A robust exploration strategy in reinforcement learning based on temporal difference error.

HAJAR, M.S., KALUTARAGE, H. and AL-KADRI, M.O.

2022

A Robust Exploration Strategy in Reinforcement Learning Based on Temporal Difference Error

Muhammad Shadi Hajar[1][0000−0002−5455−6931], Harsha Kalutarage[1][0000−0001−6430−9558], and M. Omar Al-Kadri[2][0000−0002−1146−1860]

[1] Robert Gordon University, Aberdeen AB10 7GJ, UK
{m.hajar, h.kalutarage}@rgu.ac.uk
[2] Birmingham City University, Birmingham B4 7XG, UK
omar.alkadri@bcu.ac.uk

Abstract. Exploration is a critical component in reinforcement learning algorithms. Exploration exploitation trade-off is still a fundamental dilemma in reinforcement learning. The learning agent needs to learn how to deal with a stochastic environment in order to maximize the accumulated long-term reward. This paper proposes a robust exploration strategy (RES) based on the temporal difference error. In RES, the exploration problem is modeled using Beta probability distribution to control the exploration rate. Moreover, the most promising action is selected during the exploration with a view to maximizing the accumulated reward and avoiding un-rewardable wrong actions. RES has been evaluated on the k-armed bandit problem. The simulation results show superior performance without the need to tune parameters.

Keywords: Reinforcement Learning, Exploration, Exploitation, Q-Learning, k-Armed Bandit, ε-greedy, Softmax

1 Introduction

Reinforcement learning (RL) is a branch of machine learning where a learning agent tries to map situations to actions with a view to maximizing the long-term reward. Without prior knowledge, the learning agent must discover which actions are more rewardable. Taking action at any state affects not only the immediately received reward but all the subsequent rewards. Hence, it may prevent the learning agent from converging to the global optimum. This reliance on the return feedback from the environment necessitates the learning agent to explore the whole environment to take optimal actions and maximize the long-term rewards. Therefore, RL algorithms count on exploration to obtain sufficient informative feedback from the environment to exploit the most rewardable actions.

The exploration-exploitation trade-off is still a fundamental problem. When learning agents over-explore the environment, they cannot maximize the accumulated reward

as exploratory actions may return minimum rewards. Moreover, exploiting uncertain action-value functions may yield less reward and make the convergence suboptimal. Thus, this problem is known as the exploration-exploitation dilemma, which has been widely investigated by mathematicians and is still unresolved [1]. ε-greedy and softmax exploration methods are widely used in the literature to balance exploration-exploitation. These two straightforward algorithms perform well in some cases and thus are hard to beat [2]. However, they require rigorous parameter tuning and may not hold in dynamic environments.

The main contribution of this paper is proposing a robust exploration strategy (RES) based on Temporal Difference (TD) error to effectively balance exploration-exploitation with a view to maximizing the accumulated long-term reward. The well-known k-armed bandit problem has been used to demonstrate the robustness of the proposed method.

The remainder of this paper is organized into five sections as follows. Related work is given in Section 2. The used methodology is comprehensively discussed in Section 3, followed by the evaluation results in Section 4. Finally, Section 5 concludes the paper and highlights future work.

2 Related Work

Several approaches have been proposed to produce an efficient exploration algorithm in order to balance the exploration-exploitation trade-off. However, the main two approaches are the blind exploration approach, such as ε-greedy [1], and the value-based approach, such as softmax [3].

In blind exploration approach [3], the learning agent solely explores the environment based on randomly taken actions. It is a reward-free method to explore the environment without any kind of information. Thus the action selection process during exploration is uniform. In real life, ε-greedy is always the first choice for developers due to its simplicity and near-optimal results despite the time-consuming process of tuning its single parameter [4]. In this method, the parameter $\varepsilon \in [0, 1]$ controls the exploration rate as shown in Eq. 1. Although this method is widely adopted in RL, it has some drawbacks. First is the aforementioned issue of tuning the parameter ε. Second, choosing random action during the exploration may significantly degrade the learning agent performance as taking random action at time t could affect all future rewards negatively.

$$a_t = \begin{cases} \underset{a_t \in A_t}{argmax}\ Q(s,a) & with\ probability\ \ (1-\varepsilon) \\ a\ random\ action & with\ probability\ \ \varepsilon \end{cases} \tag{1}$$

In a value-based approach, the learning agent takes informative action based on its estimation of the action-value functions. The exploration-exploitation trade-off is achieved by assigning probabilities to the available actions at time step t based on the current action-value functions. The predominant algorithm in this approach is the softmax action selection algorithm [3]. Softmax method is usually modeled using Boltzmann distribution as shown in Eq. 2, where the greedy action is always chosen with the highest probability. In contrast, other possible actions are weighted according to their corresponding action-value functions.

$$\pi(a|s) = Pr\{a_t = a|s_t = s\} = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_1^n e^{\frac{Q(s,a)}{\tau}}} \quad (2)$$

where τ is the temperature parameter, and it is used to control choosing the greedy action. When τ is decreased, the greedy action probability increases, and when $\tau \to \infty$, all possible actions will have the same probability, and hence the action will be selected randomly, as in ε-greedy. Tuning the temperature parameter is not straightforward [3]. Moreover, as the probabilities are calculated based on the actual estimated values, the action selection process is highly influenced by these values, which necessities re-tuning the temperature parameter whenever the reward function has changed, even for the same problem.

In [5] and [2], the authors proposed two methods based on ε-greedy and softmax algorithms, respectively. In both works, the authors used a method called Value-Difference Based Exploration (VDBE), which is a state-dependent exploration probability method to control the exploration-exploitation trade-off. Although the simulation results show promising results, there is still a need to tune the sensitivity parameter σ , which is used in both methods.

3 Methodology

In this section, the exploration-exploitation dilemma has been introduced, and the design requirement has been presented. Moreover, the proposed method has been discussed comprehensively.

3.1 Overview

In RL, the intelligent learning agent interacts with the environment E through a series of state-action pairs with a view to maximizing the accumulative long-term reward. The agent is modeled using the tuple (S, A, R), where S represents a set of states, A is the action set, and R is the reward function, as illustrated in Fig. 1. In value-based RL algorithms, the learning agent uses value functions to estimate the future reward of taking action a in a given state s. Therefore, the action-value function for taking action a given a state (s) and following policy π could be defined as in Eq. 3. At each time step t, the learning agent in state $s \in S$ takes an action $a \in A$, receives a reward $r_{t+1} \in R$, and moves from state s to state s'

$$q_\pi(s,a) \doteq \mathbf{E}_\pi[G_t|S_t = s, A_t = a] = \mathbf{E}_\pi[\Sigma_{n=0}^{\infty}\gamma^n R_{t+n+1}|S_t = s, A_t = a], \quad \forall s \in S \quad (3)$$

where $\mathbf{E}_\pi[G_t|S_t = s]$ is the expected discounted reward when the agent starts at state s and follows the policy π and $\gamma \in [0, 1]$ is the discount factor for the future reward R.

3.2 Q-Learning

Temporal difference (TD) algorithms are a combination of Monte Carlo methods and Dynamic programming (DP) [6]. In TD learning, the agent learns directly from the

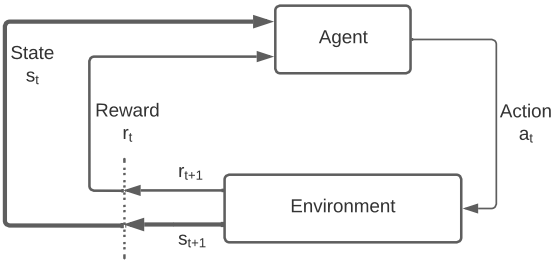

Fig. 1: RL Model

environment without any prior knowledge, and thus it is called a model-free method. Q-learning is a temporal difference algorithm widely used to predict the action-value function $q_\pi(s, a)$ without any prior knowledge using trials and errors [7]. In Q-learning, the action-value function $Q(s_t, a_t)$ is updated independently of the policy followed by the agent to approximate the optimal action-value function q_*, which makes it applicable for many problems. Q-learning is proven to converge to q_*; however, there is still a requirement that all states still need to be visited and updated.

$$a_t^{*(i)} = \underset{a_t \in A}{argmax} Q_t(s_{t+1}, a_t) \tag{4}$$

$$Q_{t+1}(s_t, a_t) \leftarrow (1 - \alpha) Q_t(s_t, a_t) + \alpha [r_{t+1}(s_{t+1}) + \gamma \underset{a \in A}{max} Q_t(s_{t+1}, a_t^*)] \tag{5}$$

where $\alpha \in [0, 1]$ is the learning step size where small values slow the learning, and higher values could cause oscillations, and $\gamma \in [0, 1]$ is the discount factor where small values make the learning agent nearsighted by ignoring the future rewards.

3.3 The Proposed Method

In RL, the environment can be modeled as a Markov decision process (MDP), as illustrated in Fig.1. The learning agent interacts with the environment at a series of discrete-time steps $t = 0, 1, 2, ..$, and receives a reward after each interaction. In order to estimate the optimal policy π_*, the learning agent must balance between exploration and exploitation. In TD algorithms, the value functions are updated using an error function representing the difference between the predicted reward at a given state and the actually received reward, as shown in Eq. 6, which represents the TD error in Q-learning.

$$\delta_t \doteq r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{6}$$

Thus, the Q-learning updating algorithm could be described as in Eq. 7.

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha \delta_t \tag{7}$$

When the Q function converges to the optimal value $Q \rightarrow Q_*$, the TD error converges to zero $\delta_t \rightarrow 0$. Therefore, the TD error could be used to estimate the exploration ratio.

The learning agent should explore more when the TD error is high and less when it is small. This approach seems reasonable. However, there are some limitations. The TD error suffers from oscillations, which means that the learning agent could over-explore the environment. Moreover, there is a need to pick the most promising action to perform during the exploration phase. Therefore, the TD error could be used to evaluate the Q values differences trend, which then will be used to choose the most promising action. The TD error trend is exponentially smoothed using Eq. 8.

$$\Delta_{t+1} = \omega\Delta_t + (1-\omega)\delta_t \tag{8}$$

RES uses a dynamic exploration rate θ. It has been evaluated using Beta probability distribution as in Eq. 9.

$$\theta_t = \frac{\alpha_t}{\alpha_t + \beta_t} \tag{9}$$

where α and β are the Beta distribution levels. After each action, the TD error δ_t and the exponentially smoothed trend Δ_t are evaluated as described in Algorithm 1. This process is modeled using Beta probability distribution to explore more when there is a promising action to explore. After each action, the learning agent compare δ_t and Δ_t. If the TD error is less than or equal to the smoothed trend, this action is considered a successful action, and the beta levels are updated using Eq. 10. Otherwise, it will be unsuccessful, and the levels will be updated using Eq. 11

$$\begin{aligned} \alpha_t &= \lambda\alpha_{t-1} + 1 \\ \beta_t &= \lambda\beta_{t-1} \end{aligned} \tag{10}$$

$$\begin{aligned} \alpha_t &= \lambda\alpha_{t-1} \\ \beta_t &= \lambda\beta_{t-1} + 1 \end{aligned} \tag{11}$$

where $\lambda \in [0, 1]$ is the longevity factor, which used to give more weight to recent observations. Adopting this method has two advantages. First, the amount of exploration will be decreased gradually for a static environment, which means over time, the learning agent will exploit the greedy action more to maximize the overall reward. Second, in a dynamic environment, this method is able to reflect the environment dynamicity into more exploration to help the algorithm re-converge to the global optimum. Therefore, higher values of λ give more weight to previous observations and fit dynamic environments, while smaller values give more weight to current observations and fit static environments.

The second important thing is choosing a promising action during the exploration instead of a random one. Randomly chosen action could be catastrophic in some applications as the chosen action at time t may affect all the future rewards. Therefore, in RES, more weight is given to actions with a high smoothed difference as they are regarded as promising actions to discover. To achieve that, RES uses the Boltzmann distribution [8] of the smoothed differences to calculate the weighted probabilities of the available actions, as shown in Eq. 12.

$$P_{\Delta_t^i} = \frac{e^{\Delta_t^i}}{\Sigma e^{\Delta_t}} : \forall i \in A \tag{12}$$

Algorithm 1: Updating Algorithm

Input: The exponentially smoothed trend: Δ^i_{t-1}
Input: The exploration threshold: θ_{t-1}
Input: The beta distribution levels: α and β
Output: The updated values $\Delta^i_t, P_{\Delta^a_t}, \theta_t$
$\delta^i_t = [r^i_{t+1} + \gamma Q^i(s_{t+1}, a_{t+1}) - Q^i(s_t, a_t)]$
$\Delta^i_t = \omega.\Delta^i_{t-1} + (1-\omega).\delta^i_t$
if $|\delta^i_t| \leq |\Delta^i_t|$ **then**
    $\alpha = \lambda\alpha + 1$
    $\beta = \lambda\beta$
else
    $\alpha = \lambda\alpha$
    $\beta = \lambda\beta + 1$
$P_{\Delta^i_t} = \frac{e^{\Delta^i_t}}{\Sigma e^{\Delta_t}} : \forall i \in A$
$\theta_t = \frac{\alpha}{\alpha+\beta}$

The aforementioned parameters are updated whether it is an exploration or exploitation cycle as shown in Algorithm 2. The uniform random number $\rho \in [0, 1]$ is drawn for each time step to choosing whether to explore or exploit. If it is an exploitation cycle, then the action will be chosen greedily. Otherwise, it will be weighted randomly chosen based on the Boltzmann distribution of the exponentially weighted TD errors.

4 Evaluation

The proposed method has been evaluated on the k-armed bandit problem, which is one of the common RL problems to evaluate the explorations exploitation methods [5]. It has been widely used to model different problems, such as economic [9] and routing problems [10, 11]. The problem represents a bandit machine with multiple arms, pulling one of which gives the player a variable reward. The reward is usually stochastic and drawn from a pre-defined probability distribution. Hence, the learning agent (player) with no prior knowledge has to learn from the pulled uncertain rewards the most rewardable arm by exploring them, which leads to a loss in the gained rewards over time.

4.1 Experimental Setup

The proposed method has been evaluated over long run to prove that it converges to the optimal solution and achieves the maximum reward. The bandit machine consists of 10 levers as described in [1]. At each lever pull, the learning agent gets a stochastic reward drawn from a randomly defined Gaussian distribution $\mathcal{N}(10, 1)$, with mean $Q_*(s, a) = 10$ and standard deviation $\sigma = 1$. The learning agent can improve its selection policy within 2000 trials. Each experiment has been repeated 1000 times, and then the results have averaged out and reported.

RES has been contrasted with ε-greedy [1], softmax action selection[3], and VDBE-Softmax [2]. To ensure a fair comparison, the exploration rate of ε-greedy and softmax

Algorithm 2: Exploration algorithm

Initialize Q values.
Initialize the differences probabilities:
$P_{\Delta_0^a} = \frac{1}{|A_0|} : \forall a \in A$
Initialize the exploration parameters:
$\alpha = 1, \beta = 1$
$\theta_0 = \frac{\alpha}{\alpha+\beta}$
while *TRUE* **do**
  $\rho \leftarrow rand(0..1)$
  if $\rho \geq \theta$ **then**
    $a_{t+1} \leftarrow \underset{a_t \in A_t}{argmax}\ Q_t(s_t, a_t)$
    Parameters updating using Algorithm 1
    $Q_{t+1}(s_t, a_t) \leftarrow (1-\alpha)Q_t(s_t, a_t) + \alpha[r_{t+1}(s_{t+1}) + \gamma \underset{a \in A}{max} Q_t(s_{t+1}, a_t)]$
  else
    $a_{t+1} \leftarrow weighted\_rand(P_{\Delta_t})$
    Parameters updating using Algorithm 1
    $Q_{t+1}(s_t, a_t) \leftarrow (1-\alpha)Q_t(s_t, a_t) + \alpha[r_{t+1}(s_{t+1}) + \gamma \underset{a \in A}{max} Q_t(s_{t+1}, a_t)]$

have been optimized. At the same time, the optimized value of the VDBE method for the $k-$armed bandit problem is adopted as reported in [5]. In RES, the longevity factor is a decay factor for the single exponential smoothing of the Beta distribution levels. Its value is set to 0.9 to slowly decrease the weight of old observations over time, which is widely used in the literature [12].

4.2 Simulation Results

In the first experiment, the simulation was run for a high number of iterations to ensure that all methods are converged as some methods show high performance at the beginning; however, over time, they perform poorly. Fig. 2 shows the average reward of the four exploration methods. Both softmax and VDBE-Softmax perform well at the beginning as both use the Q function to choose actions during exploration. ε-greedy performs poorly at the beginning but shows good performance over time. The reason behind this behavior is that randomly chosen actions during exploration need more time to build the Q table, which makes even the exploitation actions inefficient. However, the performance enhances significantly when the learning agent gets more evidence from the environment over time.

On the other hand, RES shows superior performance over the long run. As it depends on the exponentially smoothed trend of the TD error, it needs a few exploration steps to estimate Δ_t. Once it gets a few exploration outcomes from the environment, it tends to explore the most promising actions based on its observations. Unlike other exploration methods, the convergence in RES is achieved separately. The most promising Q function converges faster than others as it will be excessively chosen, as shown clearly in Fig. 3, which illustrates the smoothed trend of the TD error over time. Evidently, Δ^1 converges

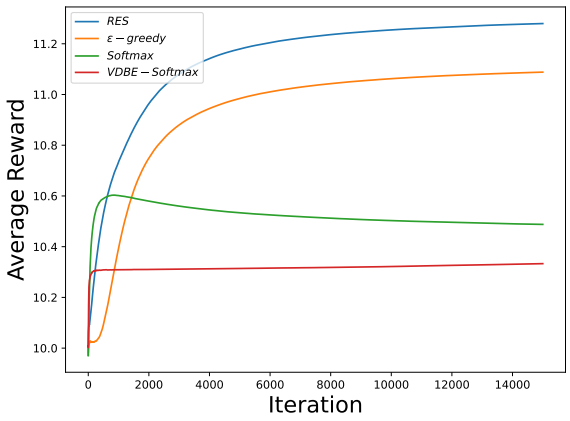

Fig. 2: The reward over time

faster to zero than others, although all other values converge over time but considerably slower than Δ^1. The reason behind this behavior is that Δ^1 is the most promising action to take. This could be obviously seen in Fig. 4, which shows the Q values for all actions over time. The most rewardable action is action one; hence, RES explored this action more at the beginning, making it converge faster than other actions. Moreover, the high oscillations of this action value confirm that RES always exploits this action to gain maximum rewards.

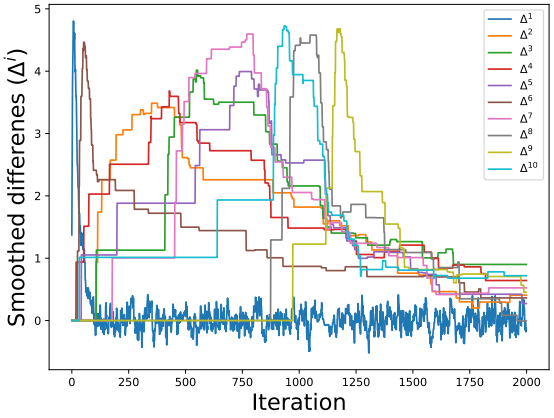

Fig. 3: The exponential smoothed differences over time

In the second experiment, the adaptability of RES is evaluated for a dynamic environment, where the action-value functions may get changed during the simulation. When the environment changes, the optimal action given a state s may become the

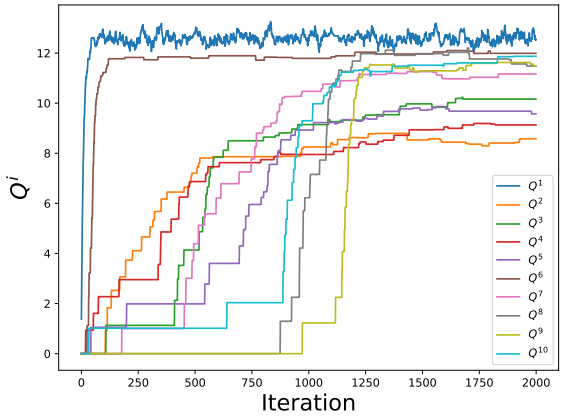

Fig. 4: Q values over time

worst. In light of that, the learning agent should act intelligently and explore the environment to re-converge to the global optimum again. After 50% of the iterations, the Gaussian distribution of the reward function has been changed randomly using the distribution $\mathcal{N}(rand(1,15), rand(1,5))$. Fig. 5 and 6 show the average static and dynamic environment rewards, respectively. For a static environment, as illustrated in Fig. 5, RES performs steadily and achieves the highest average reward over time. Moreover, when the environment has changed, as shown in Fig. 6, RES shows high adaptability to re-converge to the new global optimum. Once the change happens, it will be reflected by a change in the TD error and the exponential smoothed trend of the TD error. The former increase the exploration rate, while the latter makes the algorithm chooses the newest promising action.

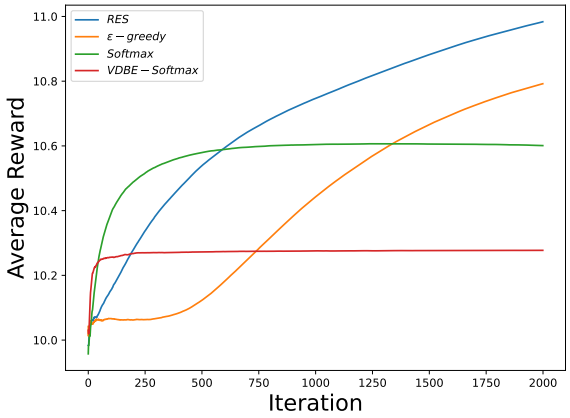

Fig. 5: The reward over time for static environment

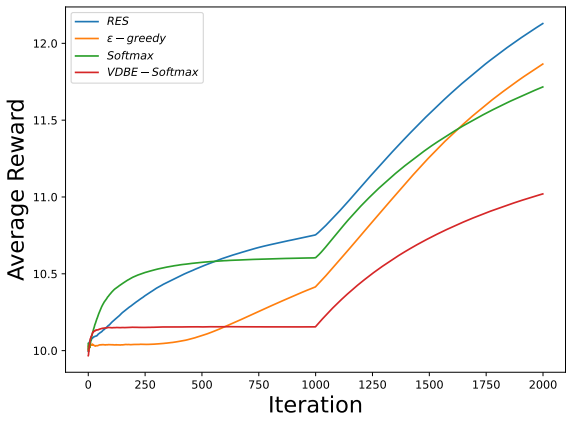

Fig. 6: The reward over time for dynamic environment

5 Conclusion and Future Work

Balancing the exploration-exploitation trade-off in reinforcement learning is still an open area of research. The dilemma lies between exploring the environment to obtain more informative data, which could be un-rewardable, or selecting actions that the learning agent can expect their reward. Our proposed action selection strategy, RES, presents a robust promising solution without the parameter tuning overhead. RES is an adaptive exploration strategy based on the exponentially smoothed trend of the temporal difference error and Beta probability distribution. This novel approach demonstrates outstanding performance on the well-known k-armed bandit problem. In the future, it will be evaluated on other temporal difference algorithms, such as SARSA. Moreover, it will be deployed on a more complicated problem to study its behavior, such as reinforcement learning-based routing protocols.

References

1. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press, 2018).
2. Tokic, M. & Palm, G. *Value-difference based exploration: adaptive control between epsilon-greedy and softmax* in *Annual conference on artificial intelligence* (2011), 335–346.
3. Amin, S., Gomrokchi, M., Satija, H., van Hoof, H. & Precup, D. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157* (2021).
4. Vermorel, J. & Mohri, M. *Multi-armed bandit algorithms and empirical evaluation* in *European conference on machine learning* (2005), 437–448.

5. Tokic, M. *Adaptive ε-greedy exploration in reinforcement learning based on value differences* in *Annual Conference on Artificial Intelligence* (2010), 203–210.
6. Apostol, K. Temporal Difference Learning. *SaluPress* **2012,** 96 (2012).
7. Jang, B., Kim, M., Harerimana, G. & Kim, J. W. Q-learning algorithms: A comprehensive classification and applications. *IEEE access* **7,** 133653–133667 (2019).
8. Tijsma, A. D., Drugan, M. M. & Wiering, M. A. *Comparing exploration strategies for Q-learning in random stochastic mazes* in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (2016), 1–8.
9. Li, J. The k-armed bandit problem with multiple priors. *Journal of Mathematical Economics* **80,** 22–38 (2019).
10. De Oliveira, T. B., Bazzan, A. L., da Silva, B. C. & Grunitzki, R. *Comparing multi-armed bandit algorithms and Q-learning for multiagent action selection: A case study in route choice* in *2018 International Joint Conference on Neural Networks (IJCNN)* (2018), 1–8.
11. Hajar, M. S., Kalutarage, H. & Al-Kadri, M. O. *DQR: A Double Q Learning Multi Agent Routing Protocol for Wireless Medical Sensor Network* in *18th EAI International Conference on Security and Privacy in Communication Networks (SecureComm)* (2022).
12. Hajar, M. S., Al-Kadri, M. O. & Kalutarage, H. *Ltms: A lightweight trust management system for wireless medical sensor networks* in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (2020), 1783–1790.