Reinforcement Learning: Algorithms to balance the bias between Exploration and Exploitation

Mansi Saxena
Department of Computer Science
North Carolina State University
Raleigh,North Carolina,United States
immansi@gmail.com

J.Jabanjalin Hilda
Department of Software Systems
Vellore Institute of Technology
Vellore, India
jabanjalin.hilda@vit.ac.in

Vinila Jinny
Department of Software Systems
Vellore Institute of Technology
Vellore, India
vinilajinny.s@vit.ac.in

***Abstract*—The main task of a reinforcement learning agent is to develop a policy that maps states to optimal actions. The agent makes decisions from the available actions and learns which of them yield the best rewards. In this way, a Reinforcement Learning agent maximizes the reward received on each action decision by exploiting its current knowledge of actions and choosing the most seemingly favorable action. However, to gain knowledge on how favorable each action is, it must explore all the available actions. This is where the conflict between exploration and exploitation exists. There are several existing algorithms that provide a balance between this bias. Some of these techniques are the Epsilon-Greedy Approach, the Optimistically High Initial Values Approach and the Upper Confidence Bound Approach. This paper presents a comparative analysis of these algorithms, their advantages and disadvantages, simulated for the K-Armed Bandit problem. Based on this research, a combination of these methods is used in maximizing rewards attained in the long term. The effect of varying the epsilon parameter is analyzed by running various simulations, and the change in performance is noted when the epsilon parameter is reduced as convergence is achieved.**

Keywords—Armed Bandit problem, exploration exploitation, Epsilon Greedy approach, Optimistic Initial Values, Upper Confidence Bound

I. INTRODUCTION

Reinforcement Learning is that part of Artificial Intelligence that deals with learning by constant feedback, ie, reinforcement. An agent is expected to learn to perform a task by making action decisions in its environment and getting feedback on these decisions. Receiving positive feedback means that the action taken is favourable in performing the task and vice versa[1].

In this way, an agent can learn tasks without using any training data, unlike supervised learning models. This makes reinforcement learning agents superior in the sense that they can learn many complex tasks such as playing chess or picking empty cans in a room. In the prior example, the environment for the agent are the chess pieces situated at various positions on the chess board. The actions available are the possible chess moves that adhere to the rules of the game. On choosing an action, i.e., making a move, the agent receives feedback on how well it played. This feedback reflects if this move made by the agent will help it in winning the chess match or not. A supervised learning model cannot learn to play chess as well as a reinforcement learning agent because creating a dataset with all possible chess piece combinations with the optimal move to take in each situation is a complex task. It is a more efficient idea to let the agent learn by experience.

The goal of a Reinforcement Learning agent is to maximise the reward it receives in the long term[2,3]. It makes use of its current knowledge to decide which action will give the most return. This is known as "exploitation" of the agent's knowledge. While exploitation is necessary, it is also important to choose random actions sometimes to find any other actions that may have become more favourable. This is known as exploration[4]. Exploration helps the agent in finding actions that may bring more return in the long term. Thus, there is a conflict between exploration and exploitation.

This paper discusses several algorithms and strategies that are useful in encouraging exploration in Reinforcement Learning problems. These models help in attaining a sophisticated balance between this exploration and exploitation dilemma. Some of these models are "Epsilon-Greedy Approach", "Optimistic Initial Values" and "Upper Confidence Bound". The effects of varying the parameters like the epsilon value and the confidence value are also analysed here.

In general, the balance between Exploration and Exploitation is a generic problem that can be applied to many domains [5–7]. Thus, this research can be used in many different ways.

Related concepts:

In this project, the Exploration-Exploitation bias is studied using the K-Armed Bandit Problem[8], as introduced in Fig. 1. It is the most basic Reinforcement Learning task, where an agent is expected to repeatedly choose between some k options. There is no concept of different states in this problem; all states all states have the same conditions and k actions, unlike a Markov Decision Process [9, 10]. Markov Decision Processes, or MDPs, are a formal way of representing any reinforcement learning problem using a finite set of states, actions and rewards along with a state transition function. Exploration in MDPs is out of the scope of this research, but is an interesting area for future research [11–13].

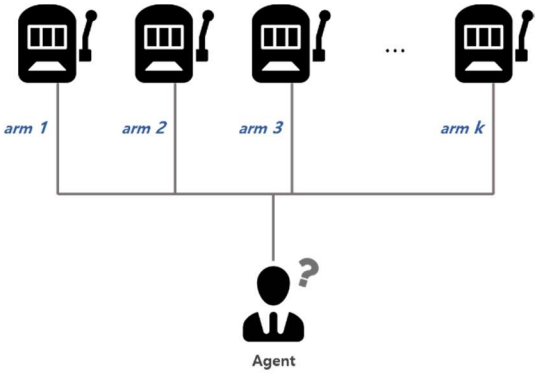


Fig. 1. The K-Armed Bandit Problem

Another assumption in this research is that the tasks are not associative. This implies that different situations have different optimal actions. Contextual bandits is one way of bringing associativity to the K-Armed Bandit Problem [14, 15]. They are intermediate between the simple K-Armed Bandit Problem, and the full-fledged Reinforcement Learning Problem, where actions affect future decisions.

In this study, only the simple K-Armed Bandit Problem is used. Here, after each action selection, a numerical reward is received for that action. The objective is to maximise the expected total reward over some given time steps [16].

For each of the actions, an expected reward called value of the action is computed based on the rewards it has yielded in the past. If the action selected and reward received at a given time t is A_t and R_t respectively, then the value of an action can be computed by the equation given below.

$$q*(a) = E[R_t \mid A_t = a] \quad (1)$$

These $q*(a)$ values are not known to us. They are estimated at each time step using the reward received, and are called the estimated value of an action, $Q_t(a)$. This estimated value can be initialised as 0 or any other value, and it must approach the expected value $q*(a)$ as more time steps pass, i.e., their values should converge.

This reward is used to update the estimated values Qt(a) of the action selected using the incremental equation. These values are useful in making action-decisions. The incremental equation is given below.

$$Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n] \quad (2)$$

The action with the highest estimated value $Q_t(a)$ value is known as the greedy action. Selecting greedy actions means exploiting your current knowledge. It is a greedy approach of making action-decisions. Choosing another action with a lower estimated value is known as exploring. Exploiting can give satisfactory short term rewards, but exploring can give larger rewards in the long run. The execution flow of iterations when agent learns from reinforcement is depicted in Fig. 2.

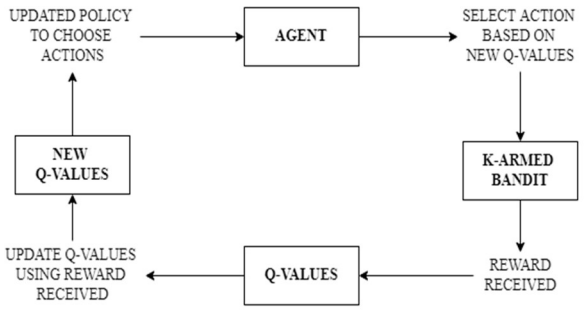


Fig. 2: Execution flow of iterations when agent learns from reinforcement

II. Existing Methods

A. Greedy Approach

The Greedy Approach involves choosing the action with the highest estimated value Qt(a) at a given time step t. This is the most basic agent as it does not explore at all. If an unusual reward from an action causes the estimated Q-value of a sub-optimal action to become greater than that of an optimal action, the greedy action will repeatedly choose the sub-optimal action, until it receives a low reward that causes its estimated Q-value to decrease.

Algorithm 1 Greedy Approach algorithm

1: **for** $A \leftarrow 1$ to k **do**
2: $Q(A) \leftarrow 0$
3: $N(A) \leftarrow 0$
4: **end for**
5: **while** $True$ **do**
6: $nextAction \leftarrow arg\ max_a Q(a)$
7: $R \leftarrow RewardFunction(nextAction)$
8: $nextAction \leftarrow nextAction + 1$
9: $Q(nextAction) \leftarrow Q(nextAction) + \frac{1}{N(nextAction)}[r - Q(nextAction)]$
10: **end while**

The algorithm for this approach is shown in Algorithm 1. First, for each action A, the estimated values Qt(a) and the arm count, i.e., number of times the action is chosen, are initialised to 0. Then, actions are chosen indefinitely in a greedy manner, and the estimated values are updated. 'RewardFunction' is a function that takes an action as a parameter and returns a reward.

B. Epsilon-Greedy Approach

This approach involves a random action to be selected with epsilon probability[17,18]. The remaining time, actions are chosen greedily. In this way, this method encourages random exploration. This prevents the agent from infinitely choosing a sub-optimal action as described in Greedy approach above. The EpsilonGreedy Approach breaks this loop by exploring random actions. One drawback of this approach is that once convergence is achieved, i.e., the estimated values Qt(a) approach q∗(a), the agent is forced to choose a random action, though it has sufficient knowledge to choose the optimal by exploitation[19]. This prevents further improvement in performance.

Algorithm 2 Epsilon-Greedy Approach algorithm

1: **for** $A \leftarrow 1$ to k **do**
2: $Q(A) \leftarrow 0$
3: $N(A) \leftarrow 0$
4: **end for**
5: **while** $True$ **do**
6: $nextAction \leftarrow \begin{cases} arg\ max_a Q(a) & with\ probability\ 1-\epsilon \\ a\ random\ action & with\ probability\ \epsilon \end{cases}$
7: $R \leftarrow RewardFunction(nextAction)$
8: $nextAction \leftarrow nextAction + 1$
9: $Q(nextAction) \leftarrow Q(nextAction) + \frac{1}{N(nextAction)}[r - Q(nextAction)]$
10: **end while**

The algorithm for this approach is shown in Algorithm 2. The ϵ parameter determines the action chosen, and balances the degree of exploration.

C. Optimistic Initial Values Approach

In this approach, the initial estimated values of each action is set optimistically high and actions are chosen greedily. Each time an action is chosen, on receiving a realistic reward is received which lowers the estimated value and makes it more realistic. This in-turn causes the estimated of other actions to be greater than that of this chosen action, thus making them more favourable to be chosen by the greedy approach. In this way, this approach encourages early exploration. One flaw in this approach is that it chooses actions greedily. Thus, once the estimated Q-values become realistic, this approach does not explore at all and behaves as a simple Greedy agent. Thus, if the optimal action were to change due to external factors, which is often encountered in real-life problems, this agent will keep choosing the previously optimal, now sub-optimal action, and never find the newly optimal action.

Algorithm 3 Optimistic Values Approach algorithm

1: **for** $A \leftarrow 1$ to k **do**
2: $Q(A) \leftarrow 0$
3: $N(A) \leftarrow an\ optimistically\ high\ value$
4: **end for**
5: **while** $True$ **do**
6: $nextAction \leftarrow arg\ max_a Q(a)$
7: $R \leftarrow RewardFunction(nextAction)$
8: $nextAction \leftarrow nextAction + 1$
9: $Q(nextAction) \leftarrow Q(nextAction) + \frac{1}{N(nextAction)}[r - Q(nextAction)]$
10: **end while**

The algorithm for this approach is shown in Algorithm 3. The estimated values Qt(a) is initialised to a high value, which encourages early exploration.

D. Upper Confidence Bound Approach

This method helps in smart exploration of the actions, unlike the Epsilon- Greedy approach, where actions are chosen randomly with no preference. This approach selects among the non-greedy actions with epsilon-probability, according to their potential for actually being optimal.

Algorithm 4 UCB Approach algorithm

1: **for** $A \leftarrow 1$ to k **do**
2: $Q(A) \leftarrow 0$
3: $N(A) \leftarrow 0$
4: **end for**
5: **while** $True$ **do**
6: $nextAction \leftarrow \begin{cases} arg\ max_a Q(a) & with\ probability\ 1-\epsilon \\ arg\ max_a \left[Q(a) + c\sqrt{\frac{\ln}{N_t(a)}}\right] & with\ probability\ \epsilon \end{cases}$
7: $R \leftarrow RewardFunction(nextAction)$
8: $nextAction \leftarrow nextAction + 1$
9: $Q(nextAction) \leftarrow Q(nextAction) + \frac{1}{N(nextAction)}[r - Q(nextAction)]$
10: **end while**

This method helps in smart exploration of the actions, unlike the EpsilonGreedy approach, where actions are chosen randomly with no preference [20]. This approach selects among the non-greedy actions with epsilon-probability, according to their potential for actually being optimal [21–23]. It considers how close the estimated values of each action is to being maximal and the uncertainties in those estimates. If Nt(a) is the number of times action a has been selected prior to time t, then while exploring, actions are chosen by the equation given below.

$$A_t = \arg_a \max[Q_t(a) + c\sqrt{\frac{lnt}{N_t(a)}}] \quad (3)$$

The square root term is the exploration term and it measures uncertainty. Each time a particular action is chosen, the denominator increases, causing the uncertainty to reduce, thus making it less likely for that action to be chosen. Similarly, if the action is not chosen for several time steps, then the numerator increases but the denominator remains the same, which increases uncertainty, making it more likely for that action to be chosen.

The exploration term is being added to get an estimate of the possible true value of the action. The action with the highest possible true value is chosen to be explored. Here, 'c' is the confidence value in this exploration term. Thus, actions with lower value estimates, or that have already been selected frequently, will be selected with decreasing frequency over time

The algorithm for this approach is shown in Algorithm 4. The action taken during exploration is not randomly chosen. Instead, the action most likely to return a high reward is chosen.

III. THE PROPOSED SIMULATION

A K-Armed Bandit is simulated here is similar to that described in [24]. There are 10 actions in this Reinforcement Learning task. The agent is expected to repeatedly choose among them and find the optimal action. The expected value $q^*(a)$ for each action is defined by randomly sampling 10 values from a normal distribution with mean 0 and variance 1. Actions are chosen according to a policy depending on the exploration-exploitation approach chosen. The reward

given for an action-decision is the sum of the estimated values $Q_t(a)$ of that action and a value sampled from a normal distribution with mean 0and variance 1.

2000 such k-armed bandits problems are simulated. That is, there are 2000 agents that learn to do the task. For each of the 2000 bandits, the optimal action is found. Then, it is checked if the agent chooses the optimal action at each time step. From the 2000 agent simulations, the percent of agents that choose optimal action at each time step is calculated and recorded.

IV. RESULTS AND DISCUSSION

In each of the graphs shown below, the percent of number of agents out of 2000 that choose an optimal action at time step 't' is plotted against the time steps.

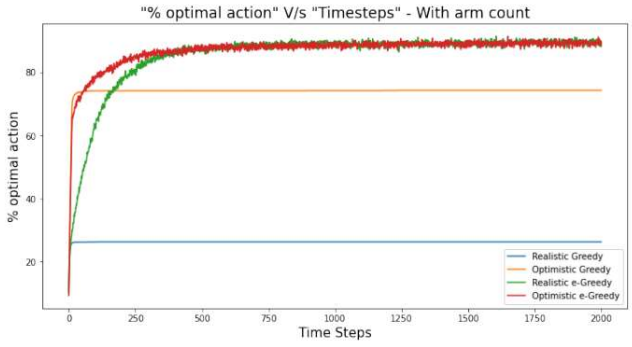
A, The significance of using optimistically high values, and random exploration using an Epsilon-value

In the first simulation, four different types of agents are visualised, each with a different policy. These policies are -

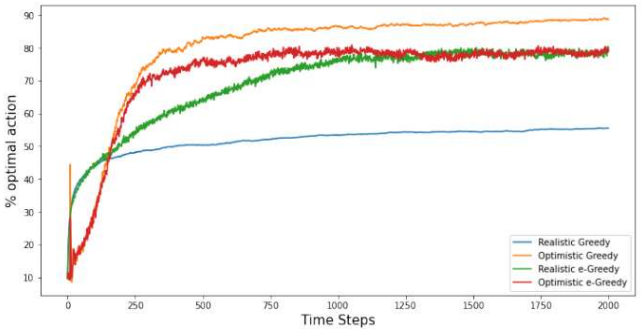
1. Realistic, Greedy (Q1 = 0, ϵ = 0)
2. Optimistic Greedy (Q1 = 5, ϵ = 0)
3. Realistic Epsilon-Greedy (Q1 = 0, ϵ = 0.1)
4. Optimistic Epsilon-Greedy (Q1 = 5, ϵ = 0.1)

These four agents are simulated for a stationary problem, where the value of n in equation 2 is the number of times a particular action has been chosen. The resultant graph is shown in Fig. 3a. It is then simulated for a non-stationary problem, where the value of $\frac{1}{n}$ in equation 2 is 0.1. The resultant graph is shown in Fig. 3b.

From the Fig. 3, it can be observed that setting optimistically high values encourages early exploration, and thus it learns faster than the Realistic approach. It can also be seen that the Epsilon-Greedy approach has an advantage over the greedy method. Also, the effect of setting optimistic values wears off, the Optimistic Epsilon-Greedy approach and Realistic Epsilon-Greedy approach level off a the same percent of times the optimal action is chosen. Another point to note is that the plots of the Greedy Approach do not fluctuate up and down, and remain at the same level, unlike the plots of the Epsilon-Greedy Approach. This is because, once the Greedy agent finds an optimal action, it does not explore and continues selecting the same action indefinitely. while the Epsilon-Greedy Approach selects a random action with ϵ probability.



(a) Stationary Problem



(b) Non-Stationary Problem

Fig. 3: The four agents, i.e., Realistic Greedy, Optimistic Greedy, Realistic Epsilon-Greedy and Optimistic Epsilon-Greedy are simulated for a Stationary and Non-Stationary Problem

B The effect of changing the Epsilon-value

In the second simulation, the effect of the parameter Epsilon is visualised. four different values are used in the simulation: 0.01, 0.1, 0.25, 0.5. The Optimistic Epsilon-Greedy agent is used to run the simulation. As seen in Fig. 4, irrespective of the epsilon value, each of the agent start at the same rate of learning. This is because, initial estimated values Qt(a) are equally and optimistically. However, depending on the value of Epsilon, the agent stops improving its performance and levels off at different percent's of the optimal action being chosen. The higher the Epsilon value, the lower the performance of the agent. This is because, even when the agent gains sufficient knowledge to decide which is the optimal action, it is made to choose another sub-optimal solution which causes its performance to decrease.

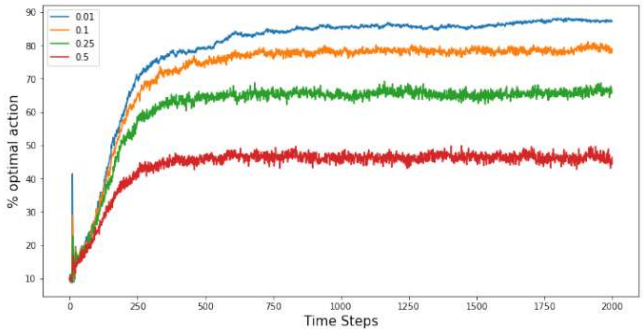


Fig. 4: Using different Epsilon values with the Optimistic Epsilon-Greedy agent Different Epsilon

C. The effect of lowering the Epsilon-value as convergence is achieved

In the third simulation, the effect of lowering the value of epsilon as convergence approaches is visualised. The epsilon value determines the degree of exploration. A lower value of epsilon would mean that the agent's current knowledge is exploited more, and random exploration is discouraged. This is suitable only when convergence approaches, as the agent has learnt about each action and has sufficient knowledge to choose the optimal action. To run the simulation, the value of epsilon is reduced by 0.02 after every 400 steps using the Optimistic Epsilon-Greedy agent.

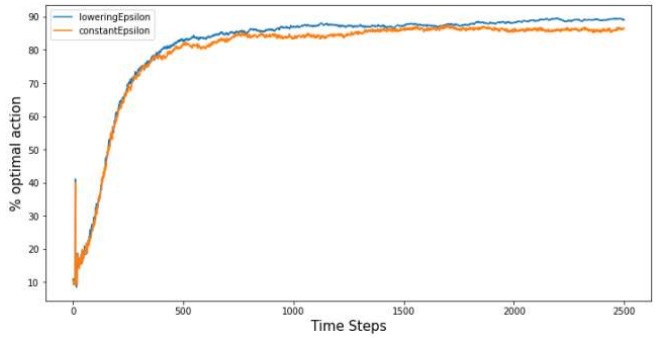


Fig. 5: Lowering the value of Epsilon as the Optimistic Epsilon-Greedy agent approaches convergence

In Fig. 5, it can be seen that the Optimistic Greedy agent that lowers its Epsilon-value as convergence approaches reaches a higher performance level than the original Optimistic Greedy agent that uses a constant Epsilon-value.

D Using Upper Confidence Bound (UCB) with the Optimistic Epsilon-Greedy Approach

The fourth simulation visualises the Upper confidence Bound algorithm, that chooses actions based on an exploration term calculated, and not randomly. the effect of lowering the value of Epsilon is seen again. To run this simulation, the Optimistic Epsilon-Greedy agent is used, and the exploration actions are chosen using the UCB algorithm. This agent is implemented using a constant Epsilon-value and a lowering Epsilon-value. These are compared to the original Optimistic Epsilon-Greedy approach that chooses its exploration actions randomly.

From Fig. **6,** it can be seen that the UCB algorithm can improve the performance of the Optimistic Epsilon-Greedy approach, and it can improve more if the value of Epsilon is lowered as convergence is approached.

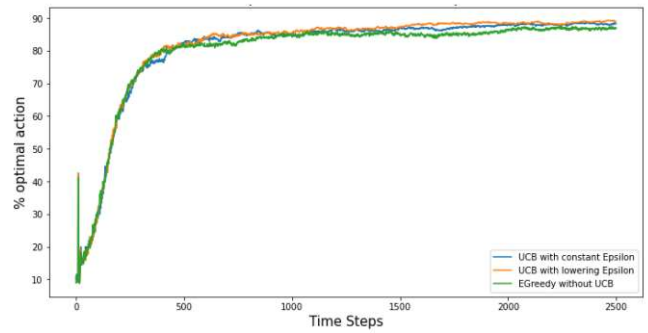


Fig. 6: Using the UCB algorithm with the Optimistic Epsilon-Greedy agent, both with constant and lowering Epsilon values

D. Policy iteration simulation

A simple Reinforcement Learning(RL) task is taken for this simulation, shown in Fig. 7.

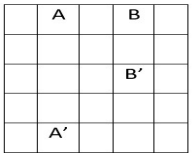


Fig. 7: Policy Iteration RL task

MDP formulation:

States: set of 25 states shown in Fig. 3. Actions:

{Top, Left, Right, Bottom}.

Dynamics Function: This task is not stochastic, ie, choosing an action in a state leads to a new state and returns a specific reward with probability 1. This can be represented as follows.

$$p(s', r \mid s, a) = 1, \text{ for all } s \text{ in } S, a \text{ in } A(s)$$

State Transitions: In any state, choosing Top, Left, Right, Bottom as action takes the agent to a new state that is directly Top, Left, Right or Bottom of the previous state, respectively. However, if taking such an action causes the agent to move out of the 5*5 board, then the state does not change and the agent remains in the same place. Taking any action in states A and B take the agent to states A' and B'.

Rewards: {If state is A, reward is +10, if state is B, reward is +5, reward is -1 if the action causes the agent to move out of the 5*5 board (here position remains unchanged). Reward is 0 everywhere else}.This has 25 states, thus to find the optimal policy by solving the Bellman Equations is not feasible. When Policy Iteration is simulated to calculate the Optimal Policy for this RL problem, the following policies are obtained at each iteration:

```
Original Policy:
(0,0) : [0.25 0.25 0.25 0.25]
(0,1) : [0.25 0.25 0.25 0.25]
(0,2) : [0.25 0.25 0.25 0.25]
(0,3) : [0.25 0.25 0.25 0.25]
(0,4) : [0.25 0.25 0.25 0.25]
(1,0) : [0.25 0.25 0.25 0.25]
(1,1) : [0.25 0.25 0.25 0.25]
(1,2) : [0.25 0.25 0.25 0.25]
(1,3) : [0.25 0.25 0.25 0.25]
(1,4) : [0.25 0.25 0.25 0.25]
(2,0) : [0.25 0.25 0.25 0.25]
(2,1) : [0.25 0.25 0.25 0.25]
(2,2) : [0.25 0.25 0.25 0.25]
(2,3) : [0.25 0.25 0.25 0.25]
(2,4) : [0.25 0.25 0.25 0.25]
(3,0) : [0.25 0.25 0.25 0.25]
(3,1) : [0.25 0.25 0.25 0.25]
(3,2) : [0.25 0.25 0.25 0.25]
(3,3) : [0.25 0.25 0.25 0.25]
(3,4) : [0.25 0.25 0.25 0.25]
(4,0) : [0.25 0.25 0.25 0.25]
(4,1) : [0.25 0.25 0.25 0.25]
(4,2) : [0.25 0.25 0.25 0.25]
(4,3) : [0.25 0.25 0.25 0.25]
(4,4) : [0.25 0.25 0.25 0.25]
```

Fig. 8: Original Policy π_0

The original policy is shown in Fig. 8. Starting from each state, represented as (row, column), the policy is a list of 4 numbers, which represented the probability that the agent will choose the actions ["top", "bottom", "right", "left"], respectively. It is also known as an Equiprobable policy, since all actions are equally likely to be chosen. Call is π0 for clarity.

The state values v5 obtained for this policy π5 are shown in Fig. 9.

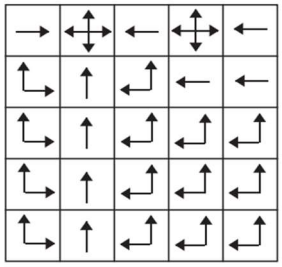
21.98	24.42	21.98	19.42	17.48
19.78	21.98	19.78	17.8	16.02
17.8	19.78	17.8	16.02	14.42
16.02	17.8	16.02	14.42	12.98
14.42	16.02	14.42	12.98	11.68

Fig. 9: v_5 values obtained after Policy Evaluation of π_5

Now, since v4 and v5 obtained are equal, the policy iteration algorithm has converged, and the 5th policy computed is the optimal policy. Thus, π5 and v5 are π* and v*. This can be represented as shown in Fig. 10.

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

Fig. 10. Optimal Policy and Optimal State Values

V. CONCLUSION

In summary, this paper performs a detailed comparison between the existing methods to balance the bias between Exploration and Exploitation. A combination of these methods is also compared to the original methods to see their effectiveness. The effect of varying the epsilon-parameter is also analysed.

From the observed behaviour of the agents, it can be concluded that for a nonstationary problem, the Optimistic Epsilon Greedy Approach that uses the UCB algorithm to make smart choices, along with a lowering Epsilon value is most effective. One drawback of this analysis is the use of the basic Reinforcement Learning problem, the KArmed Bandit Problem, where the states do not change. Due to this limitation, these results cannot be directly applied to real world problems. Future scope to this research would extend these simulations to contextual bandits and Markov Decision Process.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] Dayan, P., Niv, Y.: Reinforcement learning: the good, the bad and the ugly. Current opinion in neurobiology 18(2), 185–196 (2008)

[3] Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of artificial intelligence research 4, 237–285 (1996)

[4] Mehlhorn, K., Newell, B.R., Todd, P.M., Lee, M.D., Morgan, K., Braithwaite, V.A., Hausmann, D., Fiedler, K., Gonzalez, C.: Unpacking the exploration–exploitation tradeoff: A synthesis of human and animal literatures. Decision 2(3), 191 (2015)

[5] Zhang, L., Tang, K., Yao, X.: Explicit planning for efficient exploration in reinforcement learning. Advances in Neural Information Processing Systems 32 (2019)

[6] Gupta, A.K., Smith, K.G., Shalley, C.E.: The interplay between exploration and exploitation. Academy of management journal 49(4), 693–706 (2006) Springer Nature 2021 LATEX template 12 Exploration and Exploitation

[7] March, J.G.: Exploration and exploitation in organizational learning. Organization science 2(1), 71–87 (1991)

[8] Lavie, D., Stettner, U., Tushman, M.L.: Exploration and exploitation within and across organizations. Academy of Management annals 4(1), 109–155 (2010)

[9] Kuleshov, V., Precup, D.: Algorithms for multi-armed bandit problems. arXiv preprint arXiv:1402.6028 (2014)

[10] Garcia, F., Rachelson, E.: Markov decision processes. Markov Decision Processes in Artificial Intelligence, 1–38 (2013)

[11] White III, C.C., White, D.J.: Markov decision processes. European Journal of Operational Research 39(1), 1–16 (1989)

[12] Chang, H.S., Hu, J., Fu, M.C., Marcus, S.I.: Simulation-based Algorithms for Markov Decision Processes. Springer, ??? (2007)

[13] Tarbouriech, J., Lazaric, A.: Active exploration in markov decision processes. In: The 22nd International Conference on Artificial Intelligence and Statistics, pp. 974–982 (2019). PMLR

[14] Hallak, A., Di Castro, D., Mannor, S.: Contextual markov decision processes. arXiv preprint arXiv:1502.02259 (2015)

[15] Langford, J., Zhang, T.: The epoch-greedy algorithm for contextual multiarmed bandits. Advances in neural information processing systems 20(1), 96–1 (2007)

[16] Lu, T., P´al, D., P´al, M.: Contextual multi-armed bandits. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 485–492 (2010). JMLR Workshop and Conference Proceedings

[17] Macready, W.G., Wolpert, D.H.: Bandit problems and the exploration/exploitation tradeoff. IEEE Transactions on evolutionary computation 2(1), 2–22 (1998)

[18] Vermorel, J., Mohri, M.: Multi-armed bandit algorithms and empirical evaluation. In: European Conference on Machine Learning, pp. 437–448 (2005). Springer

[19] Agrawal, S., Goyal, N.: Analysis of thompson sampling for the multiarmed bandit problem. In: Conference on Learning Theory, pp. 39–1 (2012). JMLR Workshop and Conference Proceedings

[20] Vamplew, P., Dazeley, R., Foale, C.: Softmax exploration strategies Springer Nature 2021 LATEX template Exploration and Exploitation 13 for multiobjective reinforcement learning. Neurocomputing 263, 74–86 (2017) [20] Wilson, R.C., Bonawitz, E., Costa, V.D., Ebitz, R.B.: Balancing exploration and exploitation with information and randomization. Current opinion in behavioral sciences 38, 49–56 (2021)

[21] Bubeck, S., Cesa-Bianchi, N.: Regret analysis of stochastic and nonstochastic multi-armed bandit problems. arXiv preprint arXiv:1204.5721 (2012)

[22] Garivier, A., Moulines, E.: On upper-confidence bound policies for switching bandit problems. In: International Conference on Algorithmic Learning Theory, pp. 174–188 (2011). Springer

[23] Kaufmann, E., Capp´e, O., Garivier, A.: On bayesian upper confidence bounds for bandit problems. In: Artificial Intelligence and Statistics, pp. 592–600 (2012). PMLR

[24] Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT press; (2018).