Problem 1: Inventory Control MDP

Part a

State space

If backlogged stock units are considered as negative units, then the state space is integers from negative infinity to M, where M is the warehouse capacity. I.e.

$$S = \{ \dots, -2, -1, 0, 1, 2, \dots, M\}$$

Transition Probabilities

Let s be the number of units in period t, $s \in S$

Let a be the number of units ordered in period t, $a \in \{0,1,2, \dots, M - s\}$

Let j be the number of units in period $t + 1$

Let c be the number of units ordered by customers, $c \in \{0,1,2, \dots\}$

$$\therefore j = s + a - c \quad \rightarrow \quad j \in S$$

Let p_c be the probability of getting c customer orders.

Define q_{s+a} as the probability of customer orders exceeding or equalling $s + a$, i.e.

$$q_{s+a} = \sum_{c=s+a}^{\infty} p_c$$

Note three statements:

- The probability that j exceeds $s + a$ is 0.
- $j \leq 0$ occurs iff $c \geq s + a$, so the probability that $j \leq 0$ is q_{s+a}
- $j > 0$ occurs iff $c < s + a$

Therefore the transition probabilities can be defined as:

$$p_t(j|s,a) = \begin{cases} 0 & \text{if } M \geq j > s + a \\ q_{s+a} & \text{if } M \geq s + a \text{ and } j \leq 0 \\ p_{s+a-j} & \text{if } M \geq s + a \geq j > 0 \end{cases}$$

Expected rewards

Let $f(x)$ be the revenue for delivering x units to customers, $f(x) = 0$ if $x \leq 0$

If $s + a > 0$, revenue can be received for new customer orders, up to a maximum of $s + a$ orders. But if the number of customers orders $\geq s + a$, the maximum revenue is $f(s + a)$.

Therefore the expected present value of the revenue received in a month is:

$$F(s+a) = \sum_{i=0}^{s+a-1} f(i)p_i + f(s+a)q_{s+a}$$

Let $O(a)$ be the cost of ordering a units

Let $h(x)$ be the cost of maintaining an inventory of x units for a month, $h(x) = 0$ if $x \leq 0$

Let d be the number of units in backlog, $d = |\min(0, s)|$

Let $b(d)$ be the cost of having d units backlogged for a month

It was assumed that delivery to customers can only happen at the end of each month, even if there is a backlog.

The revenue received for filling backlogged orders is $f(\min(d, a))$

Therefore the expected reward is:

$$r_t(s,a) = F(s+a) + f(\min(d,a)) - O(a) - h(a + \max(0,s)) - b(d)\,, \quad t = 1,2,\dots,N-1$$

The value of terminal inventory is:

$$r_N(s) = g(s)\,, \quad t = N$$

Part b

The values provided are:

$$O(u) = \begin{cases} 4 + 2u & \text{if } u > 0 \\ 0 & \text{if } u = 0 \end{cases}$$

$g(u) = 0 \qquad h(u) = u \qquad b(u) = 3u \qquad f(u) = 8u \qquad M = 3 \qquad N = 3$

$$p_c = \begin{cases} \frac{1}{4} & \text{if } c = 0 \\ \frac{1}{2} & \text{if } c = 1 \\ \frac{1}{4} & \text{if } c = 2 \end{cases}$$

The expected revenue is shown in Table 1.

Table 1: Expected revenue

u	$F(u)$
0	0
1	$0 \times \frac{1}{4} + 8 \times \frac{3}{4} = 6$
2	$0 \times \frac{1}{4} + 8 \times \frac{1}{2} + 16 \times \frac{1}{4} = 8$
3	$0 \times \frac{1}{4} + 8 \times \frac{1}{2} + 16 \times \frac{1}{4} = 8$

Matlab was used to generate the expected rewards, shown in Table 2. X indicates impossible cases, where $\max(s, 0) + a$ exceeds warehouse capacity M, which in is 3. Note that s can extend to negative infinity. The code can be found in Appendix A.

Table 2: Expected rewards

	$r_t(s,a)$			
a / s	0	1	2	3
-7	-21	-20	-15	-10
-6	-18	-17	-12	-7
-5	-15	-14	-9	-4
-4	-12	-11	-6	-1
-3	-9	-8	-3	2
-2	-6	-5	0	3
-1	-3	-2	1	0
0	0	-1	-2	-5
1	5	0	-3	X
2	6	-1	X	X
3	5	X	X	X

The transition function is shown in Table 3. Note that $s + a$ can extend to negative infinity.

Table 3: Transition function

	$p_t(j\|s,a)$							
j / $s+a$	-4	-3	-2	-1	0	1	2	3
-2	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	0	0	0	0	0
-1	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	0	0	0	0
0	0	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	0	0	0
1	0	0	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	0	0
2	0	0	0	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	0
3	0	0	0	0	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$

Part c

The state space, transition probabilities and expected rewards have been stated in the previous sections.

Decision epochs:

$$T = \{1,2,\dots,N\}, \qquad N \leq \infty$$

Actions

The action is the amount of additional stock to order, which is limited by the warehouse capacity. Since existing stock s can be negative to indicate backlogged orders,

$$A_s = \{0,1,2,\dots,m\}$$

Where $m = \min(M-s, M)$

Problem 2: Queueing Control MDP

Problem 3.21 was chosen. A diagram of the system is shown in Figure 1.

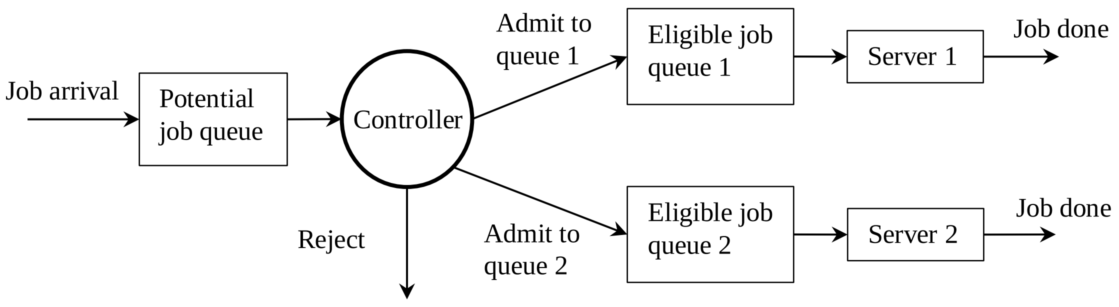

Figure 1: Admission control system

Decision epochs:

It is assumed that the controller makes decisions every $\eta > 0$ units of time.

$$T = \{0, \eta, 2\eta, \dots, N\eta\}, N \leq \infty$$

States:

Let S_1 be the number of jobs in eligible job queue 1

Let S_2 be the number of jobs in eligible job queue 2

Let S_3 be the number of jobs in the potential job queue

It is assumed that jobs pass through the server instantaneously upon completion.

$$S = \{0,1,\dots\} \times \{0,1,\dots\} \times \{0,1,\dots\} = S_1 \times S_2 \times S_3$$

Actions:

The controller can submit jobs to eligible job queues 1 and 2, but it is assumed that the controller cannot duplicate jobs and send the same job to both queues. The total number of jobs submitted to the eligible job queues cannot exceed the number of jobs in the potential job queue.

Let a_1 be the number of jobs submitted to eligible job queue 1

Let a_2 be the number of jobs submitted to eligible job queue 2

$$A_s = \{(a_1, a_2) \mid a_1 \geq 0, a_2 \geq 0, a_1 + a_2 \leq s_3\}$$

Rewards:

Let R be the constant reward received for every completed job

Let $h_i(x)$ be the holding cost for x jobs at server i and its eligible queue, $i = 1$ or 2

Let $Y_{i,t}$ be a random variable denoting the number of jobs completed by queue i at period t.

Let $f_i(n)$ be the time invariant probability of n jobs completed by queue i

$$f_i(n) = P(Y_{i,t} = n)\ , \qquad t = 0,1,\ldots$$

If the number of possible jobs completed $Y_{i,t}$ exceeds the number of eligible jobs $s_i + a_i$, only $s_i + a_i$ jobs are completed. Hence the expectation for the number of jobs completed in period t is:

$$E\{\min(Y_{i,t}, s_i + a_i)\} = \sum_{j=1}^{s_i+a_i-1} jf(j) + (s_i + a_i) \sum_{j=s_i+a_i}^{\infty} f(j)$$

Therefore the expected reward is:

$$r_t(s_1, s_2, a_1, a_2) = R\ \cdot E\{\min(Y_{1,t}, s_1 + a_1)\} + R\ \cdot E\{\min(Y_{2,t}, s_2 + a_2)\} - h(s_1 + a_1 + s_2 + a_2)$$

Transition probabilities:

Let s_1', s_2', s_3' denote the state variables in the next period

Note three statements for the eligible job queue:

- The number of jobs in the queue next state s_i' cannot exceed $s_i + a_i$
- If the maximum number of jobs that could be completed exceeds the number of jobs in the eligible job queue, the number of jobs in the next state is $s_i' = 0$
- If $s_i + a_i = 0$, then $s_i' = 0$

Define a helper probability function k as follows:

$$k(s_i' | s_i, a_i) = \begin{cases} f_i(s_i + a_i - s_i') & s_i + a_i > s_i' > 0 \\ \left[\displaystyle\sum_{j=s_i+a}^{\infty} f_i(j)\right] & s_i' = 0, s_i + a_i > 0 \\ 1 & s_i' = s_i + a_i = 0 \\ 0 & s_i' > s_i + a_i \geq 0 \end{cases}$$

Let $g(n)$ be the time invariant probability of n jobs entering the potential job queue in period t

The transition probability function is then:

$$p_t(s_1', s_2', s_3' | s_1, s_2, s_3, a_1, a_2) = k(s_1' | s_1, a_1) k(s_2' | s_2, a_2) g(s_3')$$

Possible operating policy

Depending on the job completion probability, reward and holding cost, there would be a constant number of jobs that is most likely to give the highest reward for each eligible job queue. The controller should try to allocate jobs to meet this number and rejecting excess jobs. If there is insufficient jobs, the controller should submit more jobs to the faster queue, that is, the queue with higher average $f(n)$.

Problem 3: More MDP Examples

Problem 3.26 was chosen.

Decision Epochs:

The lion can make one decision per day.

$$T = \{0,1,2,\dots\}$$

States:

The state is the energy reserve of the lion in kg of meat, with a maximum reserve of 30 kg. If $s \in S$ becomes negative, the lion dies.

$$S = \{s \in \mathbb{R} | s \le 30$$

Actions:

The lion can choose whether to hunt or not, and if hunting, the lion can choose the size of the group. Let $a \in A$ be an integer representing the action, with $a = 0$ meaning the lion chose not to hunt on that day. It is assumed that although a hunt consumes 0.5 kg of energy, there are no energy requirements restricting ability to hunt.

$$A_s = \{0,1,2,\dots\}$$

Rewards:

To maximise the probability of survival, the lion should maximise its energy reserve, so here the reward is the current energy reserve and the expected energy reserve change from going on a hunt. Let $p(n)$ be the probability of a successful hunt in a group of size $n \ge 0$

$$p(n) = \begin{cases} 0 & \text{if } n = 0 \\ 0.15 & \text{if } n = 1 \\ 0.33 & \text{if } n = 2 \\ 0.37 & \text{if } n = 3 \\ 0.40 & \text{if } n = 4 \\ 0.42 & \text{if } n = 5 \\ 0.43 & \text{if } n \ge 6 \end{cases}$$

The lion consumes 6 kg per day, 0.5 kg is consumed for a hunt, and each lion has a capacity of 30 kg. The energy reserve gained from a successful hunt is divided equally, so each lion gains $164/n$ kg where n is the group size.

$$r_t(s,a) = \begin{cases} s - 6 & a = 0 \\ \max\left(s + p(a)\left(\frac{164}{a}\right) - 6.5, 30\right) & a > 0 \end{cases}$$

Transition Probabilities:

$$p_t(j|s,a) = \begin{cases} p(a) & \text{if } a > 0 \text{ and } j = \max\left(s + \frac{164}{a} - 6.5, 30\right) \\ 1 - p(a) & \text{if } a > 0 \text{ and } j = s - 6.5 \\ 1 & \text{if } a = 0 \text{ and } j = s - 6 \end{cases}$$

Problem 5: Restless bandits

Decision epochs:

$$T = \{\mu, 2\mu, \dots, N\mu\}, \qquad N \le \infty$$

Where μ is 30 minutes

States:

$$S = S^1 \times S^2 \times S^3$$

Where S^i is the set of states for bandit i,

$$S^i = \{1,2,3\}$$

Where $1 \equiv$ deep sleep, $2 \equiv$ awake and happy, $3 \equiv$ screaming the house down

Actions:

Define an action $a \in A_s$ as a set of integers indicating which bandit(s) are selected. Number of bandits selected must be equal to or less than 2, i.e. $|a| \le 2$. Hence:

$$A_s = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}\}, \qquad s \in S$$

Rewards:

Cost = number of screaming bandits + number of feeding (active) bandits

Define

$$c(s^i) = \begin{cases} 0 & \text{if } s^i = 1 \\ 0 & \text{if } s^i = 2 \\ 1 & \text{if } s^i = 3 \end{cases}$$

Then

$$r_t\big((s^1, s^2, s^3), a\big) = -c(s^1) - c(s^2) - c(s^3) - |a|$$

Transition probabilities:

For an active bandit, the transition probability matrix is:

$$P_a = \begin{bmatrix} 0.6 & 0.1 & 0.3 \\ 0.5 & 0.2 & 0.3 \\ 0.3 & 0.6 & 0.1 \end{bmatrix}$$

For a passive bandit:

$$P_p = \begin{bmatrix} 0.6 & 0.1 & 0.3 \\ 0.1 & 0.3 & 0.6 \\ 0.2 & 0.1 & 0.7 \end{bmatrix}$$

Hence

$$p_t(u^i | s^i, a) = \begin{cases} P_{a\, s^i, u^i} & \text{if } i \in a \\ P_{p\, s^i, u^i} & \text{otherwise} \end{cases}$$

$$p_t\big((u^1, u^2, u^3) \big| (s^1, s^2, s^3), a\big) = p_t(u^1 | s^1, a) \times p_t(u^2 | s^2, a) \times p_t(u^3 | s^3, a)$$

Stationary Markov policy:

The cost is minimised by minimising the number of times a bandit is in state 3 (screaming) and the number of times a bandit is selected to be active. Since a bandit in state 3 has a high probability $(0.3 + 0.6 = 0.9)$ of going to state 1 or 2 when active, a possible policy is to select bandits that are in state 3.

I.e. the decision rule would be:

$$d_t(s^1, s^2, s^3) = \begin{cases} \emptyset & \text{if } s^1 \neq 3, s^2 \neq 3, s^3 \neq 3 \\ \{1\} & \text{if } s^1 = 3, s^2 \neq 3, s^3 \neq 3 \\ \{2\} & \text{if } s^1 \neq 3, s^2 = 3, s^3 \neq 3 \\ \{3\} & \text{if } s^1 \neq 3, s^2 \neq 3, s^3 = 3 \\ \{1,2\} & \text{if } s^1 = 3, s^2 = 3 \\ \{1,3\} & \text{if } s^1 = 3, s^2 \neq 3, s^3 = 3 \\ \{2,3\} & \text{if } s^1 \neq 3, s^2 = 3, s^3 = 3 \end{cases} \quad \forall\, t \in T$$

Matlab was used to simulate 10,000 time steps to evaluate the average cost per step of this policy. The result was found to be approximately 2 (the results varied around 1.99 to 2.01). The Matlab code can be found in Appendix B.

Appendix A: Matlab code for Problem 1

F.m

```
function output = F( u )
%F Returns expected revenue
    if u <= 0
        output = 0;
    elseif u == 1
        output = 6;
    elseif u == 2
        output = 8;
    elseif u >= 3
        output = 8;
    end
end
```

O.m

```
function cost = O( a )
%O Returns cost of ordering a units
    if a == 0
        cost = 0;
    elseif a > 0
        cost = 4 + 2*a;
    end
end
```

r.m

```
function reward = r( s, a )
%R Returns expected reward
    d = abs(min(0,s));
    reward = F(s+a) + 8*(min(d,a)) - O(a) - (a+max(0,s)) - 3*d;
end
```

makeMatrix.m

```
m=zeros(11,4);
for a=0:3,
    for s=-7:3,
        if max(s,0)+a<=3
            m(s+8,a+1)=r(s,a);
        else
            m(s+8,a+1)=999;      % 999 indicates impossible case
        end
    end
end
m
```

Appendix B: Matlab code for Problem 5

r.m

```
function reward = r( s ,a )
%R Reward function
    reward = 0;
    for i = 1:numel(s)
        if s(i) == 3
            reward = reward - 1;
        end
    end
    reward = reward - numel(a);
end
```

d.m

```
function a = d( s )
%D Decision rule. Returns action given states
    if s(1) ~= 3 && s(2) ~= 3 && s(3) ~= 3
        a = [];
    elseif s(1) == 3 && s(2) ~= 3 && s(3) ~= 3
        a = [1];
    elseif s(1) ~= 3 && s(2) == 3 && s(3) ~= 3
        a = [2];
    elseif s(1) ~= 3 && s(2) ~= 3 && s(3) == 3
        a = [3];
    elseif s(1) == 3 && s(2) == 3
        a = [1, 2];
    elseif s(1) == 3 && s(2) ~= 3 && s(3) == 3
        a = [1, 3];
    elseif s(1) ~= 3 && s(2) == 3 && s(3) == 3
        a = [2, 3];
    end
end
```

sampleResult.m

```
function u = sampleResult( s, a )
%P Samples a set of states resulting from input states and action

    % Set transition probability matrices
    Pa = [0.6 0.1 0.3; 0.5 0.2 0.3; 0.3 0.6 0.1];
    Pp = [0.6 0.1 0.3; 0.1 0.3 0.6; 0.2 0.1 0.7];

    % Initialise array of resulting states
    u = zeros(1, numel(s));

    for i = 1:numel(s)

        % Choose the appropriate transition probability matrix
        if ismember(i, a)
            P = Pa;
        else
            P = Pp;
        end

        % Generate random number, 0 < x < 1, to set resulting state
        x = rand();
        if x < P(s(i), 1)
            u(i) = 1;
        elseif x < P(s(i), 1) + P(s(i), 2)
            u(i) = 2;
        else
            u(i) = 3;
        end
    end
end
```

test.m

```
numTests = 10000;
totalReward = 0;
s = [1, 1, 1];

for i=1:numTests
    a = d(s);
    s = sampleResult(s, a);
    totalReward = totalReward + r(s, a);
end

averageCost = -totalReward / numTests;
fprintf('Average cost per step is %f\n', averageCost);
```