Question 1

(1) Pay of Job A lies in $\mathcal{N}(0,1)$ and the pay of job B lies in $\mathcal{N}(0.5,0.5)$. The distribution of both the pays are shown below.

To be stochastically dominant, the area under Job A should be more than the area under Job B at all times. The total area under the curve for a normal distribution is 1. The area within 3 standard deviation is 99.6% of the total.

Even though initially the area under the curve of Job A is greater than the area under Job B, Job B has 99.6% of its area within 2, whereas Job B has 99.6% of its area within 3. Thus Job B does not stochastically dominate Job A. The plots of both the distribution is shown in Fig1.

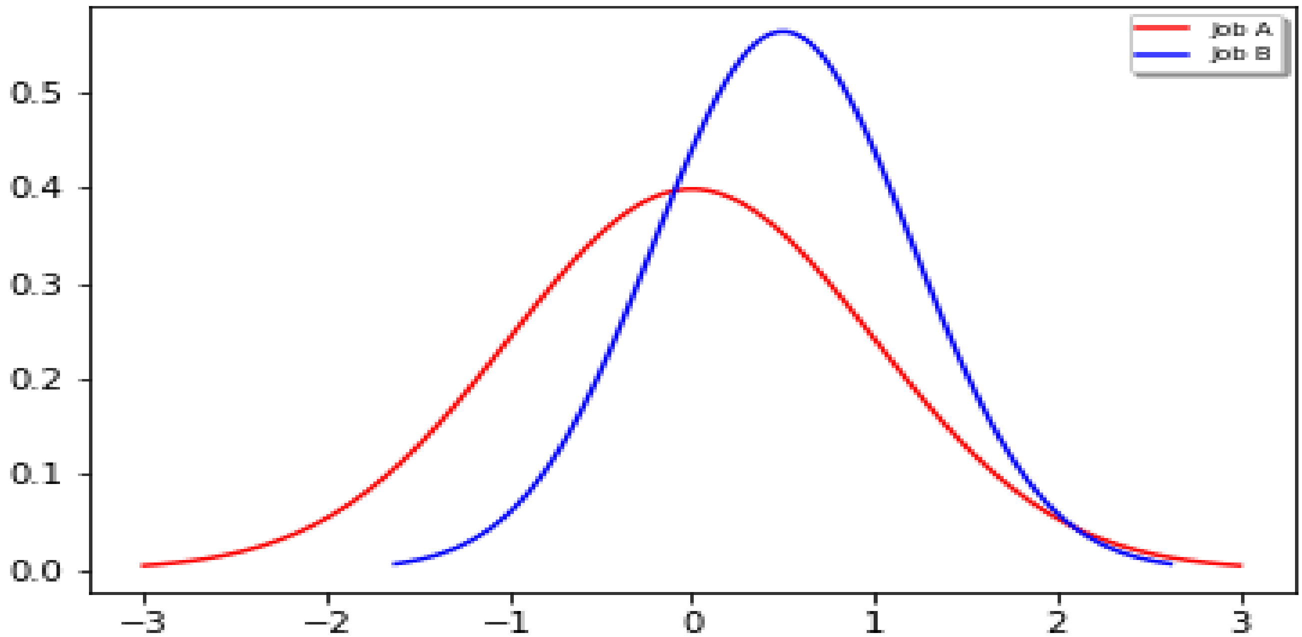

Figure 1: Distribution of Job A and Job B

(2) For Job Y to stochastically dominate Job X,

$$\int_{-\infty}^{z} \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{(t-\mu_x)^2}{2\sigma_x^2}} dt \geq \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(t-\mu_y)^2}{2\sigma_y^2}} dt$$

$$Let\ a = \frac{t-\mu_x}{\sqrt{2}\sigma_x}\ and\ b = \frac{t-\mu_y}{\sqrt{2}\sigma_y}$$

$$\int_{-\infty}^{\frac{z-\mu_x}{\sqrt{2}\sigma_x}} \frac{\sqrt{2}\sigma_x}{\sqrt{2\pi\sigma_x^2}} e^{-a^2} da \geq \int_{-\infty}^{\frac{z-\mu_y}{\sqrt{2}\sigma_y}} \frac{\sqrt{2}\sigma_y}{\sqrt{2\pi\sigma_y^2}} e^{-b^2} db$$

$$\int_{-\infty}^{\frac{z-\mu_x}{\sqrt{2}\sigma_x}} \frac{1}{\sqrt{\pi}} e^{-a^2} da \geq \int_{-\infty}^{\frac{z-\mu_y}{\sqrt{2}\sigma_y}} \frac{1}{\sqrt{\pi}} e^{-b^2} db$$

$$\implies \frac{z-\mu_x}{\sqrt{2}\sigma_x} \geq \frac{z-\mu_y}{\sqrt{2}\sigma_y}$$

If $\sigma_x = \sigma_y$ then, $\mu_y \geq \mu_x$

(3) The AUC of normal distribution is 0 at $-\infty$ and becomes 1 at ∞. Whereas, the AUC of uniform distribution U(a,b) becomes greater than zero at a and becomes 1 at b. Therefore the normal distribution will always have more area initially and the uniform distribution will always reach area=1 before. So, neither uniform distribution stochastically dominates uniform distribution nor vice versa.

Question 2

Since initially we don't know the machine, therefore all the machines are equally likely. Moreover we can skip the -4$ since it is common in all equations.

- SLOT(X) $\rightarrow$ U(X) $= \frac{10}{100} \times 20 + \frac{30}{100} \times 5 + \frac{40}{100} \times 1 + \frac{20}{100} \times 0 = 2 + 1.5 + 0.4 + 0 = 3.9$
- SLOT(Y) $\rightarrow$ U(Y) $= \frac{5}{100} \times 40 + \frac{25}{100} \times 4 + \frac{30}{100} \times 2 + \frac{40}{100} \times 0 = 2 + 1 + 0.6 + 0 = 3.6$
- SLOT(Z) $\rightarrow$ U(Z) $= \frac{25}{100} \times 10 + \frac{25}{100} \times 5 + \frac{25}{100} \times 2 + \frac{25}{100} \times 0 = 2.5 + 1.25 + 0.5 + 0 = 4.25$

$\implies U(XorYorZ) = \frac{1}{3} \times 3.9 + \frac{1}{3} \times 3.6 + \frac{1}{3} \times 4.25 = \frac{11.75}{3}$

Now, identifying one of the slot machines will give rise to three cases:

1. SLOT(X) is identified.
   $\implies U(YorZ) = \frac{3.6+4.25}{2} = \frac{7.85}{2} = 3.925$
   Since $U(YorZ) > U(X)$, the best action would be to play on the remaining two machines.
2. SLOT(Y) is identified.
   $\implies U(XorZ) = \frac{3.9+4.25}{2} = \frac{8.15}{2} = 4.075$
   Since $U(XorZ) > U(Y)$, the best action would be to play on the remaining two machines.
2. SLOT(Z) is identified.
   $\implies U(XorY) = \frac{3.9+3.6}{2} = \frac{7.5}{2} = 3.75$
   Since $U(Z) > U(XorY)$, the best action would be to play on the identified Z machine.

Each of the aboe cases have equal probability of happening. Therefore the utility of getting one of the slot machine identified is,
$\implies U = \frac{1}{3} \times 3.925 + \frac{1}{3} \times 4.075 + \frac{1}{3} \times 4.25 = \frac{12.25}{3}$

Difference in Utility $= U - U(XorYorZ) = \frac{12.25}{3} - \frac{11.75}{3} = \frac{0.5}{3} = 0.166667$
Therefore, we should be willing to pay 0.166667$ to get the identification done.

Question 3

$Reward(s) =$

	50	
	0	-3
-50	-1	-10
	-3	-2

$\gamma = 0.8 \quad Probability(a) = \{0.7, 0.15, 0.15\}$

(1)

$InitialState =$

	50	
	4	8
-50	7	4
	9	5

$FinalState =$

	50	
	33.87	17.3
-50	11.43	-0.38
	2.70	-1.18

$InitialPolicy =$

	50	
	↑	↑
-50	→	←
	→	→

$FinalPolicy =$

	50	
	↑	←
-50	↑	↑
	↑	←

$Num\_Iterations = 4$

(2)

$InitialState =$

	50	
	0	0
-50	0	0
	0	0

$FinalState =$

	50	
	33.68	16.72
-50	11.04	-1.38
	1.47	-2.97

$InitialPolicy =$

	50	
	↑	↑
-50	→	←
	↑	↑

$FinalPolicy =$

	50	
	↑	←
-50	↑	↑
	↑	←

$Num\_Iterations = 4$

(3) From (1) and (2) we have two sets of utilities. When we use Bellman's equations, then for every update we have

$$\|U_0 - U_0'\| \geq \gamma \, \|U_1 - U_1'\| \tag{1}$$

In this question, let $\|U_0 - U_0'\| = \sum_{cell}(U_0[cell] - U_0'[cell])$.
Thus the difference in utility at every iteration is [37, 30.04, 17.51, 10.21, 5.20].
Figure 2 shows the plot of $\|U_0 - U_0'\| \ and \ \gamma \, \|U_1 - U_1'\|$

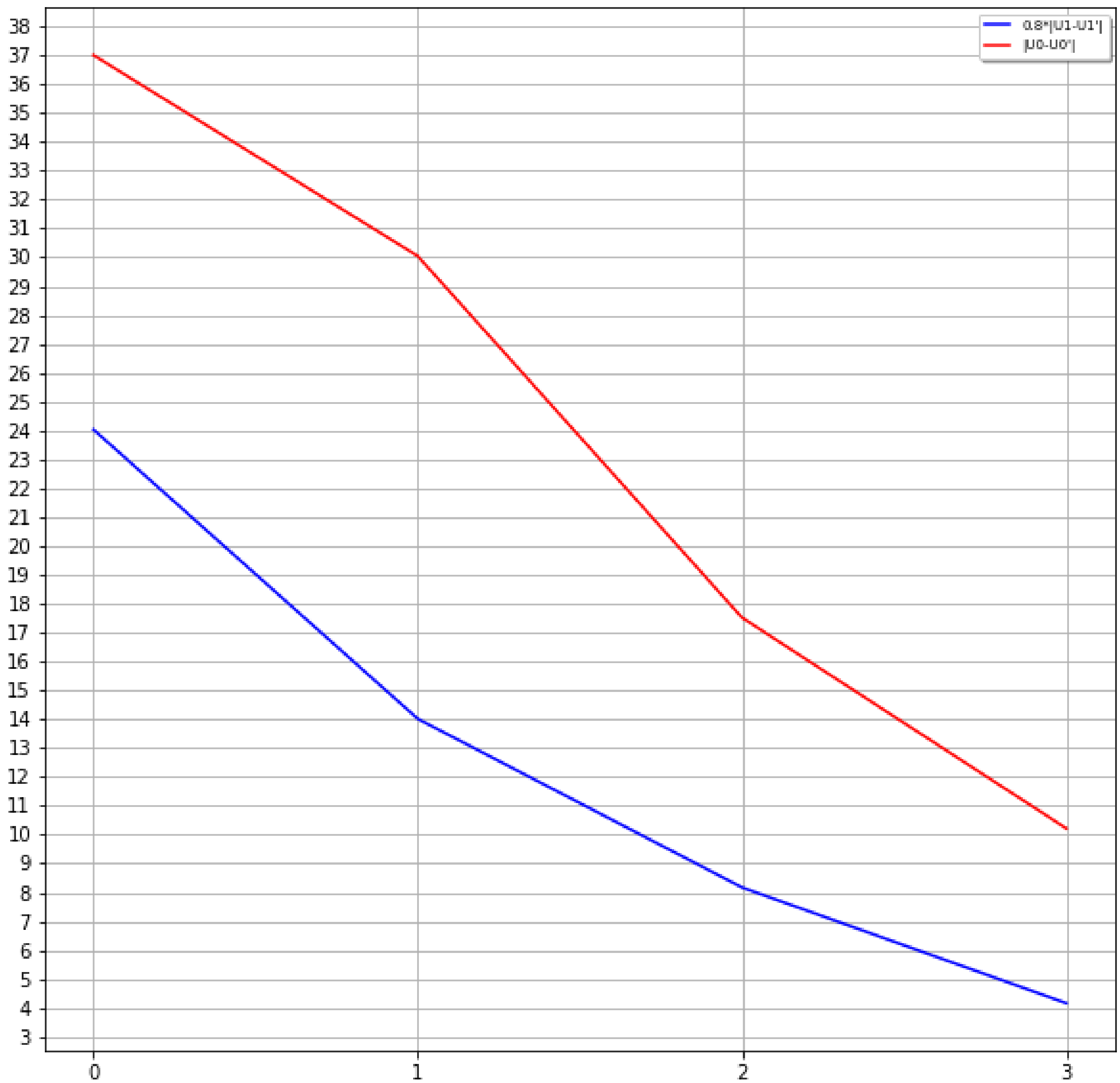

Figure 2: $\|U_0 - U_0'\| \ and \ \gamma \, \|U_1 - U_1'\|$

Question 4

$InitialPolicy =$

	50	
	↑	↑
-50	↑	↑
	↑	↑

$U(0,1) = 50$
$U(1,1) = -1 + 0.8(0.7 \times U(0,1) + 0.15 \times U(1,1) + 0.15 \times U(1,2))$
$U(1,2) = -1 + 0.8(0.7 \times U(1,2) + 0.15 \times U(1,1) + 0.15 \times U(1,2))$
$U(2,0) = -50$
$U(2,1) = -1 + 0.8(0.7 \times U(1,1) + 0.15 \times U(2,0) + 0.15 \times U(2,2))$
$U(2,2) = -1 + 0.8(0.7 \times U(1,2) + 0.15 \times U(2,1) + 0.15 \times U(2,2))$
$U(3,1) = -1 + 0.8(0.7 \times U(2,1) + 0.15 \times U(3,1) + 0.15 \times U(3,2))$
$U(3,2) = -1 + 0.8(0.7 \times U(2,2) + 0.15 \times U(3,1) + 0.15 \times U(3,2))$
Solving this system of linewar equations we get the value of utilities for each cell.

$Utility =$

	50	
	34.38	18.78
-50	12.53	2.29
	4.70	1.03

Then we again get the best policy for these utilities and repeat until the policies converge.

The change in policies and utilities in every iteration are given below.

	50	
	↑	↑
-50	↑	↑
	↑	↑

$\rightarrow$

	50	
	↑	←
-50	↑	←
	↑	←

$\rightarrow$

	50	
	↑	←
-50	↑	↑
	↑	←

	50	
	32.19	2.69
-50	10.30	-8.28
	1.982	-7.27

$\rightarrow$

	50	
	34.31	18.29
-50	12.096	-0.988
	4.336	0.352

$\rightarrow$

	50	
	34.38	18.78
-50	12.53	2.29
	4.70	1.03

$FinalPolicy =$

	50	
	↑	←
-50	↑	↑
	↑	←

$FinalUtility =$

	50	
	34.38	18.78
-50	12.53	2.29
	4.70	1.03

$Num\_Iterations = 3$

The code for the above calulations is shown in the next page.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Apr  8 10:54:39 2019

@author: ghosh128
"""

#MDP
import copy
import numpy as np

ACTIONS = []
ACTIONS.append([-1, 0])
ACTIONS.append([1, 0])
ACTIONS.append([0, -1])
ACTIONS.append([0, 1])

rows = 4
columns = 3
reward = [[None, 50, None],
          [None, 0, -3],
          [-50, -1, -10],
          [None, -3, -2]]
terminal_states = [[0, 1], [2, 0]]
state = [[None, 50, None], [None, 0, 0], [-50, 0, 0], [None, 0, 0]]
policy = [[None, 50, None],
          [None, ACTIONS[0], ACTIONS[0]],
          [-50, ACTIONS[0], ACTIONS[0]],
          [None, ACTIONS[0], ACTIONS[0]]]
probability = [0.7, 0.15, 0.15]
gamma = 0.8
index_list = [[0,1], [1,1], [1,2], [2,0], [2,1], [2,2], [3,1], [3,2]]

def action_list(action):
    if not action[1]:
        return [action, [0, -1], [0, 1]]
    else:
        return [action, [-1, 0], [1, 0]]

def apply_action(action, i, j):
    actions = action_list(action)
    value = 0
    for step,action in enumerate(actions):
        new_i = i+action[0]
        new_j = j+action[1]
        if new_i<0 or new_i>=rows or new_j<0 or new_j>=columns or reward[new_i][new_j] is None:
            new_i = i
            new_j = j
        value += probability[step]*state[new_i][new_j]
    return value

def get_equation(i, j):
    a = np.zeros(len(index_list))
    b = reward[i][j]
    a[index_list.index([i, j])] = 1
    if [i, j] in terminal_states:
        return a, b
    else:
        actions = action_list(policy[i][j])
        for step, action in enumerate(actions):
            new_i = i+action[0]
            new_j = j+action[1]
            if new_i<0 or new_i>=rows or new_j<0 or new_j>=columns or reward[new_i][new_j] is None:
```

```
                new_i = i
                new_j = j
            a[index_list.index([new_i, new_j])] += (-1*gamma*probability[step])
        return a, b

def get_best_action(i,j):
    value_up = apply_action(ACTIONS[0], i, j)
    value_down = apply_action(ACTIONS[1], i, j)
    value_left = apply_action(ACTIONS[2], i, j)
    value_right = apply_action(ACTIONS[3], i, j)
    return np.argmax([value_up, value_down, value_left, value_right])

state_all = []
policy_all = []
iter = 0
count = len(index_list)
while count != 0:
    policy_all.append(copy.deepcopy(policy))
    A = np.zeros(len(index_list))
    B = []
    for index in index_list:
        a, b = get_equation(index[0], index[1])
        A = np.vstack((A, np.reshape(a, (1,-1))))
        B.append(b)
    A = A[1:,:]
    x = np.linalg.solve(A, B)

    for i, index in enumerate(index_list):
        if index not in terminal_states:
            state[index[0]][index[1]] = x[i]

    state_all.append(copy.deepcopy(state))
    new_policy = copy.deepcopy(policy)
    count=0
    for i, index in enumerate(index_list):
        if index not in terminal_states:
            new_policy[index[0]][index[1]] = ACTIONS[get_best_action(index[0], index[1])]
            if new_policy[index[0]][index[1]][0] is not policy[index[0]][index[1]][0] or
            new_policy[index[0]][index[1]][1] is not policy[index[0]][index[1]][1]:
                print(new_policy[index[0]][index[1]], policy[index[0]][index[1]])
                count += 1
    policy = copy.deepcopy(new_policy)
    iter += 1
    print(iter, count)
```

Question 5

Let, $b(s) = \begin{array}{|c|c|}\hline \text{a} & \text{b} \\ \hline \text{c} & \text{d} \\ \hline\end{array}$, $\quad Probability(a) = \{0.7, 0.15, 0.15\} \quad P(e|s) = \begin{array}{|c|c|}\hline 0.3 & 0 \\ \hline 0.9 & 0.2 \\ \hline\end{array} \quad b(s') = \begin{array}{|c|c|}\hline \text{A} & \text{B} \\ \hline \text{C} & \text{D} \\ \hline\end{array}$

$$b'(s') = \alpha P(e|s') \sum P(s'|s,a)b(s)$$

Suppose the chosen action is "LEFT". Then,
$A = 0.3 \times (0.85 \times a + 0.7 \times b + 0.15 \times c + 0 \times d)$
$B = 0 \times (0 \times a + 0.15 \times b + 0 \times c + 0.15 \times d)$
$C = 0.9 \times (0.15 \times a + 0 \times b + 0.85 \times c + 0.7 \times d)$
$D = 0.2 \times (0 \times a + 0.15 \times b + 0 \times c + 0.15 \times d)$

Similarly if the chosen action is "DOWN". Then,
$A = 0.3 \times (0.15 \times a + 0.15 \times b + 0 \times c + 0 \times d)$
$B = 0 \times (0.15 \times a + 0.15 \times b + 0 \times c + 0 \times d)$
$C = 0.9 \times (0.7 \times a + 0 \times b + 0.85 \times c + 0.15 \times d)$
$D = 0.2 \times (0 \times a + 0.7 \times b + 0.15 \times c + 0.85 \times d)$

R(s) $= c - a - 4 \times b - 2 \times d$
The sequence of actions for this problem can be

1 LEFT, DOWN $\Longrightarrow \begin{array}{|c|c|}\hline 0.2 & 0.3 \\ \hline 0 & 0.5 \\ \hline\end{array} \xrightarrow{\text{LEFT}} \begin{array}{|c|c|}\hline 0.237 & 0 \\ \hline 0.7125 & 0.0499 \\ \hline\end{array} \xrightarrow{\text{DOWN}} \begin{array}{|c|c|}\hline 0.0144 & 0 \\ \hline 0.94533 & 0.0403 \\ \hline\end{array} \Longrightarrow$ REWARD $= 0.8504$

2 LEFT, LEFT $\Longrightarrow \begin{array}{|c|c|}\hline 0.2 & 0.3 \\ \hline 0 & 0.5 \\ \hline\end{array} \xrightarrow{\text{LEFT}} \begin{array}{|c|c|}\hline 0.237 & 0 \\ \hline 0.7125 & 0.0499 \\ \hline\end{array} \xrightarrow{\text{LEFT}} \begin{array}{|c|c|}\hline 0.1318 & 0 \\ \hline 0.866 & 0.0021 \\ \hline\end{array} \Longrightarrow$ REWARD $= 0.7299$

3 DOWN, LEFT $\Longrightarrow \begin{array}{|c|c|}\hline 0.2 & 0.3 \\ \hline 0 & 0.5 \\ \hline\end{array} \xrightarrow{\text{DOWN}} \begin{array}{|c|c|}\hline 0.065 & 0 \\ \hline 0.564 & 0.370 \\ \hline\end{array} \xrightarrow{\text{LEFT}} \begin{array}{|c|c|}\hline 0.0579 & 0 \\ \hline 0.926 & 0.0152 \\ \hline\end{array} \Longrightarrow$ REWARD $= 0.838$

4 DOWN, DOWN $\Longrightarrow \begin{array}{|c|c|}\hline 0.2 & 0.3 \\ \hline 0 & 0.5 \\ \hline\end{array} \xrightarrow{\text{DOWN}} \begin{array}{|c|c|}\hline 0.0656 & 0 \\ \hline 0.564 & 0.37 \\ \hline\end{array} \xrightarrow{\text{DOWN}} \begin{array}{|c|c|}\hline 0.0048 & 0 \\ \hline 0.86 & 0.13 \\ \hline\end{array} \Longrightarrow$ REWARD $= 0.5946$

Thus, {LEFT, DOWN} is the best sequence of actions that can be taken.