Exploratory Project

Pick and Place Robot using Q-Learning Algorithm

under the guidance of

Dr. Lakshmanan Kailasam

by

Jaideep Sharma
20074017

Siddharth Verma
20075086

Sridharan N
20075109

Acknowledgement

We would like to express our sincere gratitude to our respected professor Dr. Lakshmanan Kailasam, who gave us the golden opportunity to do this wonderful exploratory project in Reinforcement Learning and provided support and guidance. The project enabled us to learn many concepts related to major algorithms involved in Reinforcement Learning and simulate a robot using Pybullet.

Abstract

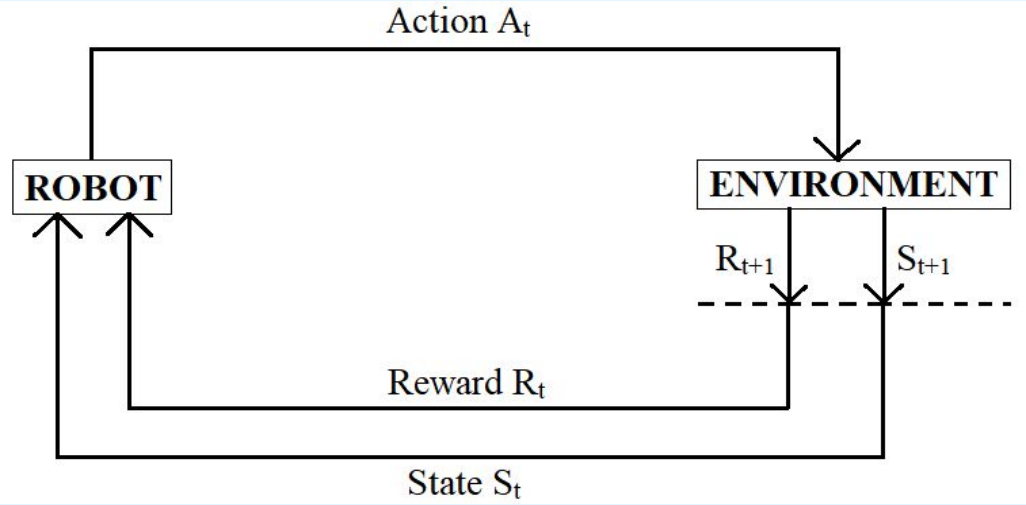
An *n x n* square grid has *m* objects randomly misplaced in its n^2 cells. The objective is to train a robot to pick the misplaced objects and place them in their correct positions in the grid in the minimum possible time, monitoring the robot's battery levels. We implement the Q-learning algorithm of Reinforcement Learning to achieve this objective.

Assumptions

- The initial and final positions of an object do not overlap with those of another object.
- The robot already knows the initial and final positions of each object.

The Agent-Environment Interface

- The decision-maker and learner agent is the pick-and-place robotic arm
- Environment: n*n grid, with objects to be arranged, and the robot's battery.
- The battery level cannot increase unless the robot decides to recharge.



The interaction between the robotic arm and the environment

The Agent-Environment Interface

- The robot and environment interact at discrete time steps, t = 0, 1, 2, 3,. .., and at each time step t, the robot receives representation of the environment's **state**, $S_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of possible states.
- The robot selects an **action** $A_t \in \mathcal{A}(S_t)$, where $\mathcal{A}(S_t)$ is the set of actions available in state S_t.
- At time t+1, the robot receives a reward $R_{t+1} \in \mathfrak{R} \subset \mathbb{R}$, and enters into new state S_{t+1}.
- At each time step, the robot implements a policy, denoted by π_t , where $\pi_t(a|s)$ is the probability that $A_t = a$ if $S_t = s$.

State of the Interface

- At any given time t, the state S_t is depicted by the grid configuration consisting of ‘m-1’ objects located at random positions,
- One of the ‘m’ objects being picked up and carried by the robotic arm, and the robot’s battery level.
- At time t = 0, all objects lie in the incorrect positions, and finally they lie in their correct positions in the ‘m’ squares of the grid.

Initial and Final States of Grid

```
Initial environment :
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 2 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
Final environment :
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
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 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0
0 0 0 0 0 10 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Markov Property

"The future is independent of the past given the present"

- The state captures all relevant information from the history and gives no information about future. Thus, the state satisfies Markov property.
- In general case, we can define the environment's dynamics only by specifying the complete probability distribution $p_{general}$:

$$p_{general}(s', r \mid s, a) = Pr\{R_{t+1} = r, S_{t+1} = s' \mid S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t\} \; \forall$$

reward $r \in \mathbb{R}$, next state s', and all possible values of the past events:

$$S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t.$$

Markov Property

- Since the state satisfies Markov property, then the environment's response at a time t+1 is dependent only on the state and action representations at time t. Thus, we can define the environment's dynamics by specifying only p_{markov} :

  $p_{markov}(s', r \mid s, a) = Pr\{R_{t+1} = r, S_{t+1} = s' \mid S_t , A_t\}$ $\forall$ reward $r \in \mathbb{R}$, next state s', $A_t \in \mathcal{A}(S_t)$, $R_{t+1} \in \mathfrak{R}$.
- For a Markov state, it follows that $p_{general} = p_{markov}$.
- In this case, the environment and task also satisfy the Markov property.

Markov Decision Process (MDP)

- Any task in reinforcement learning fulfilling the Markov property is called a Markov Decision Process (MDP), which is a tuple {S, A, R, T}, where,
- S is the state space
- A is the action space of the agent
- R is the reward function
- T is the transition function

State - Diagrammatic Representation

0	**1**	**2**		
3	**OBJECT X**	**4**		
5	**6**	**7**		
				FINAL

Let X denote the object number from 1 to m among m objects, and FINAL be the correct position of X in the grid.

The next positions where object will be taken to by the robotic arm are labeled from 0 to 7, denoting the next state and stored in position[] array. The Euclidean distances between each labeled cell starting from 0 and the 'FINAL' cell are stored in a distance[] array.

distance[] = {5.000, 4.242, 3.605, 4.472, 2.828, 4.123, 3.162, 2.236}
position[] = {0, 1, 2, 3, 4, 5, 6, 7 }

Action and Policy

- The distance[] is sorted in increasing order:

  distance[] = {2.236, 2.828, 3.162, 3.605, 4.123, 4.242, 4.472, 5.000}

  position[] = {7, 4, 6, 2, 5, 1, 3, 0 }
- Let L be the length of the distance[] array. L denotes the number of possible subsequent states for an object. If the object 'X' is present at the corner of the grid, then L = 3; if it is present at an edge of the grid excluding the corners, then L = 5; if it is not present at any of the edges, then L = 8.
- The **action** performed by the robot is to select a position so that the reward is maximized. The better the quality of the action, the greater the reward will be.
- The method implemented for choosing actions in subsequent states is the robot's **policy.**

Action and Policy

- Let us define three arrays storing the index of the selected position in the position[] array as-

index8[] = {8} , used if L = 8

index5[] = {5} , used if L = 5

index3[] = {3} , used if L = 3

- Initially, the robot appends a random index from 0 to L - 1 until there are at least ‘x’ unique elements present in the corresponding index array.

Action and Policy

- The learning rate ‘α’ of the robot influences the value of ‘x’.
- With increasing values of ‘α’, ‘x’ decreases, so robot learns faster to choose better indices to obtain a greater reward. ‘x’ is a function of ‘α’ is defined as -

$$x = 1/(10 * \alpha) + 1 \text{ , if } L = 8$$

$$x = 1/(20 * \alpha) + 1 \text{ , if } L = 5$$

$$x = 1/(30 * \alpha) + 1 \text{ , if } L = 3$$

- Once there are at least ‘x’ unique elements present in the array of selected indices, the robot has learned which index leads to a better state.
- The index arrays contain chosen indices from 0 to t-1.
- The robot then selects a random index from 0 to min - 1 in a state of ‘L’ following possible states. (‘min’ is the minimum element in index array)

Action and Policy

- The battery level required to safely place an object from the current position to its final position is ⌈distance[C]⌉ + d, where

  ⌈.⌉ denotes the ceil function

  ‘C’ is the chosen index of position array

  d is discharge rate

- Robot needs negligible battery to reach an object for pick up.

Action and Policy

- If battery < required and warnings < y, then reward decreases by 1 and warning is given.
- If battery becomes 0%, then the picked-up object is dropped and reward decreases by 4, with another warning being given.
- The robot must be manually recharged to resume the task in the latter case. Here, the value of 'y' depends on the learning rate 'α' as -

$$y = 1/(10 * \alpha)$$

- If battery < required and warnings >= required, then the robot recharges itself up to the necessary level instead of complete recharging, thereby saving the time by resuming the task as fast as possible, and the reward increases by 1.
- The higher the learning rate, the lower the number of warnings required by the robot to maintain sufficient battery levels.

Reward Function

- Let 'R_B' be the reward obtained due to battery level 'B' in state S_t. The reward is calculated by the function -

$$\text{reward} = (L - C)/L + R_B \text{ , where } R_B \in \{-5, -1, 0, 1\}$$

- Thus, the lesser the chosen index and more suitable the battery, the greater will be the reward.

Reward Function - Example

Let C = 2, d = 3, and B = 6

The object ‘X’ will be taken to position labeled ‘6’, because position[C] = 6 in the modified array.

- distance[C] = 3.162 and the required battery level is ⌈distance[C]⌉ + d = 4 + 3, i.e., 7.
- Thus, B < 7, L = 8, and if the warnings given are less than ‘y’, then R_B= -1. So,

  reward = (8 − 2)/8 + (− 1) ⇒ reward = 3/4 − 1 ⇒ reward = −0.25
- If the warnings have reached the value ‘y’, the robot will recharge by one level, making B = 7 and R_B = 1. So,

  reward = (8 − 2)/8 + 1 ⇒ reward = 3/4 + 1 ⇒ reward = 1.75

Transition Function

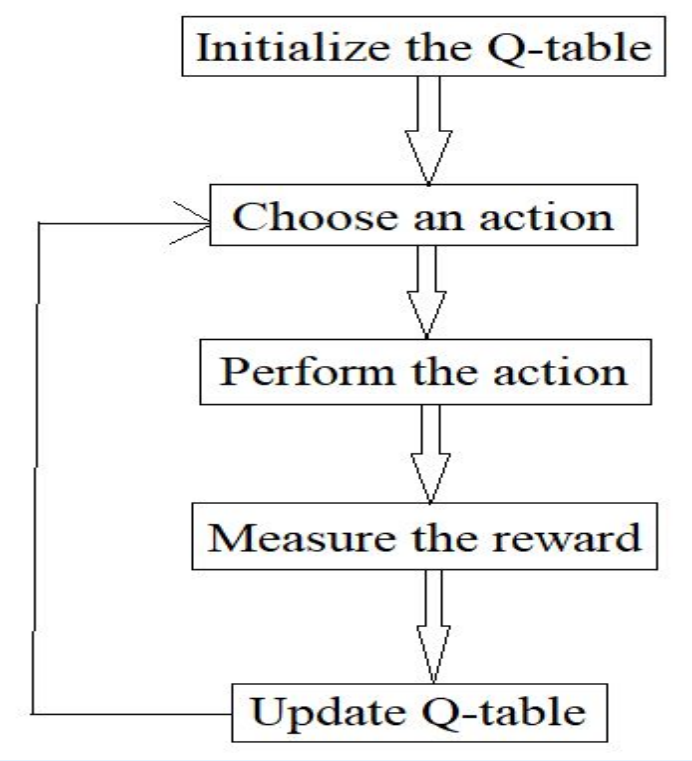
0	**1**	**2**		
3	**OBJECT X**	**4**		
5	**6**	**7**		
				FINAL

0'	**1'**	**2'**		
3'	**OBJECT X**	**4'**		
5'	**6'**	**7'**		**FINAL**

State S_t → State S_{t+1}

Transition is the movement of a picked-up object from one position in state S_t to the next position in state S_{t+1}. If the index chosen in state S_t was 2, then the state transition occurs as shown above, since position[2] = 6.

Q-learning Algorithm



Sequence of steps in Q-learning

- The Q-learning is an exemplary reinforcement learning algorithm that aims to determine the best action to take, given the current state of the problem.
- It intends to learn a policy that maximizes the total reward received.

Computation of Q-values

- The Q-value of state-action pair denoted by Q(s,a) is calculated as -

  $Q(s,a) = (1 - \alpha)*Q(s,a) + \alpha*[R(s, a) + \gamma*\max Q(s', a')]$, $0 \leq \alpha, \gamma \leq 1$

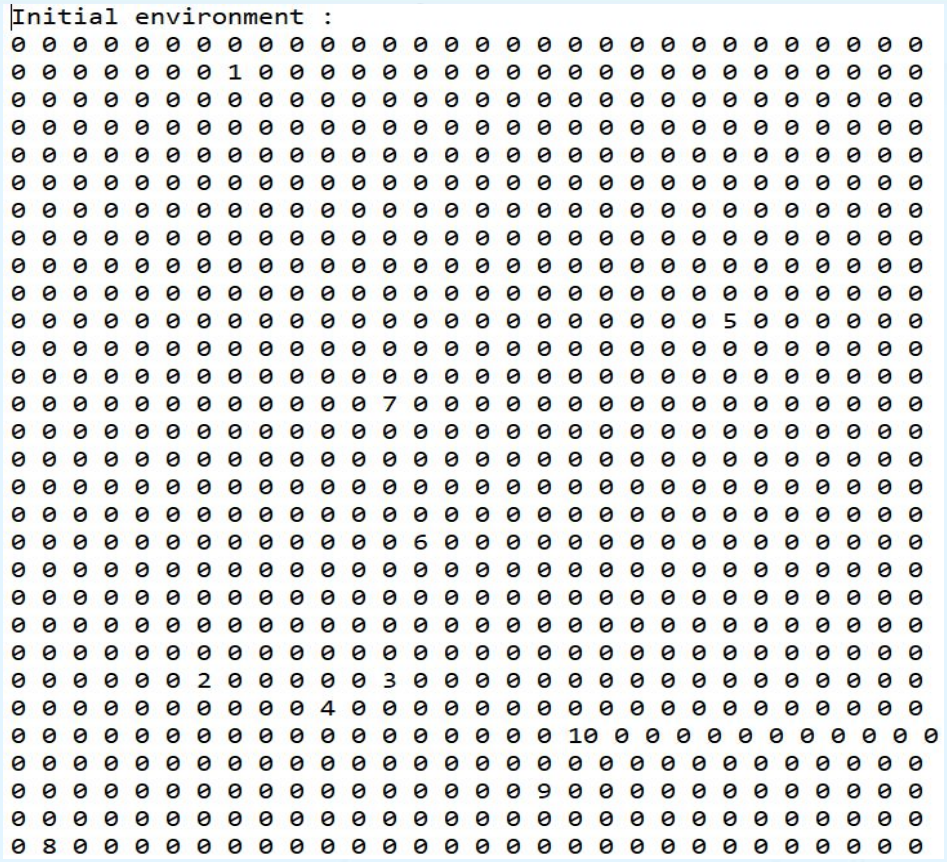
  where α is the learning rate, R(s,a) is the immediate reward, Q(s,a) is the current Q-value, Q(s',a') is the maximum Q-value in next state, and γ is the discount rate.
- We initialize the Q-values for all objects to zero and observe that the maximum Q-value in the next state is given as -

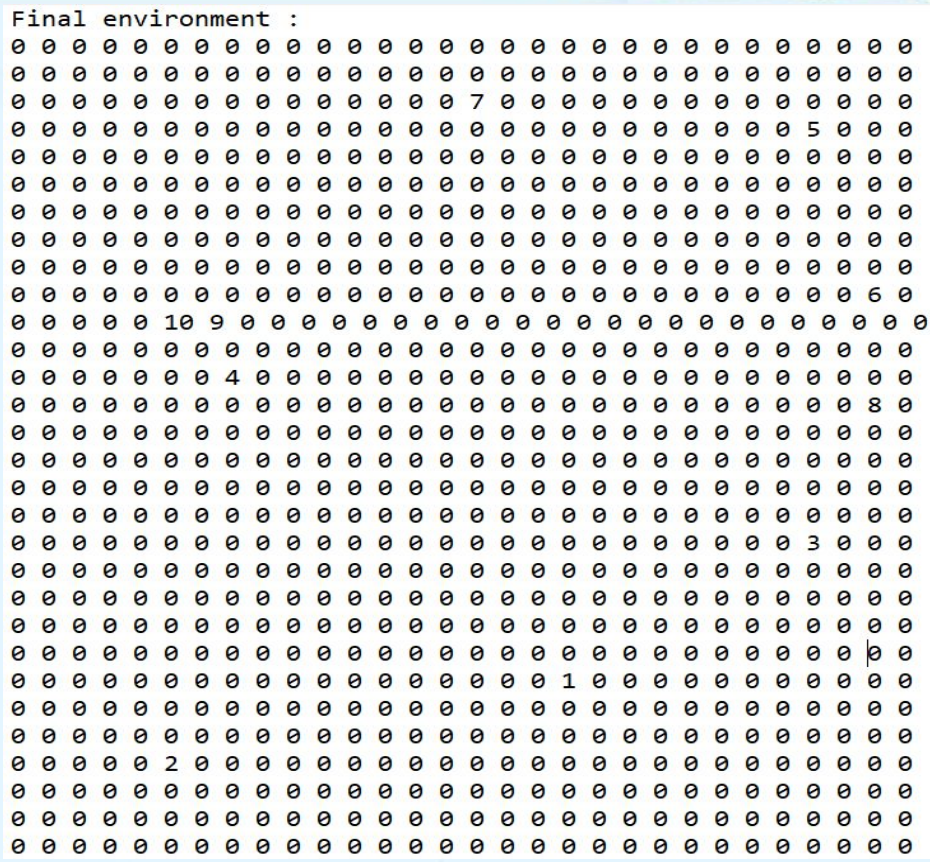
  $\max Q(s',a') = [Q(s,a) - R(s,a)] / \gamma$
- Hence, to maximize max Q(s', a'), R(s, a) must be minimized, which will occur on the maximum possible value of index 'C'. The worst possible reward due to the battery level will occur at the poorest index chosen.

The Experiment

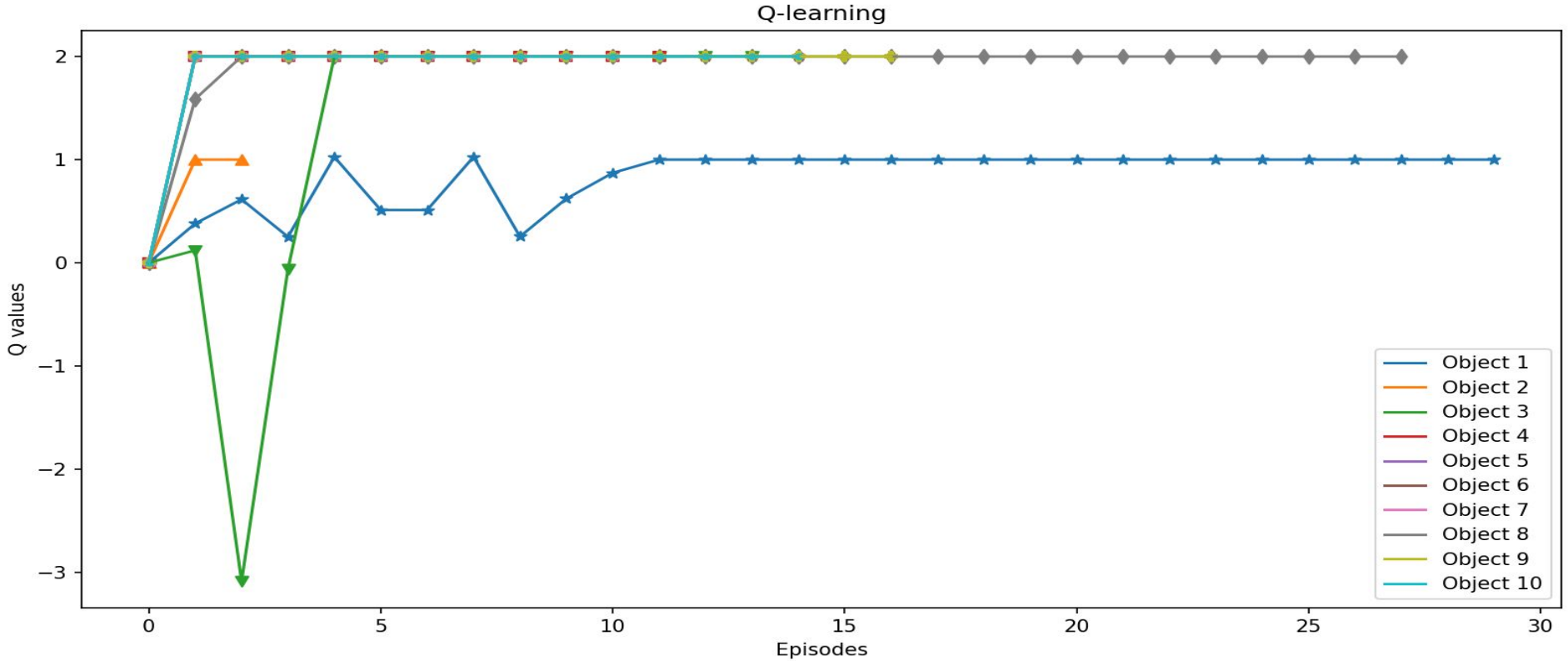
- An experiment of the pick-and-place task of the robotic arm was performed keeping m=10, n=30, d=3, α = 0.03, and γ = 0.5.
- Thus, the number of objects in the 30*30 grid is 10, the battery's discharge rate is 3%, the learning rate is 0.03, and the discount-rate parameter is 0.5.

Initial and Final States of Grid



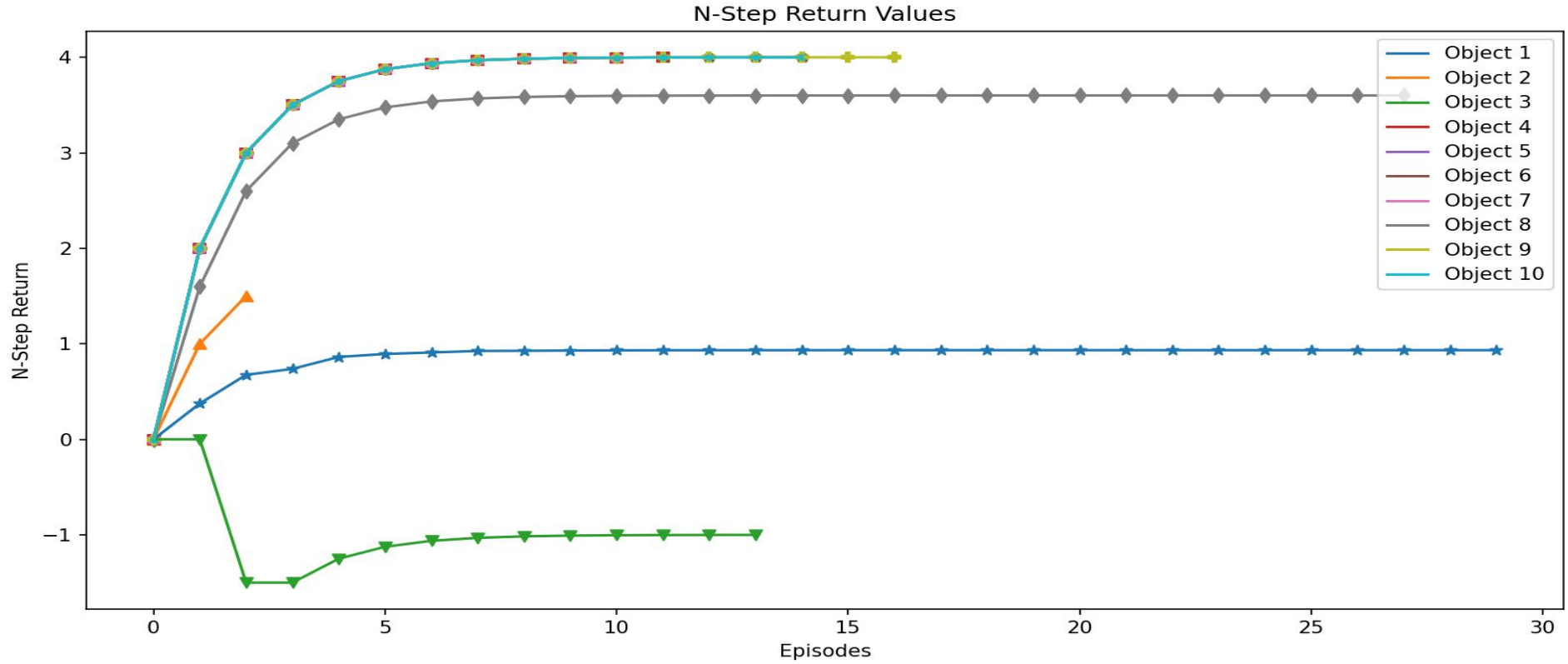


Q-values



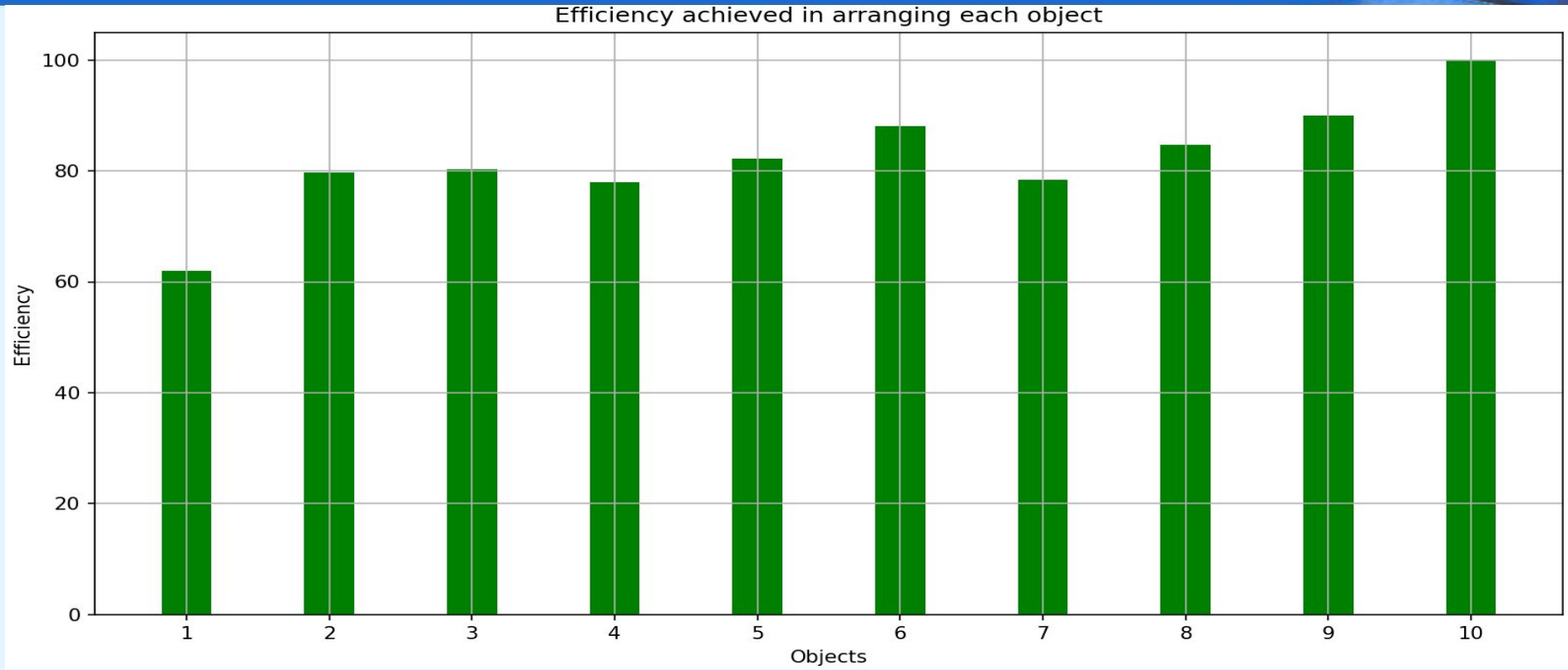
Q-values for placing 10 objects to their places in successive episodes

N-step Returns



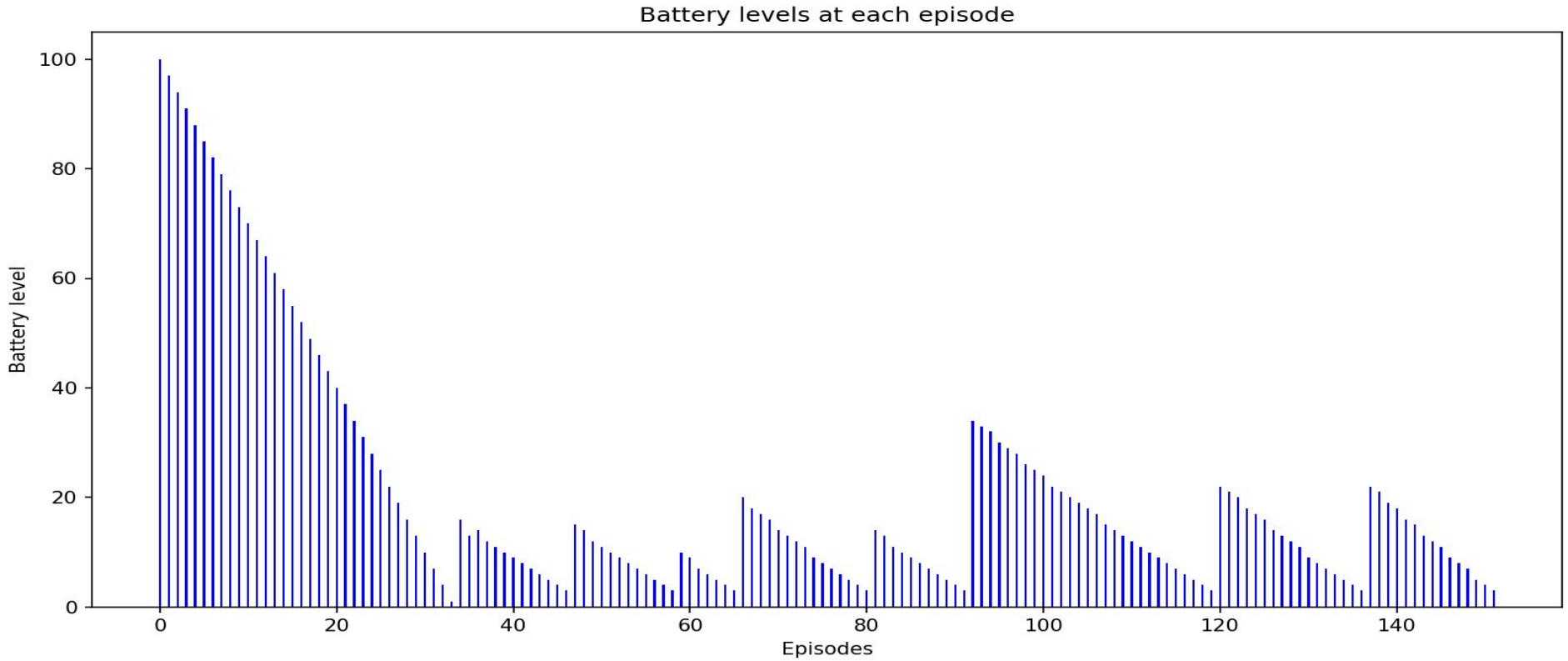
N-step return values for placing each object

Percentage Efficiency



Percentage efficiency of placing each object

Battery Levels



Battery levels at various episodes during the whole task

Definitions and Computations

- The N-step return G_t is the sum of discounted rewards at time step t, given by: $G_t = \Sigma \gamma^k R_{t+k}$, k ranges from 0 to N - 1.
- The percentage efficiency is the distance between the initial and final positions of an object divided by the number of steps taken by the robot to arrange it. The object arranged with highest efficiency is given a percentile 100, with relative evaluation for others.
- The battery level is initially kept at 100% and a discharge rate ‘d’ is decided. The battery decreases by this rate at each action.

Comparison of Graphical Outcomes with Different Learning Rates

In the previous experiment, we kept the learning rate α at 0.03.

Now, let us observe the behavior of our pick-and-place robotic arm

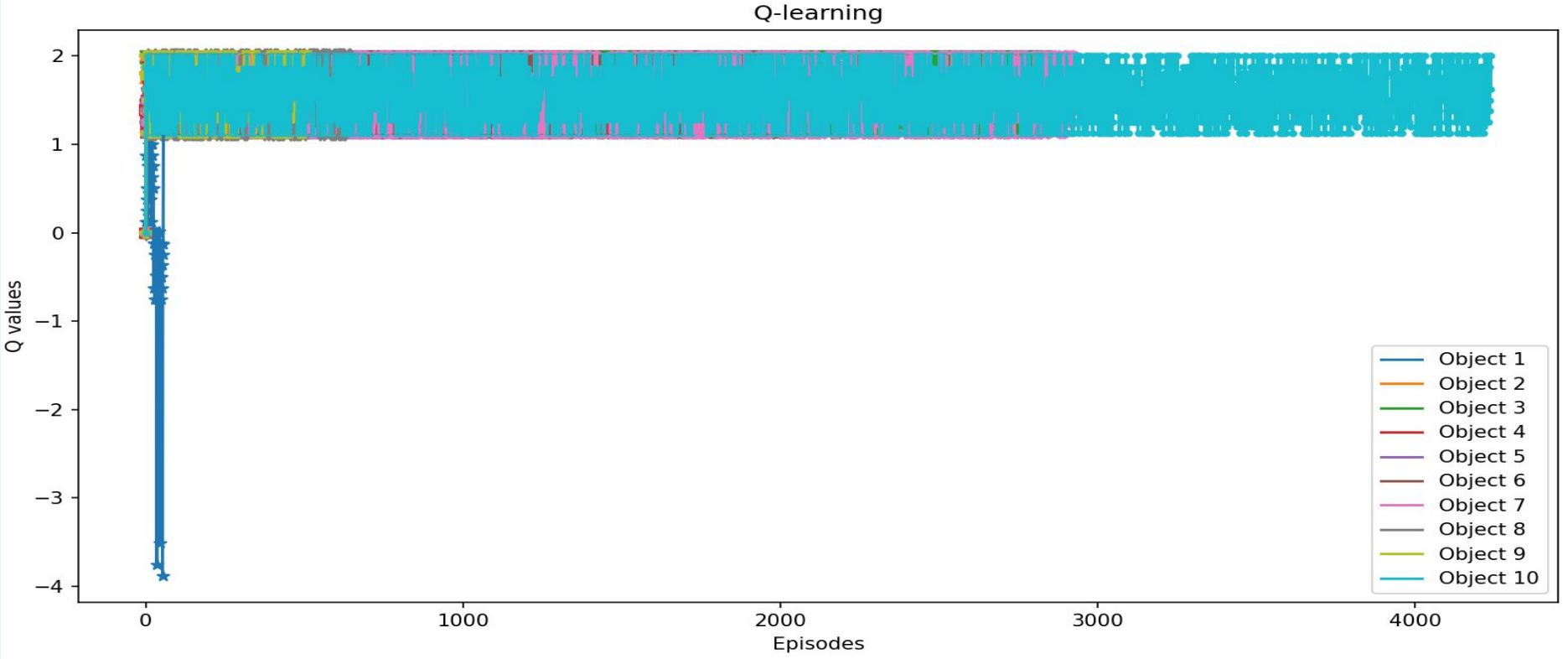
with a learning rate $\alpha' = (1/10)*\alpha$, i.e., $\alpha' = 0.003$.

Initial and Final States of Grid

```
Initial environment :
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0
0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 9 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

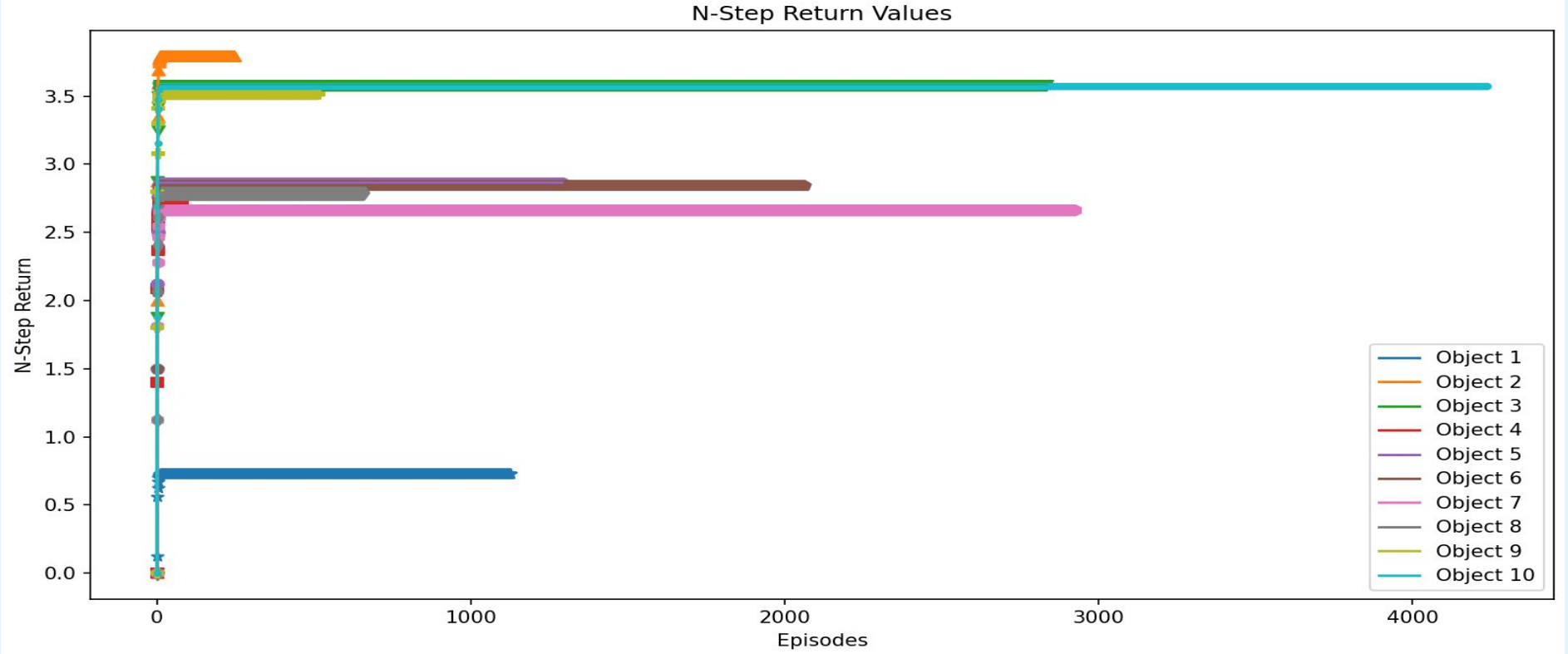
```
Final environment :
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0
0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Q-values



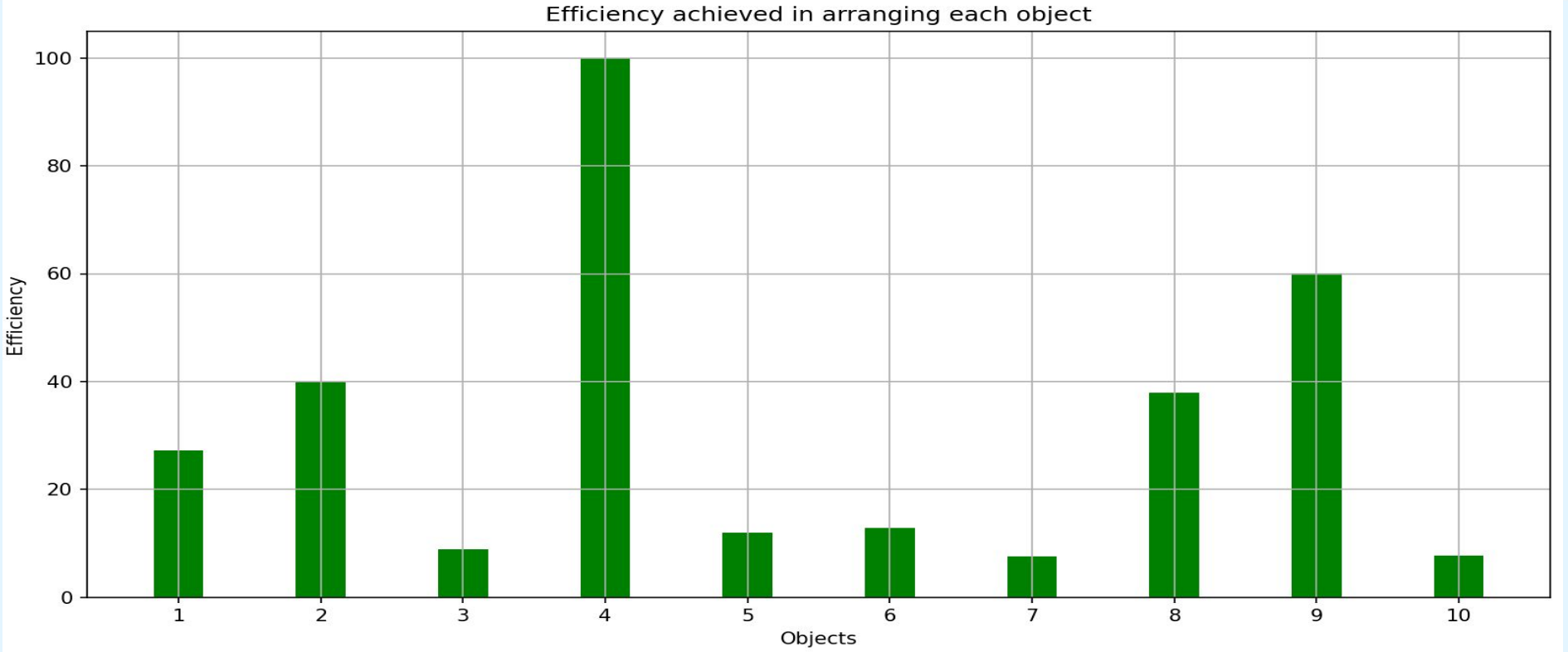
Q-values for placing 10 objects to their places in successive episodes

N-step Returns



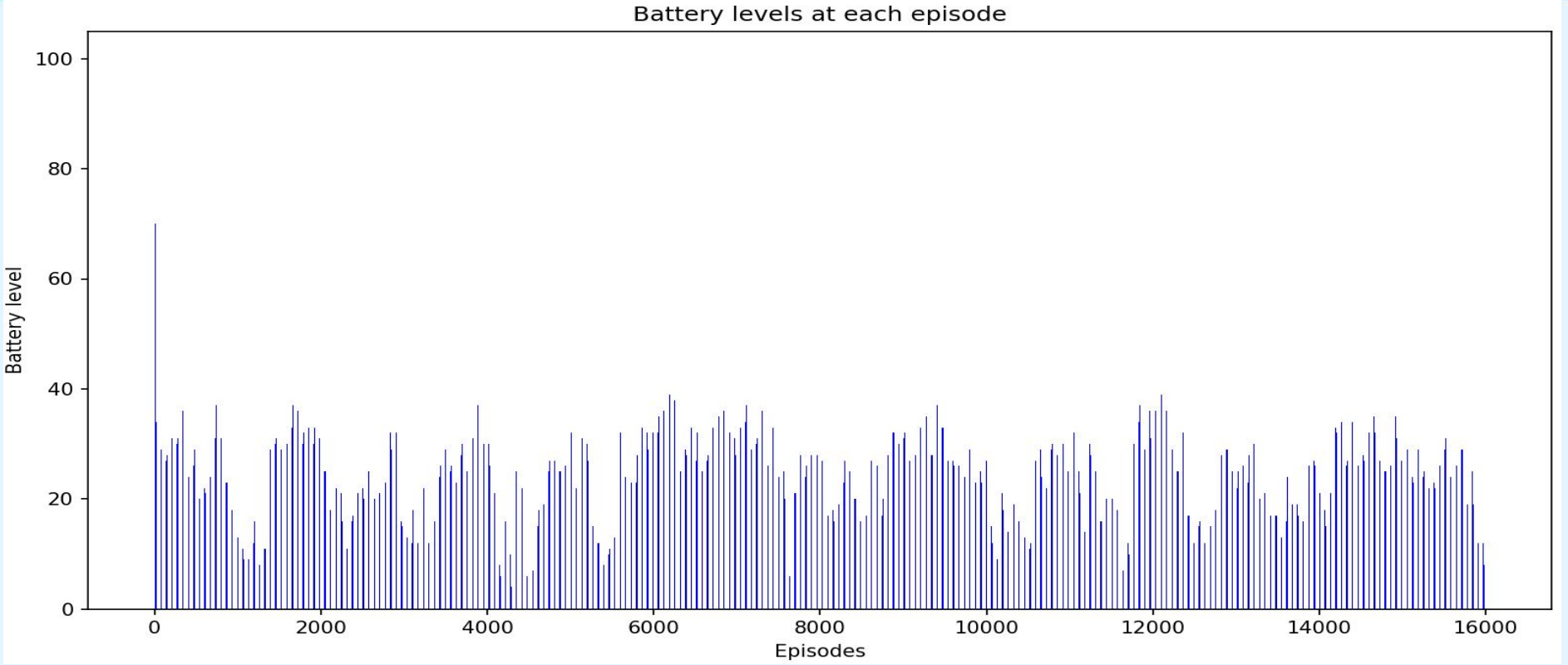
N-step return values for placing each object

Percentage Efficiency



Percentage efficiency of placing each object

Battery Levels



Battery levels at various episodes during the whole task

Experiments Performed with Different Learning Rates

Learning rate α	Number of episodes to finish the task	Average efficiency (in percentage)
0.9	105	83.87
0.5	119	81.81
0.1	139	81.84
0.05	159	85.84
0.01	14932	23.13
0.005	25749	23.64
0.001	28229	21.79
0.0001	37200	16.17

Expected Value of Efficiency

- The average efficiency for a particular learning rate varies slightly on executing the same task with different initial and final configurations of the grid. Therefore, we compute the average efficiency more accurately for a task with a defined learning rate using the Q-learning algorithm.
- We deduce from the Law of Large Numbers that by completing the task of our experiment a large number of times, we can get the expected value of efficiency.
- We define the values required in the experiment as -

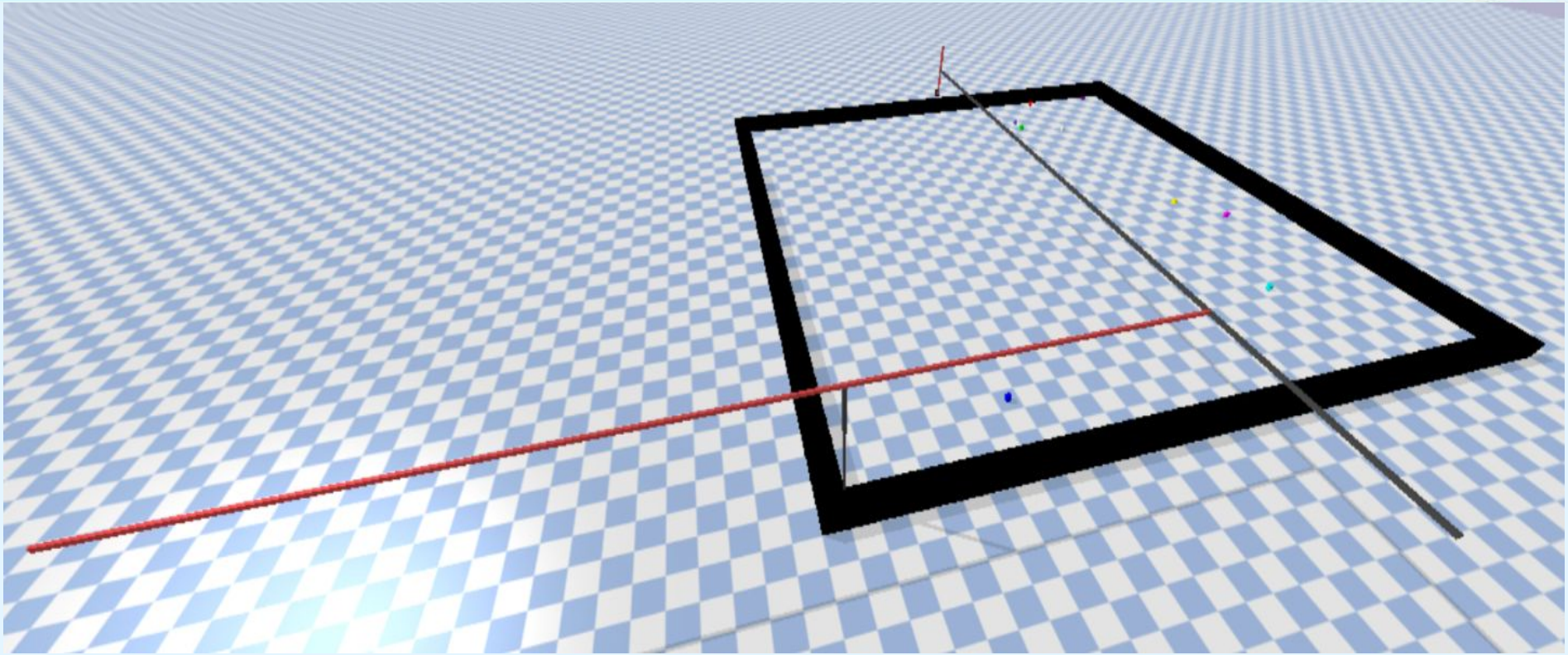
  m=10, n=30, d=3, α = 0.03 and γ = 0.5.
- On executing the task 100,000 times, the average efficiency converges to its expected value at nearly 81.10%.

Expected Values of Efficiency for Different Learning Rates

Learning rate α	Average efficiency (in percentage)
1	82.93
0.5	82.91
0.1	82.90
0.05	82.46
0.04	81.66
0.02	80.19

The maximum achievable expected efficiency occurs at the learning rates greater than or equal to 0.1, i.e., nearly 83%.

Simulation through Pybullet



Pick and place robotic arm

Pybullet

- PyBullet is a OpenGL based python module
- It helps facilitating simulation to mimic the physics involved in the real world.
- It provides forward and inverse kinematics, inverse dynamics computation, forward dynamics simulation, and collision detection.
- We connect to the GUI Physics server and set the gravity to -10 in the vertical direction.

The Universal Robot Description File (URDF)

- The Universal Robot Description File (URDF) is a file containing all the details regarding the structure of a robot or any particular shape, which includes the orientation and location of the links and the joints connecting any two links, the geometry, color, and collision, visual and inertial properties of the links.
- Among any two links attached by a joint, one link is called the parent link, and another is the child link. We can load a robot or a shape into the simulation space by sending commands to the physics server, which uses the format specified in the URDF file concerned.

The Universal Robot Description File (URDF)

The URDF files of some robots or shapes are included in the PyBullet module by default, whereas others have to be constructed as per requirement. In this simulation, we have used four different URDFs for these objects -

1. The Pick and Place Robot
2. Ten objects denoted by different colored small cuboids.
3. Black-colored large cubes marking the boundary of the 30*30 grid.
4. The plane on which all objects, including the robot, are placed.

The URDF of Pick and Place Robot

We construct the URDF for the robot as elucidated in the following points -

- Six links denote the robot's parts connected by five prismatic joints. A prismatic joint facilitates a sliding movement between two links, and the moving link has one degree of freedom, i.e., it can move only along one axis specified.
- The base or the first link remains fixed and perpendicular to the plane. It is connected to the second link.
- The second link is perpendicular to the base link and can move horizontally to the plane, along one pair of opposite sides of the grid, covering its entire length (or breadth). It is connected to the third link.
- The third link is perpendicular to the second link and can move horizontally to the plane, along another pair of opposite sides of the grid, covering its entire breadth (or length). It is connected to the fourth link.

The URDF of Pick and Place Robot

- The fourth link is perpendicular to the third link and can move vertically to the plane. It is connected to the fifth link.
- The fifth link is perpendicular to the fourth link and can move horizontally to the plane. It is connected to the sixth link.
- The sixth link is perpendicular to the fifth link and can move vertically to the plane. It functions as a lock to rigidly hold the picked-up objects.

The URDFs of Objects in the Grid

The objects are differentiated from each other by their color. We construct the URDFs of the ten objects to be arranged by the robot as described in the following points -

- Four cuboidal links constitute an object connected by three fixed joints. A fixed joint does not permit any movement between two links, i.e., each link has zero degrees of freedom.
- The base link lies horizontal to the plane. It is connected to the left and right links.
- The left and right links lie horizontally on the base link and are at some distance apart, thus leaving a tunnel-shaped hole across the object, passing through its center of mass. The right link is connected to the top link, which lies horizontally to the plane, on the left and right links.
- The size of the objects is less than that of the square so that they can be clearly identifiable when placed at the center of a square.

Reaching the Given Coordinates

The robot must first reach the coordinates of an object in the grid to pick or place it. The robot uses the hole present across each object to pick it. Since this tunnel-shaped hole lies horizontal to the plane, the robot maintains a small distance from the object's center of mass upon reaching the square containing it.

Reaching the coordinates is effectively carried out by the functions defined in PyBullet to control the target positions, maximum velocity, force, and damping oscillations of various joints between the two links they connect.

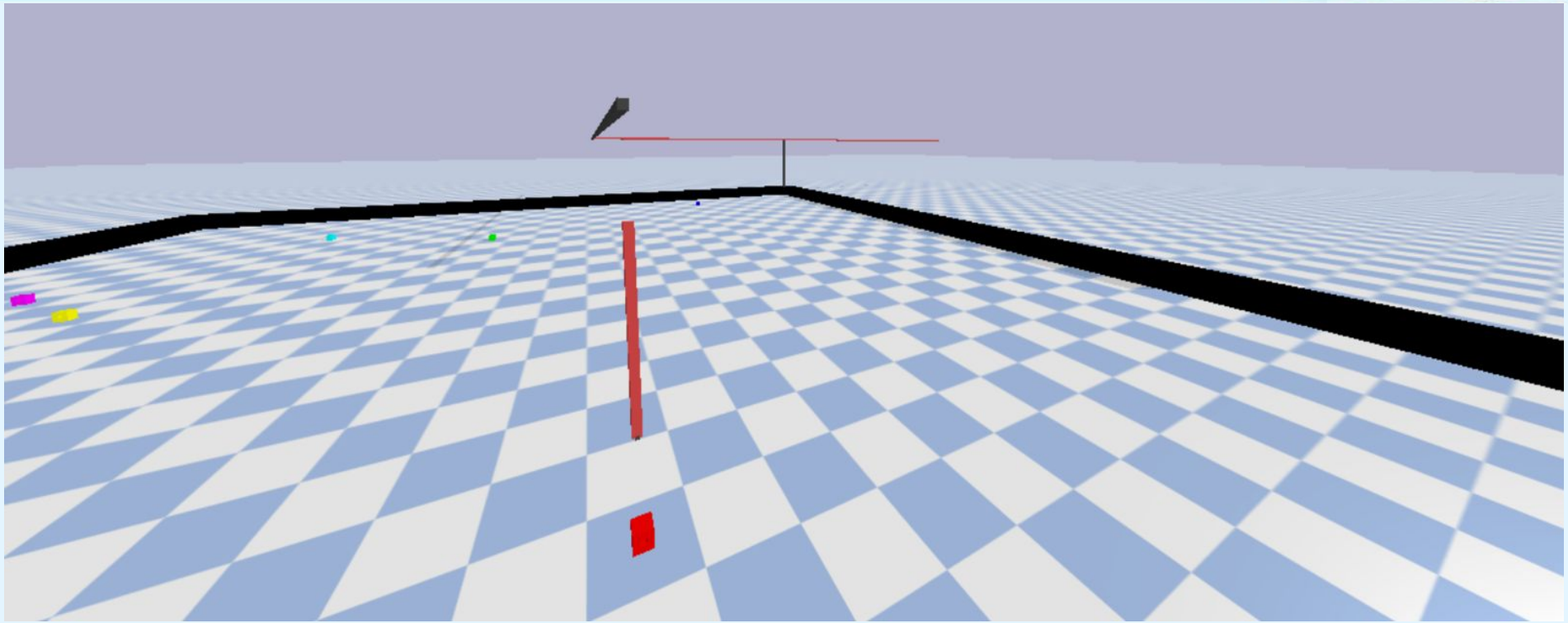
The second and third links attempt to reach the object's 'x' and 'y' coordinates, respectively. During this journey, the sixth link, i.e., the lock, is constantly applied an upward force to prevent it from sliding down due to gravity. This force is applied to the lock throughout the simulation except when the object is being locked, carried, or released.

Picking Mechanism

After reaching the square containing the object, the robot picks it up by a sequence of movements of links by controlling the associated joints as described-

- The fourth link descends up to a certain distance such that the fifth link comes in level with the hole of the object.
- The fifth link then moves horizontally and enters the hole. The sixth link moves upwards and locks the object to prevent it from sliding while carrying at another position.
- The fourth link ascends back to its position, and the robot successfully picks up the object.

Picking Mechanism



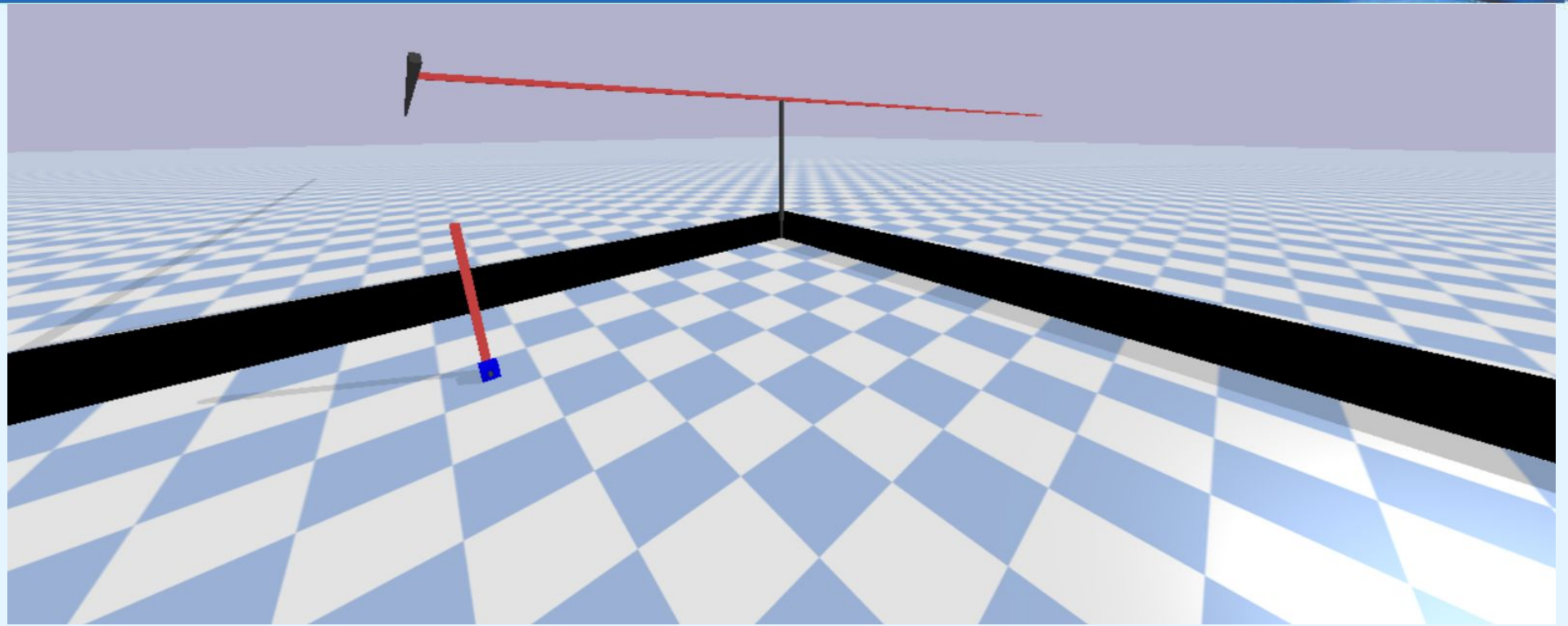
The robot picking an object according to the defined mechanism.

Placing Mechanism

After picking up the object, the robot must reach its correct position to place it. This reaching is implemented by the Q-learning algorithm used to train the robot in the Markov Decision Process as described in previous slides. The sequence of movements of links by controlling the associated joints is -

- The fourth link descends up to a certain distance such that the object's base comes in contact with the plane.
- The sixth link moves downwards to unlock the object.
- The fifth link moves horizontally out of the hole, back to its position.
- The object is now successfully placed in its final position.
- The fourth link ascends back to its position, and the robot executes the picking and placing processes for the remaining objects.

Placing Mechanism



The robot placing an object according to the defined mechanism.

Conclusions and Discussion

We focused on training the Pick and Place robot using the Q-learning algorithm, a classic reinforcement learning algorithm to solve a problem of the Markov Decision Process. The robot learned an optimal policy based on rewards from the environment to arrange all the misplaced objects in the least possible time and with sufficient battery level while carrying each object, without any external supervision.

The two major parameters in applying the Q-learning algorithm were the learning rate α and discount rate γ. The learning rate affected the Q-values, the rewards obtained in each episode, the efficiency achieved for arranging each object, and the average efficiency of the whole task. The discount rate affected the Q-values and the N-step return at each episode.

The robot achieved the maximum expected efficiency of nearly 83% at learning rates greater than or equal to 0.1.

Contributions

The pick and place robot in this project can have many valuable contributions to the industries, hospitals, and homes, such as -

- The robot integrated with computer vision technology can organize and select objects based on their size, shape, color, or barcode and place them in their desired location or can modify their orientation. It can also pick and remove defective products before they reach the subsequent production phase.
- The robot can be employed to clean a room by collecting useless objects lying on the floor and placing them in the bin. It can also be instructed to sort valuable things accidentally dropped in a bin and put them in their desired locations.
- The robot can find use in assembling incoming objects on a conveyor belt in industries, where it will pick an object and join or insert it into another thing.

Contributions

- The robot can be used in the packaging process, where it shall grab the material and pack it into a container or place it on a conveyor belt for transportation purposes.
- The robot can be of significant use in transporting medicines and first-aid kits to the patients' rooms in the hospital by pick and place mechanisms.
- The robot can be operated in households to arrange misplaced things in a room or transport the stuff from one room to another.

Advantages

The prominent advantages in all the above tasks using the pick and place robot are -

- The robot learns to execute the job in the least possible time without prior knowledge of optimal actions.
- It tracks its battery level and executes the task only with a safe level defined to avoid damaging the picked-up objects due to a sudden drop in case of a fully drained battery.
- It relieves human beings from highly mundane and tiresome jobs, enabling them to concentrate on more complicated work and alleviating the risk of any strain or injury while doing a recurring task.

Future Directions

There are many other reinforcement learning algorithms applicable to train this robot, namely, State-Action-Reward-State-Action (SARSA), Deep Q-Network (DQN), Deep Deterministic Policy Gradient (DDPG), which can be possibly applied to train the robot, and can be the prospects of this project.

The robot can also be equipped with a vision-guided system to find the objects on its own without specifying their initial positions. For example, in our experiment, the robot enabled with computer vision can take the image of the grid containing the objects and then analyze it to predict the coordinates of various things to execute the pick and place task. The reward can be decreased for inaccurately computing an object's coordinates while keeping the rest of the algorithm the same.

THANK YOU