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) $$c_1(x_1) + f_2(i_0 + x_1 - d_1)$$

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

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) $$f_n(i) = \min_{x_n} \;(c(x_n))$$

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 $x_r(i)$.

<u>Stage 1</u> - calculation of f_4

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

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

<u>Stage 2</u> - calculation of f_3

In this case 1.3 becomes

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

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

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 had 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) $g_1(i) = c(i + d_1)$

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

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 a~~ Allow backordering, up to $-B$
(iii) Add a "smoothing" penalty.

Knapsack Problem

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

Problem

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

subject to

$$\sum_{j=1}^{n} w_j x_j \leq W$$

$x_j \geq 0$ and integer, $j = 1, 2, \dots, n$.

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

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

$x_1, \dots, x_4 \geq 0$ 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 $2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 12$

$x_1, \dots x_4 \geq 0$ 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 $x_r(w) = 0$

$$\left.\begin{array}{l} x_4(12) > 0 \\ x_4(7) > 0 \\ x_4(2) = 0 \end{array}\right\} \rightarrow x_4(12) = 2$$

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

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

Complexity of above algorithm $= O(nW)$

(2) Let $f(w) = \max . \; c_1x_1 + \dots + c_nx_n$

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

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

Let $\mu = \min\limits_j w_j$ then

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

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

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 (A) 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

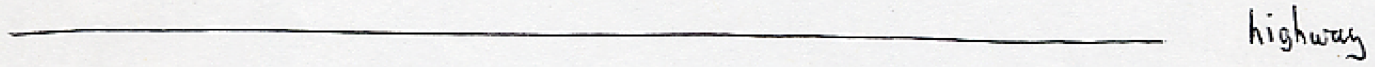
$$f_n(T,h) = A + f_n(0,h).$$

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 $[i, i+1, \cdots j]$.

How should the stick be broken in order to maximise the total value.

Example



v_{ij} = 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 $f(j,r)$, $j = 1, 2, \cdots k$, $r = 0, 1, \cdots L$.

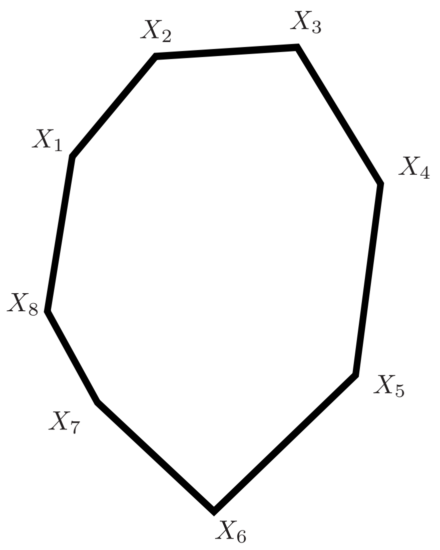
$$f(j,r) = 0 \qquad j > r$$

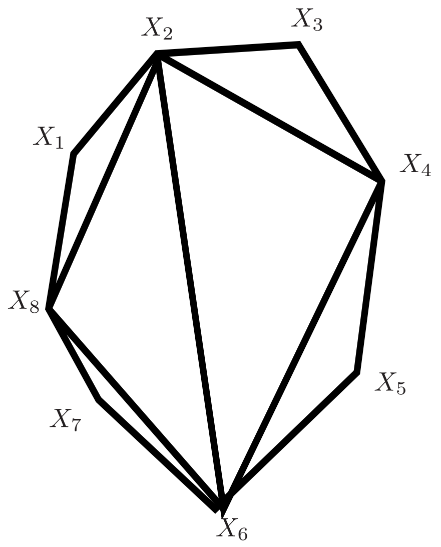
$$= \max_{0 \le i < r} \left[f(j-1, i) + v_{i,r} \right]$$

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

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

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

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

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 $\pi$.
2.  Compute $y$ as in (1).
3.  If (2) holds – current $\pi$ is optimal, stop.
4.  If (2) doesn't hold then
5.          compute $\lambda$ 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:

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

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.

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

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

This represents the total cost of a "tour through $[n]$ in the order $1, \pi(1), \pi^2(1), \ldots, \pi^n(1), 1$. There are $(n-1)!$ distinct tours (each tour, as a set of directed edges of $\vec{K}_n$, arises from n distinct permutations.)
With DP we can solve the problem in $O(n^2 2^n)$ time. For $1 \in S \subseteq [n]$ and $x \in S$, let $f(x,S)$ denote the minimum cost of a path that begins at 1, ends at x and visits each vertex in S exactly once. Then, $f(x,S) = 0$ for $S = \{1\}$ and

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

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

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

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