Journal of Computer Engineering, Electronics and Information Technology (COELITE)

Journal homepage: https://ejournal.upi.edu/index.php/COELITE

# Analysis of Model-Free Reinforcement Learning Algorithm for Target Tracking

*Muhammad Fikry[1, *], Rizal Tjut Adek[2], Z. Zulfhazli[3], Subhan Hartanto[4], T. Taufiqurrahman[5], Dyah Ika Rinawati[6]*

[1,2]Department of Informatics, Universitas Malikussaleh, Indonesia
[3]Department of Civil Engineering, Universitas Malikussaleh, Indonesia
[4]Department of Computer Science, Universitas Pat Petulai, Indonesia
[5]Department of Informatics, Universitas Sumatera Utara, Indonesia
[6]Department of Industrial Engineering, Universitas Diponogoro, Indonesia
Correspondence: E-mail: muh.fikry@unimal.ac.id

## ABSTRACT

Target tracking is a process that can find points in different domains. In tracking, some places contain prizes (positive or negative values) that the agent does not know at first. Therefore, the agent, which is a system, must learn to get the maximum value with various learning rates. Reinforcement learning is a machine learning technique in which agents learn through interaction with the environment using reward functions and probabilistic dynamics to allow agents to explore and learn about the environment through various iterations. Thus, for each action taken, the agent receives a reward from the environment, which determines positive or negative behavior. The agent's goal is to maximize the total reward received during the interaction. In this case, the agent will study three different modules, namely sidewalk, obstacle, and product, using the Q-learning algorithm. Each module will be training with various learning rates and rewards. Q-learning can work effectively with the highest final reward at a learning rate of 0.8 for 500 rounds with an epsilon of 0.9.

## ARTICLE INFO

***Article History:***
*Submitted/Received 30 Jan 2022*
*First Revised 28 Feb 2022*
*Accepted 17 Mar 2022*
*First Available online 01 Apr 2022*
*Publication Date 01 Apr 2022*

***Keyword:***
*Algorithm,*
*Machine learning,*
*Probabilistic,*
*Q-Learning,*
*Reinforcement learning,*
*Target tracking.*

## 1. INTRODUCTION

Determination of distribution channels can determine the optimal path with distance, time, and capacity as aspects of consideration [1]. Many methods can be used to optimize the path, such as the Heuristic Algorithm, the Tabu Search Method for delivery [2], the Ant Colony Optimization (ACO) method for mobile robot navigation [3], and others. Getting the optimal path will provide many advantages in terms of time, speed, cost and avoid losses caused by certain conditions such as congestion. The optimal path is determined from one place to a particular destination, and it has a starting point and stops at a predetermined endpoint with various conditions encountered. We try to approach the initial situations at any point and the stopping point in more than one place, meaning that it has several places as temporary stops and one final stop that occurs anywhere (random location).

In this study, we consider the problem of tracking multiple objects in an environment. An environment is a rectangular space bounded by a sidewalk. We use model-free reinforcement learning to train the tracker to detect the product and avoid any obstacle in its movement. In other words, the agent must find the best way to achieve the highest reward. However, one of the challenges in reinforcement learning is the problem of exploration vs. exploitation, which is a trade-off between obtaining a tip from a choice that is considered safe and exploring other possibilities that may be profitable. The investigation allows increasing agent knowledge, which leads to long-term benefits. At the same time, exploitation chooses the action to get the most rewards but leads to less than optimal behavior. Our learning is done through python simulations to observe the system's characteristics. The agent will explore specific paths to get information. Then, the agent uses this information in the exploitation to choose the best path for branching lanes. The agent will take action using greedy epsilon. Therefore, it is essential to get the best level of learning for agents to bring efficiency in taking steps.

## 2. METHODS

### 2.1. Reinforcement Learning

As part of machine learning methods, reinforcement learning (RL) [4-5] has been widely used in several disciplines to find optimal policies in an uncertain environment. Many studies have been carried out to develop RL techniques to find optimal feedback solutions, such as the zero-sum game problem [6] RL works by interacting with the environment. The RL algorithm works like the human brain when making several decisions and helps to take decisions sequentially. **Figure 1.** shows the interaction of agents with the environment in taking action to maximize reward.

RL studied effective strategies for agents from experimental trials and received feedback. With an optimal design, agents can actively adapt to the environment to maximize future rewards. The agent can remain in one of the many states ($s \in S$) in the background and choose to take action ($a \in A$) to move from one state to the next. The state where the agent will arrive is determined by the transition probability between states (P). After the action is taken, the environment rewards ($r \in R$) as feedback. In this study, state (s) is the location, activity (a) is the action (movement) that must be taken, and the reward (r) is the initial value given when the move occurs.

DOI: https://doi.org/10.17509/coelite.v1i1.43795
p- ISSN 2829-4157 e- ISSN 2829-4149

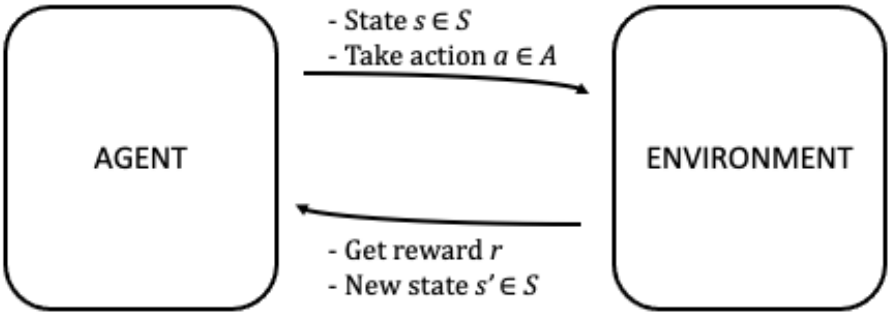


**Figure 1.** Interaction of agent with the environment.

In defining the reward function and transition probability. If the model is known, it means planning with perfect information, i.e., doing model-based RL. We can find the optimal solution with Dynamic Programming (DP) when we fully see the environment. However, if the model is unknown, that means learning with incomplete information, i.e., doing RL model-free or learning the model explicitly as part of the algorithm.

The model describes the environment, and we can study or infer how the environment will interact with models and provide feedback to agents. This model has two main parts, the transition probability function P and the reward function R. In Equation [1], the transition function P records the transition probability from state (s) to ( s') after performing an action and obtaining (a) temporary reward (r).

$$P(s', r|s, a)=\mathbb{P}[S_{t+1} = s', R_{t+1}|S_t = s, A_t = a] \quad (1)$$

Thus, the transition function between states can be defined in Equation [2] as a function of P(s',r|s,a).

$$P_{ss'}^{a} = P(s'|s, a) = \mathbb{P}[S_{t+1} = s', R_{t+1}|S_t = s, A_t = a] = \sum_{r\in R} P(s', r|s, a) \quad (2)$$

And the reward function R in Equation [3] predicts the next reward based on an action. The agent's only goal is to maximize the total reward he receives in the long run. Therefore, this reward function is important in mapping each state in the environment to a numerical reward.

$$R(s, a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] = \sum_{r\in R} r \sum_{r\in R} P(s', r|s, a) \quad (3)$$

### 2.2. Q-Learning

In defining the reward function and transition probability. If the model is known, it means planning with perfect information, i.e., doing model-based RL. We can find the optimal solution with Dynamic Programming (DP) when we fully see the environment. However, if the model is unknown, that means learning with incomplete information, i.e., doing RL model-free or learning the model explicitly as part of the algorithm.

Q-learning is one of the important breakthroughs in reinforcement learning, which is the development of the Temporal Difference algorithm introduced by Watkins in 1989 [7-8] but The Q learning algorithm in its early use is flawed in several aspects and its application is limited [9]. Nowadays, some improved method of Q-Learning which combine with the Deep

DOI: https://doi.org/10.17509/coelite.v1i1.43795
p- ISSN 2829-4157 e- ISSN 2829-4149

Learning method and named Deep Q-Learning and has been implemented in robotic application [10-12]. The Q in q-learning stands for quality, which indicates how useful a given action is in getting rewards in the future. Equation [4] below is in the form of the Q-learning formula.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma\, max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t)\right) \quad (4)$$

Q-learning takes action with the highest reward because Q-learning does not use a behavior policy to select the additional action At+1. Instead, estimate the expected future return in the update rule as max A Q (St+1, A). The max operator used follows a greedy policy. Q-Learning will converge to the optimal solution assuming that, after generating experience and training, it switches to a greedy policy. In Equation [4], St is state S at time t, At which is action A at time t, while St+1 is the next state which will be the target of movement, and action At+1 and reward Rt+1. Therefore, the transition probability function P and the reward function R affect the next reward that will be obtained. We define the set of states S as {1, 2, …, 40} which means we have 40 location points, the group of actions A {up, down, left, right} which means that the possible moves are up, down, left, right, while for the positive R reward while {1, 10, 100} and the negative R reward while {-1, -10, -100} [13-20].

### 2.3. System Performance

The flow of system performance is shown in **Figure 2.** We will first create a table of n columns and m rows, where n is the number of actions and m is the number of states. Then assign zero values to all columns and rows. In the next step, we will select action an in state (s) based on the q-table, but as mentioned earlier, at the start, all values are 0. So, we use the greedy epsilon method to select actions randomly (up, down, left, right).

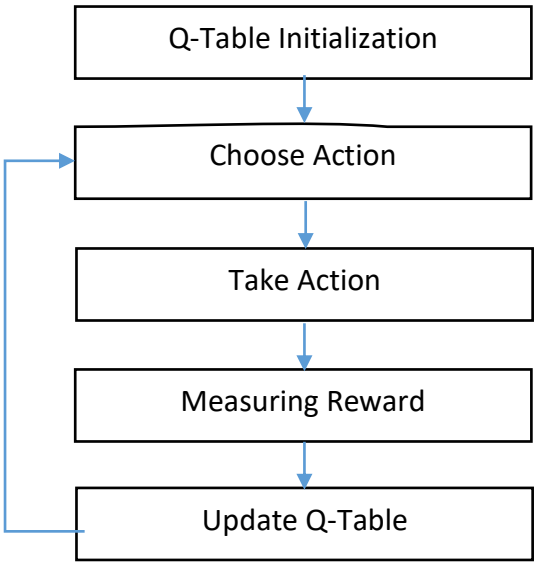


**Figure 2.** System performance scenario using Q-learning.

DOI: https://doi.org/10.17509/coelite.v1i1.43795
p- ISSN 2829-4157 e- ISSN 2829-4149

The system will execute the chosen action based on the greedy epsilon. After the move is obtained, the rewards and results are observed, then update the Q(s, a) function according to Equation [4]. After that, we will return to step two to do the next round until the condition stops. Table Q helps us find the best action in each state, and this is useful for maximizing the expected reward by choosing the best of all possible activities. Q(s, a) returns the expected future reward of the action on that state. When the q-table is ready, the agent will exploit the environment and take better steps.

## 3. RESULTS AND DISCUSSION

**Figure 3.** shows that when the agent moves, the agent will learn the best path to the end of the sidewalk through the reward function. Agent (A) starts at position (1,1). Sidewalks consist of rows and columns, not sidewalks denoted by a line (-). Obstacles (x) and Product (o) are placed randomly.


**Figure 3.** Location of sidewalk, obstacles and product points.

In this case, the state is a collection of locations, and actions are moves that the agent can perform. The state-transition function is the probability of an agent moving from one state to another through several actions. As an algorithm for learning with incomplete information, q-learning is chosen in this study for the application of RL, which is used to train agents in the environment. When q-learning is applied, we create a matrix that follows the form [state, action] and assigns an initial value (q-value) of zero. We then update and store the q-value after each round. The matrix, called q-table, becomes a reference table for agents to choose

DOI: https://doi.org/10.17509/coelite.v1i1.43795
p- ISSN 2829-4157 e- ISSN 2829-4149

the best action based on the q-value. The state values for each possible action are entered into the q-table. The most significant matter is obtained according to the action that will give the maximum value from the next step.

We will train agents on each module and combine them linearly. The selected action is determined by the greedy method; the agent will use the appropriate q-table to choose the action with the maximum expected value. However, the steps are not deterministic, so it is possible that the agent will not initially take the best move for a given state to encourage exploration. The sidewalk module is trained to give positive rewards to agents when they reach the sidewalk's end and give them negative rewards when they move out of the sidewalk. In the obstacles module, the agent has a window with 3×3 dimensions. If the object is in the window in a specific state, the agent will get punishment based on the reward function based on the distance between (0 – 1). With this, if the agent is outside the sidewalk and is in window obstacles, the negative value obtained is even greater. It may result in the agent starting training again from the beginning. The opposite happens if the agent is in the product window in a particular state, the agent will get a reward based on the distance-based reward function (0 – 1).

Modules are trained with various learning rates and rewards. Learning rate determines the extent to which information can override old details. **Figure 4.** shows the last reward for each module in each training round. **Figure 5.** is a heatmap of the q-table on the sidewalk module for the “up” action. The darker areas show a lower q-value, while the lighter areas show a higher q-value. We display a heatmap from the q-table with the three highest learning rates for each module. **Figure 5.** shows that agents are learning to stay away from the sidewalk.

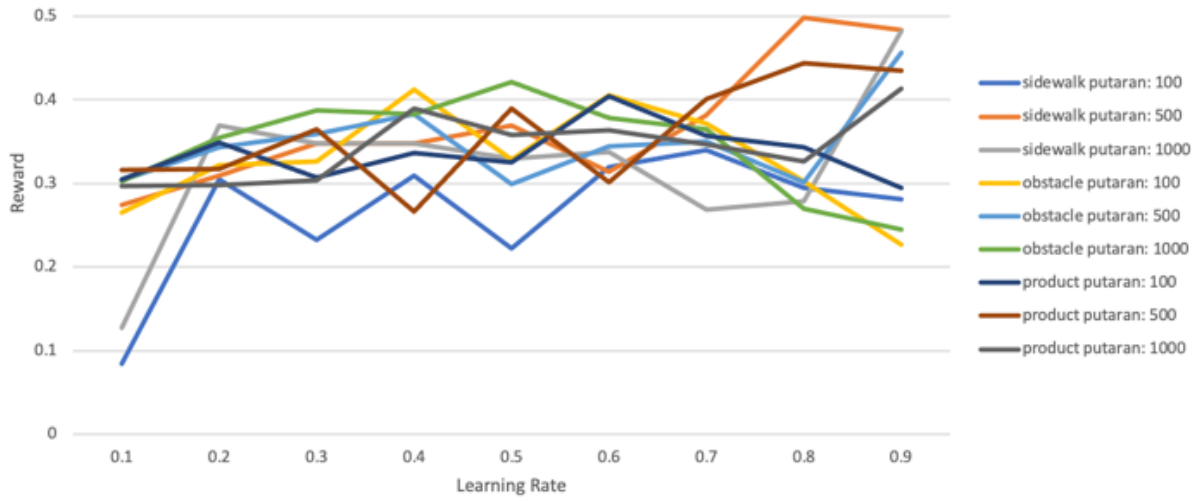


**Figure 4.** The last reward for each module.

DOI: https://doi.org/10.17509/coelite.v1i1.43795
p- ISSN 2829-4157 e- ISSN 2829-4149

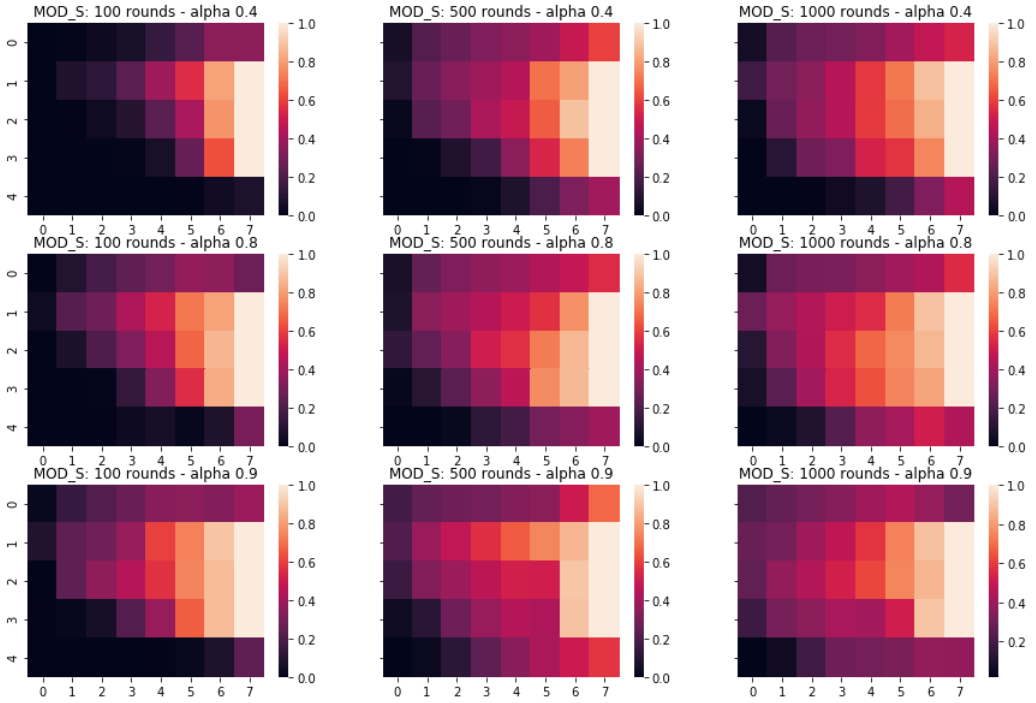


**Figure 5.** Q-table on the sidewalk module.

Furthermore, the obstacle module is trained with varying alpha values and rewards after 1000 training rounds. **Figure 6.** is a heatmap of the Q-table for the obstacle module for the "up" action. **Figure 6.** shows that the agent is learning to avoid obstacles.

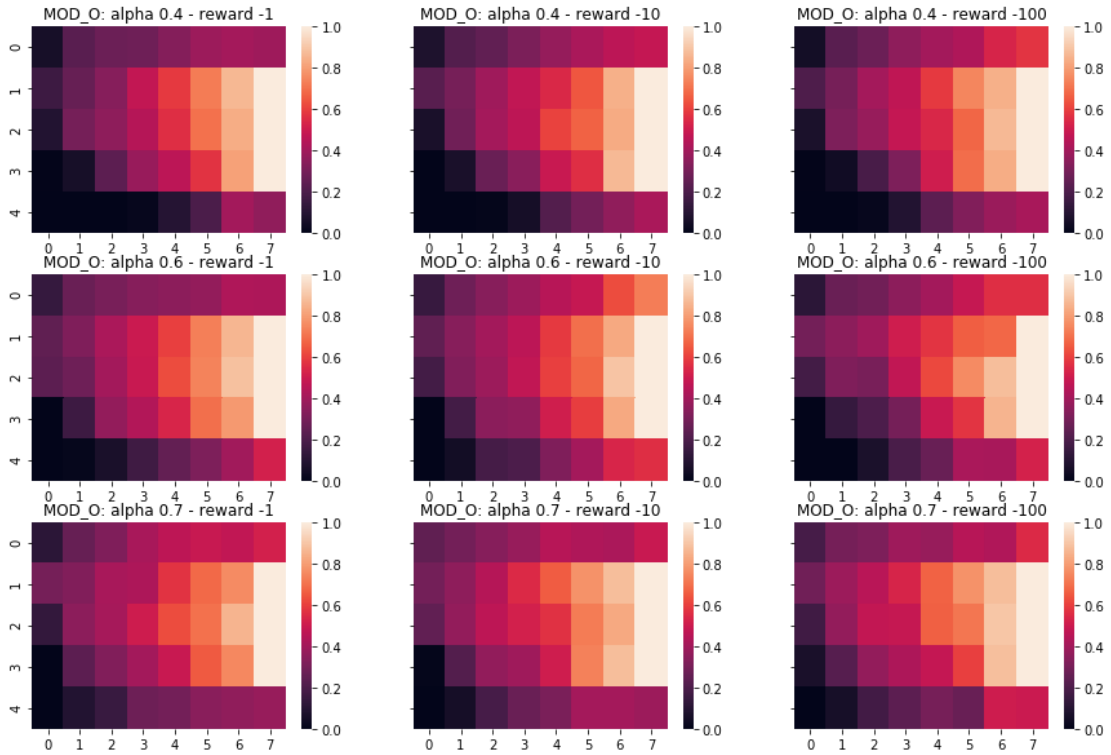


**Figure 6.** Q-table on the obstacle module.

Finally, the product module is trained on various alpha values and rewards after 1000 training rounds, and can be seen in **Figure 7.**

DOI: https://doi.org/10.17509/coelite.v1i1.43795
p- ISSN 2829-4157 e- ISSN 2829-4149

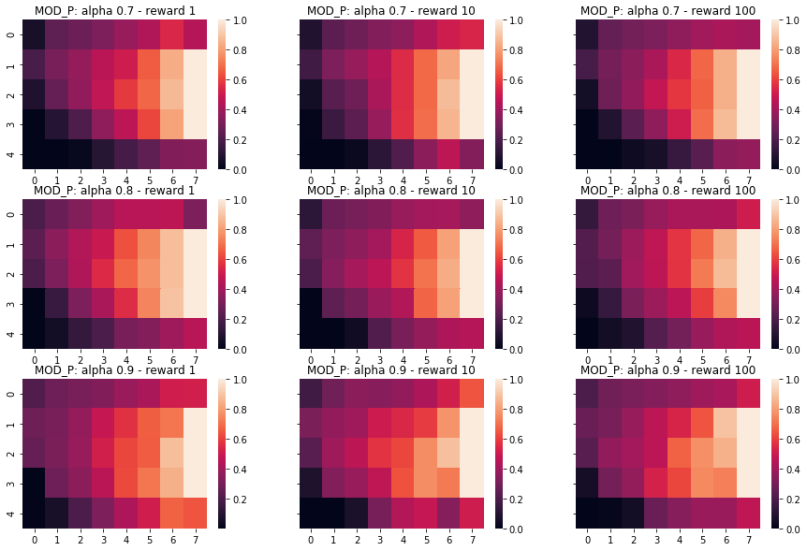


**Figure 7.** Q-table on the product module.

**Figure 8.** is showing a heatmap of the linear combinations of the three q-tables for the "up" action.

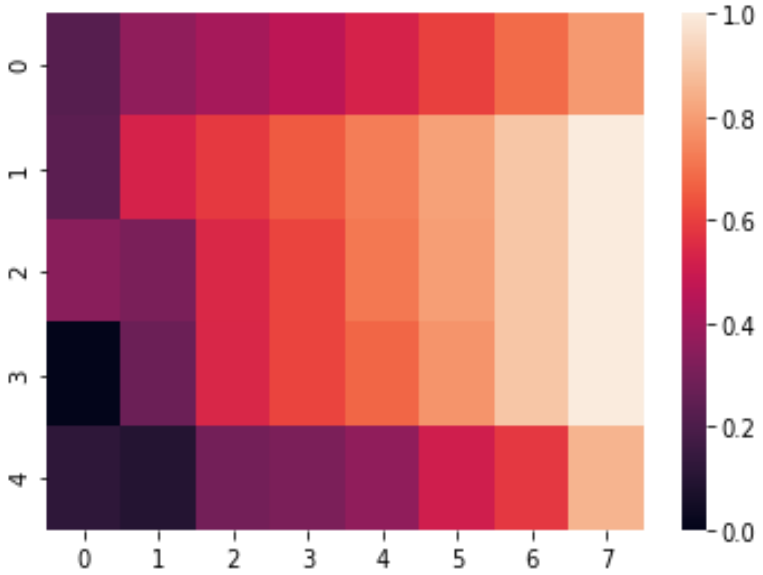

**Figure 8.** Q-Table combination module.

To balance exploration and exploitation during training, we used the epsilon-greedy method. We give a value of epsilon=0.9, which means with a probability of 0.9, an action is chosen randomly from the action space. With a probability of 0.1, an effort is selected greedily based on argmax(Q). The highest final reward is obtained with a learning rate of 0.8 in 500 rounds. More training provides more learning opportunities for agents, and this is the more stable the reward value will be. In other words, the agent is getting better at tracking, determining the action to the next state to get a positive reward.

DOI: https://doi.org/10.17509/coelite.v1i1.43795
p- ISSN 2829-4157 e- ISSN 2829-4149

## 4. CONCLUSION

Q-learning is a practical algorithm for RL problems with incomplete information. This works best when the agent has a limited number of actions to perform in an environment with random conditions. Moreover, it effectively divides the problem into separate modules that can be adjusted and linearly combined. The agent will learn more to get the best reward with the higher learning rate value used. Still, the risk that arises is the possibility of error also increasing because it affects the agent to ignore previous knowledge to explore options. The higher the epsilon value, it allows the agent to expand his knowledge for each action, which leads to long-term benefits, increases the accuracy of value estimates, and allows the agent to make more informed decisions in the future, but may result in reduced chances of getting the most rewards. RL with the q-learning algorithm can also be used to determine the shortest route or optimize the movement of the robot in collecting rewards and avoiding certain states that can result in losses. The knowledge obtained from this research is expected to be a suggestion for system improvement or identification of influential variables related to tracking.

## 5. AUTHORS' NOTE

The authors declare that there is no conflict of interest regarding the publication of this article. Authors confirmed that the paper was free of plagiarism.

## 6. REFERENCES

[1] Goli, A., Khademi-Zare, H., Tavakkoli-Moghaddam, R., Sadeghieh, A., Sasanian, M., and Malekalipour-Kordestanizadeh, R. (2021). An integrated approach based on artificial intelligence and novel meta-heuristic algorithms to predict demand for dairy products: a case study. *Network: Computation in Neural Systems*, *32*(1), 1-35.

[2] Sivaram, M., Batri, K., Amin Salih, M., and Porkodi, V. (2019). Exploiting the local optima in genetic algorithm using tabu search. *Indian Journal of Science and Technology*, *12*(1), 1-13.

[3] Yang, B., Guo, L., Guo, R., Zhao, M., and Zhao, T. (2020). A novel trilateration algorithm for RSSI-based indoor localization. *IEEE Sensors Journal*, *20*(14), 8164-8172.

[4] Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., and Hassabis, D. (2019). Reinforcement learning, fast and slow. *Trends in Cognitive Sciences*, *23*(5), 408-422.

[5] Dabbaghjamanesh, M., Moeini, A., and Kavousi-Fard, A. (2020). Reinforcement learning-based load forecasting of electric vehicle charging station using q-learning technique. *IEEE Transactions on Industrial Informatics*, *17*(6), 4229-4237.

[6] Faruk, A., and Cahyono, E.S. (2018). Prediction and classification of low birth weight data using machine learning techniques. *Indonesian Journal of Science and Technology*, *3*(1), 18-28.

[7] Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020). Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, *21*(1), 5183-5244.

[8] Netrapalli, P. (2019). Stochastic gradient descent and its variants in machine learning. *Journal of the Indian Institute of Science*, *99*(2), 201-213.

[9] Jang, B., Kim, M., Harerimana, G., and Kim, J. W. (2019). Q-learning algorithms: a comprehensive classification and applications. *IEEE Access*, *7*(1), 133653-133667.

DOI: https://doi.org/10.17509/coelite.v1i1.43795
p- ISSN 2829-4157 e- ISSN 2829-4149

[10] Peng, Z., Luo, R., Hu, J., Shi, K., Nguang, S. K., and Ghosh, B. K. (2021). Optimal tracking control of nonlinear multiagent systems using internal reinforce q-learning. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(8), 4043-4055.

[11] Liu, R., Nageotte, F., Zanne, P., de Mathelin, M., and Dresp-Langley, B. (2021). Deep reinforcement learning for the control of robotic manipulation: a focused mini-review. *Robotics*, *10*(1), 1-22.

[12] Zhang, T., and Mo, H. (2021). Reinforcement learning for robot research: a comprehensive review and open issues. *International Journal of Advanced Robotic Systems*, *18*(3), 1-22.

[13] Jiang, S., Huang, Z., and Ji, Y. (2020). Adaptive UAV-assisted geographic routing with q-learning in vanet. *IEEE Communications Letters*, *25*(4), 1358-1362.

[14] Jiang, L., Huang, H., and Ding, Z. (2019). Path planning for intelligent robots based on deep q-learning with experience replay and heuristic knowledge. *IEEE/CAA Journal of Automatica Sinica*, *7*(4), 1179-1189.

[15] Dittrich, M. A., and Fohlmeister, S. (2020). Cooperative multi-agent system for production control using reinforcement learning. *CIRP Annals*, *69*(1), 389-392.

[16] Li, Q., Meng, X., Gao, F., Zhang, G., and Chen, W. (2021). Approximate cost-optimal energy management of hydrogen electric multiple unit trains using double q-learning algorithm. *IEEE Transactions on Industrial Electronics*, *69*(9), 9099-9110.

[17] Vimal, S., Khari, M., Crespo, R. G., Kalaivani, L., Dey, N., and Kaliappan, M. (2020). Energy enhancement using multiobjective ant colony optimization with double q-learning algorithm for IoT based cognitive radio networks. *Computer Communications*, *154*(1), 481-490.

[18] Boussakssou, M., Hssina, B., and Erittali, M. (2020). Towards an adaptive e-learning system based on q-learning algorithm. *Procedia Computer Science*, *170*(1), 1198-1203.

[19] Genders, W., and Razavi, S. (2019). Asynchronous n-step q-learning adaptive traffic signal control. *Journal of Intelligent Transportation Systems*, *23*(4), 319-331.

[20] Qiu, C., Yao, H., Yu, F. R., Xu, F., and Zhao, C. (2019). Deep q-learning aided networking, caching, and computing resources allocation in software-defined satellite-terrestrial networks. *IEEE Transactions on Vehicular Technology*, *68*(6), 5871-5883.

DOI: https://doi.org/10.17509/coelite.v1i1.43795
p- ISSN 2829-4157 e- ISSN 2829-4149