Northeastern University
College of Engineering

EECE 5698 - ST: Reinforcement Learning Instructor: Prof. Mahdi Imani

Project #2

## Problem 1

Consider a maze shown on the second page. This maze consists of several walls that the agent cannot enter and bumps and oils that moving to them have negative rewards. For simplicity, consider an $18 \times 18$ matrix, where each element is associated with one of the following:

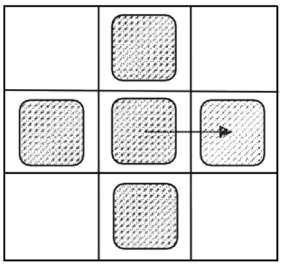
- *Empty*
- *Full (Wall)*
- *Bump*
- *Oil*

**State Space ($S$):** The state-space contains all cells in the maze except the walls, where the agent can possibly be there $(18 \times 18 - 76(walls) = 248)$.

**Action Space (A):** The agent can take one of the four possible actions at any given state: *up (U), down (D), right (R), and left (L).*

**Transition Probabilities:** After choosing an action, the agent will either move to one of the neighborhood cells or stay in its current cell. After taking any action, with a probability of 1-p, the agent moves to the anticipated state and, with an equal probability of p/3, will move to one of the other neighboring cells.

Consider the following example:

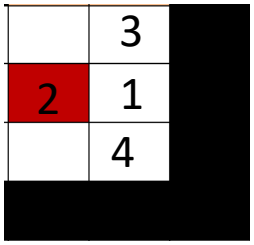


If action *right (R)* is selected, the agent: $\begin{cases} \text{moves to the state in right with probability } 1-\text{p} \\ \text{moves to the state in left with probability p}/3 \\ \text{moves to the the state in up with probability p}/3 \\ \text{moves to the state in down with probability p}/3 \end{cases}$

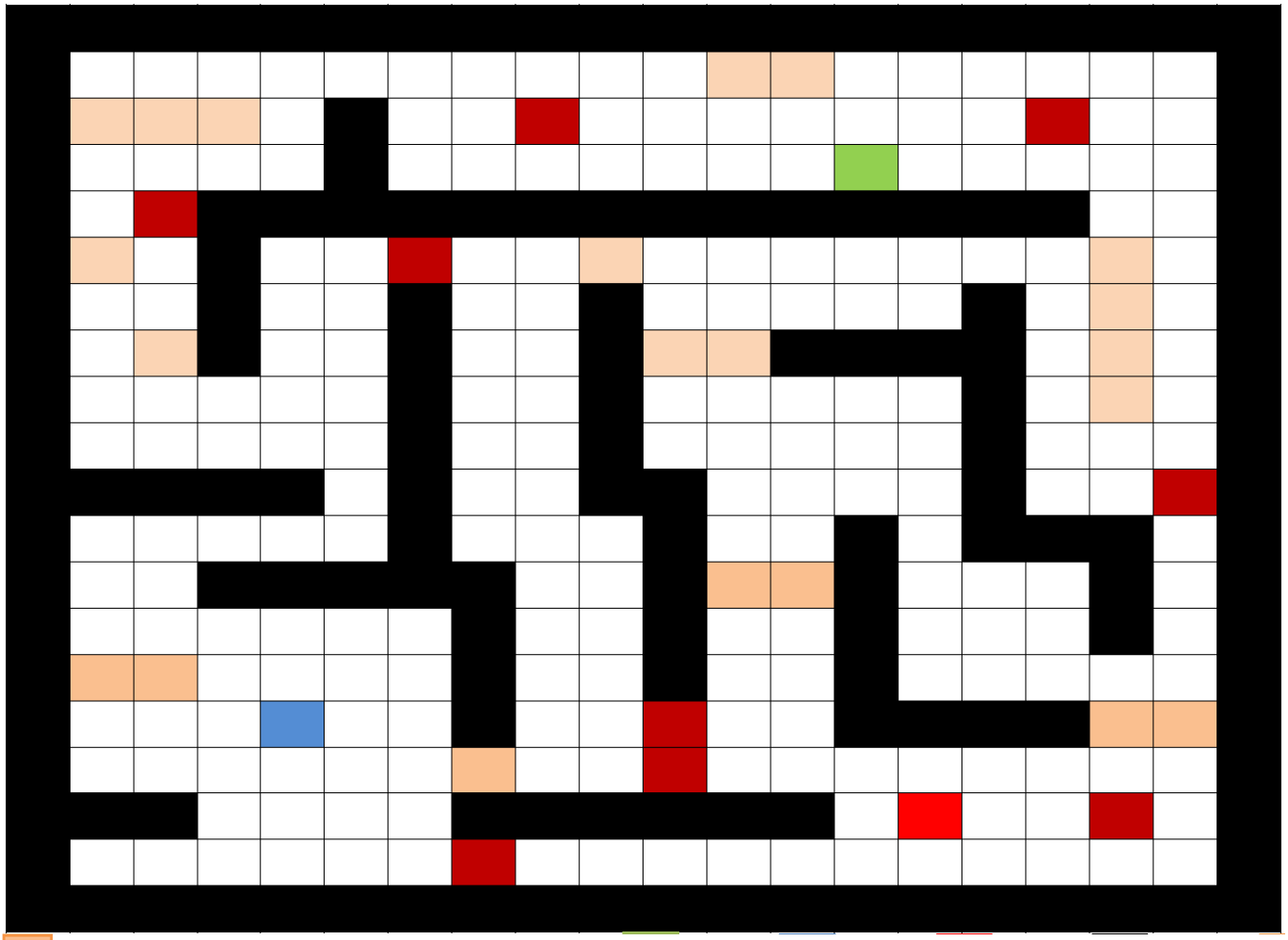
Notice that if any of the neighboring cells are wall, the agent stays in the current cell.

**Reward Function:**

The primary objective is to find the optimal policy which leads the agent to the goal state with the maximum accumulated reward. Therefore, we define the reward of taking any action -1 (delay), the reward for ending up to the oil states -5, the reward for ending up the bump states -10, and the reward for the goal state +200. For instance, this can be expressed for the following example as:



$$\begin{cases} R(1, L, 2) = -6 & -5 \text{ for oil and } -1 \text{ for taking an action} \\ R(1, L, 3) = -1 & \text{for taking an action} \\ R(1, L, 4) = -1 & \text{for taking an action} \\ R(1, L, 1) = -1 & \text{for taking an action} \end{cases}$$



Bump

Wall

Oil

Start

End

1- Use vector-form Policy Iteration technique to find the optimal policy and the optimal state values. Choose action Left in all states as your initial policy for the Policy Iteration method. Solve the problem for the following cases:

a. $p = 0.02,\ \gamma = 0.95,\ \theta = 0.01$ (Base Scenario)
b. $p = 0.5,\ \gamma = 0.95,\ \theta = 0.01$ (Large Stochasticity Scenario)
c. $p = 0.02,\ \gamma = 0.55,\ \theta = 0.01$ (Small Discount Factor Scenario)

Report the optimal value functions, the optimal policy, and the optimal path from start to end. Meanwhile, compare the results obtained in three parts (see the next page for an example of results that need to be provided).

2- Repeat the same process as part 1 with the vector-form Value Iteration method. Set initial state values to zero. Compare the results obtained from Value Iteration and Policy Iteration techniques.

**Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$**

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))\left[r + \gamma V(s')\right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   *policy-stable* $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   *old-action* $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

**Value Iteration, for estimating $\pi \approx \pi_*$**

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$

**Example of Results:** For each combination of method and scenario, you need to provide the following set of plots:

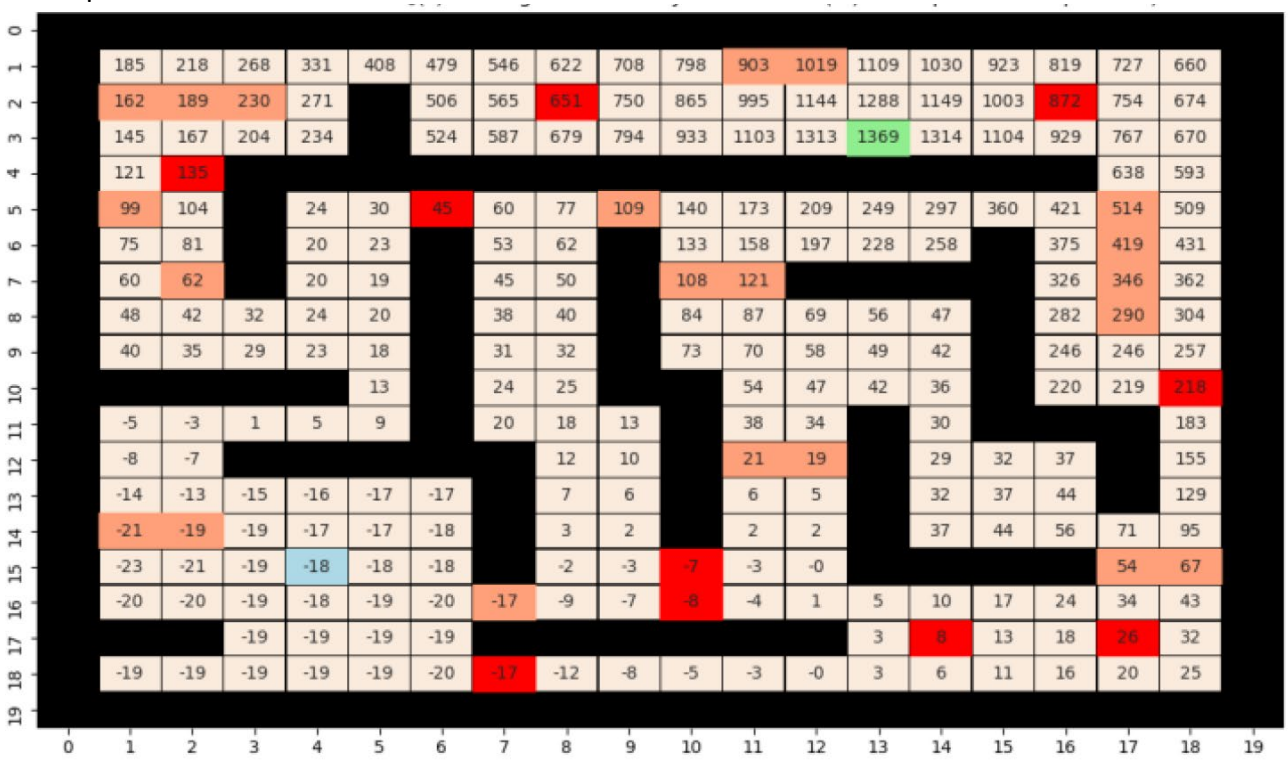



*Figure 1 The optimal value function values at all states for the base scenario under the Policy Iteration* **method.**

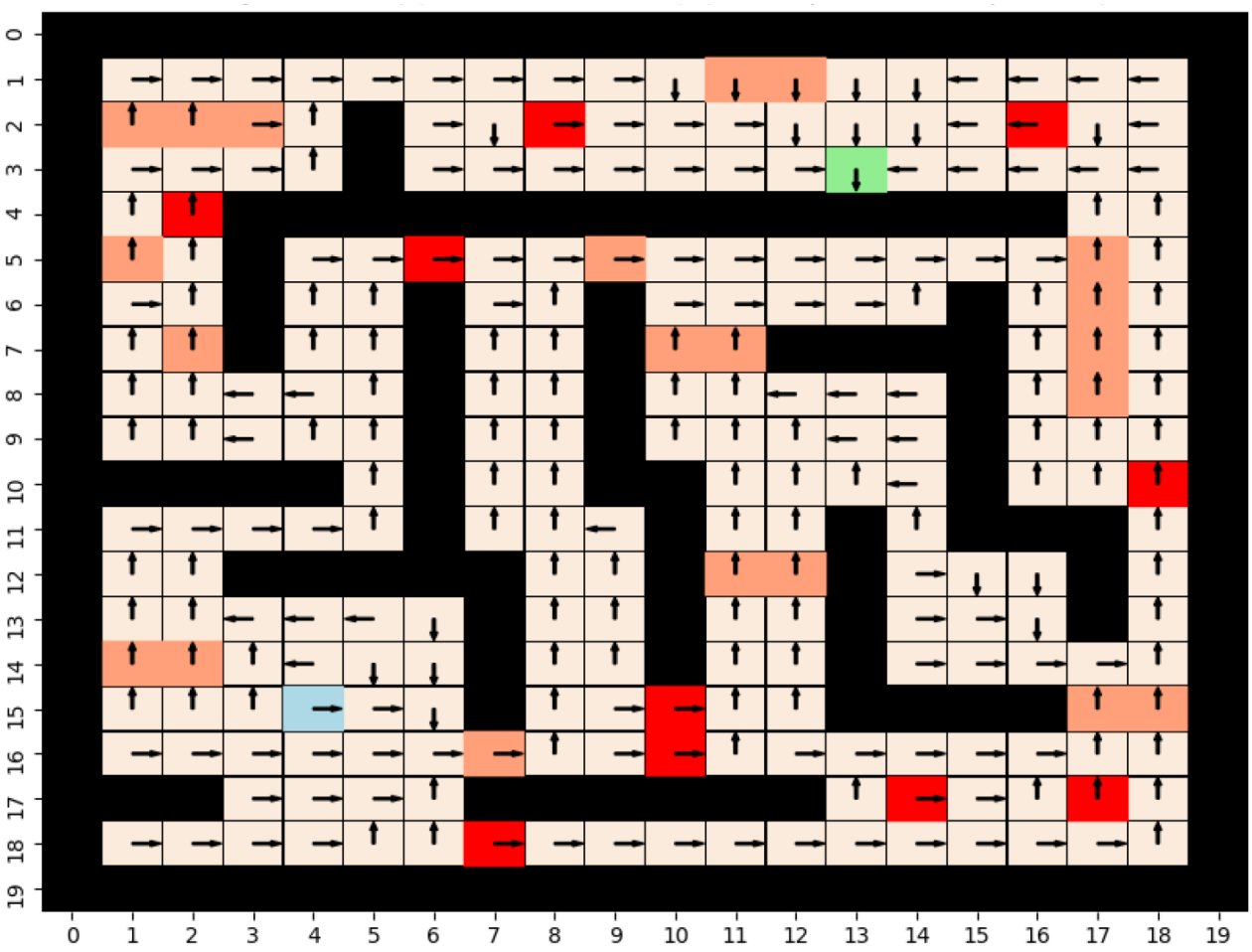



*Figure 2 The optimal policy for all states for the base scenario under the Policy Iteration method.*

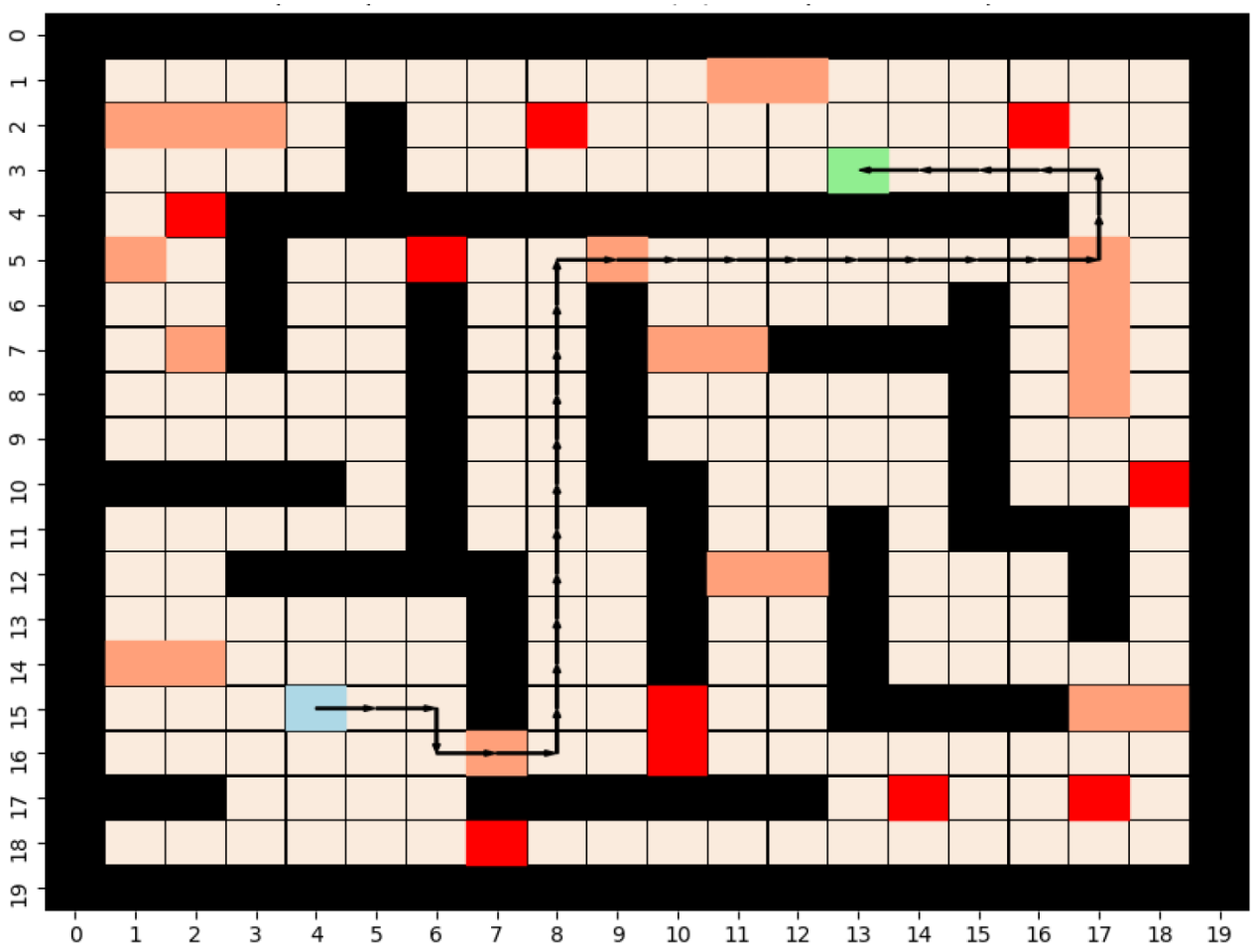



*Figure 3 The optimal path from the start state to the end state for the base scenario under the Policy Iteration method.*

You also need to compare the results obtained in all three scenarios and explain the difference and your thoughts on the obtained results. Please also report the number of iterations of Policy Iteration method (i.e., number of times that policy improvement methods has been performed) and the number of interation of Value Iteration method.

## Problem 2

The left plot of Fig. 1 represents the pathway diagram for a well-known biological network (the p53-Mdm2 negative feedback loop network). The network consists of 4 components/genes, represented in the state vector $\boldsymbol{s}_k$=[ATM, p53, Wpi, MDM2]$^T$.

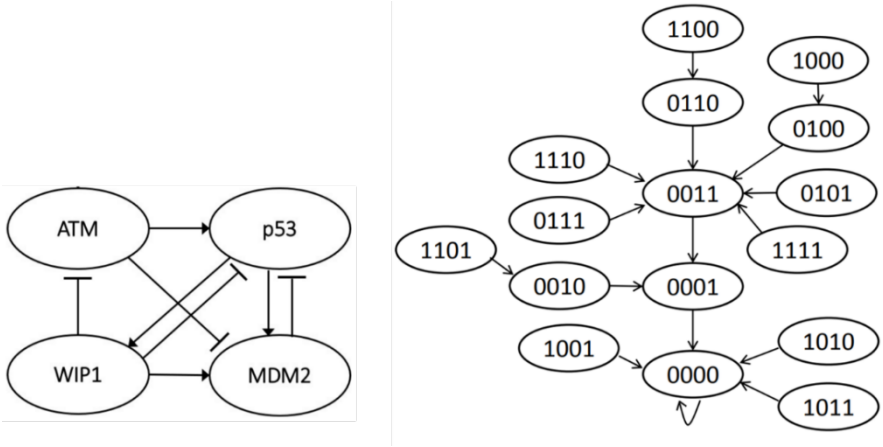



*Figure 1 Pathway diagram for the p53-MDM2 network (left), and state transition diagram (right).*

At any given time, the state value of each gene can be one of the following two values: 0 (OFF) and 1 (ON). Thus, the state space consists of $2^4 = 16$ states described below:
$\boldsymbol{s}^1 = [0,0,0,0]^T, \boldsymbol{s}^2 = [0,0,0,1]^T, \boldsymbol{s}^3 = [0,0,1,0]^T, \boldsymbol{s}^4 = [0,0,1,1]^T$, …, $\boldsymbol{s}^{15} = [1,1,1,0]^T, \boldsymbol{s}^{16} = [1,1,1,1]^T$.
The action space is $A = \{\boldsymbol{a}^1 = [0,0,0,0]^T, \boldsymbol{a}^2 = [1,0,0,0]^T, \boldsymbol{a}^3 = [0,1,0,0]^T, \boldsymbol{a}^4 = [0,0,1,0]^T, \boldsymbol{a}^5 = [0,0,0,1]^T\}$, meaning that a single action at any time should be selected frm the action space, i.e., $\boldsymbol{a}_{k-1} \in A = \{\boldsymbol{a}^1, \boldsymbol{a}^2, \boldsymbol{a}^3, \boldsymbol{a}^4, \boldsymbol{a}^5\}$.
The dynamics (state transition probabilities) of this system is governed by the following state transition:

$$\boldsymbol{s}_k = \overline{\underbrace{\begin{bmatrix} 0 & 0 & -1 & 0 \\ +1 & 0 & -1 & -1 \\ 0 & +1 & 0 & 0 \\ -1 & +1 & +1 & 0 \end{bmatrix}}_{\text{C: Connectivity Matrix}} \boldsymbol{s}_{k-1}} \oplus \boldsymbol{a}_{k-1} \oplus \boldsymbol{n}_k \quad \text{Eq. (1)}$$

where $\overline{\boldsymbol{v}}$ maps the element of vector greater than 0 to 1, and smaller or equal to zero to 0, $\boldsymbol{a}_{k-1}$ is a action input and $\boldsymbol{n}_k$ is the state transition noise. The module-2 addition "$\oplus$" is like a component-wise XOR, such that

$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

We assume that four elements of $\boldsymbol{n}_k$ are identically distributed through a Bernoulli noise with parameter $p$. This can be expressed as: $\boldsymbol{n}_k(j) \sim \begin{cases} 1 & with\ prob\ p \\ 0 & with\ prob\ 1-p \end{cases}$, for $j = 1, \ldots, 4$.
The transition matrices for this system, which represent moving from any state to other states under different control inputs, are $16 \times 16$ matrices. The elements in the $i$th row and $j$th column of this matrix associated with action $\boldsymbol{a}$ can be represented by

$$\big(M(\boldsymbol{a})\big)_{ij} = p^{||\boldsymbol{s}^j - (\overline{C\boldsymbol{s}^i} \oplus \boldsymbol{a})||_1} \times (1-p)^{4-||\boldsymbol{s}^j - (\overline{C\boldsymbol{s}^i} \oplus \boldsymbol{a})||_1}$$

where again $\big|\big|\boldsymbol{v}\big|\big|_1 = \sum_i |\boldsymbol{v}(\boldsymbol{i})|$ sums the absolute value of the elements of the vector $\boldsymbol{v}$.

For the system under no control, $\boldsymbol{a} = [0,0,0,0]^T$, the state transition diagram (without noise) is shown in the right plot of Fig. 1. It can be seen that the system without any process noise (deterministic case) ends up at state "0000" after a few transitions, meaning that the system spends most of its time in that state. The objective is to control/perturb the system so that the state values of all components become

1 (ON) at all time steps (i.e., continuing task). In other words, the system is desired to have as many active components as possible (ideally, spend most of the time on the state "1111"). This can be expressed through the following reward function:

$$R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') = 5\, \boldsymbol{s}'^{(1)} + 5\, \boldsymbol{s}'^{(2)} + 5\, \boldsymbol{s}'^{(3)} + 5\boldsymbol{s}'^{(4)} - |\boldsymbol{a}|$$

where $|\boldsymbol{a}|$ sums the absolute value of the elements of the vector $\boldsymbol{a}$. According to this reward function, the activation of each gene has the reward $+5$ and actions $\boldsymbol{a}^2$ to $\boldsymbol{a}^5$ have reward of -1.

**Dynamic Programming:** After constructing the controlled transition matrices, you need to use matrix-form Dynamic Programming with $\gamma = 0.95$.

a) For $p = 0.05$, use matrix-form Value Iteration method for computing the optimal control policy $\pi^*$, which is a vector of size 16, specifying the best action at any given state. Use all-zero initial state values and $\theta = 0.01$. Compare the average activation of genes, $AvgA$ (definition is provided below), under the obtained optimal policy with $AvgA$ under no control policy (taking $\boldsymbol{a}^1$in all states).

b) Repeat part (a) for $p = 0.2$ and $p = 0.45$. What is your observation? Compare the obtained optimal policy and $AvgA$ for two noise parameters with the results of part (a).

c) Similar to part (a), use $p = 0.05$, and action $\boldsymbol{a}^1$ in all states as the initial policy. Perform matrix-form Policy Iteration method for computing the optimal control policy $\pi^*$. Compare the results with part (a). Are the obtained policies the same?

- Please also report the number of iterations of Policy Iteration method (i.e., number of times that policy improvement methods has been performed) and the number of interation of Value Iteration method.

- In addition to the final value function and policy (similar to problem 1), you need to evaluate the performance of the final obtained policy during the test in all parts. In episode $i$, start with a random initial state $\boldsymbol{s}_0 \in \{\boldsymbol{s}^1, \dots, \boldsymbol{s}^{16}\}$; perform the action prescribed by the optimal policy, $\boldsymbol{a}_0 = \pi^*(\boldsymbol{s}_0)$; use the transition matrix associated with $\boldsymbol{a_0}$ to compute $\boldsymbol{s}_1$ ($\boldsymbol{s}_1$ is a sample from $p(\boldsymbol{s}'|\boldsymbol{s}_0, \pi^*(\boldsymbol{s}_0))$). Continue this to get the trajectory $D^i = \{(\boldsymbol{s}_0, \boldsymbol{a}_0 = \pi^*(\boldsymbol{s}_0), \boldsymbol{s}_1, \boldsymbol{a}_1 = \pi^*(\boldsymbol{s}_1), \boldsymbol{s}_2, \dots, \boldsymbol{a}_{199} = \pi^*(\boldsymbol{s}_{199}), \boldsymbol{s}_{200})\}$. The average activation rate in episode $i$ is $A^i = \frac{1}{200}\sum_{k=1}^{200} ||\boldsymbol{s}_k||_1$, where the $0 \le A^i \le 4$, where the closer to 4 corresponds to perfect activation of all four genes at all times, whereas the values close to 1 denotes that a single gene (among 4) has been activated on average.
  Repeating the same process for 100 episodes, the average activation rate can be computed as:

$$AvgA = \frac{1}{100}\sum_{i=1}^{100} A^i$$

**Questions about the project should be directed to TA, Mohammad Alali, at alali.m@northeastern.edu.**

## Value Iteration – Matrix Form

Transition matrices $M(a)$, $a \in A$, Reward $R^a_{ss'}$, Number of states $N$.

- Reward: $R^a_s = \left(M(a) \odot R^a_{ss'}\right) 1_{N\times 1}$, for $a \in A$. ($\odot$: Hadamard Product)
- $V_0 = 0_{N\times 1}$, $k = 0$

Repeat

- Value Iteration Backup: $V_{k+1} = \max\limits_{a\in A} \left\{ R^a_s + \gamma M(a) V_k \right\}$ (max: Row-wise)
- $k = k+1$

Until $\max\limits_{i\in\{1,\dots,N\}} |V_k(i) - V_{k-1}(i)| < \theta$

- optimal state values: $V^* = V_k$
- optimal policy: $\pi^* = \arg\max\limits_{a\in A} R^a_s + \gamma M(a) V^*$ (arg max: Row-wise)

## Policy Iteration: Matrix Form

Initialization: arbitrary policy $\pi$, transition matrices $M(a)$, Reward $R^a_{ss'}$

- Reward: $R^a_s = \left(M(a) \odot R^a_{ss'}\right) 1_{N\times 1}$, for $a \in A$. ($\odot$: Hadamard Product)

Policy Evaluation

- $M(\pi) = \begin{bmatrix} \text{Row 1 of } M(\pi(s)) \\ \vdots \\ \text{Row N of } M(\pi(s)) \end{bmatrix}$, $R^\pi_s = \begin{bmatrix} \text{Element 1 of } R^{\pi(s')}_s \\ \vdots \\ \text{Element N of } R^{\pi(s')}_s \end{bmatrix}$
- $V^\pi = \left(I - \gamma M(\pi)\right)^{-1} R^\pi_s$

Policy Improvement

- $\pi' = \arg\max\limits_{a\in A} R^a_s + \gamma M(a) V^\pi$ (arg max: Row-wise)

If $\pi \neq \pi'$, $\pi = \pi'$ and go back to policy Evaluation step.

Else

- optimal policy: $\pi^* = \pi'$
- optimal state values: $V^* = V^\pi$