assignment a will be called σ-optimal if for any $a' \in A_k$ with we have $c(a) \leq c(a')$.

Theorem

Let $a \in A_k$ be σ-optimal

(i) If C is a cycle in $G(a)$ then $\ell(C) \geq 0$

(ii) Let P be a shortest path from v_{k+1} to w_{k+1}. Then $a*P$ is also σ-optimal.

Proof

(i) Now $a*C \in A_k$ and so $c(a*C) = c(a) + \ell(C) \geq c(a)$ by assumption. Hence $\ell(C) \geq 0$.

(ii) Let $\hat{a}$ be any $k+1$ - assignment. We show that there exist node disjoint cycles $C_1,\ldots,C_q$ $q \geq 0$ and a path Q from v_{k+1} to w_{k+1} such that

$$\hat{a} = a * C_1 * \ldots * C_q * Q$$

(strictly speaking we need brackets on RHS of the above). Then

$$c(\hat{a}) = c(a) + \sum_{i=1}^{q} \ell(C_i) + \ell(Q)$$

$$\geq c(a) + \ell(Q) \qquad \text{by (i)}$$

$$\geq c(a) + \ell(P) \qquad \text{by assumption}$$

$$= c(a*P)$$

To construct $C_1,\ldots,C_q$ and Q consider the subset X of arcs of $G(a)$ defined by

$$X = \{(v_i, w_{\hat{a}(i)}) : i = k+1 \text{ or } i \leq k \text{ and } a(i) \neq \hat{a}(i)\}$$
$$\cup\{(w_{a(i)}, v_i) : i \leq k \text{ and } a(i) \neq \hat{a}(i)\}$$

One then checks that the digraph $(V_{k+1} \cup W_{k+1}, X)$ satisfies the conditions of the lemma, that $d^+(v_{k+1}) = 1$, $d^-(v_{k+1}) = 0$ and $d^+(w_{k+1}) = 0$, $d^-(w_{k+1}) = 1$ and that all other vertices are isolated or satisfy $d^+(v) = d^-(v) = 1$. The cycles and path of the lemma are what we need. □

Assignment Algorithm

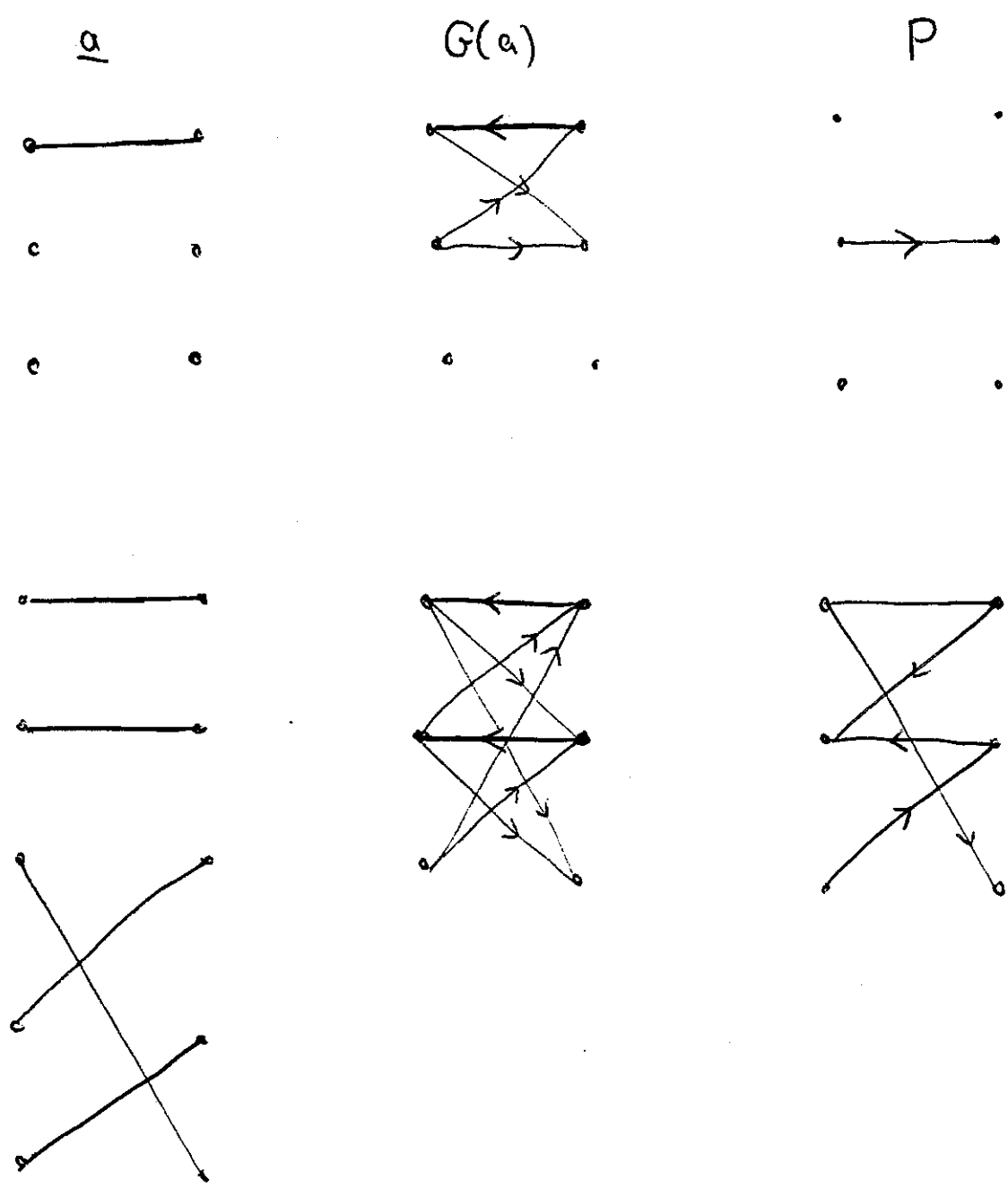
(Initial Description)

```
begin
  a(1) = 1,
  for k = 2 to m do
  begin
    construct G(a);
    find a shortest path P from v_k to w_k;
    a = a*P
  end
end
```

By Theorem 1 a remains σ-optimal and after m-iterations we will have solved the problem. As arc lengths could be negative, each shortest path problem could take $O(m^3)$ steps and so the time complexity of the whole algorithm is $O(m^4)$. ~~We will subsequently show how to reduce this to $O(m^3)$.~~

Example

$$n = 3 \begin{bmatrix} 6 & 4 & 5 \\ 4 & 2 & 4 \\ 7 & 2 & 5 \end{bmatrix} \leftarrow \|c_{ij}\|$$


a
G(a)
P

Linear Programming Formulation

$$\text{ALP} = \text{minimise} \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij} x_{ij}$$

subject to

$$(1) \qquad \sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1,2,\ldots,m$$

$$(2) \qquad \sum_{j=1}^{m} x_{ij} = 1 \qquad i = 1,2,\ldots,m$$

$$(3) \qquad x_{ij} \geq 0 \quad \forall\ i,j.$$

Let $X_F = \{\underline{x}:$ (1) - (3) hold$\}$

and

$$X_I = \{\underline{x} \in X_F: \ x_{ij} = 0 \text{ or } 1, \ \forall\ i,j\}.$$

Assignment Problem is equivalent to minimise $c^T x$ subject to $x \in X_I$.

Proof

If **a** is an assignment then putting

$$x_{i\ a(i)} = 1 \qquad i = 1,2,\ldots,m$$

$$x_{ij} = 0 \qquad j \neq a(i)$$

gives a solution to ALP with value $c(\mathbf{a})$.

Conversely, if $x \in X_I$ define $\mathbf{a}$ by

$$\mathbf{a}(i) = \underline{\text{unique}}\ \ j \ \text{ such that } \ x_{ij} = 1$$

(uniqueness follows from (1))

$\mathbf{a}$ is a permutation for if $\mathbf{a}(r) = \mathbf{a}(s) = k$ then

$$\sum_{i=1}^{m} x_{ik} \geq x_{rk} + x_{sk} = 2, \quad \text{contradiction.} \qquad \square$$

We show that

$$z_I = \min c^T x \ \text{ subject to } \ x \in X_I = [\min\{c(\mathbf{a}): \mathbf{a} \in A\}]$$
$$= \min c^T x \ \text{ subject to } \ x \in X_F = z_F \ , \text{ say.}$$

Proof

Consider dual problem

$$\text{DALP} = \text{maximise} \ \sum_{i=1}^{m} y_i + \sum_{j=1}^{m} z_j$$

$$\text{subject to } \ y_i + z_j \leq c_{ij} \ \ \forall\ i,j$$

We show that if $\mathbf{a}^*$ is the assignment constructed by our algorithm then there exists (y^*, z^*) satsifying

(4) $$y_i^* + z_j^* \leq c_{ij} \quad 1 \leq i,j \leq m$$

(5) $$y_i^* + z_{a^*(i)}^* = c_{ia^*(1)} \quad 1 \leq i \leq m$$

Now (4) implies y^*,z^* is a dual solution and so

(6) $$z_F \geq \sum_{i=1}^{m} y_i^* + \sum_{j=1}^{m} z_i^*. \qquad \begin{matrix}[z_F \text{ is max. dual} \\ [\text{value}\end{matrix}$$

But, from (5)

$$\sum_{i=1}^{m} c_{ia^*(i)} = \sum_{i=1}^{m} y_i^* + \sum_{i=1}^{m} z_{a^*(i)}^*$$

(7) $$= \sum_{i=1}^{m} y_i^* + \sum_{j=1}^{m} z_j^*$$

since a^* is a permutation.

Hence $z_F \geq c(a^*) \geq z_F$, where the first inequality follows from (6), (7) and the second from duality.

To construct y^*,z^* let $\hat{a}$ be the (m-1)-assignment produced just before a^*. For $x \in V \cup W$ let $d(x)$ = length of shortest path from v_m to x in $G(\hat{a})$.

Let

$$y_i^* = -d(v_i) \quad i = 1,2,\ldots,m$$

$$z_j^* = d(w_j) \quad j = 1,2,\ldots,m.$$

Observe first that

$$d(v_i) = d(w_{\hat{a}(i)}) + \ell(w_{\hat{a}(i)}, v_i) \qquad 1 \leq i \leq m-1$$

since $(w_{\hat{a}(i)}, v_i)$ is the unique arc entering v_i in $G(\hat{a})$. Equivalently

$$-y_i^* = z_{\hat{a}(i)}^* - c_{i,\hat{a}(i)}. \tag{8}$$

This verifies (5) whenever $\hat{a}(i) = a^*(i)$, and also (4) for all $i,j = \hat{a}(i)$. But if $a^*(i) \neq \hat{a}(i)$ then the last path P found contains the arc $(v_i, w_{a^*(i)})$ and for an arc of P we have

$$d(w_{a^*(i)}) = d(v_i) + \ell(v_i, w_{a^*(i)})$$

or

$$z_{a^*(i)}^* = -y_i + c_{i,a^*(i)}$$

and (5) is verified for all i.

We only now need check (4) for arcs of $F(\hat{a})$ which are not on P. But for such arcs we have

$$d(w_j) \leq d(v_i) + \ell(v_i, w_j).$$

□

Notes on optimization

August 21, 2018

1 Optimization Problems

We consider the following problem:

$$\text{Minimize } f(\mathbf{x}) \text{ subject to } \mathbf{x} \in S, \tag{1}$$

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

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

Local versus Global Optima: $\mathbf{x}^*$ is a *global minimum* if it is an actual minimizer in (1).

$\mathbf{x}^*$ 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*.

2 Convex sets and functions

2.1 Convex Functions

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

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

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

$$f(y) \geq f(x) + f'(x)(y - x) \text{ for all } x, y. \tag{2}$$

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 (2).

Now suppose that (2) 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 $f(\mathbf{y}) \geq f(\mathbf{x}) + (\nabla f(\mathbf{x}))^T(\mathbf{y} - \mathbf{x})$ 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 (2). □

F5 It follows from F3 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

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

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.

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

□

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

$$\frac{a_1 + a_2 + \cdots + a_m}{m} \geq (a_1 a_2 \cdots a_m)^{1/m}. \tag{3}$$

$-\log(x)$ is a convex function for $x \geq 0$. So, applying (3),

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

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

2.2 Convex Sets

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

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

See Diagram 3 at the end of these notes.

Examples of convex sets:

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

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

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

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

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

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

Operations on convex sets:

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

O2 S, T convex implies that $A = S \cap T$ is convex. $\mathbf{x}, \mathbf{y} \in A$ implies that $\mathbf{x}, \mathbf{y} \in S$ and so $L = L(\mathbf{x}, \mathbf{y}) \subseteq S$. 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 $\mathbf{z}_i = \alpha \mathbf{x}_i + \beta \mathbf{y}_i \in T, i = 1, 2$ then

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

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 2.1. *Let f, S both be convex in* (1)*. 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 2.2. *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 $\mathbf{x}, \mathbf{y} \in S$ i.e. $f(\mathbf{x}), f(\mathbf{y}) \leq b$. Then for $0 \leq \lambda \leq 1$

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

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

3 Algorithms

3.1 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) $f(a_2) \le f(a_3)$. 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

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

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 $\alpha^t D \le \delta$ where D is our initial uncertainty.

3.2 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

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

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

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

Then we have

$$\begin{aligned} |\mathbf{x}_{k+1} - \mathbf{x}^*|^2 &= |\mathbf{x}_k - \mathbf{x}^*|^2 - 2\alpha_k \nabla f(\mathbf{x}_k)^T(\mathbf{x}_k - \mathbf{x}^*) + \alpha_k^2 |\nabla f(\mathbf{x}_k)^2| \\ &\le |\mathbf{x}_k - \mathbf{x}^*|^2 - 2\alpha_k(f(\mathbf{x}_k) - f(\mathbf{x}^*)) + \alpha_k^2 |\nabla f(\mathbf{x}_k)|^2. \end{aligned} \tag{4}$$

The inequality comes from F3.

Applying (4) repeatedly we get

$$|\mathbf{x}_k - \mathbf{x}^*|^2 \le |\mathbf{x}_0 - \mathbf{x}^*|^2 - 2\sum_{i=1}^{k} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) + \sum_{i=1}^{K} \alpha_i^2 |\nabla f(\mathbf{x}_k)|^2. \tag{5}$$

Putting $R = |\mathbf{x}_0 - \mathbf{x}^*|$, we see from (5) that

$$2\sum_{i=1}^{k} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) \le R^2 + \sum_{i=1}^{K} \alpha_i^2 |\nabla f(\mathbf{x}_k)|^2. \tag{6}$$

On the other hand,

$$\sum_{i=1}^{k} \alpha_i (f(\mathbf{x}_i) - f(\mathbf{x}^*)) \ge \left(\sum_{i=1}^{k} \alpha_i\right) \min\{f(\mathbf{x}_k) - f(\mathbf{x}^*) : i \in [k]\} = \left(\sum_{i=1}^{k} \alpha_i\right) (f(\mathbf{x}_{min} - f(\mathbf{x}^*)), \tag{7}$$

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

Combining (6) and (7) we get

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

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

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

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

4 Separating Hyperplane

See Diagram 7 at the end of these notes.

Theorem 4.1. *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) $\mathbf{a}^T\mathbf{x} \ge b$ and (ii) $C \subseteq \{\mathbf{y} \in \mathbb{R}^n : \mathbf{a}^T\mathbf{y} \le b\}$.*

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 $\mathbf{y} \in C$ such that $\mathbf{a}^T\mathbf{y} > b$. Let $\mathbf{w} \in C$ be the closest point to $\mathbf{x}$ on the line segment $L(\mathbf{y}, \mathbf{z}) \subseteq C$. 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 $\mathbf{y} \in \bar{C}$ we have $\mathbf{a}_n^T \mathbf{y} \leq b_n$ for all n. Taking limits we see that $\mathbf{a}^T \mathbf{y} \leq b$. Furthermore, for $\mathbf{y} \notin \bar{C}$ we see that for large enough n, $\mathbf{a}_n^T \mathbf{y} > b_n$. taking limits we see that $\mathbf{a}^T \mathbf{y} \leq b$. □

Corollary 4.2. *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 4.1 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 4.3 (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 $\mathbf{u} \in \mathbb{R}^m$ such that $\mathbf{u}^T A \geq \mathbf{0}$ and $\mathbf{u}^T \mathbf{b} < 0$.

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

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

Suppose then that (i) fails to hold. Let $S = \{\mathbf{y} : \mathbf{y} = A\mathbf{x} \text{ for some } \mathbf{x} \geq \mathbf{0}\}$. 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). □

4.1 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 4.4. *$conv(S)$ is a convex set.*

Proof. Let $\mathbf{x} = \sum_{i \in I} \lambda_i \mathbf{x}_i, \mathbf{y} = \sum_{j \in J} \mu_j \mathbf{y}_j \in conv(S)$. 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

$$\alpha\mathbf{x} + (1-\alpha)\mathbf{y} = \sum_{i \in K} (\alpha\lambda_1 + (1-\alpha)\mu_i)\mathbf{x}_i \text{ and } \sum_{i \in K} (\alpha\lambda_1 + (1-\alpha)\mu_i) = 1$$

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

Lemma 4.5. *If S is convex, then $S = conv(S)$.*

Proof. Exercise. □

Corollary 4.6. *$conv(conv(S)) = conv(S)$ for all $S \subseteq \mathbb{R}^n$.*

Proof. Exercise. □

4.1.1 Extreme Points

A point $\mathbf{x}$ of a convex set S is said to be an *extreme point* if **THERE DO NOT EXIST** $\mathbf{y}, \mathbf{z} \in S$ 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 4.7. *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 $H = \{\mathbf{y} : \mathbf{a}^T\mathbf{y} = b\}$ be the hyperplane defined in Theorem 4.1. Let $b^* = \max\{\mathbf{a}^T\mathbf{y} : \mathbf{y} \in S\}$. We have $b^* < \infty$ since S is bounded. Let $H^* = \{\mathbf{y} : \mathbf{a}^T\mathbf{y} = b^*\}$ 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 4.8. *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.

5 Lagrangean Duality

See Diagram 9 at the end of these notes.

Here we consider the *primal problem*

$$\text{Minimize } f(\mathbf{x}) \text{ subject to } g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, m, \tag{8}$$

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

$$\text{Maximize } \phi(\boldsymbol{\lambda}) \text{ subject to } \boldsymbol{\lambda} \geq 0 \text{ where } \phi(\boldsymbol{\lambda}) = \min_{\mathbf{x}\in\mathbb{R}^n} L(\mathbf{x}, \boldsymbol{\lambda}). \tag{9}$$

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

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

D2

Weak Duality: If $\boldsymbol{\lambda}$ is feasible for (9) and $\mathbf{x}$ is feasible for (8) then $f(\mathbf{x}) \geq \phi(\boldsymbol{\lambda})$.

$$\phi(\boldsymbol{\lambda}) \leq L(\mathbf{x}, \boldsymbol{\lambda}) \leq f(\mathbf{x}) \text{ since } \lambda_i \geq 0, g_i(\mathbf{x}) \leq 0,\ i = 1, 2, \ldots, m. \tag{10}$$

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 (10).

Theorem 5.1. *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 4.2 there exists $\boldsymbol{\lambda}, \gamma, b$ such that $(\boldsymbol{\lambda}, \gamma) \neq \mathbf{0}$ and

$$b \leq \min\left\{\boldsymbol{\lambda}^T\mathbf{u} + \gamma t : (\mathbf{u}, t) \in \mathcal{A}\right\}. \tag{11}$$
$$b \geq \max\left\{\boldsymbol{\lambda}^T\mathbf{u} + \gamma t : (\mathbf{u}, t) \in \mathcal{B}\right\}. \tag{12}$$

We deduce from (11) that $\boldsymbol{\lambda} \geq 0$ and $\bar{g} \geq 0$. If $\gamma < 0$ or $\lambda_i < 0$ for some i then the minimum in (11) is $-\infty$. We deduce from (12) that $\gamma t < b$ for all $t < f^*$ and so $\gamma f^* \leq b$. And from (11) that

$$\gamma f(\mathbf{x}) + \sum_{i=1}^m \lambda_i g_i(\mathbf{x}) \geq b \geq \gamma f^* \qquad \text{for all } \mathbf{x} \in \mathbb{R}^n. \tag{13}$$

If $\gamma > 0$ then we can divide (13) 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 (13) we see that $\sum_{i=1}^m \lambda_i g_i(\mathbf{x}^*) \geq 0$ which then implies that $\boldsymbol{\lambda} = 0$, contradiction. □

6 Conditions for a minimum: First Order Condition

6.1 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

$$\nabla f(\mathbf{a}) = 0 \tag{14}$$

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 (14) 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 6.1. *If f is convex then* (14) *is also a sufficient condition.*

Proof. This follows directly from F3. □

6.2 Constrained problem

We will consider Problem (8), but we will not assume convexity, only differentiability. The condition corresponding to (14) 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 (8) is that there exists $\boldsymbol{\lambda}$ such that

$$\nabla f(\mathbf{a}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{a}) = 0. \tag{15}$$

$$\lambda_i \geq 0 \qquad 1 \leq i \leq m. \tag{16}$$

$$\lambda_i g_i(\mathbf{a}) = 0, \qquad 1 \leq i \leq m. \qquad \text{Complementary Slackness} \tag{17}$$

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 (15), (16) and (17) are sufficient for a global minimum.

6.2.1 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 4.3 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 $A\boldsymbol{\lambda} = -\nabla f(\mathbf{a})$.

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

Diagram 1

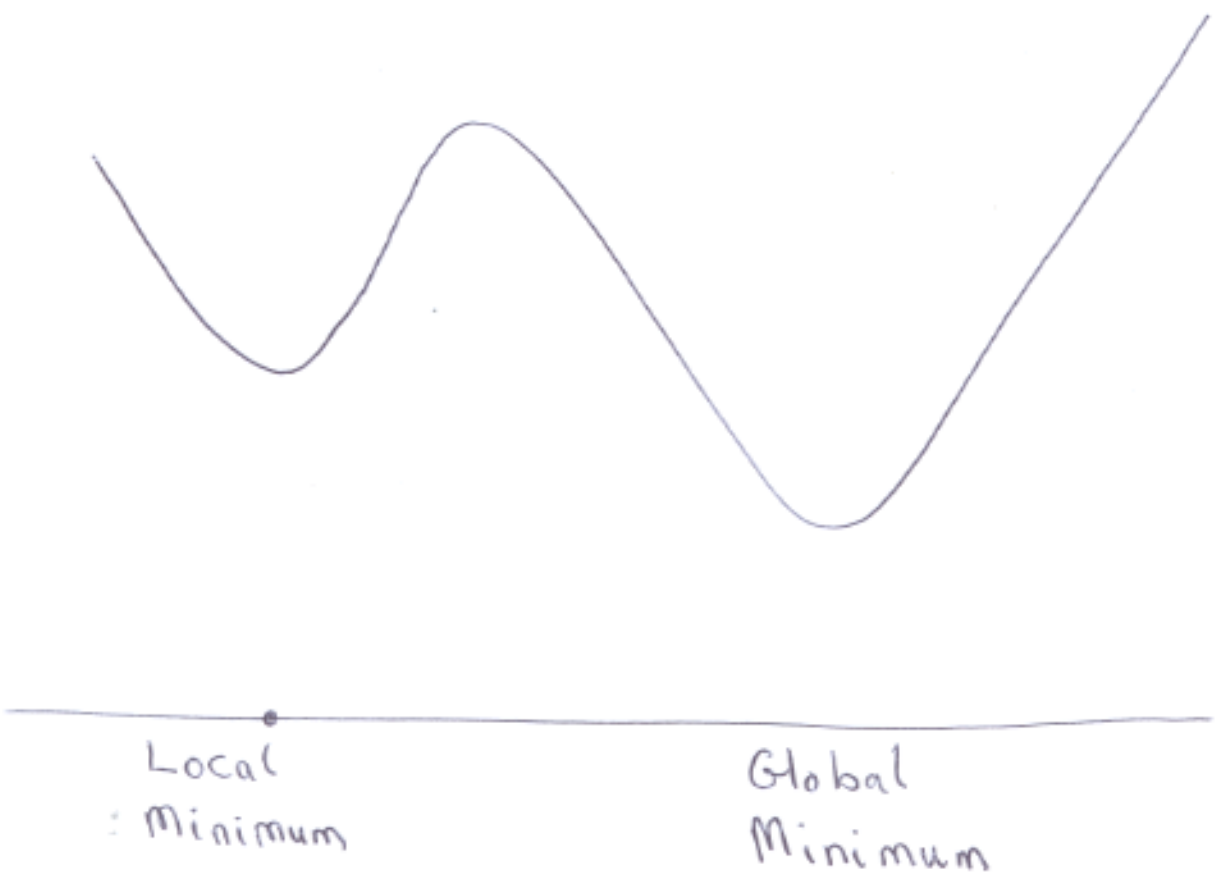
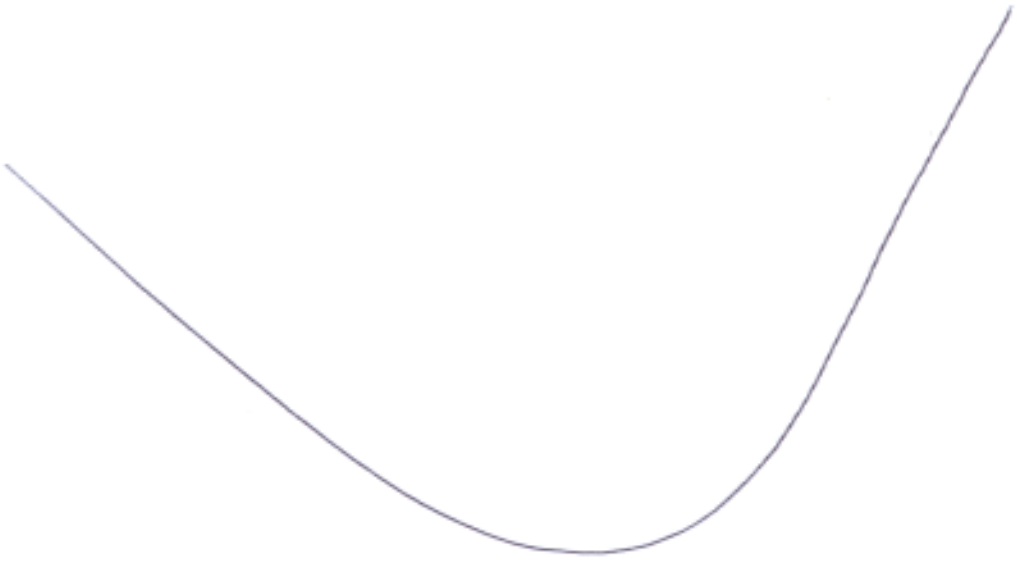
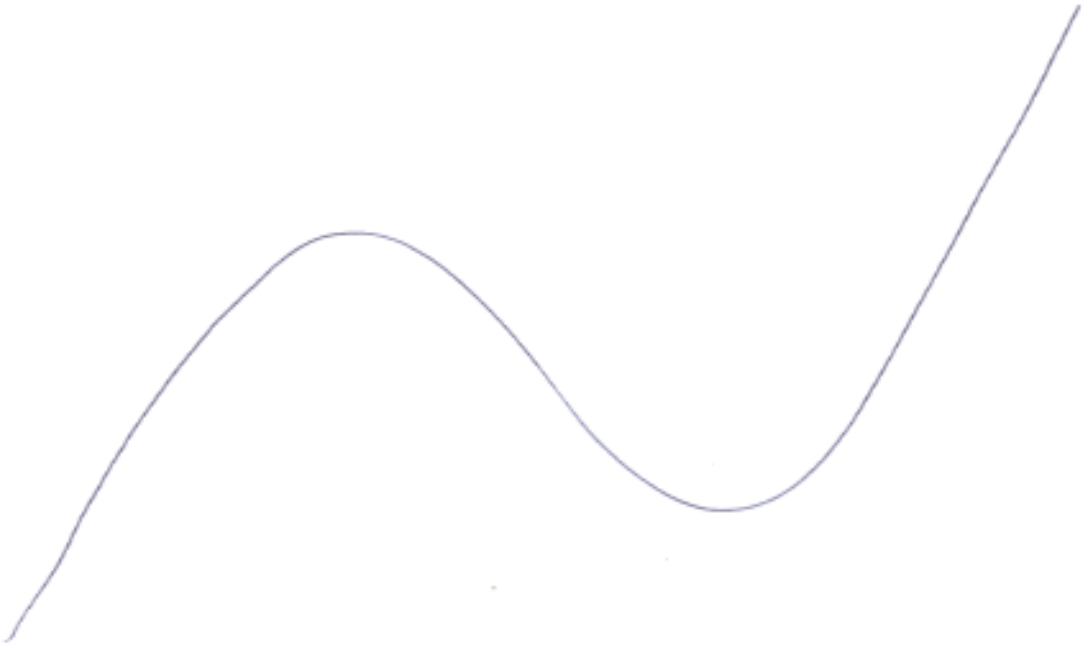


Diagram 2

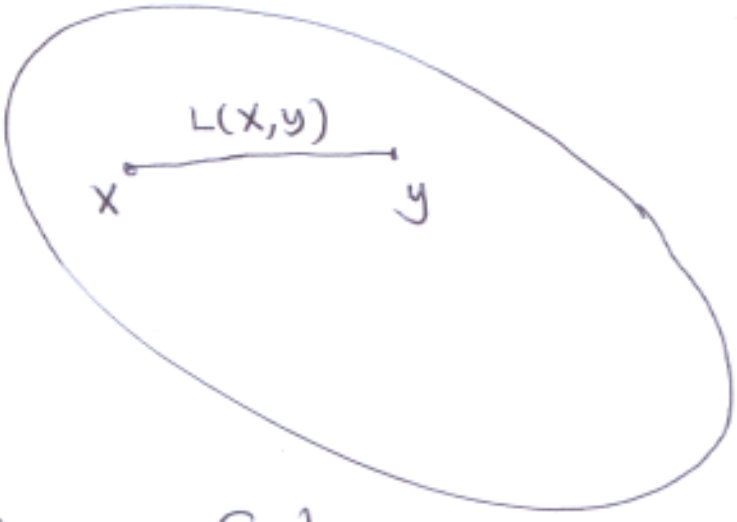


Convex Function

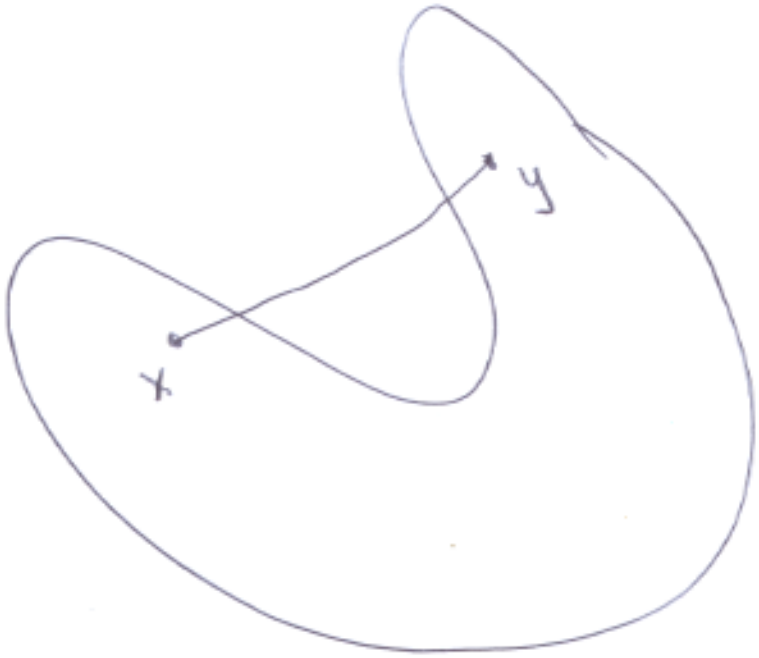


Non-Convex Function

Diagram 3



Convex Set



Non-Convex Set

Diagram 4

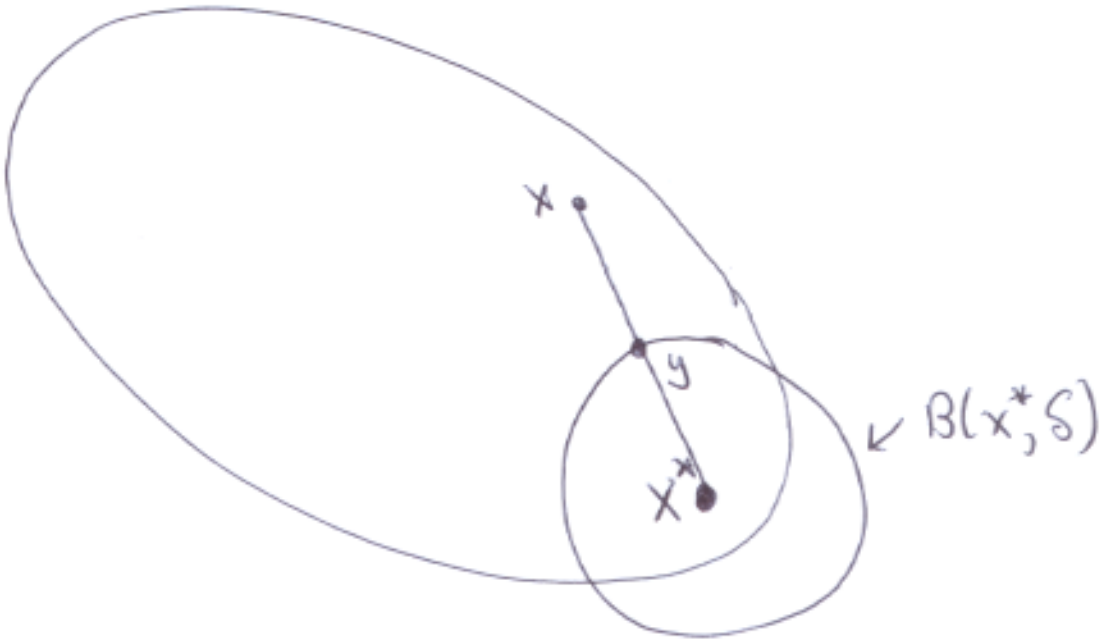


Diagram 5

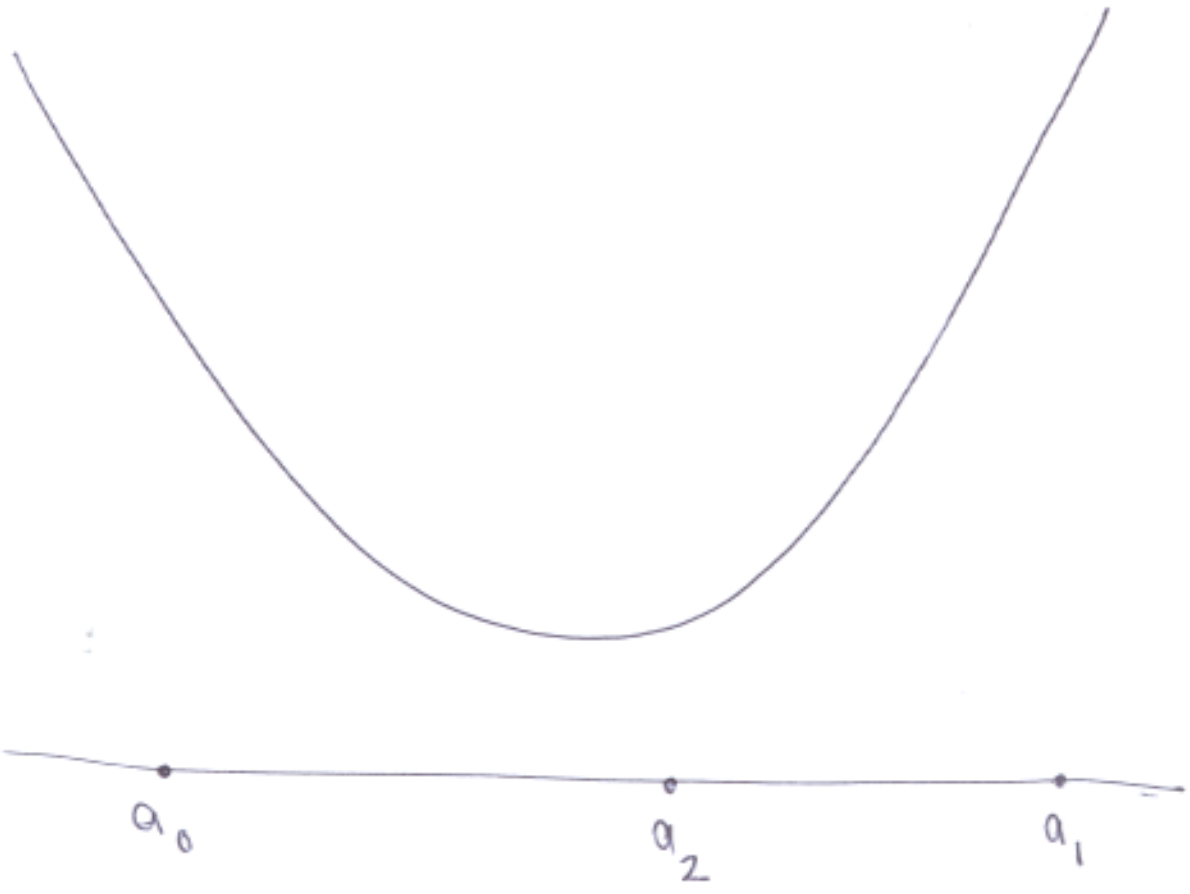


Diagram 6

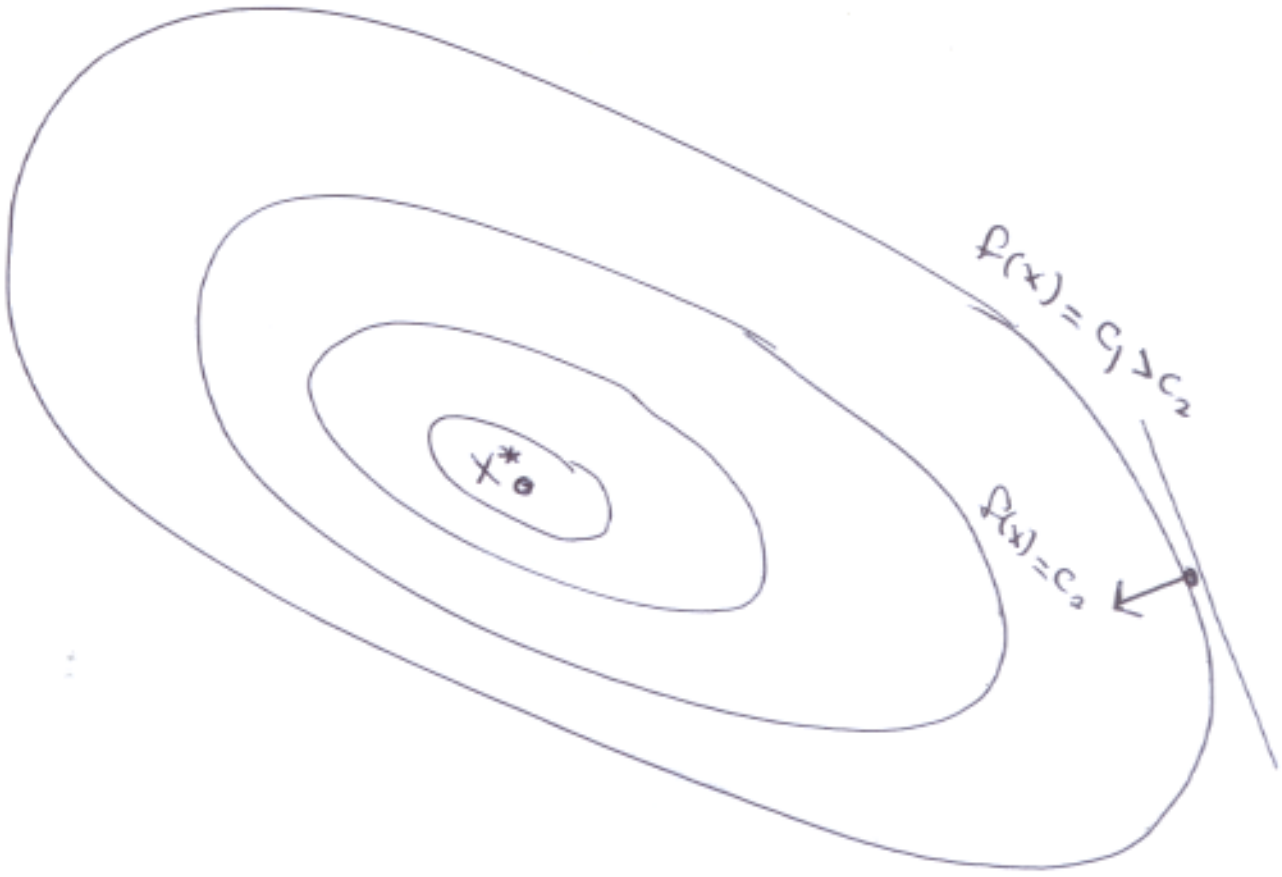


Diagram 7

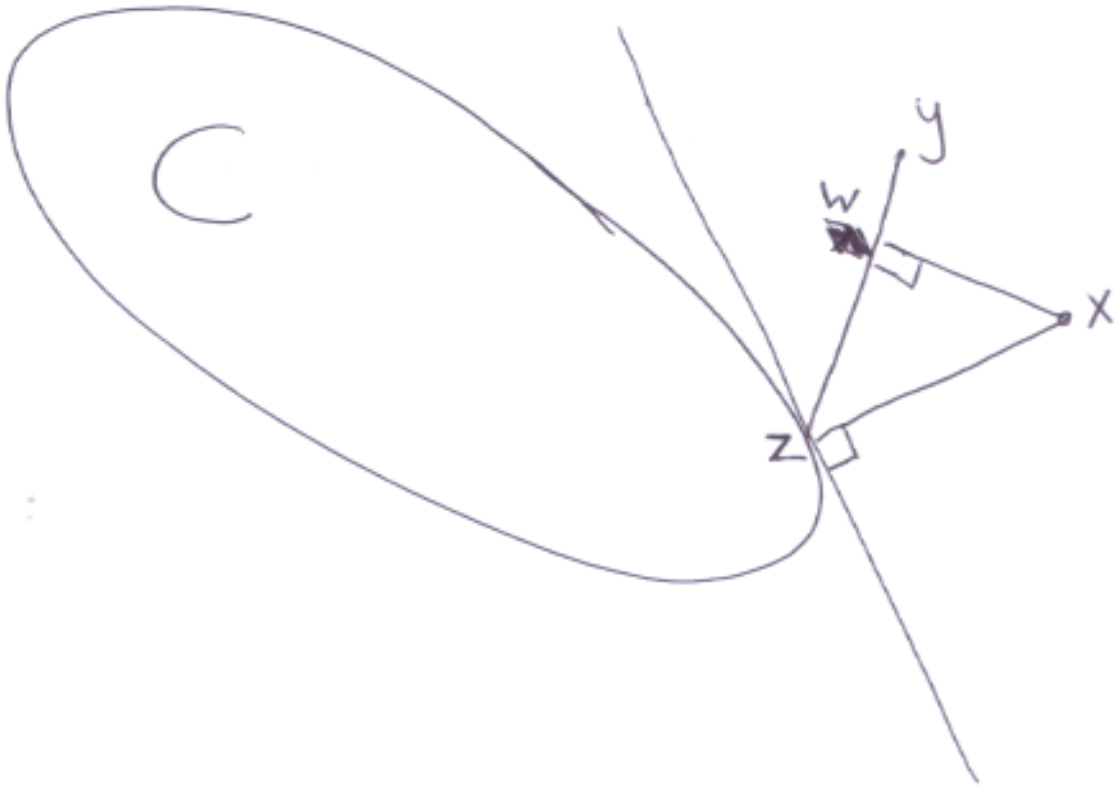


Diagram 8

Diagram 9

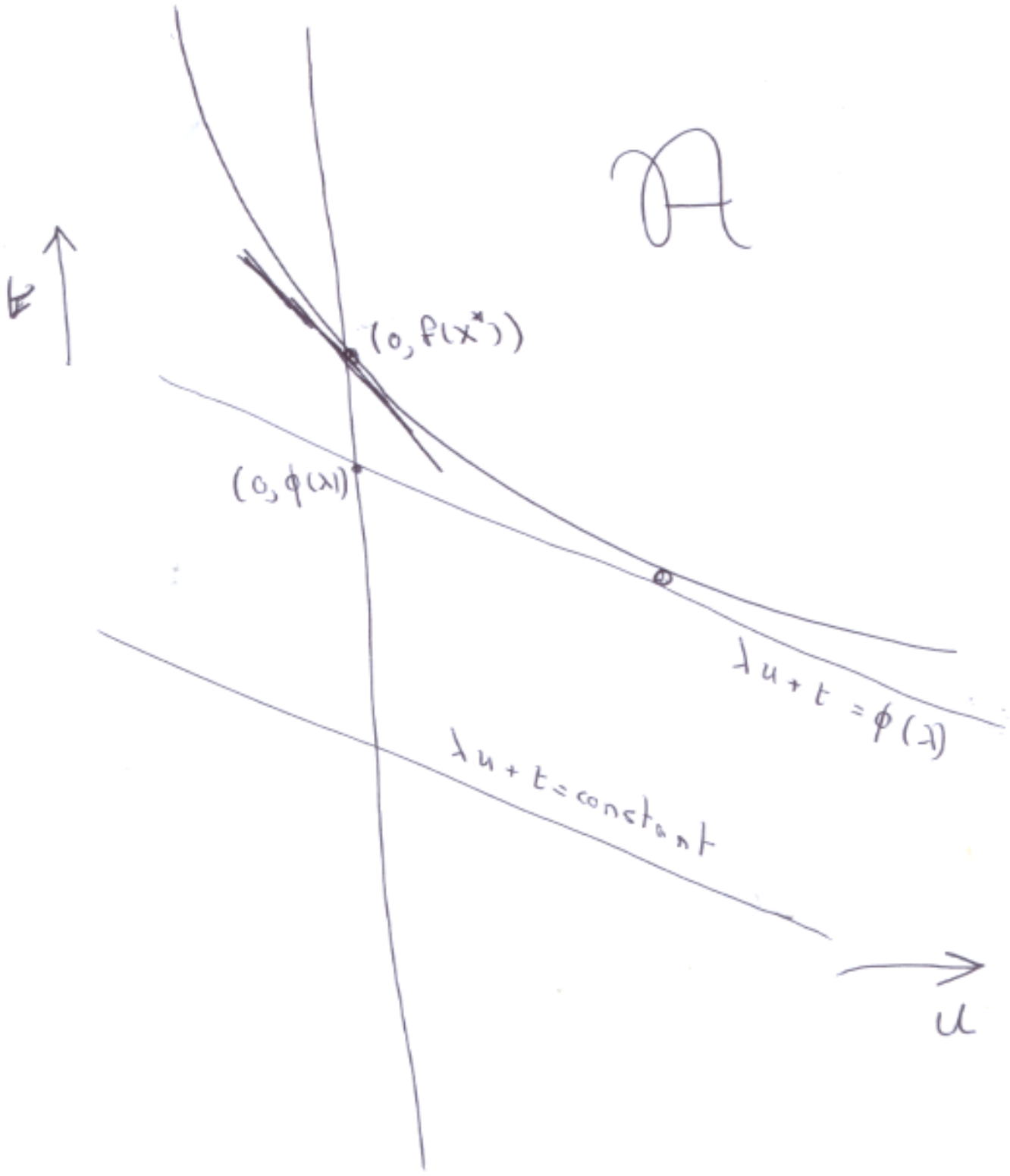
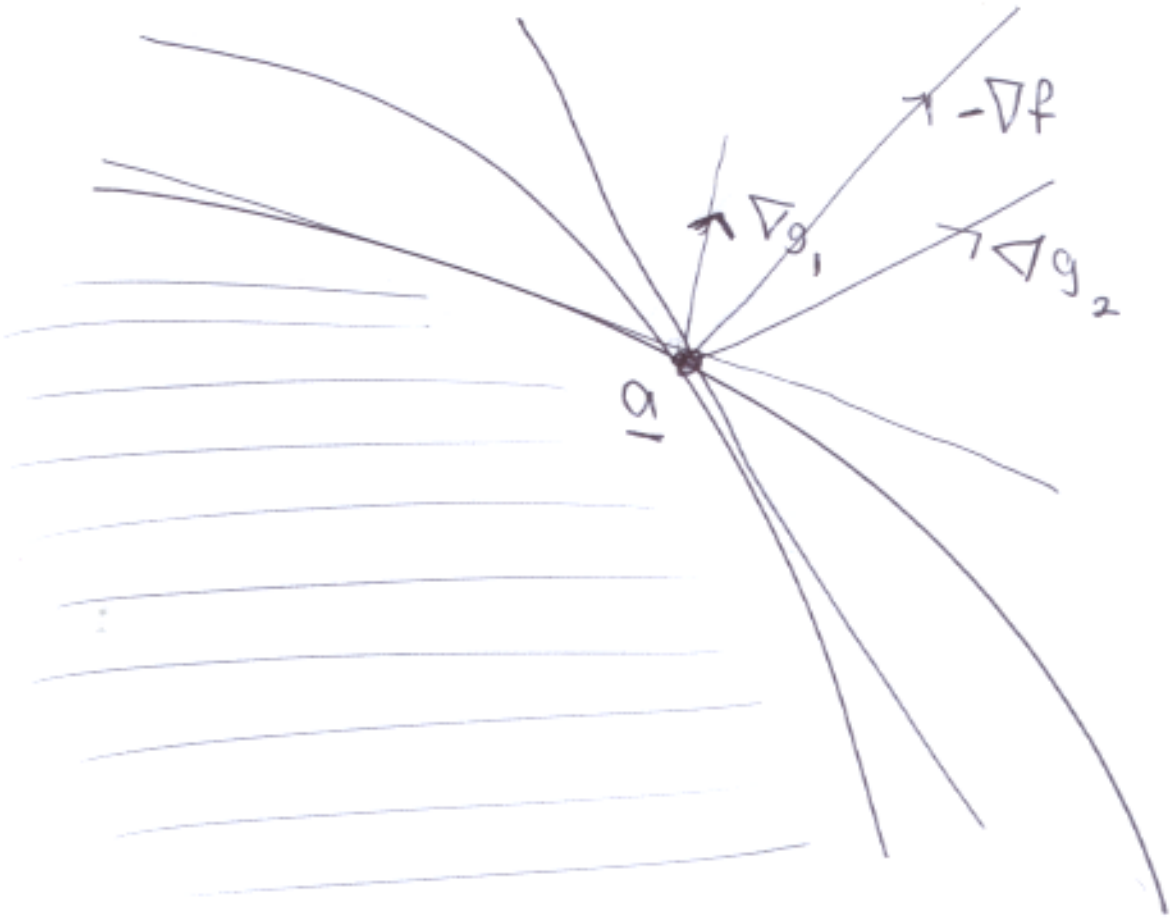


Diagram 10



Integer Linear Programming

This is the name given to L.P. problems which have the extra constraint that some or all variables have to be integer.

Examples

1. 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_{ij} in time period i for i=1,...m. The capital available in time period i is b_i. The problem of minimising revenue subject to the budget constraints can be formulated as follows: let $x_j = 0$ or 1 correspond to not proceeding or respectively proceeding with project j then we have to

$$\text{Maximise} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i = 1,\ldots m$$

$$0 \leq x_j \leq 1 \qquad x_j \text{ integer} \qquad j = 1,\ldots n$$

2. 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 transport cost of supplying the whole of customer j's requirements over the given planning period from potential site i is c_{ij}. 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 m 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_{ij} to be the fraction of customer j's requirements supplied from depot i in a given solution. The problem can then be expressed.

$$\text{Minimise} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i$$

$$\text{subject to} \quad \sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1,\ldots n \qquad *$$

$$x_{ij} \leqslant y_i \qquad i=1,\ldots m,\ j=1,\ldots n$$

$$x_{ij} \geqslant 0 \quad 0 \leqslant y_i \leqslant 1 \quad y_i \text{ integer} \qquad \begin{matrix} i=1,\ldots m \\ j=1,\ldots n \end{matrix}$$

Note that if $y_i = 0$ then $f_i y_i = 0$ and there is no contribution to the total cost. Also $x_{ij} \leqslant y_i$ implies $x_{ij} = 0$ for $j = 1,\ldots n$ and so no goods are distributed from site i. This corresponds exactly to no depot at site 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 $x_{ij} \leqslant y_i$ becomes $x_{ij} \leqslant 1$ which holds anyway from the constraints *.

3. Set Covering

Let $S_1,\ldots S_n$ 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 $S = \bigcup_{j \in I} S_j$. Assume that each subset S_j has cost $c_j > 0$ associated with it. We define the cost of a cover to be ths 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 specififed as follows: define the m x n matrix $A = || a_{ij} ||$ by

$a_{ij} = 1$ if $i \in S_j$

$= 0$ otherwise

Let x_j 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

$$\text{minimise} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad i = 1,\ldots m \tag{15.1}$$

$$x_j = 0 \text{ or } 1$$

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

As an example consider the following simplified airline crew scheduling problem. An airline has m scheduled flight-legs 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 $j = 1,\ldots n$ 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

covering problem. Note that if crews are not allowed to be passengers on a flight, i.e. so that they can be flown to their next flight, then we have to make 15.1 an equality - the set partitioning problem.

General Terminology :

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

$$\text{minimise} \quad x_0 = \underline{c} \cdot \underline{x}$$

subject to

$$A\underline{x} = \underline{b}$$

$$x_j \geq 0 \qquad j = 1 \cdots n$$

$$x_j \text{ integer for } j \in IN$$

where IN is some subset of $N_0 = \{0, 1, \cdots n\}$.

When $IN = N_0$ we have what is called a pure integer programming problem. For such a problem one generally has all given quantities c_j, a_{ij}, b_i integer. One has to be careful here. Consider for example

$$\text{minimise} \quad x_0 = -\tfrac{1}{3}x_1 - \tfrac{1}{2}x_2$$

subject to

$$\tfrac{2}{3}x_1 + \tfrac{1}{3}x_2 \leq \tfrac{1}{3}$$

$$\tfrac{1}{2}x_1 - \tfrac{3}{2}x_2 \leq \tfrac{2}{3}$$

$$x_1, x_2 \geq 0 \text{ and integer}$$

As defined this is not a pure problem. For a start x_0 will not necessarily be integer and neither will the slack variables. If we want to use an algorithm for solving pure problems we must scale the objective and constraints to give

minimise $x_0 = -2x_1 - 3x_2$

subject to

$$2x_1 + x_2 + x_3 = 4$$
$$3x_1 - 9x_2 + x_4 = 4$$
$$x_1, \ldots x_4 \geq 0 \text{ and integer.}$$

A final class of problems is the <u>pure 0-1 programming problem</u>

maximise $x_0 = \underline{c} \cdot \underline{x}$

subject to

$$A\underline{x} \leq \underline{b}$$
$$x_j = 0 \text{ or } 1 \quad \text{for } j = 1 \ldots n.$$

<u>Further Uses of Integer Variables</u>

(1) If a variable x can only take a finite number of values $p_1, \ldots p_m$ we can replace x by the expression

$$p_1 w_1 + \ldots + p_m w_m$$

where $w_1 + \ldots + w_m = 1$

and $w_i = 0 \text{ or } 1 \qquad i = 1, \ldots m$

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

$$c_1 w_1 + c_2 w_2 + c_3 w_3$$

where c_i is the cost of a small plant etc.

(2) 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

e.g.

(a) $0 \leq x \leq M$

$0 \leq x \leq 1$ or $x \geq 2$

can be expressed

$$x \leq 1 + (M - 1)\delta$$

$$x \geq 2 + M(\delta - 1)$$

$$x \geq 0 \quad \delta = 0 \text{ or } 1$$

$x \leq M$ is a notional upper bound to make this approach possible.

(b) $x_1 + x_2 \leq 4$

$x_1 \geq 1$ or $x_2 \geq 1$ but not both ≥ 1

$x_1, x_2 \geq 0$

can be expressed

$$x_1 + x_2 \leq 4$$

$$x_1 \geq \delta$$

$$x_2 \geq 1 - \delta$$

$$x_1 \leq (1-\delta) + 4\delta$$

$$x_2 \leq \delta + 4(1-\delta)$$

$$\delta = 0 \text{ or } 1$$

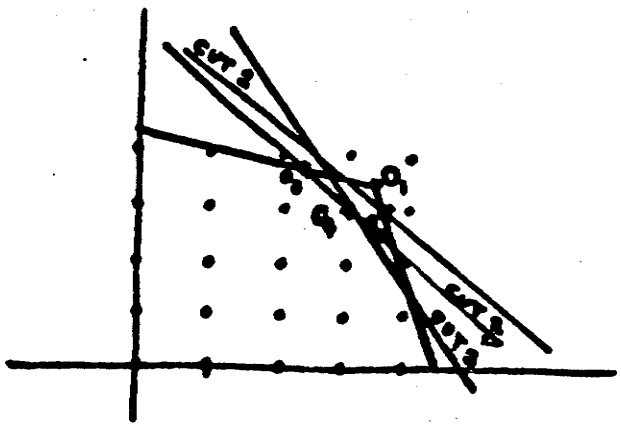
Integer programming problems generally take much longer to solve then the corresponding linear program obtained by ignoring integrality.. It is wise therefore to consider the possibility of solving as a straight forward L.P. and then rounding e.g. in the trim-loss problem. This is not always possible for example if x_1 is a 0 - 1 variable such that $x_1 = 0$ means do not build a plant and $x_1 = 1$ means build a plant then rounding $x_1 = \frac{1}{2}$ is not very satisfactory.

A cutting plane algorithm for the pure problem

The rationale behind this approach is:-

1) Solve the continuous problem as an L.P. i.e. ignore integrality.
2) If by chance the optimal basic variables are all integer then the optimum solution has been found. Otherwise:-
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.
4) Add this new constraint and go to (1).

The idea of such an approach is illustrated below:-



It is straight forward to show that if at any stage the current L.P. solution x is integer it is the optimal integer solution. This is because 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

(15.2) $$a_1 x_1 + \ldots + a_n x_n = b$$

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

For a real number ξ we define $\lfloor \xi \rfloor$ it to be the largest integer $\leq \xi$. Thus $\xi = \lfloor \xi \rfloor + \varepsilon$ where $0 \leq \varepsilon < 1$.

$$\lfloor 6\tfrac{1}{2} \rfloor = 6 \qquad \lfloor 3 \rfloor = 3 \qquad \lfloor -4\tfrac{1}{2} \rfloor = -5$$

Now let $a_j = \lfloor a_j \rfloor + f_j$ and $b = \lfloor b \rfloor + f$ in (15.2) then we have

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

and hence

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

Now for $x \in S$ the right hand side of 15.3 is clearly integer and so $\xi = \sum f_j x_j - f$ is integer for $x \in S$. Since $x \geq 0$ for $x \in S$ we also have $\xi \geq -f > -1$ and since ξ is integer we deduce that $\xi \geq 0$ and that

$$\sum_{j=1}^{n} f_j x_j \geq f \qquad \text{for } x \in S$$

Suppose now that one has solved the continuous problem in (1) of our cutting plane algorithm and the solution is not integer. Therefore there is a basic variable x_i with

$$x_i + \sum_{j \notin I} b_{ij} x_j = b_{io}$$

where b_{io} is not integer.

Putting $f_i = b_{ij} - \lfloor b_{ij} \rfloor$ and $f = b_{io} - \lfloor b_{io} \rfloor$ and we deduce that

$$\sum_{j \notin I} f_j x_j \geq f \tag{15.4}$$

for all integer solutions to our problem.

Now $f > 0$ since b_{io} is not integer and so (15.4) is not satisfied by the current L.P. solution since $x_j = 0$ for $j \notin I$ and so (15.4) is a cut.

Statement of the Algorithm

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

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 15.4 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 is used to re-optimise.

Example.

(15.5) Maximize $x_1 + 4x_2$

Subject to

$$2x_1 + 4x_2 \leq 7$$
$$10x_1 + 3x_2 \leq 14$$
$$x_1, x_2 \geq 0$$

B.V.	x_1	x_2	x_3	x_4	y_1	y_2	R.H.S.
x_0	-1	-4					0
x_3	2	[4]	1				7
x_4	10	3		1			14
~							
x_0	1		1				7
x_2	$\frac{1}{2}$	1	$\frac{1}{4}$				$1\frac{3}{4}$ *
x_4	$\frac{17}{2}$		$-\frac{3}{4}$	1			$3\frac{3}{4}$
y_1	[$-\frac{1}{2}$]		$-\frac{1}{4}$		1		$-3\frac{1}{4}$
x_0			$\frac{1}{2}$		2		$11/2$
x_2		1			1		1
x_4			[-5]	1	17		-4
x_1	1		$\frac{1}{2}$		-2		$3/2$
~							
x_0				$\frac{1}{10}$	$3\frac{7}{10}$		$5\frac{1}{10}$ *
x_2		1			1		1
x_3			1	$-\frac{1}{5}$	$-1\frac{7}{5}$		$4/5$
x_1	1			$\frac{1}{10}$	$-\frac{3}{10}$		$1\frac{1}{10}$
y_2				[$-\frac{1}{10}$]	$-\frac{7}{10}$	1	$-\frac{1}{10}$
x_0					3		5
x_2		1			1		1
x_3			1		-1	-2	1
x_1	1				-1	1	
x_4				1	7	-10	

	x_1	x_2	x_3	x_4	x_5	x_6	R
0	-1	-4					0
	2	[4]	1				7
4	10	3		1			14

↓ 'continuous relaxation' solved by one pivot, solution non-integer, add a cut.

x_0	1		1				7
x_2	$\frac{1}{2}$	1	$\frac{1}{4}$				$\frac{7}{4}$ *
x_4	$\frac{17}{2}$		$-\frac{3}{4}$	1			$\frac{35}{4}$
x_5 (6)	$\frac{1}{2}$		[$\frac{1}{4}$] (b)		-1		$\frac{3}{4}$ K

↑ current solution infeasible, solve a phase 1 problem.

x_0	-1				4		4
x_2		1			1		1
x_4	[10]				-3		11
x_3	2		1		-4		3

↓ 2nd 'continuous' problem solved, but non-integer, add a cut.

x_0				$\frac{1}{10}$	$\frac{37}{10}$		$\frac{5}{10}$ *
x_2		1			1		1
x_1	1			$\frac{1}{10}$	$-\frac{3}{10}$		$\frac{11}{10}$
x_3			1	$-\frac{1}{5}$	$-\frac{17}{5}$		$\frac{4}{5}$
x_5 (c)				[$\frac{1}{10}$]	$\frac{7}{10}$	-1	$\frac{1}{10}$

↑ current solution infeasible, solve a phase 1 problem.

x_0					3		5
x_2		1			1		1
x_1	1				-1	1	1
x_3			1		-1	-2	1
x_4					7	-10	1

(a) Cut generated is $\frac{1}{2}x_1 + \frac{1}{4}x_2 \geq \frac{3}{4}$ or $\frac{1}{2}x_1 + \frac{1}{4}x_2 - x_5 = \frac{3}{4}$

(All artificial ξ to this, in order to get a basic feasible solution to an augmented t of equations, gives $\frac{1}{2}x_1 + \frac{1}{4}x_3 - x_5 + \xi = \frac{3}{4}$.

(b) Choose a pivot that will reduce ξ (pivoting in col. 1 is allowable, but the arithmetic is worse).

(c) Cut generated is $\frac{1}{10}x_4 + \frac{7}{10}x_6 - x_5 = \frac{1}{10}$

One can show that the Gomory cuts $\sum f_j x_j > f$ when expressed in terms of the original non-basic variables have the form $\sum w_j x_j \leq W$ where the w_j, W are integer and the value of $\sum w_j x_j$ after solving the current continuous problem is $W + \varepsilon$ where $0 < \varepsilon < 1$ assuming the current solution non-integer. Thus the cut is obtained by moving a hyperplane parallel to itself to an extend 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.

Further Remarks

1) After adding a cut and carrying out one iteration of the dual simplexalgorithm the slack variable corresponding to this cut becomes nonbasic. 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.

2) 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 value. 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.

3) 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.

It is useful to see what has happened graphically. We first express the cuts in terms of x_1, x_2.

....

Cut no.1

$$\tfrac{1}{2}\ x_1 + \tfrac{1}{4}x_3 \geq \tfrac{3}{4}$$

since

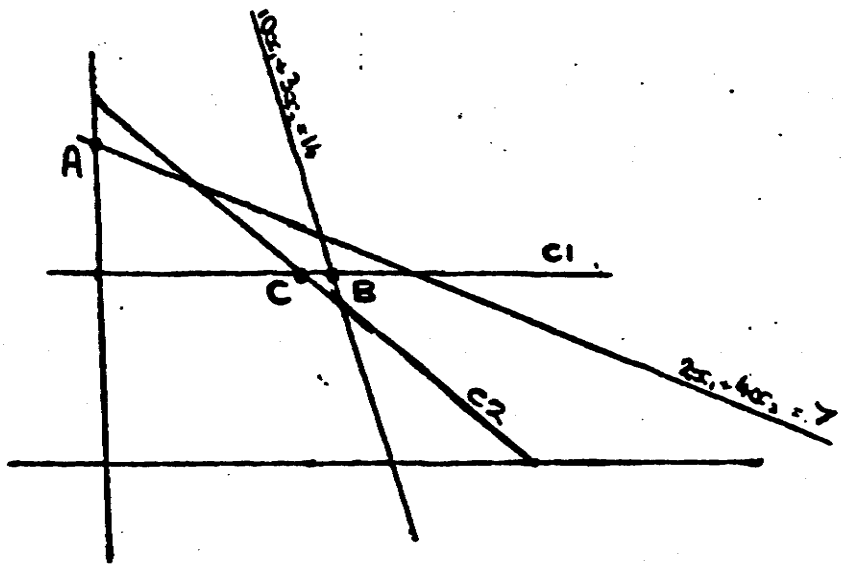
$x_3 = 7 - 2x_1 - 4x_2$ this becomes

$x_2 \leq 1$.

Cut no.2

After re-arranging, this becomes

$$x_1 + x_2 \leq 2$$



A is optimal solution ignoring integrality

B is optimal solution after adding cut C1

C is optimal integer solution found after adding C2.

Branch and Bound Method

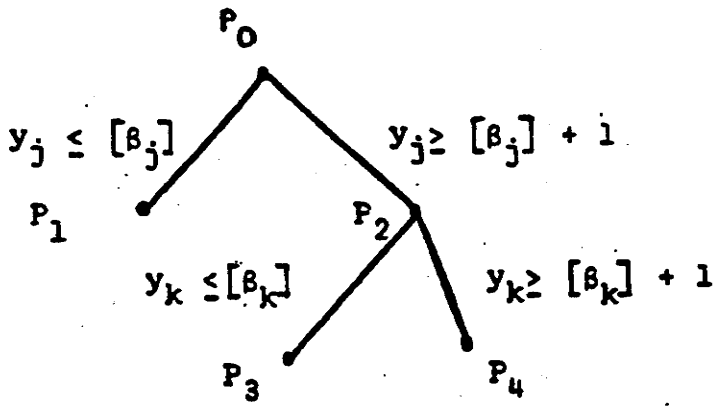
The method to be described in this section constitutes the most successful method applied to date. The idea is quite general and has been applied to many other discrete optimisation problems, e.g. travelling salesman, job shop scheduling.

Let us assume we are trying to solve the mixed integer problem 12.2. Let us call this problem P_0. The first step is to solve the 'continuous' L.P. problem obtained by ignoring the integrality constraints. If in the optimal solution, one or more of the integer variables turn out to be non-integer, we choose one such variable and use it to split the given problem P_0 into two 'sub-problems' P_1 and P_2. Suppose the variable chosen is y_j and it takes the non-integral value β_j in the continuous optimum. Then P_1 and P_2 are defined as follows:

$$P_1 \equiv P_0 \text{ with the added constraint } y_j \leq [\beta_j]$$

$$P_2 \equiv P_0 \text{ with the added constraint } y_j \geq [\beta_j] + 1$$

Now any solution to P_0 is either a solution of P_1 or P_2 and so P_0 can be solved by solving P_1 and P_2. We continue by solving the L.P. problems associated with P_1 and P_2. We then choose one of the problems and if necessary split it into two sub-problems as was done with P_0.



This process can be viewed as the construction of a binary tree of sub-problems whose terminal (pendant) nodes correspond to the problems that remain to be solved.

In an actual computation one keeps a list of the unsolved problems into which the main problem has been split. One also keeps a note of the objective value MIN of the best integer solution found so far.

Step 0

Initially the list consists of the initial problem P_0. Put MIN equal to either the value of some known integer solution, or if one is not given equal to some upper bound calculable from initial data, if neither possibility is possible put MIN = ∞.

Solve the L.P. problem associated with P_0. If the solution has integral values for all integer variables terminate, otherwise

Step 1

Remove a problem P from the list whose optimal continuous objective function value x_0 is less than MIN. If there are no such problems terminate. The best integer solution found so far is optimal. If none have been found the problem is infeasible.

Step 2

Amongst the integer variables in problem P with non-integer values in the optimal continuous solution for P select one for branching. Let this variable be y_p and let its value in the continuous solution be β.

Step 3

Create two new problems P' and P" by adding the extra restrictions $y_p \leq [\beta]$ and $y_p \geq [\beta] + 1$ respectively. Solve the L.P. problems associated with P' and P" and add these problems to the list. If a new and improved integer solution is found store it and update MIN. The new L.P. problems do not have to be solved from scratch but can be re-optimised using the dual algorithm (or parametrically altering the bound on y_p). If during the re-optimisation of either L.P. problem the value of the objective function exceeds MIN this problem may be abandoned. Go to step 1.

If one assumes that each integer variable in P_0 has a finite upper bound (equal to some large number for notionally unbounded variable) then the algorithm must terminate eventually, because as one proceeds further down the tree of problems the bounds on the variables become tighter and tighter, and these would eventually become exact if the L.P. solutions were never integer.

As an example we show a possible tree (Fig 1) for solving

Minimise $20 - 3x_1 - 4x_2$

Subject to

$$\frac{2}{5} x_1 + x_2 \leq 3$$

$$\frac{2}{5} x_1 - \frac{2}{5} x_2 \leq 1$$

$x_1, x_2 \geq 0$ and x_1, x_2 integer

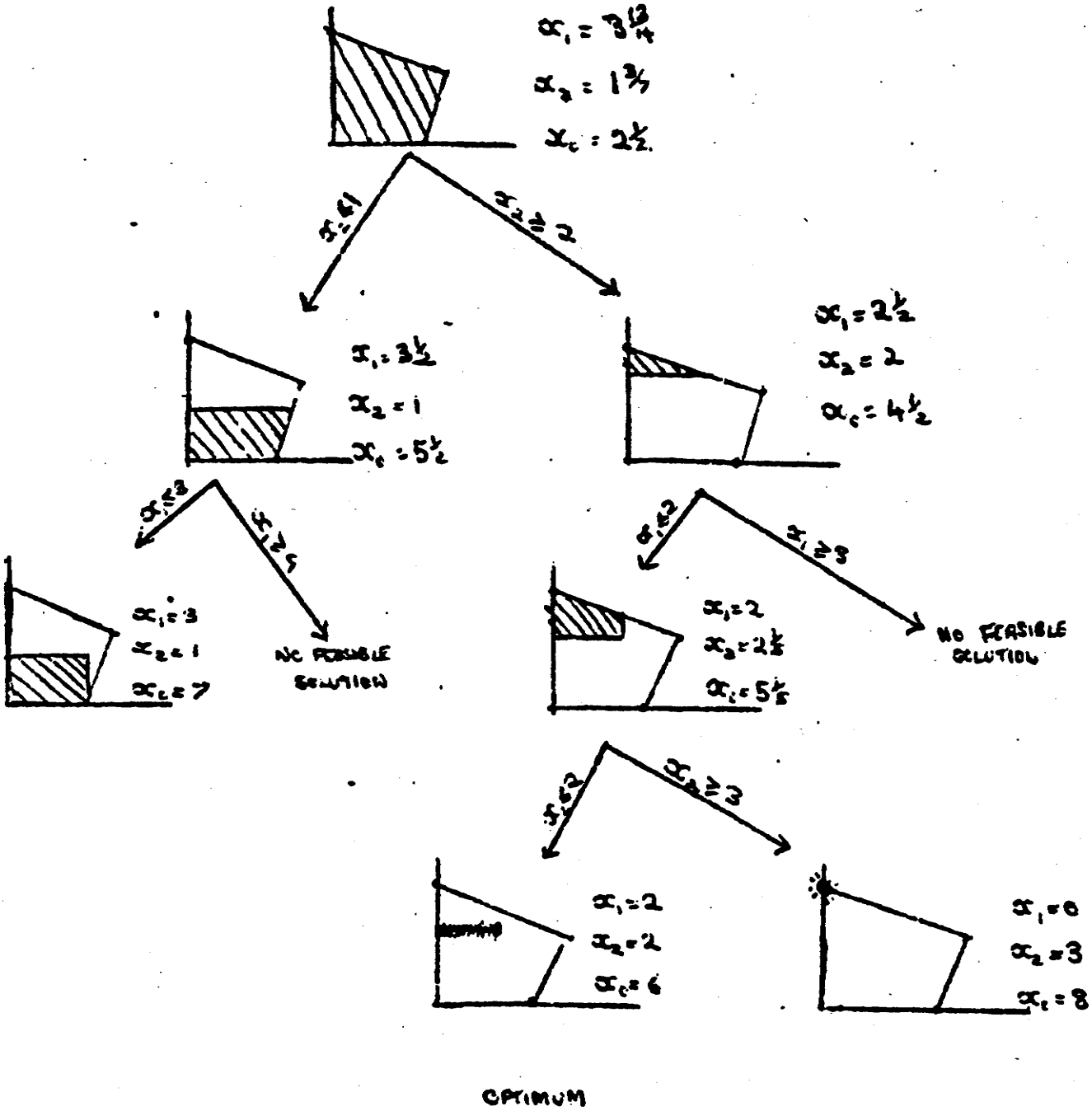

$x_1 = 3\frac{13}{14}$
$x_2 = 1\frac{3}{7}$
$x_c = 2\frac{1}{2}$
$x_2 \leq 1$
$x_2 \geq 2$
$x_1 = 3\frac{1}{2}$
$x_2 = 1$
$x_c = 5\frac{1}{2}$
$x_1 = 2\frac{1}{2}$
$x_2 = 2$
$x_c = 4\frac{1}{2}$
$x_1 \leq 3$
$x_1 \geq 4$
$x_1 = 3$
$x_2 = 1$
$x_c = 7$
NO FEASIBLE SOLUTION
$x_2 \leq 2$
$x_1 \geq 3$
$x_1 = 2$
$x_2 = 2\frac{1}{5}$
$x_c = 5\frac{1}{5}$
NO FEASIBLE SOLUTION
$x_2 \leq 2$
$x_2 \geq 3$
$x_1 = 2$
$x_2 = 2$
$x_c = 6$
$x_1 = 0$
$x_2 = 3$
$x_c = 8$
OPTIMUM

Fig 4

Reformulation of integer programming problems

Consider the problem

(15.6) maximise $x_1 + 4x_2$

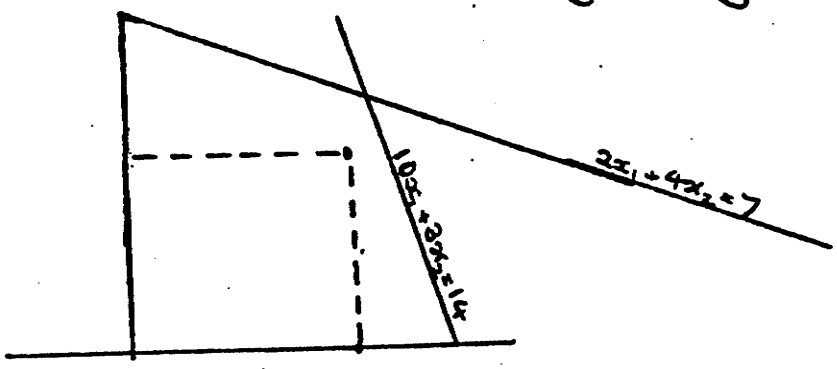
subject to $x_1 \leq 1$

$x_2 \leq 1$

$x_1, x_2 \geq 0$ and integer

It's solution $x_1 = 1, x_2 = 1$ is the same as that of problem (15.5). Indeed for any objective function these 2 problems will have the same solution because they have the same set of integer solutions. – (0,0), (1,0), (0,1), (1,1)

However problem (15.6) is much easier to solve – the continuous solution to (15.6) is always integer.



This illustrates an important point: for a given problem let us denote by IFR the set of integer solutions from which we are searching for the optimum. There are an infinite number of LP problems whose integer solution sets are <u>precisely</u> IFR. For such an LP let CFR(LP) $\supseteq$ IFR denote the set of continuous solutions

to the LP.

There will be a unique problem LP^* such that all vertices (basic feasible solutions) of $CFR(LP^*)$ are members of IFR. Thus if we could always identify the corresponding LP^* we could solve it using the simplex algorithm and we would know that its continuous solution would also be the integer solution.

We cannot in general easily identify LP^*. But suppose we have a formulation LP_1 for which the available algorithms are not proving satisfactory, a 'tighter' reformulation to LP_2 where $CFR(LP_1) \supset CFR(LP_2) \supseteq IFR$ can sometimes dramatically improve things.

If you are lucky a reformulation can simply involve adding some extra easily identified constraints satisfied by points in IFR but not by all points in $CFR(LP_1)$.

Implicit Enumeration

Simple B&B algorithm for pure 0-1 problems:

Ex: minimize $z = -7x_1 + 3x_2 + 2x_3 - x_4 + 2x_5$

s.t. $4x_1 + 2x_2 - x_3 + 2x_4 + x_5 \geq 3$

$4x_1 + 2x_2 + 4x_3 - x_4 - 2x_5 \geq 7$

$x_j = 0$ or 1

Top Node: (i) Lower Bound: -3 $x_4 = x_5 = 1$

— NOT FEASIBLE THOUGH $x_1 = x_2 = x_3 = 0$

(ii) Feasibility: $x_1 = x_2 = x_4 = x_5 = 1$ makes (i) feasible

$x_1 = x_2 = x_3 = 1$ makes (ii) feasible.

Branch $x_1 = 1$

or

$x_1 = 0$

$x_1 = 1$: minimize $7 + 3x_2 + 2x_3 - x_4 - 2x_5$

s.t.

$$2x_2 - x_3 + 2x_4 + x_5 \geq -1$$

$$2x_2 + 4x_3 - x_4 - 2x_5 \geq 3$$

LB : 4 NOT FEASIBLE

$x_2 = x_4 = x_5 = 1$ (i) ✓

$x_2 = x_3 = 1$ (ii) ✓

§17. 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 an amount a_{ij} – the payoff – given in the matrix below. (If $a_{ij} < 0$ this is equivalent to A paying B $-a_{ij}$).

$$\begin{array}{c} \\ A \end{array} \begin{array}{c} \begin{array}{cccc} B & & & \end{array} \\ \begin{bmatrix} 2 & 4 & 2 & 1 \\ -2 & 5 & 1 & -1 \\ 1 & -5 & 3 & 0 \\ 6 & 2 & -3 & -2 \end{bmatrix} \end{array}$$

This is a two person zero sum game, zero sum because the algebraic sum of the players' winnings is always zero.

Game 2 (Penalty Kicks)

Suppose 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 Middle of the goal. A can Dive to his Right or Dive to his left or Stay where he is. If A correctly guesses where B will kick the ball he will make a save.

The payoff to A is given by the following matrix:

A \ B	R	L	M
DR	2	-1	-2
DL	-1	2	-2
S	-1	-1	1

We shall be considering m×n generalisations of game 1 and other games like game 2 which can be reduced to this form.

Thus there is given some m×n payoff matrix $\|a_{ij}\|$. In a play of the game A chooses $i \in M = \{1,2,\cdots m\}$ and B chooses $j \in N = \{1,2,\cdots n\}$. These choices are made independently without either player knowing what the other has chosen. They then announce their choices and B pays a_{ij} 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 maximise his average winnings from the match and B's objective is to minimise his average losses.

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 $S_A = \{(1),\cdots (m)\}$ and $S_B = \{(1),\cdots (n)\}$ 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. S_A and S_B will always be such that 'average payment' is meaningful. Thus if $u=(i)$ and $v=(j)$ then $PAY(u,v) = a_{ij}$.

Stable Solutions

$(u_0, v_0) \in S_A \times S_B$ is a stable solution if

(17.1) $PAY(u, v_0) \leq PAY(u_0, v_0) \leq PAY(u_0, v)$.

If (17.1) holds then neither A nor B has any incentive to change strategy if each assumes his opponent is not going to change his.

The subsequent analysis is concerned with finding a stable solution.

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

$$ROWMIN(u) = \min_{v \in S_B} PAY(u,v) \quad \text{for } u \in S_A$$

and

$$COLMAX(v) = \max_{u \in S_A} PAY(u,v) \quad \text{for } v \in S_B$$

Suppose now that A chooses $\hat{u}$. We assume that after some finite time B will be able to deduce this. B will then choose his strategy v to minise $PAY(\hat{u}, v)$. Thus if A chooses u then he can expect his average winnings to be $ROWMIN(u)$.

Similarly if B chooses v he can expect his

average losses to be COLMAX(v).

Thus if $P_A = \text{ROWMIN}(u_0) = \max_{u \in S_A} \text{ROWMIN}(u)$ and

$P_B = \text{COLMAX}(v_0) = \min_{v \in S_B} \text{COLMAX}(v)$ then A can by choosing u_0 ensure his winnings average P_A and B by choosing v_0 can ensure his losses average P_B. If $P_A = P_B$ this seems to 'solve' the game but is $P_A = P_B$ always?

Theorem 17.1

(a) $P_A \leq P_B$

(b) $S_A \times S_B$ contains a stable solution if and only if $P_A = P_B$

Proof

(a)

(17.2) $$P_A = \text{ROWMIN}(u_0) \leq \text{PAY}(u_0, v_0) \leq \text{COLMAX}(v_0) = P_B$$

(b)

Suppose first that $(\hat{u}, \hat{v})$ is stable. Then from (17.1) we have

$$\text{COLMAX}(\hat{v}) = \text{PAY}(\hat{u}, \hat{v}) = \text{ROWMIN}(\hat{u})$$

and hence

$$P_B \leq \text{COLMAX}(\hat{v}) = \text{ROWMIN}(\hat{u}) \leq P_A$$

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

Conversely if $P_A = P_B$ then from (17.2) we deduce that $\text{ROWMIN}(u_0) = \text{PAY}(u_0, v_0) = \text{COLMAX}(v_0)$ which implies (17.1).

Q.E.D.

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

For game 1 we have $P_A = P_B = 1 = a_{14}$ and hence A plays (1) and B plays (4) solves the game: 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 <u>saddle point</u> (i_0, j_0) which means (i_0), (j_0) satisfies (17.1).

For a game whose matrix does not have saddle point things are more complex. Consider for example game 2. $P_A = -1$ and $P_B = 1$. It follows from theorem 17.1 that no pair of pure strategies solves the game. A knows he 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 B plays (1) A will play (1) and so on.

<u>Mixed Strategies</u>

To break this seeming deadlock we allow the players to choose <u>mixed strategies</u>. A mixed strategy for A is a vector of probabilities $(p_1, \cdots p_m)$ 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 $(q_1, \cdots q_n)$ satisfying $q_j \geq 0$ for $j \in N$ 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

(17.3) $$S_A = \{ \underline{p} \in \mathbb{R}^m : \underline{p} \geq 0 \text{ and } p_1 + \cdots + p_m = 1 \}$$
$$S_B = \{ \underline{q} \in \mathbb{R}^n : \underline{q} \geq 0 \text{ and } q_1 + \cdots + q_n = 1 \}$$

For $\underline{p} \in S_A$, $\underline{q} \in S_B$ it is straightforward to show that

$$PAY(\underline{p}, \underline{q}) = \sum_{i \in M} \sum_{j \in N} a_{ij} p_i q_j$$

We now show using the duality theory of linear programming that $S_A \times S_B$ as defined in (17.3) contains a stable solution.

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

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

Lemma 17.2

(17.5) $$\min_{\underline{q} \in S_B} \left(\sum_{j=1}^{n} \xi_j q_j \right) = \min(\xi_1 \cdots \xi_n)$$

Proof

Let $\xi_p = \min(\xi_1 \cdots \xi_n)$ and let L = LHS of (17.5). Putting $\hat{q}_j = 0$ for $j \neq t$ and $\hat{q}_t = 1$ we have $\hat{\underline{q}} \in Q$ and $\sum_{j=1}^{m} \xi_j \hat{q}_j = \xi_t$. Thus $L \leq \xi_t$. However for any $\underline{q} \in Q$ we have

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

Q.E.D.

It follows from the lemma and (17.4) that

$$P_A = \max_{\underline{p} \in S_A} \left(\min \left(c_1(\underline{p}), \cdots c_n(\underline{p}) \right) \right)$$

$$= \max \quad \min \left(c_1(\underline{p}), \cdots c_n(\underline{p}) \right)$$

subject to

$$p_1 + \cdots + p_m = 1$$

$$p_1, \cdots p_m \geq 0$$

(17.6)
$$= \max \quad \xi$$

$$\xi \leq \sum_{i=1}^{m} a_{ij} p_i \qquad j = 1, \cdots n$$

$$p_1 + \cdots + p_m = 1$$

$$p_1, \cdots p_m \geq 0$$

where LP (17.6) is derived as in §A of the ORJ notes. Using similar arguments we can show that

(17.7)
$$P_B = \min \quad \eta$$

$$\eta \geq \sum_{j=1}^{n} a_{ij} q_j \qquad i = 1, \cdots m$$

$$q_1 + \cdots + q_n = 1$$

$$q_1, \cdots q_n \geq 0$$

We note next that (17.6) and (17.7) are a pair of dual linear programs. They are both feasible and hence $P_A = P_B$ and stable solutions exist. In fact if $\underline{p}^0$ solves (17.6) and $\underline{q}^0$ solves (17.7) then $(\underline{p}^0, \underline{q}^0)$ is stable as $PAY(\underline{p}^0, \underline{q}^0) = P_A = P_B$.

We describe next a simple transformation which reduces the LP problems by one row and one variable.

Consider first (17.6). We assume that $a_{ij} > 0$. If not we can first add a quantity λ to each a_{ij} so that $a_{ij}+\lambda > 0$. One can see that for any u, v $PAY(u,v)$ is also increased by λ and so this has no effect on the strategies.

We can now assume that $\xi > 0$ in the optimum solution to (17.6). We can the re-write (17.6) as

$$\text{minimise} \quad \frac{1}{\xi} = \left(\frac{p_1}{\xi}\right) + \cdots + \left(\frac{p_m}{\xi}\right)$$

$$\text{subject to} \quad \sum_{i=1}^{m} a_{ij}\left(\frac{p_i}{\xi}\right) \geq 1 \qquad j = 1, \cdots n$$

(17.8)
$$\left(\frac{p_1}{\xi}\right) + \cdots + \left(\frac{p_m}{\xi}\right) = \frac{1}{\xi}$$

$$\left(\frac{p_1}{\xi}\right), \cdots \left(\frac{p_m}{\xi}\right) \geq 0$$

Letting $x_i = p_i / \xi$ for $i = 1, \cdots m$ this becomes

(17.9)
$$\text{minimise} \quad x_1 + \cdots + x_m$$

$$\text{subject to} \quad \sum_{i=1}^{m} a_{ij} x_i \geq 1 \qquad j = 1, \cdots m$$

$$x_1, \cdots x_m \geq 0$$

where equation (17.8) is redundant.

The optimal strategy $\underline{p}^0$ can be recovered from an optimal solution $\underline{x}^0$ to (17.9) as follows:

let $\xi = 1/(x_1^0 + \cdots + x_m^0)$ and then $p_i^0 = x_i^0 \xi$ for $i = 1 \cdots m$.

The equivalent transformation for (17.7) gives

(17.10) maximise $y_1 + \cdots + y_n$

subject to $$\sum_{j=1}^{n} a_{ij} y_j \leq 1 \qquad i = 1 \cdots m$$

$$y_1, \cdots y_n \geq 0$$

which dual to (17.9)

We now use the above theory to solve game 2.

We first add 3 to each element of the matrix and then solve either (17.9) or (17.10)

Problem (17.10) is

maximise $y_1 + y_2 + y_3$

subject to $5y_1 + 2y_2 + y_3 \leq 1$

$2y_1 + 5y_2 + y_3 \leq 1$

$2y_1 + 2y_2 + 4y_3 \leq 1$

$y_1, y_2, y_3 \geq 0$

The optimal solution can be found to be $y_1 = \frac{1}{8}$ $y_2 = \frac{1}{8}$ $y_3 = \frac{1}{8}$ and $y_1 + y_2 + y_3 = \frac{3}{8}$ implying that $\eta = \frac{8}{3}$ and $q_1 = q_2 = q_3 = \frac{1}{3}$. The solution to (17.9) is $x_1 = \frac{1}{12}$ $x_2 = \frac{1}{12}$ $x_3 = \frac{5}{24}$ implying that $\xi = \frac{8}{3}$ and $p_1 = \frac{1}{7}$ $p_2 = \frac{1}{7}$ and $p_3 = \frac{5}{7}$. Since we added 3 to each element initially we see that the actual value of the game to A is $\frac{8}{3} - 3 = -\frac{1}{3}$.

Dominated Strategies

In the matrix game below

$$\begin{bmatrix} 1 & 3 & -3 & -2 \\ 3 & 2 & 3 & 3 \\ 2 & 3 & 2 & -1 \end{bmatrix}$$

we see that strategy(3) is better for A than strategy(1) for any choice of strategy by B and consequently strategy(1) can be ignored by A and the game reduced to

$$\begin{bmatrix} 3 & 2 & 3 & 3 \\ 2 & 3 & 2 & -1 \end{bmatrix}$$

We see now that strategy(4) is better than either of strategies (1),(3) for B for either of A's strategies. Thus columns 1, 3 can be deleted to reduce the game to

$$\begin{bmatrix} 2 & 3 \\ 3 & -1 \end{bmatrix}$$

We have used the idea of dominated strategies to reduce the size of the game to be considered. Thus

Strategy(i) <u>dominates</u> strategy 'i') for the row player A if

$a_{ij} \geq a_{i'j}$ for all j.

Strategy(j) dominates strategy(j') for the column player B' if

$a_{ij} \leq a_{ij'}$ for all i.

Thus strategies(i')(j') above can be ignored. Successive applications of these rules can reduce the size of a game significantly.

Random Payoffs

We finally note that the above analysis goes through unchanged if A,B having selected tactics i,j the payoff to A is a random variable whose expected value is a_{ij}

Simple aspects of games

① Dominance

② Latin Square Games: $\underline{p} = \frac{1}{m}\underline{e}$ $\quad \underline{q} = \frac{1}{m}\underline{e}$

Magic Square Games: $\quad \downarrow \quad S = \text{row sum}$

$$\max \sum x_i$$
$$Ax \le \underline{1}$$
$$x_i \ge 0$$

$$\underline{x} = \frac{1}{S}\underline{e}$$

③ Non-singular games: A is non-singular $\quad \underline{1}^T A^{-1} \underline{1} \neq 0$

$V = \dfrac{1}{\underline{1}^T A^{-1} \underline{1}}$ solution: $\underline{p}^T = V \underline{1}^T A^{-1}$, $\underline{q} = V A^{-1} \underline{1}$ provided

$p \ge 0$, $q \ge 0$

④ Symmetric Games: $A^T = -A$: optimal value is 0,

If A uses $\underline{p}$ and B uses $\underline{p}$

$$\underline{p}^T A \underline{p} = \underline{p}^T A \underline{p} = 0$$

so

$$\text{PAY}(p,q) \begin{bmatrix} 0 & & & & \\ & 0 & & & \\ & & 0 & & \\ & & & 0 & \\ & & & & 0 \end{bmatrix}$$

$p_A \not\equiv 0$, $p_B \not\equiv 0$ $\Rightarrow p_A = p_B = 0$

Appendix 2: Existence of Equilibria in Finite Games

We give a proof of Nash's Theorem based on the celebrated Fixed Point Theorem of L. E. J. Brouwer. Given a set C and a mapping T of C into itself, a point $z \in C$ is said to be a fixed point of T, if $T(z) = z$.

Brouwer's Fixed Point Theorem. *Let C be a nonempty, compact, convex set in a finite dimensional Euclidean space, and let T be a continuous map of C into itself. Then there exists a point $z \in C$ such that $T(z) = z$.*

The proof is not easy. You might look at the paper of K. Kuga (1974), "Brower's fixed point Theorem: An Alternate Proof", *SIAM Journal of Mathematical Analysis*, **5**, 893-897. Or you might also try Parthasarathy and Raghavan (1971), Chapter 1.

Now consider a finite n-person game with the notation of Section III.2.1. The pure strategy sets are denoted by $X_1, \ldots, X_n$, with X_k consisting of $m_k \geq 1$ elements, say $X_k = \{1, \ldots, m_k\}$. The space of mixed strategies of Player k is given by X_k^*,

$$X_k^* = \{\boldsymbol{p}_k = (p_{k,1}, \ldots, p_{k,m_k}) : p_{k,i} \geq 0 \text{ for } i = 1, \ldots, m_k, \text{ and } \textstyle\sum_{i=1}^{m_k} p_{k,i} = 1\}. \quad (1)$$

For a given joint pure strategy selection, $\boldsymbol{x} = (i_1, \ldots, i_n)$ with $i_j \in X_j$ for all j, the payoff, or utility, to Player k is denoted by $u_k((i_1, \ldots, i_n)$ for $k = 1, \ldots, n$. For a given joint mixed strategy selection, $(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n)$ with $\boldsymbol{p}_j \in X_j^*$ for $j = 1, \ldots, n$, the corresponding expected payoff to Player k is given by $g_k(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n)$,

$$g_k(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n) = \sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} p_{1,i_1} \cdots p_{n,i_n} u_k(i_1, \ldots, i_n). \quad (2)$$

Let us use the notation $g_k(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n | i)$ to denote the expected payoff to Player k if Player k changes strategy from $\boldsymbol{p}_k$ to the pure strategy $i \in X_k$,

$$g_k(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n | i) = g_k(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k-1}, \boldsymbol{\delta}_i, \boldsymbol{p}_{k+1}, \ldots, \boldsymbol{p}_n). \quad (3)$$

where $\boldsymbol{\delta}_i$ represents the probability distribution giving probability 1 to the point i. Note that $g_k(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n)$ can be reconstructed from the $g_k(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n | i)$ by

$$g_k(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n) = \sum_{i=1}^{m_k} p_{k,i} g_k(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n | i) \quad (4)$$

A vector of mixed strategies, $(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n)$, is a strategic equilibrium if for all $k = 1, \ldots, n$, and all $i \in X_k$,

$$g_k(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n | i) \leq g_k(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n). \quad (5)$$

Theorem. *Every finite n-person game in strategic form has at least one strategic equilibrium.*

Proof. For each k, X_k^* is a compact convex subset of m_k dimensional Euclidean space, and so the product, $C = X_1^* \times \cdots \times X_n^*$, is a compact convex subset of a Euclidean space of dimension $\sum_{i=1}^n m_i$. For $z = (p_1, \ldots, p_n) \in C$, define the mapping $T(z)$ of C into C by

$$T(z) = z' = (p_1', \ldots, p_n') \tag{6}$$

where

$$p_{k,i}' = \frac{p_{k,i} + \max(0, g_k(p_1, \ldots, p_n|i) - g_k(p_1, \ldots, p_n))}{1 + \sum_{j=1}^{m_k} \max(0, g_k(p_1, \ldots, p_n|j) - g_k(p_1, \ldots, p_n))}. \tag{7}$$

Note that $p_{k,i} \geq 0$, and the denominator is chosen so that $\sum_{i=1}^{m_k} p_{k,i}' = 1$. Thus $z' \in C$. Moreover the function $f(z)$ is continuous since each $g_k(p_1, \ldots, p_n)$ is continuous. Therefore, by the Brouwer Fixed Point Theorem, there is a point, $z' = (q_1, \ldots, q_n) \in C$ such that $T(z') = z'$. Thus from (7)

$$q_{k,i} = \frac{q_{k,i} + \max(0, g_k(z'|i) - g_k(z'))}{1 + \sum_{j=1}^{m_k} \max(0, g_k(z'|j) - g_k(z'))}. \tag{8}$$

for all $k = 1, \ldots, n$ and $i = 1, \ldots, m_n$. Since from (4) $g_k(z')$ is an average of the numbers $g_k(z'|i)$, we must have $g_k(z'|i) \leq g_k(z')$ for at least one i for which $q_{k,i} > 0$, so that $\max(0, g_k(z'|i) - g_k(z')) = 0$ for that i. But then (8) implies that $\sum_{j=1}^{m_k} \max(0, g_k(z'|j) - g_k(z')) = 0$, so that $g_k(z'|i) \leq g_k(z')$ for all k and i. From (5) this shows that $z' = (q_1, \ldots, q_n)$ is a strategic equilibrium. ■

Remark. From the definition of $T(z)$, we see that $z = (p_1, \ldots, p_n)$ is a strategic equilibrium if and only if z is a fixed point of T. In other words, the set of strategic equilibria is given by $\{z : T(z) = z\}$. If we could solve the equation $T(z) = z$ we could find the equilibria. Unfortunately, the equation is not easily solved. The method of iteration does not ordinarily work because T is not a contraction map.

Decision Analysis

Company GFB owns a plot of land.

Alternatives	Payoffs: Oil	Dry
Drill	700k	-100k
Sell Land	90k	90k
Probability	.25	.75

Decision Strategies:

(i) Choose alternative that maximises minimum payoff

$\max\{-100k, 90k\}$ i.e. Sell

(ii) Choose alternative that maximises payoff under most likely alternative. Again Sell

(iii) Choose alternative that maximises expected profit.

Drill : $.25 \times 700 - .75 \times 100 = 100^{*}$ Drill

Sell : $.25 \times 90 + .75 \times 90 = 90$

Now introduce third alternative:

Do a Seismic Study (SS) and then make decision. Cost of survey is 30k.

Outcome: FSS ≡ Favorable

USS ≡ Unfavorable

Data:

$P[FSS|Oil] = .6$ $\quad P[USS|Oil] = .4$

$P[FSS|Dry] = .2$ $\quad P[USS|Dry] = .8$

Bayes Computation:

We need

$$P[Oil \mid FSS] = \frac{P[Oil \wedge FSS]}{P[FSS]}$$

$$= \frac{P[FSS \mid Oil]\, P[Oil]}{P[FSS \mid Oil]\, P[Oil] + P[FSS \mid Dry]\, P[Dry]}$$

$$= \frac{.6 \times .25}{.6 \times .25 + .2 \times .75}$$

$$= \frac{1}{2}$$

Similar calculations give:

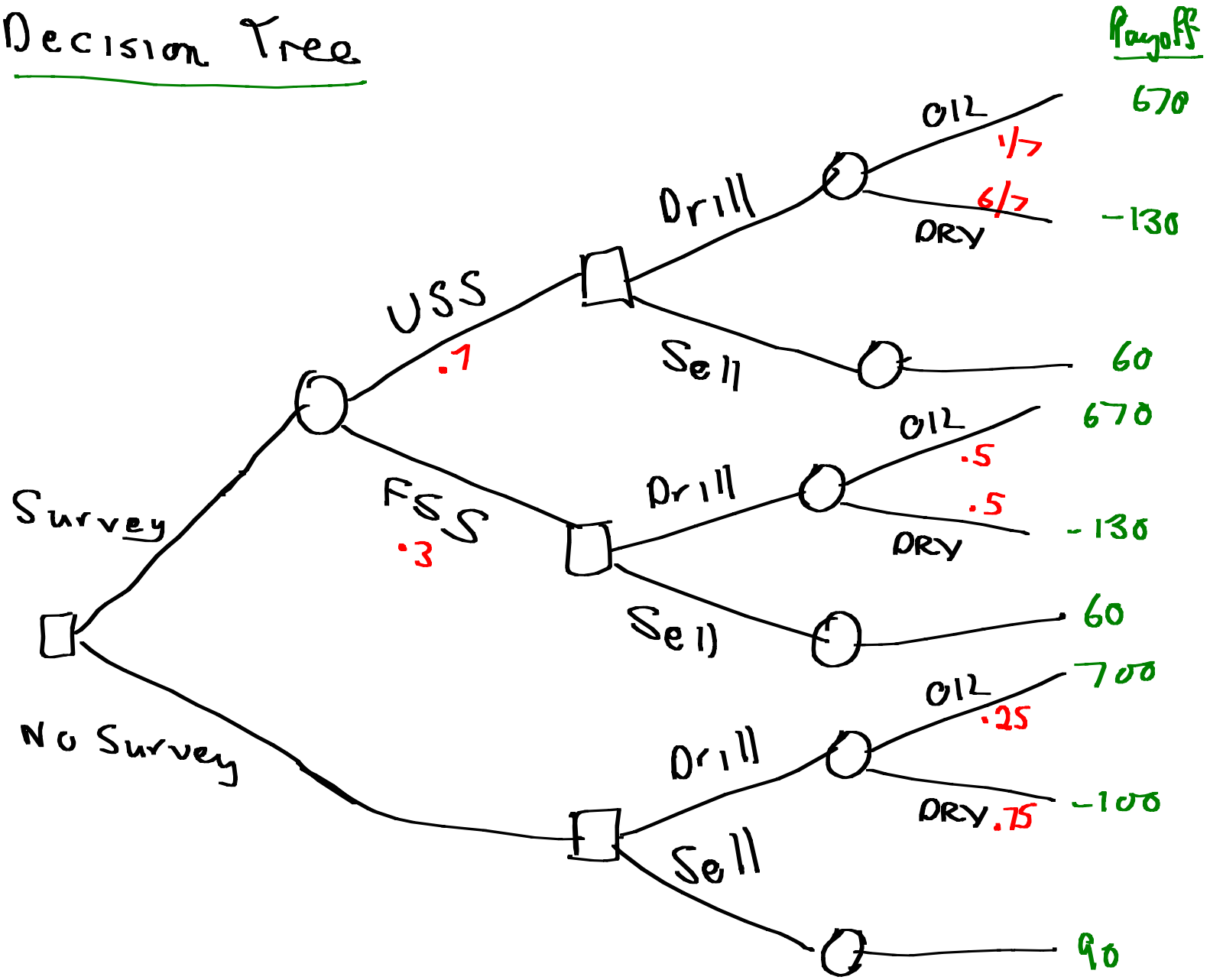
$$P[Oil \mid FSS] = \frac{1}{2}$$

$$P[Dry \mid FSS] = \frac{1}{2}$$

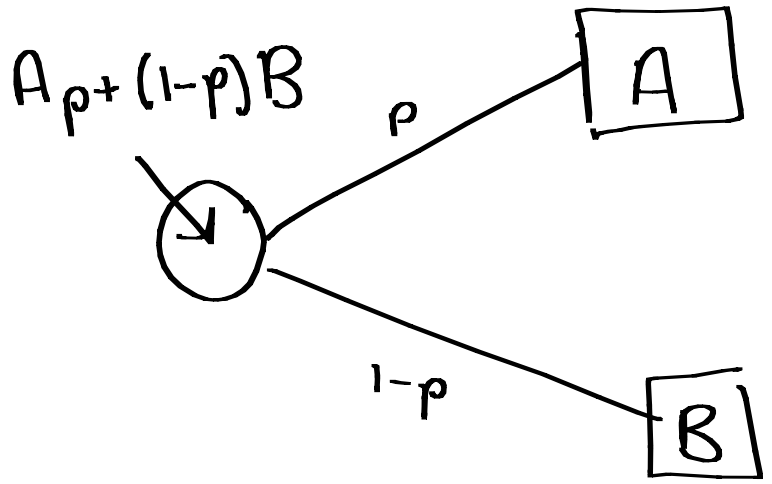
$$P[Oil \mid VSS] = \frac{1}{7}$$

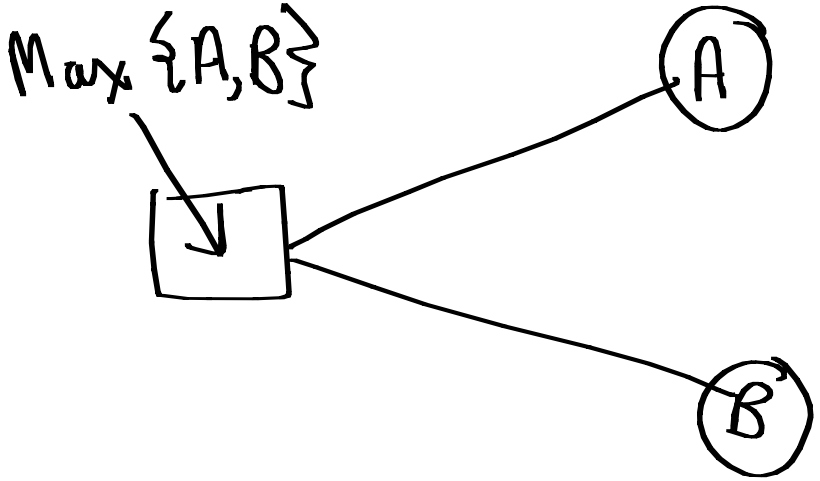
$$P[Dry \mid VSS] = \frac{6}{7}$$

Decision Tree

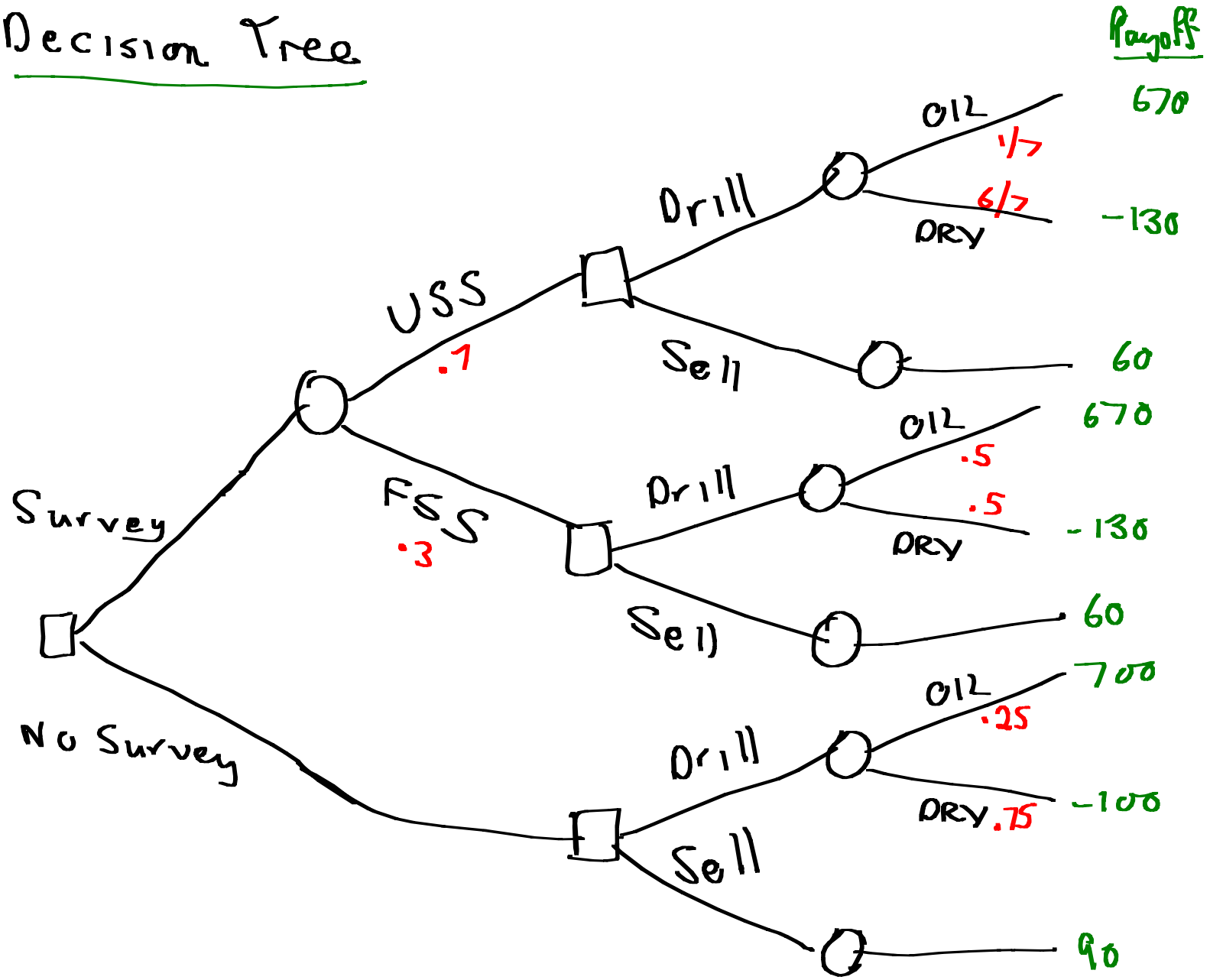


Evaluating Nodes of Tree





Decision Tree



Inventory Control

§1. - Introduction

We are mainly concerned here with the control of the inventory (stock) levels in an organization.

We shall develop models for a range of different situations and use a variety of techniques for their solution. The number of situations studied can only be a sample of possible problem structures and there is no all powerful "inventory management system" which can be generally applied.

Why keep Inventories?

On the face of it inventories represent idle resources and are wasteful. Money tied up in stocks could be invested elsewhere. They are, nevertheless, necessary to uncouple inputs and outputs, and usually arise from essentially "batch" inputs - either from buying in dozens to sell singly, or from naturally arising production processes. Some common example of inventories are:

1) Stocks of spare parts for cars, locomotives or any industrial machinery
2) Stocks of fuel at power stations, petrol stations
3) Stocks of semi-finished goods waiting for next stage in a production line
4) Stocks of blood in a blood bank
5) Money kept for withdrawals by banks, building societies etc.

Costs Involved

(a) Inventory Systems

(i) basic item cost, e.g. cost of purchase from supplier: often constant, but may be effected by quantity discount structures, and in dynamic situations, may change with time.

(ii) The cost of making an order excluding cost (i), e.g. cost of paper work involved: ranges from very small costs to substantial ones in some special cases.

(iii) Inventory holding costs: includes capital insurance, buildings etc. usually taken as proportional to the value of inventory held.

(iv) Stock out Costs, i.e. cost of being out of stock when an item is demanded: this is generally difficult to assess especially with regard to customer satisfaction. In some cases back orders can be 'backlogged' i.e. filled later on, while in others, non-fulfilled demands represent 'lost sales'.

Types of Inventory System

The two most common types of inventory system are:

(a) Lot Size - Re-order point systems

Orders for fixed quantities are placed when the stock level falls to a preset figure. This requires a constant monitoring of stock levels, every time an item is taken from stock.

(b) Cyclical Review Models

Stocks of a particular item are examined at fixed intervals of time and an amount is ordered depending on the stock level at that time. Often the rule is to order up to a predetermined level i.e. if stock level is s and one orders up to level L, then one orders L-s

Lead Time

There is usually a delay between making an order, and the time when the order actually arrives. This time is called the lead time. It may be constant, dependent on order size, or even a stochastic variable.

The questions we shall concern ourselves with are:

(1) When to order?

(2) How much to order?

§2 Deterministic Single Product Models

In this section we examine 3 models of an inventory system. They are necessarily very simple but illustrate some of the ideas involved.

Model 1

The demand for the product is constant and λ per period. There is a zero lead time and no stockouts 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 minimise the average cost per period of running the system.

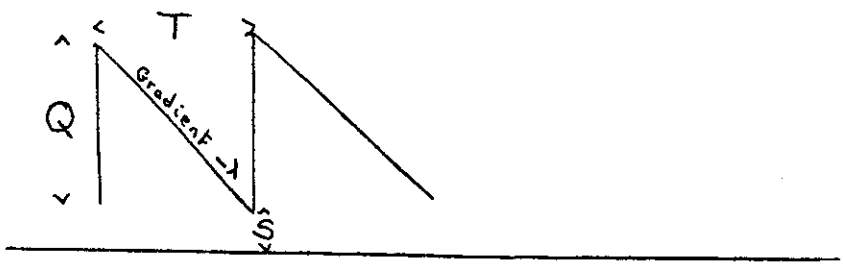
Notation

A = the fixed cost associated with making an order (a(ii))

C = unit cost per item.

I = inventory carrying charges. If one unit of stock is kept for t periods then the inventory charge is It units.

Let $S \geq 0$ 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 affect 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.

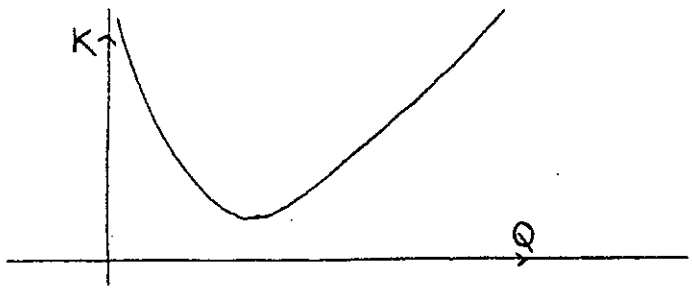
The average stock level is $\frac{1}{2}(S+(S+Q))=S+Q/2$.

$\therefore$ the average holding cost per period is

$$I(S+Q/2)$$

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

2.1 $$K = A\lambda/Q + I(Q/2+S)$$



K is to minimised for Q and $S \geq 0$. Now for given Q we minimise K by taking S = 0. To find the optimal Q we differentiate 2.1 (with S = 0).

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

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

2.2 $Q_w \simeq (2\lambda A/I)^{\frac{1}{2}}$

The right hand side of 2.2 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

$$T_w = (2A/\lambda I)^{\frac{1}{2}}$$

$$K_w = (2\lambda A I)^{\frac{1}{2}}$$

Discrete Case

Suppose that one can only order discrete quantities $q, 2q, \ldots$ and that Q_w is not an exact multiple of q. In this case we simply compare the costs for $Q = [Q_w/q]$ and $Q = [Q_w/q] + q$ and take the smaller.

Ex

$A = 10$, $\lambda = 100$, $I = .1$ and $q = 10$

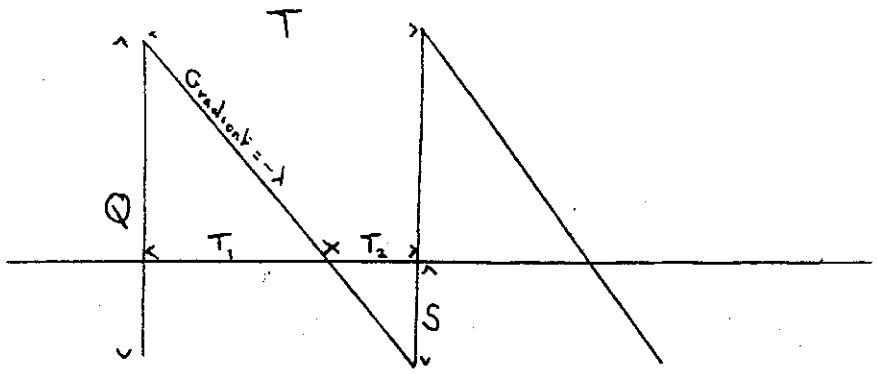
then $Q_w = 141.4$ and $K_w = 14.141$

Also $K(140) = 14.143$ and $K(150) = 14.167$

Therefore the optimal value of Q in the discrete case is $Q = 140$.

Model 2

This model is the same as for model 1 except that items out of stock can be back-ordered 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 inventory level will follow the pattern shown 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 minimised.

The following relations are apparent from the diagram

$$Q = \lambda T$$

$$S = \lambda T_2$$

$$Q-S = \lambda T_1$$

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 inventory level is $(Q-S)/2$ and so the average holding cost per period is

$$I(Q-S)T_1/2T$$

But from above we have that $T_1/T = (Q-S)/Q$ and so this cost is in fact

$$I(Q-S)^2/2Q$$

Back-Order Cost

The proportion of time that the system is out of stock is T_2/T. Because of the given form of back-order 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 back ordered is S/2. Therefore the average back-order cost is

$$\pi ST_2/2T$$
$$= \pi S^2/2Q \quad \text{from above}$$

The total variable annual cost K is then

$$K = \lambda A/Q + I(Q-S)^2/2Q + \pi S^2/2Q$$

In order to minimise K we solve the equations

$$\frac{\partial K}{\partial Q} = \frac{\partial K}{\partial S} = 0$$

yielding

2.3 $$-\lambda A/Q^2 - I(Q-S)^2/2Q^2 + I(Q-S) - \pi S^2/2Q^2 = 0$$

and

2.4 $$-I(Q-S)/Q + \pi S/Q = 0$$

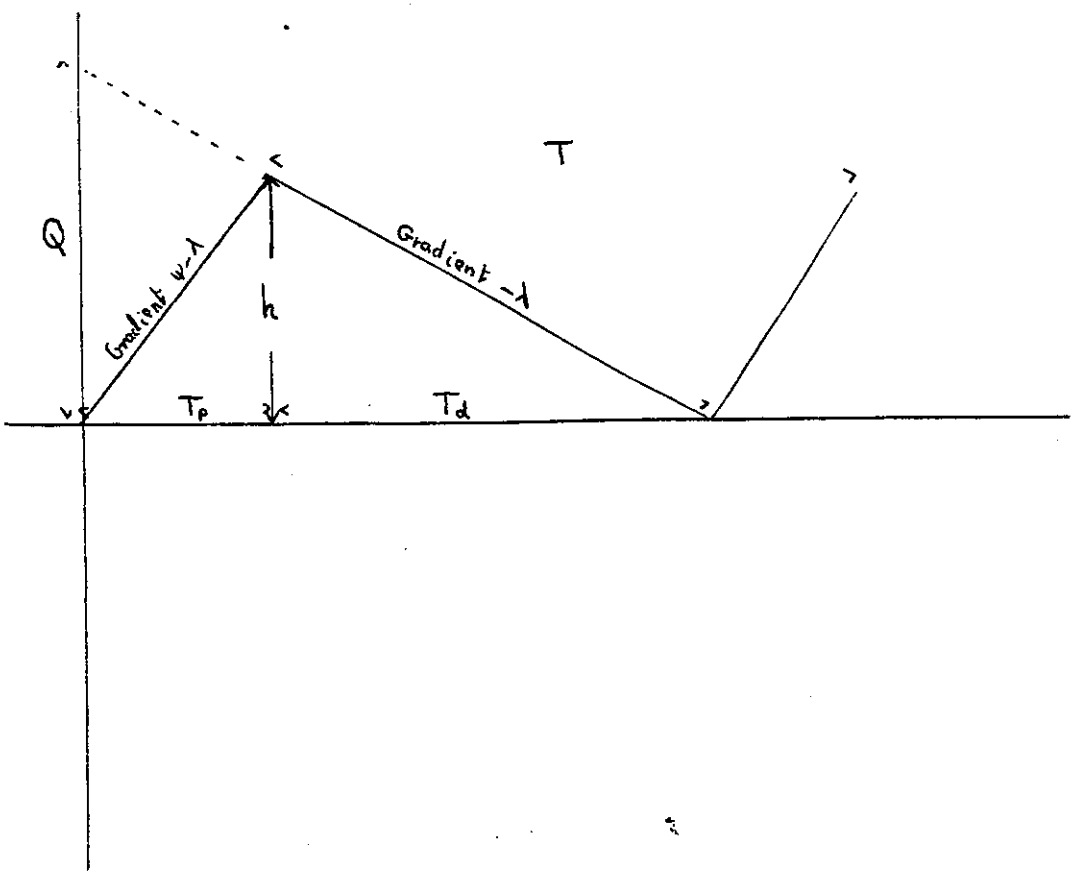
These can be solved by using 2.4 to express Q in terms of S, substitute in 2.3 to give an equation in S and then solving giving

$$S = (2\lambda AI/\pi(\pi+I))^{\frac{1}{2}}$$
$$Q = Q_W((\pi+I)/\pi)^{\frac{1}{2}}$$
$$K = K_W(\pi/(\pi+I))^{\frac{1}{2}}$$

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

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 inventory level assuming no back ordering will then have the pattern below:



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 time between orders is T and a total amount Q is produced at a time then we must have $Q = \lambda T$ otherwise stock levels will 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 is

$$A\lambda/Q$$

Holding Cost

If h is the maximum stock level during a cycle then the average stock is 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

$$\tfrac{1}{2}\, I(1-\lambda/\psi)\, Q$$

and the average total variable cost is

$$K = \lambda A/Q + \tfrac{1}{2}\, IQ(1-\lambda/\psi)$$

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

$$Q = Q_w(\psi/(\psi-\lambda))^{\frac{1}{2}}$$

with minimal cost

$$K = K_w(\psi-\lambda)/\psi)^{\frac{1}{2}}$$

Thus total cost has decreased as against model 1 due to decrease in inventory costs. Note that taking $\psi = \infty$ gives model 1 again.

Fixed Lead Time

If the lead time τ between making an order and receiving it is a fixed constraint then its effect on the above 3 models and indeed any model is limited. We need only alter our ordering policy so that if in a certain model with zero lead time we make orders at time t_1, t_2,... then in the corresponding model with fixed lead time τ we make orders at time $t_1-\tau$, $t_2-\tau$,... and then in other respects i.e. costs and inventory levels etc., the two models will be the same.

§3 Deterministic Multi-Item Problems

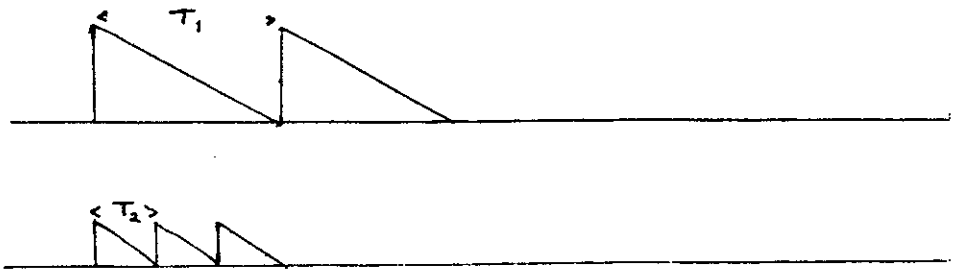
We consider here some problems when more than one type of item is being stored and when the inventory costs of the items interact. If there is no interaction we can consider each item separately.

Models 4 and 5 of this section are generalisations of model 1 although we could equally well have generalised model 2 or model 3. Model 6 however is specifically a generalisation of model 3.

Model 4

Assume that there are n distinct types of item whose 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.

Suppose then that there is a demand λ_j and holding cost I_j for item j. Let the optimal policy be to order item j at period T_j.



We show that $T_1 = T_2 = \ldots = T_n$. For let $T = T_k = \min_j(T_j)$ Then if $T_j > T$ for some j by increasing the frequency of ordering of j to that of k and ensuring that item j is ordered at the same time as k we

(i) eliminate any extra ordering costs for j over those of k

(ii) decrease the average stock level of j.

Thus $T_j > T$ implies non-optimality and the result follows. The problem then is to find the optimal value for T.

Ordering Cost

The average number of orders per period is 1/T 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

3.1 $$Q_j = \lambda_j T$$

Now the average inventory for item j is $Q_j/2$ and so the average inventory cost per period is $I_j Q_j/2$. Using 3.1 we see therefore that the average total inventory cost is

$$(\tfrac{1}{2} \sum_j \lambda_j I_j)T$$

and so the average total cost per period K is given by

$$K = A/T + (\tfrac{1}{2} \sum_j \lambda_j I_j)T$$

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

$$T = (2A/(\sum_j \lambda_j I_j))^{\frac{1}{2}}$$

The optimal lot sizes Q_j and minimal cost K can now be calculated.

Model 5

In the previous model the items were able to share a facility

(ordering) without increase of 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 e.g. 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 assume that there are n items whose individual characteristics are those of model 1 but that the lot-sizes Q_j have to satisfy the constraint

3.2 $$\Sigma f_j Q_j \leq f$$

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

3.3 $$K = \sum_j (\lambda_j A_j / Q_j + I_j Q_j / 2)$$

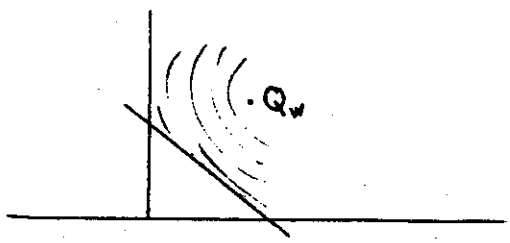
The problem is therefore to minimise K subject to 3.2 and we can proceed as follows:

(i) we first find the minimum of K ignoring constraint 3.2. Now to minimise K overall we can simply minimise each term in the summation 3.3 separately. This will naturally lead to the Wilson Lot-Sizes

$$Q_{jw} = (2\lambda A_j / I_j)^{\frac{1}{2}}$$

If these values satisfy the constraint 3.2 then we have solved the problem, otherwise

(2)



in these circumstances we may assume that constraint 3.2 is active at the optimum $(Q^*_1,\ldots Q^*_n)$ i.e.

$$\Sigma f_j Q^*_j = f$$

(this follows because we may express

$$K(Q) = K(Q^*) + \sum_j \frac{\partial K}{\partial Q_j}(Q_j - Q^*_j) + \ldots$$

and if $\Sigma f_j Q^*_j < f$ then $(Q_1 - \theta \frac{\partial K}{\partial Q_1}, \ldots Q_n - \theta \frac{\partial K}{\partial Q_n})$ is both feasible and better than Q^* for small enough $\theta > 0$ unless $\frac{\partial K}{\partial Q_j} = 0$ for $j = 1,..n$ but this would mean $Q_j = Q_{jw}$ which we have already discounted).

So we now tackle the problem

3.4 minimise K

subject to

3.5 $F = \Sigma f_j Q_j - f = 0$

(Strictly speaking $Q_j \geq 0$ is also a constraint but the solution to 3.4 always gives $Q_j > 0$ so we can ignore it in this case).

Problem 3.4 is in a form suitable for application of the classical Lagrange Multiplier Technique.

This would state in our case that there exists θ such that at the optimum for 3.4

(3.6) $$\frac{\partial K}{\partial Q_j} = \theta \frac{\partial F}{\partial Q_j} \text{ for } j = 1,\ldots n$$

(a non rigourous argument for this is that if 3.6 does not hold at the optimum Q^* then there exists q such that $q.\nabla F = 0$ and $q.\nabla K < 0$

at Q^* and so for small $\theta > 0, Q^* + \theta q$ is both feasible and better than Q^*).

On differentiation we see that 3.6 states

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

or

$$(3.7)\quad Q_j = (2\lambda A_j/(I_j - 2\theta f_j))^{\frac{1}{2}} \qquad \text{for } j = 1,\ldots n$$

Now the value for θ can be calculated by substituting 3.7 into $F = 0$ giving

$$(3.8)\quad \sum_j f_j (2\lambda_j A_j/(I_j - 2\theta f_j))^{\frac{1}{2}} = f$$

In general 3.8 may be solved using any numerical technique e.g. Newton Raphson or Bisection (Exercise: find a range for θ)

Example

A company manufactures 3 types of washing machine with characteristics as below:

Washing Machine	Price(£)	Sales/Week
1	80	100
2	120	80
3	150	50

The inventory charge per week is worked out as 1% of the price of the item. The fixed costs A_j for making an order are all the same at £100. The company at present manufactures the machines in batches calculated through the Wilson Lot Size formula. There is concern however at the amount of money tied up in stock under this policy and it has been decided to cut the average value

of stock held by 20%. How can this be done at minimum increase in total cost of ordering and inventory?

In this case constraint 3.2 will be of the form

3.9 $$\Sigma\, C_j\, Q_j \leq C$$

where C_j is the cost of each item. Also we will have $I_j = IC_j$ for some I - in the example .01. We see then that the optimal Q_j as given by 3.7 are

$$\begin{aligned} Q_j &= (2\lambda_j A_j/(IC_j - 2\theta C_j)) \\ &= (2\lambda A_j/IC_j)^{\frac{1}{2}}(1-2\theta I^{-1})^{-\frac{1}{2}} \\ &= \alpha\, Q_{jw} \qquad \text{where } \alpha = (1-2\theta I^{-1})^{-\frac{1}{2}} \end{aligned}$$

Thus instead of calculating θ we can calculate α and since $\Sigma C_j Q_j$ is to be cut by 20% we see that $\alpha = 0.8$. Thus the optimal lot sizes are

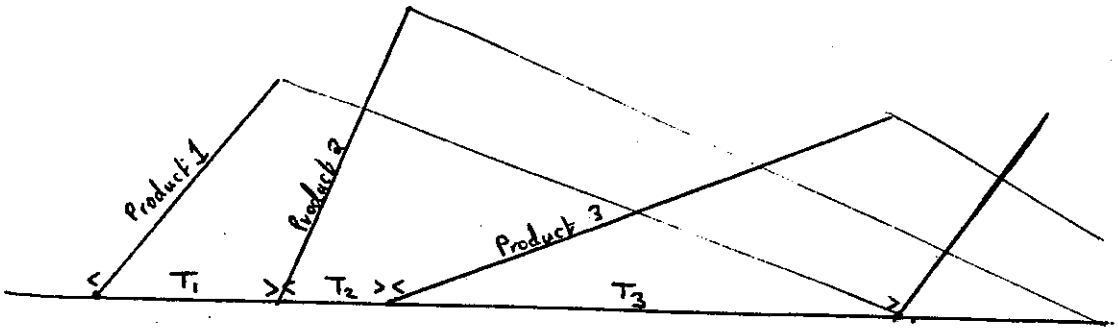
$$Q_1 = 40 \qquad Q_2 \simeq 29 \qquad Q_3 \simeq 6.5$$

The value of θ can be calculated from $\alpha = (1-2\theta I^{-1})^{-\frac{1}{2}}$ giving $\theta = -.0028$. This value has a similar significance to that of the shadow prices of linear programming (§6) i.e. for a small increase δC in the right hand side of 3.9 the saving in cost will be $.0028\, \delta C$.

Model 6

Suppose now that a factory manufactures n products with the characteristics of model 3 i.e. constant demand λ_j and rate of production ψ_j for product j. The products all share the same manufacturing facility and so cannot be manufactured simultaneously.

Instead they must be produced in series in batches i.e. first a batch of product j_1, then a batch of product j_2,... and so on.



This sequence of batches of products $j_1,\ldots j_n$ is to be repeated indefinitely at intervals of time T. Note that it is possible for there to be some idle time between a batch of product j_n being completed and a new batch of j_1 being started due to excess production capacity. Instead of a fixed set up cost for each product we have a change-over cost C_{ij} if a batch of product j is produced immediately after product i. The problem is to find the ordering of batches and the lengths of time interval T so that set up costs and inventory costs are minimised. Fortunately the selection of sequence and calculation of T can be done separately i.e. first find optimal sequence and then calculate T.

Finding an optimal sequence is non-trivial if the number of jobs is not small. It is in fact what is called a travelling salesman problem and the general solution of such problems is discussed elsewhere. For the example given below we simply evaluate the total change-over cost of all possible sequence.

Assume therefore that we have found the best sequence and that the total change-over cost is A.

Now let $Q_1,\ldots Q_n$ be the batch sizes of each product produced

in each production cycle of length T and let $T_1,\ldots T_n$ be the time spent in each cycle producing these batches. Then we have

$$Q_j = \lambda_j T$$

and

$$Q_j = \psi_j T_j$$

for reasons as given in model 3.

We therefore have

3.10 $\quad T_j = \rho_j T \qquad$ where $\rho_j = \lambda_j/\psi_j$

Summing 3.10 for j = 1,...n we have that

$$\sum_j T_j = T \sum \rho_j$$

and since $\sum_j T_j \leq T$ we must have $\sum \rho_j \leq 1$ else it is not possible to produce the products as described above. If $\sum \rho_j < 1$ then there is time to spare and so another product could possibly be fitted in. Assuming then that this production method is feasible we calculate the average total change-over and inventory costs. Now the inventory levels of each individual product will be as in model 3 and so the average inventory cost for product j will be

$$I_j(1-\lambda_j/\psi_j)Q_j/2$$

$$= \quad I_j \lambda_j(1-\rho_j)T/2$$

Therefore the average total cost is K where

3.11 $\quad K = A/T + BT \qquad$ where $B = \frac{1}{2} \sum_j I_j\lambda_j(1-\rho_j)$

The optimal value for T is then found from $\frac{dK}{dT} = 0$ i.e.

$$T = (A/B)^{\frac{1}{2}}$$

and the values K, T_j and Q_j can then be evaluated.

Example

Suppose there are 4 products with data

Product	λ	ψ	I
1	200	600	2
2	100	400	1
3	80	400	3
4	50	250	2

Changeover costs C_{ij} are given by the matrix

-	20	30	40
30	-	20	50
40	30	-	10
30	20	40	-

The possible production sequences are (arbitrarily choose 1 as beginning of cycle)

Cycle	Changeover Cost
1 2 3 4	20 + 20 + 10 + 30 = 80
1 2 4 3	20 + 50 + 40 + 40 =150
1 3 2 4	30 + 30 + 50 + 30 =140
1 3 4 2	30 + 10 + 20 + 30 = 90
1 4 2 3	40 + 10 + 20 + 40 =110
1 4 3 2	40 + 40 + 30 + 30 =140

Thus the optimal sequence is 1-2-3-4 giving A = 80. We note that $\Sigma\rho_j = 59/60$ and so a schedule is possible. Direct calculation gives

$$B = 133.3 + 37.5 + 96 + 40 = 406.8$$

Therefore the optimal value of $T \simeq .44$.

From this we then calculate

j	1	2	3	4
T_j	.15	.11	.09	.09
Q_j	90	44	36	22

$$K = 2(AB)^{\frac{1}{2}} = 361$$

The models discussed in §2 and §3 are obviously much simplified ignoring the major factor of stochastic variation. In spite of this however the batch sizes as calculated do find a use in some practical models.

Inventory Problems

1) A firm manufactures and sells a single product. The demand for the product is 150 units/year. The value of the stock for calculating inventory costs is £10 per unit. The set up cost for a run of the product is £100. The penalty cost is considered to be £5 per unit per year out of stock. The inventory carrying charge is £1 per unit per period held.

Find the optimal re-order policy if

a. Production time is negligible and no stock-outs are allowed.

b. Production time is negligible and stock-outs can be back-ordered.

c. Production is at a rate of 400 units per year and no stock-outs are allowed.

2) A firm manufactures and sells 3 different products, production time being negligible. Data on costs and demands are given in the table below. The maximum average value of inventory that can be held at any one time is £4000. Find the optimal re-order policy under these circumstances if the inventory charge is 10% of the items value per period.

Product	Demand/year	Value/item	Set up cost
1	2000	10	50
2	1500	15	40
3	3000	20	60

3) Assuming the demand and cost data for Q2 find the optimal production policy if

(a) Production rates for products 1,2,3 are 6000, 5000, 12000 items per period respectively.

(b) Changeover costs are given by

i \ j	1	2	3
1	-	150	200
2	100	-	250
3	200	150	-

(4) If the different items can be ordered at the same time at a cost of 50 and other parameters are as in Q2 find the optimum order policy,

(5) Extend model 3 to take account of the possibility of having back-orders as in model 2.

(6) In an inventory system the cost of having q units in inventory for t units of time is Rqt^3. Demand is uniform at a constant rate of λ units per unit time. No stockouts are allowed. The fixed order cost is A. Solve the system

Model 5 - Fixed order level

It may be that it is unrealistic to assume that each additional order means an additional cost. If we consider a situation in which the sales department can deal with up to a certain number of orders per period, then it is reasonable to assume that the cost of orders up to this level remains constant. Suppose then that n items are stocked by the firm and that the parameters I, C_j, λ_j are as in the previous example. Suppose also that up to L orders per period can be dealt with. Then if x_j is the number of orders for items made per period we have the constraint

$$(1.52) \qquad \sum_{j=1}^{n} x_j \leq L$$

For simplicity let us assume that goods arrive in batches and no stock-outs are allowed. Then we only need to calculate the inventory cost per period. The amount of items ordered at a time Q must be such that all demand λ_j in that period is met.

$$Q_j = \lambda_j / x_j$$

The average stock level of items will be $\frac{1}{2}Q_j$ and so the total (inventory) cost per period is

$$(1.53) \qquad K = \sum_{j=1}^{n} \tfrac{1}{2} I C_j \lambda_j / x_j$$

The problem we must solve is that of minimising 1.53 subject to 1.52. Now one can see that no optimal solution to this problem will give a strict inequality in 1.52 because we would be able to obtain a cheaper solution by increasing x_1 for example. So we can

assume equality in 1.52 and we can use a Lagrange multiplier θ as in the previous model. Define

$$J = K + \theta(\Sigma x_j - L)$$

then we have $\frac{\partial J}{\partial x_j} = 0$ in an optimal solution.

This becomes on differentiation

$$-\tfrac{1}{2} IC_j\lambda_j/x_j^2 + \theta = 0$$

$$x_j = (IC_j\lambda_j/2\theta)^{\frac{1}{2}} \tag{1.54}$$

and we choose θ so as to make

$$\Sigma(IC_j\lambda_j/2\theta)^{\frac{1}{2}} = L$$

We take the following example with 5 different items

j	C_j	λ_j	$(C_j\lambda_j)^{\frac{1}{2}}$
1	25	50	35.03
2	5	100	22.36
3	30	150	67.08
4	2	300	24.24
5	1	600	24.24

The inventory charge is $\frac{1}{2}\%$ giving I = .05 and the maximum number of orders per period L is 30. We have calculated the quantities $(C_j\lambda_j)^{\frac{1}{2}}$ because as can be seen from 1.54, the optimal x_j are proportional to them, and consequently we have

1.55 $x_j = L(C_j\lambda_j)^{\frac{1}{2}}/\Sigma\ (C_j\lambda_j)^{\frac{1}{2}}$

From the table we have $\Sigma(C_j\lambda_j)^{\frac{1}{2}} \cong 173$ and therefore

$x_1 = 6.1,\ x_2 = 3.9,\ x_3 = 11.6,\ x_4 = 4.2,\ x_5 = 4.2$

If we want an integer solution, we can start by rounding the optimal solution above giving

$x_1^* = 6,\ x_2^* = 4,\ x_3^* = 12,\ x_4^* = 4,\ x_5^* = 4$

We then try to improve this solution by seeing if the total inventory cost can be decreased by increasing the number of orders for one item by one and simultaneously decreasing the number of orders of another item by one.

j	α_j	β_j
1	20.80	14.90
2	20.80	12.50
3	17.10	14.40
4	25.00	15.00
5	25.00	15.00

α_j is the increase in the average stock value of item j if one less of item j is ordered per period.
β_j is the decrease in the average stock value of item j if one more of item j is ordered per period.

We can see from this table that any such change will increase the average total value of the stock held and consequently we have found the optimal integer solution.

While it is true in general that there is some way of finding the

optimal integer solution by rounding the optimal 'continuous' solution, not all roundings will do. For example if $x_1 = .25$ then rounding to $x_1 = 0$ will not be feasible.

Auctions

Bids made on a single item.

N bidders.

Each has own private valuation of item for sale.

(a) English Auction.

Bids increase. Last person to bid pays last bid.

(b) Dutch Auction

Start at high price. First bidder wins. Pays bid.

(c) Sealed Bid First Price

Bids placed in envelopes.

Highest price wins at that price.

(d) Sealed Bid Second Price

Bids placed in envelopes.

Highest price wins at second price.

$$a \equiv d$$

Optimal to bid evaluation.

[More discussion later]

$$b \equiv c$$

Basically same auction.

Analysis of Symmetric Nash Equilibrium in (c) and (d).

Bidder i values object at value x_i.

$$Pr[x_i \leq x] = F(x), \quad x \in [0, w]$$

Bids $\beta_i(x_i)$.

Symmetric if $\beta_1 = \beta_2 = \cdots = \beta_N$

Second Price Auction

$$\text{Payoff to } i = \begin{cases} x_i - \max_{j \neq i} b_j & b_i > \max_{j \neq i} b_j \\ 0 & \text{otherwise} \end{cases}$$

Prop $\beta^{II}(x) = x$ is dominant strategy

Proof

$i = 1$. $p_1 = \max_{j \neq 1} b_j$

Bid $z_1 < x_1$		$x_1 > z_1 \geq p_1$	$p_1 > x_1 > z_1$	$x_1 > p_1 > z_1$
Profit	z_1	$x_1 - p_1$	0	0
	x_1	$x_1 - p_1$	0	$x_1 - p_1$

Bid $z_1 > x_1$		$z_1 > x_1 \geq p_1$	$p_1 > z_1 > x_1$	$z_1 > p_1 > x_1$
Profit	z_1	$x_1 - p_1$	0	$x_1 - p_1$
	x_1	$x_1 - p_1$	0	0

$$y_1 = \max\{x_2, x_3, \cdots, x_n\}.$$

$$P[y_1 \le y] = F(y)^{N-1} = G(y).$$

Expected Payment

$$m^{II}(x) = G(x)\, E[y_1 \mid y_1 < x].$$

First Price Auction

$$\text{Payoff to } i = \begin{cases} x_i - b_i & \text{if } b_i > \max_{j \neq i} b_j \\ 0 & \text{otherwise} \end{cases}$$

Assume $j \neq 1$ follow $\beta \nearrow$

$$y_1 = \max_{i \neq 1} x_j$$

Bidder 1 bids b.

(i) $b \leq \beta(w)$.

(ii) $\beta(0) = 0$.

$$E(\text{payoff}) = \underset{\nearrow}{G}(\beta^{-1}(b)) \cdot (x - b)$$

$$P_1[\beta(y_1) < b]$$

Maximising w.r.t. b. $\quad G' = g$

$$\frac{g(\beta^{-1}(b))}{\beta'(\beta^{-1}(b))}(x-b) - G(\beta^{-1}(b)) = 0$$

$y = \beta^{-1}(b)$; $b = \beta(y)$; $\frac{db}{dy} = \beta'(y) = \beta'(\beta^{-1}(b)) = \frac{1}{\frac{dy}{db}}$.

Heuristic: $b = \beta(x)$ in symmetric equilibrium

$$G(x)\beta'(x) + g(x)\beta(x) = xg(x)$$

$$\frac{d}{dx}\left(G(x)\beta(x)\right) = xg(x)$$

$$\beta(x) = \frac{1}{G(x)} \int_{y=0}^{x} y g(y)\, dy$$

$$= \mathbb{E}(y_1 \mid y_1 < x).$$

Prop $\beta^I(x) = E(y_1 \mid y_1 < x)$ defines a symmetric equilibrium.

Proof

Assume $2, 3, \ldots, N$ use this strategy

Expected payoff to bidder 1 if value is 1 is x and he bids b.

$z = \beta^{-1}(b)$.

$$\Pi(b, x) = \underbrace{G(z)}_{P_1(b \text{ wins})}(x - \beta(z))$$

$$= G(z)x - G(z)E(y_1 \mid y_1 < z)$$

$$= G(z)x - \int_0^z y\, g(y)\, dy$$

$$= G(z)x - G(z)z + \int_0^y G(y)\, dy.$$

$$\Pi(\beta(x), x) - \Pi(b, x)$$

$$= G(x)x - G(x)x + \int_0^x G(y)\,dy$$

$$-\left(G(z)x - G(z)z + \int_0^z G(y)\,dy\right)$$

$$= G(z)(z-x) - \int_x^z G(y)\,dy$$

$$= \int_x^z (G(z) - G(y))\,dy \geq 0.$$

□

$$\beta^I(x) = \frac{1}{G(x)} \int_0^x y g(y)\,dy = \frac{1}{G(x)} \left[[yG(y)]_0^x - \int_0^x G(y)\,dy \right]$$

$$= x - \int_0^x \frac{G(y)}{G(x)}\,dy$$

$$= x - \int_0^x \left(\frac{F(y)}{F(x)}\right)^{N-1} dy.$$

Examples

(i) $F(x) = x, \quad x \in [0,1]$ Uniform $[0,1]$

$$\beta^I(x) = \frac{N-1}{N} x$$

(ii) $F(x) = 1 - e^{-\lambda x}, \quad N = 2$

$$\beta^I(x) = x - \int_0^x \frac{F(y)}{F(x)} dy$$

$$= x - \frac{1}{1-e^{-\lambda x}} \left[y + \lambda^{-1} e^{-\lambda y} \right]_0^x$$

$$= \frac{1}{\lambda} - \frac{x e^{-\lambda x}}{1 - e^{-\lambda x}}$$

$\lambda = 2$: Never bids more than $1/2$

[Neither does other bidder]

Revenue

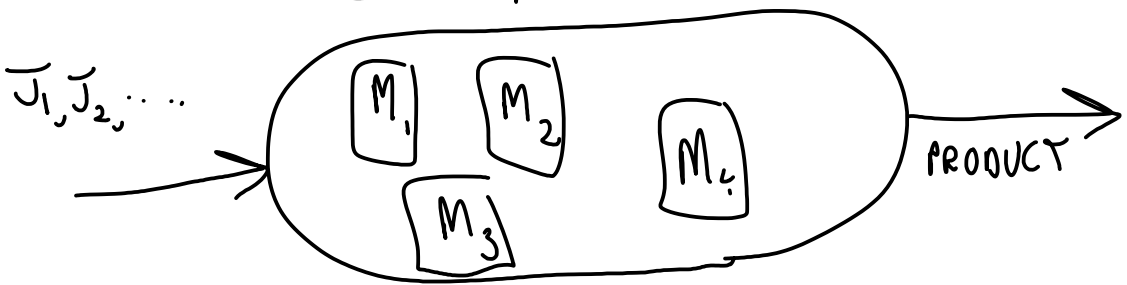
$$m^{I}(x) = G(x)\,E(y_1 \mid y_1 < x) = m^{II}(x).$$

10\20\14

Job Shop Scheduling

Various optimization problems

Machine Shop



Example 1: $1 \mid \cdot\cdot \mid \sum_j \omega_j C_j$

1 machine

restrictions, comments

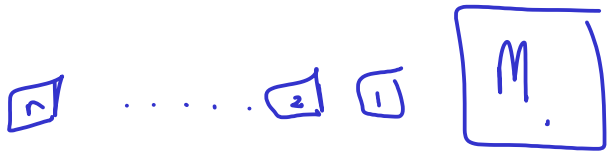
objective function

C_j = completion time of job j

P_j = processing time

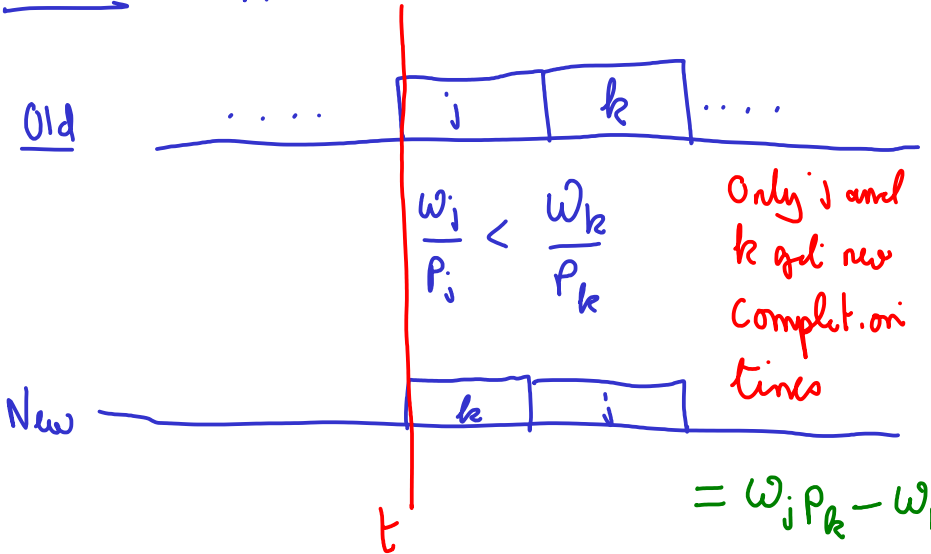
? What is the best ordering of the jobs to minimise the objective?

Optimal Ordering



$$\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq \cdots \geq \frac{w_n}{p_n}$$

Proof: Suppose we do not use this order. There exist



Only j and k get new completion times

New − Old =

$w_k(t+p_k) +$

$w_j(t+p_k+p_j)$

$- w_j(t+p_j)$

$- w_k(t+p_k+p_j)$

$= w_j p_k - w_k p_j < 0.$ □

Ex 2 $1 \mid \cdots \mid L_{max}$

Job j has a due date of d_j.

$L_j = (C_j - d_j)^+$

$L_{max} = \max_j L_j$

Sort so that: $d_1 \leq d_2 \leq \cdots \leq d_n$

Proof: Suppose we do not use this order.

Contribution of j & k: old: $\max\left(\left[C_j^{old} - d_j\right]^+, \left[C_k^{old} - d_k\right]^+\right)$

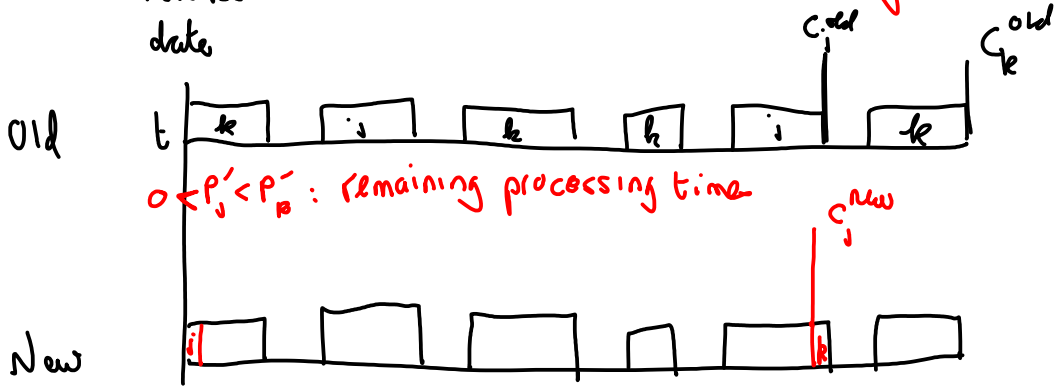
New: $\max\left[\left(C_j^{new} - d_j\right)^+, \left[C_k^{new} - d_k\right]^+\right) = \left[C_k^{old} - d_k\right]^+$

$\leq \left(C_k^{old} - d_k\right)^+$ ↙ ≤

Ex3 $1 \mid r_j, \text{ preemption} \mid \sum_j C_j$

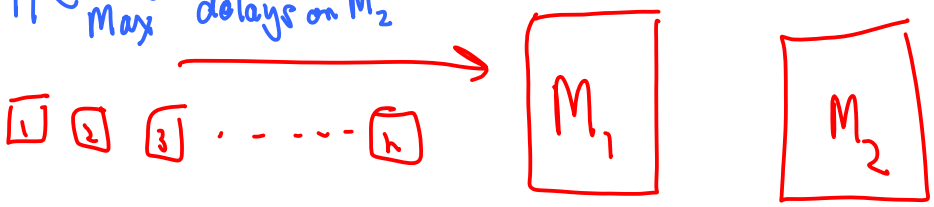
r_j: release date

SRPT rule: Shortest Remaining Processing Time



Two machine Flow Shop – Johnson's Rule

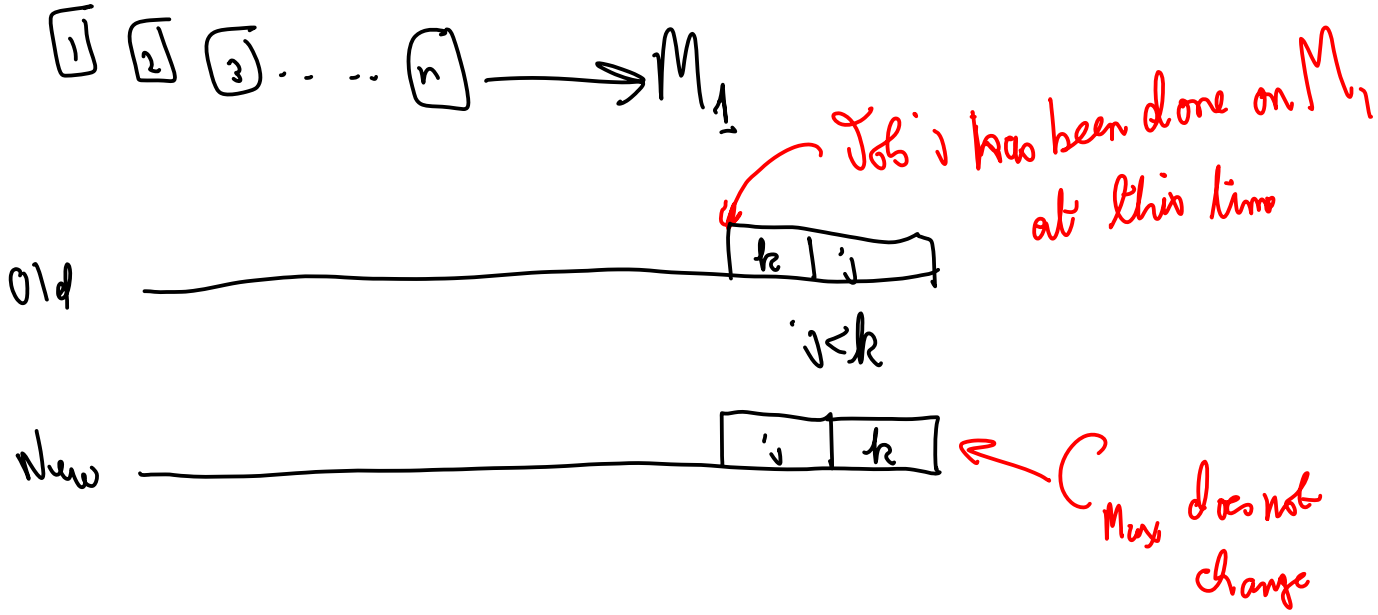
$F2 \| C_{max}$ ← minimise delays on M_2



Flow shop: every job is processed first by M_1 and then by M_2.

Permutation Flow Shop: same order each machine.

We can assume a permutation schedule:

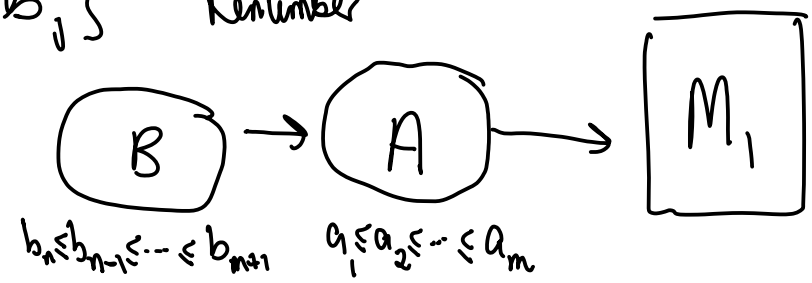


Processing times on M_1 are $a_1, a_2, \ldots, a_n$

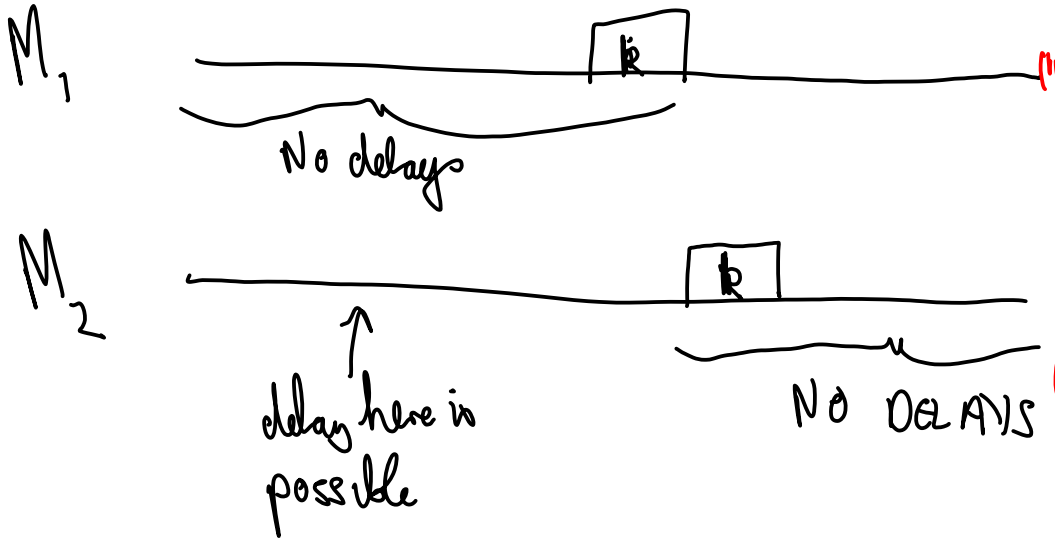
M_2 are $b_1, b_2, \ldots, b_n$

$A = \{ j : a_j \leq b_j \}$ Renumber

$B = [n] \setminus A$



$\forall$ permutation schedules $\exists i$:



(I) C_{max} = sum of $n+1$ job times

(II) Reducing every a_j, b_j by same amount ρ does not change optimal order.

(III) $a_i \leftarrow 0 \Rightarrow$ i should go first

$b_i \leftarrow 0 \Rightarrow$ i should go last

this yields J. rule.