Problem 1: Inventory Control

Set $N = 5,\ K = 5,\ c(u) = 3u,\ g(u) = 0,\ h(u) = 2u,\ M = 2,\ f(u) = 16u,$
$p_j = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}\right)$ for $j = 0,1,2$ respectively
This gives the state and action space as:

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

The expected revenue is $(u) = \sum_{j=0}^{u-1} f(j)p_j + f(u)q_u$. The calculated values are shown in Table 1

Table 1: Expected revenue

u	$F(u)$
0	0
1	$0 \times \frac{1}{4} + 16 \times \frac{3}{4} = 12$
2	$0 \times \frac{1}{4} + 16 \times \frac{1}{4} + 32 \times \frac{1}{2} = 20$

The expected reward is calculated from $r(s,a) = F(s+a) - O(a) - h(s+a)$ as follows:

$s = 0, a = 0,1,2$:

$$r(0,0) = F(0) - O(0) - h(0) = 0 - 0 - 0 = 0$$
$$r(0,1) = F(1) - O(1) - h(1) = 12 - (5+3) - 2 = 2$$
$$r(0,2) = F(2) - O(2) - h(2) = 20 - (5+6) - 4 = 5$$

$s = 1, a = 0,1$:

$$r(1,0) = F(1) - O(0) - h(1) = 12 - 0 - 2 = 10$$
$$r(1,1) = F(2) - O(1) - h(2) = 20 - (5+3) - 4 = 8$$

$s = 2, a = 0$:

$$r(2,0) = F(2) - O(0) - h(2) = 20 - 0 - 4 = 16$$

This is summarised in Table 2.

Table 2: Expected rewards

	$r_t(s,a)$		
a	0	1	2
s			
0	0	2	5
1	10	8	X
2	16	x	X

The transition probabilities are shown in Table 3.

Table 3: Transition function

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

Define $u_t^*(s,a) = r(s,a) + \sum_{j\in S} p(j|s,a)u_{t+1}^*(j)$

Backward induction:

Set $t = 6$ and $u_6^*(s) = g(s) = 0$

Set $t = 5$

$$u_5^*(s) = \max_{a\in A_s}\left\{r(s,a) + \sum_{j\in S} p(j|s,a)u_6^*(j)\right\}, \qquad s = 0,1,2$$

$$u_5^*(s) = \max_{a\in A_s}\{r(s,a)\}$$

Hence $u_5^*(s)$ is easily determined for $s = 0,1,2$ from Table 2. A summary for $t = 5$ is given in Table 4.

Table 4: Optimality equation solutions for t = 5

s	$u_5^*(s)$	$A_{s,5}^*$
0	5	2
1	10	0
2	16	0

Set $t = 4$

$$u_4^*(0,0) = r(0,0) + p(0|0,0)u_5^*(0) + p(1|0,0)u_5^*(1) + p(2|0,0)u_5^*(2)$$
$$u_4^*(0,0) = 0 + 1\times 5 + 0 + 0 = 5$$

$$u_4^*(0,1) = r(0,1) + p(0|0,1)u_5^*(0) + p(1|0,0)u_5^*(1) + p(2|0,0)u_5^*(2)$$
$$u_4^*(0,1) = 2 + \frac{3}{4}\times 6 + \frac{1}{4}\times 10 + 0 = 9$$

$$u_4^*(0,2) = r(0,2) + p(0|0,2)u_5^*(0) + p(1|0,2)u_5^*(1) + p(2|0,2)u_5^*(2)$$
$$u_4^*(0,2) = 5 + \frac{1}{2}\times 5 + \frac{1}{4}\times 10 + \frac{1}{4}\times 16 = 14$$

$$u_4^*(1,0) = r(1,0) + p(0|1,0)u_5^*(0) + p(1|1,0)u_5^*(1) + p(2|1,0)u_5^*(2)$$
$$u_4^*(1,0) = 10 + \frac{3}{4}\times 5 + \frac{1}{4}\times 10 + 0 = 16.25$$

$$u_4^*(1,1) = r(1,1) + p(0|1,1)u_5^*(0) + p(1|1,1)u_5^*(1) + p(2|1,1)u_5^*(2)$$
$$u_4^*(1,1) = 8 + \frac{1}{2} \times 5 + \frac{1}{4} \times 10 + \frac{1}{4} \times 16 = 17$$

$$u_4^*(2,0) = r(2,0) + p(0|2,0)u_5^*(0) + p(1|2,0)u_5^*(1) + p(2|2,0)u_5^*(2)$$
$$u_4^*(2,0) = 16 + \frac{1}{2} \times 5 + \frac{1}{4} \times 10 + \frac{1}{4} \times 16 = 25$$

A summary for $t = 4$ is given Table 5.

Table 5: Summary for t = 4

	$u_4^*(s,a)$				
s	$a = 0$	$a = 1$	$a = 2$	$u_4^*(s)$	$A_{s,4}^*$
0	5	9	14	14	2
1	16.25	17	X	17	1
2	25	X	X	25	0

Set $t = 3$

$$u_3^*(0,0) = r(0,0) + p(0|0,0)u_4^*(0) + p(1|0,0)u_4^*(1) + p(2|0,0)u_4^*(2)$$
$$u_3^*(0,0) = 0 + 1 \times 14 + 0 + 0 = 14$$

$$u_3^*(0,1) = r(0,1) + p(0|0,1)u_4^*(0) + p(1|0,0)u_4^*(1) + p(2|0,0)u_4^*(2)$$
$$u_3^*(0,1) = 2 + \frac{3}{4} \times 14 + \frac{1}{4} \times 17 + 0 = 16.75$$

$$u_3^*(0,2) = r(0,2) + p(0|0,2)u_4^*(0) + p(1|0,2)u_4^*(1) + p(2|0,2)u_4^*(2)$$
$$u_3^*(0,2) = 5 + \frac{1}{2} \times 14 + \frac{1}{4} \times 17 + \frac{1}{4} \times 25 = 22.5$$

$$u_3^*(1,0) = r(1,0) + p(0|1,0)u_4^*(0) + p(1|1,0)u_4^*(1) + p(2|1,0)u_4^*(2)$$
$$u_3^*(1,0) = 10 + \frac{3}{4} \times 14 + \frac{1}{4} \times 17 + 0 = 24.75$$

$$u_3^*(1,1) = r(1,1) + p(0|1,1)u_4^*(0) + p(1|1,1)u_4^*(1) + p(2|1,1)u_4^*(2)$$
$$u_3^*(1,1) = 8 + \frac{1}{2} \times 14 + \frac{1}{4} \times 17 + \frac{1}{4} \times 25 = 25.5$$

$$u_3^*(2,0) = r(2,0) + p(0|2,0)u_4^*(0) + p(1|2,0)u_4^*(1) + p(2|2,0)u_4^*(2)$$
$$u_3^*(2,0) = 16 + \frac{1}{2} \times 14 + \frac{1}{4} \times 17 + \frac{1}{4} \times 25 = 33.5$$

Table 6: Summary for t = 3

	$u_3^*(s,a)$				
s	$a=0$	$a=1$	$a=2$	$u_3^*(s)$	$A_{s,3}^*$
0	14	16.75	22.5	22.5	2
1	24.75	25.5	X	25.5	1
2	33.5	X	X	33.5	0

Set $t = 2$

$$u_2^*(0,0) = r(0,0) + p(0|0,0)u_3^*(0) + p(1|0,0)u_3^*(1) + p(2|0,0)u_3^*(2)$$
$$u_2^*(0,0) = 0 + 1 \times 22.5 + 0 + 0 = 22.5$$

$$u_2^*(0,1) = r(0,1) + p(0|0,1)u_3^*(0) + p(1|0,0)u_3^*(1) + p(2|0,0)u_3^*(2)$$
$$u_2^*(0,1) = 6 + \frac{3}{4} \times 22.5 + \frac{1}{4} \times 25.5 + 0 = 29.25$$

$$u_2^*(0,2) = r(0,2) + p(0|0,2)u_3^*(0) + p(1|0,2)u_3^*(1) + p(2|0,2)u_3^*(2)$$
$$u_2^*(0,2) = 5 + \frac{1}{2} \times 22.5 + \frac{1}{4} \times 25.5 + \frac{1}{4} \times 33.5 = 31$$

$$u_2^*(1,0) = r(1,0) + p(0|1,0)u_3^*(0) + p(1|1,0)u_3^*(1) + p(2|1,0)u_3^*(2)$$
$$u_2^*(1,0) = 10 + \frac{3}{4} \times 22.5 + \frac{1}{4} \times 25.5 + 0 = 33.25$$

$$u_2^*(1,1) = r(1,1) + p(0|1,1)u_3^*(0) + p(1|1,1)u_3^*(1) + p(2|1,1)u_3^*(2)$$
$$u_2^*(1,1) = 8 + \frac{1}{2} \times 22.5 + \frac{1}{4} \times 25.5 + \frac{1}{4} \times 33.5 = 34$$

$$u_2^*(2,0) = r(2,0) + p(0|2,0)u_3^*(0) + p(1|2,0)u_3^*(1) + p(2|2,0)u_3^*(2)$$
$$u_2^*(2,0) = 16 + \frac{1}{2} \times 22.5 + \frac{1}{4} \times 25.5 + \frac{1}{4} \times 33.5 = 42$$

Table 7: Summary for t = 2

	$u_2^*(s,a)$				
s	$a=0$	$a=1$	$a=2$	$u_2^*(s)$	$A_{s,2}^*$
0	22.5	29.25	31	31	2
1	33.25	34	X	34	1
2	42	X	X	42	0

Set $t = 1$

$$u_1^*(0,0) = r(0,0) + p(0|0,0)u_2^*(0) + p(1|0,0)u_2^*(1) + p(2|0,0)u_2^*(2)$$
$$u_1^*(0,0) = 0 + 1 \times 31 + 0 + 0 = 31$$

$$u_1^*(0,1) = r(0,1) + p(0|0,1)u_2^*(0) + p(1|0,0)u_2^*(1) + p(2|0,0)u_2^*(2)$$
$$u_1^*(0,1) = 2 + \frac{3}{4} \times 31 + \frac{1}{4} \times 34 + 0 = 33.75$$

$$u_1^*(0,2) = r(0,2) + p(0|0,2)u_2^*(0) + p(1|0,2)u_2^*(1) + p(2|0,2)u_2^*(2)$$
$$u_1^*(0,2) = 5 + \frac{1}{2} \times 31 + \frac{1}{4} \times 34 + \frac{1}{4} \times 42 = 39.5$$

$$u_1^*(1,0) = r(1,0) + p(0|1,0)u_2^*(0) + p(1|1,0)u_2^*(1) + p(2|1,0)u_2^*(2)$$
$$u_1^*(1,0) = 10 + \frac{3}{4} \times 31 + \frac{1}{4} \times 34 + 0 = 41.75$$

$$u_1^*(1,1) = r(1,1) + p(0|1,1)u_2^*(0) + p(1|1,1)u_2^*(1) + p(2|1,1)u_2^*(2)$$
$$u_1^*(1,1) = 8 + \frac{1}{2} \times 31 + \frac{1}{4} \times 34 + \frac{1}{4} \times 42 = 42.5$$

$$u_1^*(2,0) = r(2,0) + p(0|2,0)u_2^*(0) + p(1|2,0)u_2^*(1) + p(2|2,0)u_2^*(2)$$
$$u_1^*(2,0) = 16 + \frac{1}{2} \times 31 + \frac{1}{4} \times 34 + \frac{1}{4} \times 42 = 50.5$$

Table 8: Summary for t = 1

	$u_1^*(s,a)$				
s	$a = 0$	$a = 1$	$a = 2$	$u_1^*(s)$	$A_{s,1}^*$
0	31	33.75	39.5	39.5	2
1	41.75	42.5	X	42.5	1
2	50.5	X	X	50.5	0

Since $t = 1$, stop.

The optimal policy and expected total rewards are shown in Table 9. Due to the high reward of selling inventory, the optimal policy appears to be order as many units as possible, except in the final turn.

Table 9: Optimal policy and expected total reward function

s	$d_1^*(s)$	$d_2^*(s)$	$d_3^*(s)$	$d_4^*(s)$	$d_5^*(s)$	$v_6^*(s)$
0	2	2	2	2	2	39.5
1	1	1	1	1	0	42.5
2	0	0	0	0	0	50.5

Problem 2: Threshold policy in inventory control

The backward induction algorithm for an inventory control problem was implemented in C++. The source code is supplied in Appendix A. The output of this program is the decision rule at each time step, for example, the output for trial 10 is shown in Table 10. The parameters used for each trial is shown in Table 11, where N is the time horizon, M is the warehouse capacity, f is the profit per unit sold, c is the ordering cost per unit, K is the base ordering cost, h is the holding cost per unit, and g is the terminal value per unit. p_j is the probability for selling 0, 1, 2, … units at a time step.

Table 10: Software output for trial 10

s	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15
0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	0
4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11: Trial parameters

Trial	N	M	f	c	K	h	g	p_j
1	15	2	16	3	5	2	0	0.25, 0.25, 0.5
2	15	5	30	5	10	5	0	0.2, 0.2. 0.2, 0.2, 0.1, 0.1
3	15	5	30	2	30	5	0	0.2, 0.2. 0.2, 0.2, 0.1, 0.1
4	15	5	40	5	50	5	0	0.2, 0.2. 0.2, 0.2, 0.1, 0.1
5	15	5	20	1	15	5	0	0.2, 0.2. 0.2, 0.2, 0.1, 0.1
6	15	10	30	5	20	5	0	All 1/11
7	15	5	8	2	4	1	0	0.1, 0.2, 0.3, 0.4, 0.2, 0.1
8	15	5	8	2	4	1	5	0.1, 0.2, 0.3, 0.4, 0.2, 0.2
9	15	7	8	2	4	1	0	All 1/8
10	15	7	8	2	4	1	0	0.2, 0, 0, 0, 0.4, 0, 0, 0.4

A threshold (σ, Σ) policy is one where if the inventory drops below σ units, order a sufficient number of units to raise the inventory to Σ units. It is obvious from Table 10 that $\sigma = 5, \Sigma = 7$. This may not be optimal at the last few time steps, where the behaviour depends on the terminal value. The σ and Σ values for each trial is summarised in Table 12.

Table 12: σ and Σ values for each trial

Trial	σ	Σ
1	2	2
2	3	4
3	2	5
4	2	5
5	2	4
6	5	9
7	3	5
8	3	5
9	4	7
10	5	7

If the ordering cost is not linear with respect to the number of units ordered, the threshold policy may not be the optimal policy. For example, replacing the ordering cost function with

$O(u) = cu^2 + 1\{u > 0\}K$ results in a non-threshold policy. The policy from using the same parameters as in trial 9 but with the new ordering cost function is shown in Table 13. New stock is ordered when the inventory drops below 4, but the maximum number of units ordered at once is 2.

Table 13: Decision rules for non-linear ordering cost function

s	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15
0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0
1	2	2	2	2	2	2	2	2	2	2	2	2	2	1	0
2	2	2	2	2	2	2	2	2	2	2	2	2	1	0	0
3	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Problem 3: Optimal Markov Deterministic Policies

Theorem: An optimal deterministic Markovian policy exists when the state space S is finite or countable and the action space A_s is finite for each $s \in S$.

Proof:

Let u_t^* be the solution to the optimality equation, $t = 1, \dots, N$

$$u_t^*(h_t) = \sup_{a \in A_{s_t}} \left\{ r_t(s_t + a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(h_t, a, j) \right\}$$

Since $u_N^*(h_N) = u_N^*(h_{N-1}, a_{N-1}, s) = r_N(s)$, then $u_N^*(h_N) = u_N^*(s_N)$. Then by the induction hypothesis $u_t^*(h_t)$ depends on h_t only through s_t, i.e.

$$u_t^*(h_t) = \sup_{a \in A_{s_t}} \left\{ r_t(s_t + a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j) \right\} = u_t^*(s_t)$$

Since A_s is finite, there exists an action a' such that

$$r_t(s, a') + \sum_{j \in S} p_t(j|s_t, a') u_{t+1}^*(j) = \sup_{a \in A_s} \left\{ r_t(s + a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j) \right\}$$

Since S is finite u_t^* exists for $t = 1, \dots, N$and hence a decision rule d_t^* also exists. Therefore an optimal deterministic Markovian policy $\pi^* = (d_1^*, d_2^*, \dots, d_{N-1}^*) \in \Pi^{MD}$ exists.

Problem 4: The Secretary Problem

Part 1

The figures were plotted in Matlab and are shown in Figure 1 and Figure 2. Source code can be found in Appendix B.

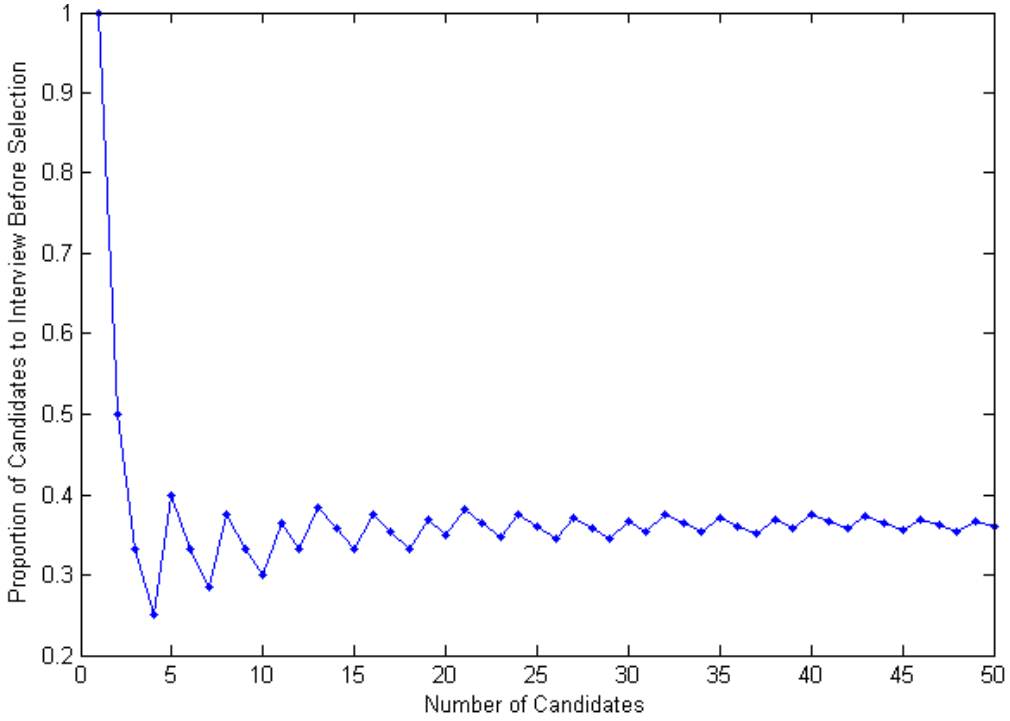

Figure 1: Proportion of candidates to interview before selection vs. number of candidates

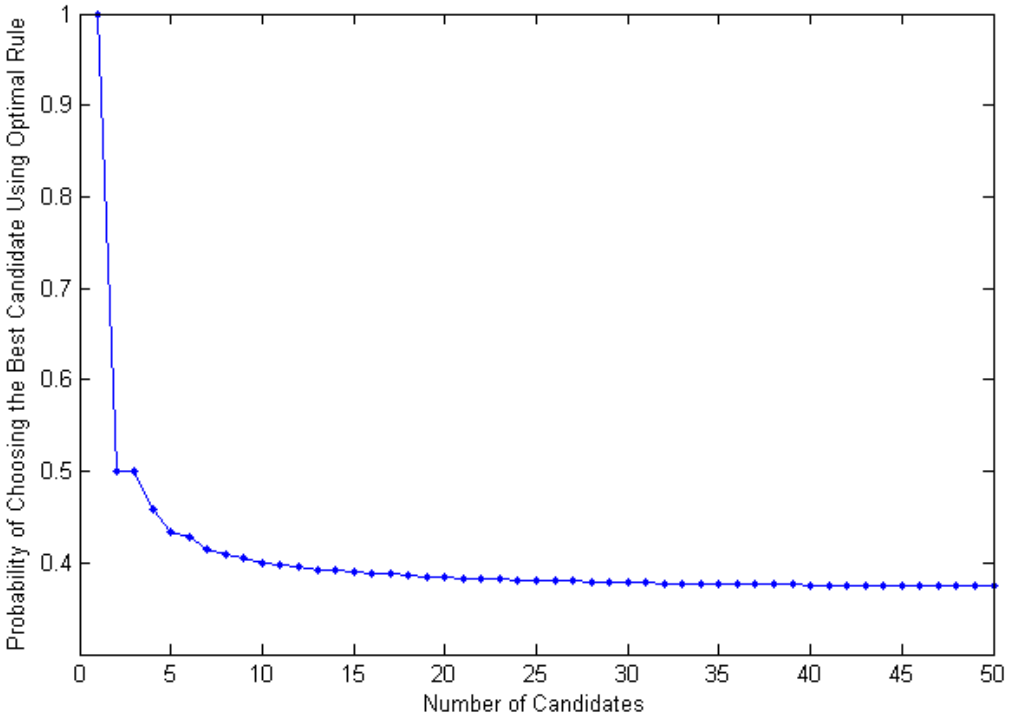

Figure 2: Probability of choosing the best candidate using optimal rule vs. number of candidates

Part 2

$$1 \approx \left[\frac{1}{\tau(N)} + \frac{1}{\tau(N)+1} + \cdots + \frac{1}{N-1}\right] \approx \int_{\tau(N)}^{N} \frac{1}{x}\,dx$$

$$\int_{\tau(N)}^{N} \frac{1}{x}\,dx = [\log x]_{\tau(N)}^{N} = \log N - \log \tau(N) = \log\left(\frac{N}{\tau(N)}\right)$$

So for large N, $\log\left(\frac{N}{\tau(N)}\right) \approx 1$

$$\rightarrow \lim_{N\to\infty} \frac{\tau(N)}{N} = e^{-1}$$

$\therefore$ For large N, $\tau(N) = Ne^{-1}$

This rule means that the first Ne^{-1} candidates should always be rejected where N is the total number of candidates. The first candidate who is better than those interviewed before should then be accepted. Possible uses for this rule of thumb in life include choosing a parking spot, a wife/husband, a gas station, and when to buy or sell a car. A restriction with this rule is that the number of options must be known. In real life applications, there may be a cost involved in searching. Furthermore, there is sometimes the option of going back and accepting an option that was previously rejected, which would change the dynamic of the problem.

Part 3

A modification of the secretary problem is rather than aiming to pick the best candidate, the goal is to pick the second best candidate. This modified problem is called the postdoc variant of the secretary problem by Vanderbei (n.d.), as the motivating story is that when trying to hire a postdoc, the best candidate will accept an offer from Harvard. The probability of success is $\frac{k_0(n-k_0)}{n(n-1)}$ where $k_0 = \left\lfloor\frac{n}{2}\right\rfloor$. As n goes to infinity, this probability tends to 1/4, which is less than the e^{-1} limit for the classical secretary problem. This illustrates the fact that it is easier to pick the best candidate rather than the second best.

Appendix A: C++ Code for Problem 2

main.cpp

```cpp
#include <vector>
#include <map>
#include <string>
#include <iostream>
#include <fstream>

std::string filename = "output10.csv";

std::map<int, double> p;    // number of units -> probability of selling that number
std::map<int, std::vector<double>> uStar;// time -> solution to optimality equation
std::map<int, std::vector<int>> aStar;          // time -> best action

int N = 15;                 // Number of time steps
int M = 7;                  // Maximum warehouse capacity

double profitPerUnit = 8;
double orderCostPerUnit = 2;
double orderCostBase = 4;
double holdCostPerUnit = 1;
double terminalValuePerUnit = 0;

// Present value of revenue for u units
double f(int u) {
	return profitPerUnit * u;
}

// Probability of supply exceeding demand u
double q(int u) {
	double sum = 0;
	std::map<int, double>::iterator it;
	for (it = p.begin(); it != p.end(); it++) {
		if (it->first >= u) {
			sum += it->second;
		}
	}
	return sum;
}

// Returns cost for ordering u units
double O(int u) {
	if (u > 0) {
		return orderCostBase + orderCostPerUnit * u;
	}
	return 0;
}

// Returns holding/maintenance cost for u units
double h(int u) {
	return holdCostPerUnit * u;
}

double F(int u) {
	double sum = 0;
	for (int j = 0; j <= u - 1; j++) {
		if (p.find(j) != p.end()) {
			sum += f(j) * p[j];
		}
```

```
	}
	return sum + f(u) * q(u);
}

// Reward function for state s, action a
double r(int s, int a) {
	return F(s + a) - O(a) - h(s + a);
}

// Terminal value of number of units u
double g(int u) {
	return terminalValuePerUnit * u;
}

// Transition probability function
double tranProb(int j, int s, int a) {
	std::map<int, double>::iterator it;
	if (j > s + a || s + a > M) {
		return 0;
	} else if (j > 0) {
		it = p.find(s + a - j);
		if (it != p.end()) {
			return it->second;
		} else {
			return 0;
		}
	} else {
		return q(s + a);
	}
}

void step(int t) {
	uStar[t] = std::vector<double>();
	aStar[t] = std::vector<int>();
	for (int s = 0; s <= M; s++) {
		double bestReward = -999999;
		int bestAction;
		for (int a = 0; a <= M - s; a++) {
			double reward = r(s, a);
			for (int j = 0; j <= M; j++) {
				reward += tranProb(j, s, a) * uStar[t + 1][j];
			}
			if (bestReward < reward) {
				bestReward = reward;
				bestAction = a;
			}
		}
		uStar[t].push_back(bestReward);
		aStar[t].push_back(bestAction);
	}
}

int main() {

	// Populate selling probabilities
	p[0] = 0.2;
	p[1] = 0;
	p[2] = 0;
	p[3] = 0;
	p[4] = 0.4;
	p[5] = 0;
	p[6] = 0;
```

```
	p[7] = 0.4;

	// Populate uStar and aStar for t = N + 1 (i.e. populate with terminal rewards)
	int t = N + 1;
	uStar[t] = std::vector<double>();
	for (int s = 0; s <= M; s++) {
		uStar[t].push_back(g(s));
	}
	t--;

	// Loop until t == 1
	while (t >= 1) {
		step(t);
		t--;
	}

	// Output results in csv format
	std::ofstream output;
	output.open (filename);
	output << "s";
	for (int t = 1; t <= N; t++) {
		output << ",d" << t;
	}
	output << ",v";
	for (int s = 0; s <= M; s++) {
		output << "\n" << s;
		for (int t = 1; t <= N; t++) {
			output << "," << aStar[t][s];
		}
		output << "," << uStar[1][s];
	}
	return 0;
}
```

Appendix B: Matlab Code for Problem 4

calculateTau.m

```
function tau = calculateTau( N )
%CALCULATETAU Calculates tau given N
if N <= 2
    tau = 1;
else

    for t = 1:(N - 1)
        sum = 0;
        for x = t:(N - 1)
            sum = sum + 1/x;
        end
        if (sum < 1)
            break
        end
    end
    tau = t - 1;
end
```

calculateProb.m

```
function probability = calculateProb( N )
%CALCULATEPROB Probability of choosing best using optimal rule
if (N == 1)
    probability = 1;
else
    sum = 0;
    tau = calculateTau(N);
    for x = tau:(N - 1)
        sum = sum + (1 / x);
    end
    probability = sum * tau / N;
end
end
```

graph.m

```
maxN = 50;
proportions = zeros(1, maxN);
probabilities = zeros(1, maxN);
N = 1:maxN;
for x = 1:maxN
    proportions(x) = calculateTau(x)/x;
    probabilities(x) = calculateProb(x);
end
figure();
plot(N, proportions,'- .');
figure();
plot(N, probabilities,'- .');
```

References

Vanderbei, RJ n.d. *The Postdoc Variant of the Secretary Problem*, viewed 8 September 2014, <http://www.princeton.edu/~rvdb/tex/PostdocProblem/PostdocProb.pdf>.

Puterman, ML 2005, *Markov Decision Processes*, John Wiley & Sons, Inc., Hoboken.