©2025 Published in 11th International Symposium on Intelligent Technologies in Engineering and Science 16-18 May 2025 (ISITES2025 Diyarbakır - Türkiye)

Comparative Analysis of TD-Based Reinforcement Learning Algorithms

[1]Ahmet KALA, [2]Süleyman UZUN, and [2,3]Cem ÖZKURT
*[1]Department of Information Technologies, Sakarya University of Applied Sciences, Türkiye
[2]Department of Computer Engineering, Sakarya University of Applied Sciences, Türkiye
[3]AI and Data Science Research and Application Center, Sakarya University of Applied Sciences, Türkiye

Abstract

Reinforcement Learning has emerged as a fundamental framework in artificial intelligence, enabling agents to optimize decision-making strategies through interaction with their environment. Among RL techniques, Temporal-Difference (TD) learning stands out due to its efficiency in updating value functions incrementally, combining the strengths of Monte Carlo methods and Dynamic Programming. This study focuses on analyzing and comparing the performance of TD(0), State Action Reward State Action (SARSA), and Expected SARSA algorithms in various reinforcement learning scenarios. By conducting experiments in dynamic environments such as the Sliding Block and Windy Gridworld problems, we evaluate the adaptability, stability, and efficiency of these TD-based methods. The results demonstrate that Expected SARSA exhibits superior stability and learning performance compared to SARSA and Q-learning, particularly in high-variance environments. Our findings provide valuable insights into the effectiveness of TD-based algorithms and contribute to the ongoing development of reinforcement learning strategies for complex decision-making tasks.

Key words: Reinforcement Learning (RL), Temporal-Difference learning (TD), State Action Reward State Action (SARSA), Monte Carlo methods

1. Introduction

Reinforcement Learning (RL) has emerged as a fundamental paradigm in artificial intelligence, enabling agents to learn optimal decision-making strategies through interactions with their environment [1]. Unlike supervised learning, which relies on labeled datasets, RL enables learning through trial and error, where an agent receives feedback in the form of rewards and adjusts its actions accordingly. This approach has been widely applied in areas such as robotics, game playing, and autonomous systems [2-6].

Among RL techniques, Temporal-Difference (TD) learning has gained significant attention due to its ability to combine the strengths of Monte Carlo (MC) methods and Dynamic Programming (DP). Introduced by Sutton [7], TD learning estimates value functions incrementally by updating predictions based on observed rewards and future estimates rather than waiting for complete episodes. This key characteristic allows TD learning to be more data-efficient and adaptable to online learning scenarios, distinguishing it from MC methods, which require full episode completion before updating estimates [1].

The effectiveness of TD learning has been demonstrated in numerous applications. One of the most notable examples is TD-Gammon, an AI system developed by Tesauro [8] that utilized TD learning to achieve superhuman performance in backgammon. Moreover, theoretical studies have provided

*Corresponding author: Address: Department of Information Technologies, Sakarya University of Applied Sciences, 54187, Sakarya TURKİYE. E-mail address: ahmetkala@subu.edu.tr, Phone: +0264 616 0054

insights into the convergence properties of TD learning. Tsitsiklis and Van Roy [9] analyzed the stability of TD(0) with function approximation and established its convergence under specific stochastic conditions. Further empirical studies have shown that TD methods tend to generalize better than MC methods in RL problems [10-12].

In the control domain, TD learning serves as the foundation for various on-policy and off-policy RL algorithms. One of the most widely used on-policy methods is State Action Reward State Action (SARSA), introduced by Rummery and Niranjan (1994) [13]. SARSA learns action-value functions based on the current policy, making it suitable for dynamic and uncertain environments. However, one limitation of SARSA is its high variance in updates, which led to the development of Expected SARSA, a refined version proposed by van Seijen et al. [14]. Expected SARSA mitigates variance by computing the expected value of the next state-action pair, leading to smoother updates and improved stability. Empirical evaluations, such as the Cliff Walking problem, have shown that Expected SARSA converges faster and exhibits lower variance compared to standard SARSA [14]. The theoretical and empirical results from various studies support its efficacy, particularly when combined with extensions such as eligibility traces and hybrid control frameworks [15-19].

This study aims to analyze and compare the performance of TD(0), SARSA, and Expected SARSA in RL tasks. Specifically, it investigates the advantages of TD-based prediction methods over MC methods and evaluates the efficiency of SARSA and Expected SARSA in control tasks. By conducting experimental comparisons under varying learning conditions, this research provides insights into the effectiveness of TD-based algorithms in real-world RL scenarios. Furthermore, the study explores the impact of different hyperparameter settings, such as learning rates and exploration strategies, on the convergence and stability of these methods.

2. Related Works

RL has been a fundamental topic in artificial intelligence, with TD learning emerging as one of its most influential techniques [1]. TD learning integrates the sampling nature of MC methods with the bootstrapping capability of Dynamic Programming (DP), making it effective for both offline and online learning [2]. Unlike MC, TD updates value functions based on the next observed reward rather than waiting until an episode ends, leading to faster learning and adaptability [1].

The effectiveness of TD learning has been widely demonstrated. Tesauro [8] developed TD-Gammon, a backgammon-playing AI, showcasing TD learning's real-world potential. Theoretical properties of TD(0) were explored by Tsitsiklis & Van Roy [9], proving its convergence under stochastic conditions, while Szepesvári [10] experimentally showed that TD learning converges faster than MC methods. Comparisons between these methods were further investigated by Sutton [1], who found that TD learning exhibits greater stability in Random Walk problems, and by Doya [20], who explored its applicability in continuous-time RL. Additionally, Bertsekas [2] examined its role in optimal control problems, emphasizing its ability to generalize in dynamic environments.

Several TD-based control algorithms have been introduced to enhance learning efficiency.

Rummery & Niranjan (1994) developed SARSA, an on-policy learning method successfully applied in environments requiring continuous adaptation, such as Windy Gridworld [13]. Later, van Seijen et al. [14] proposed Expected SARSA, which reduces variance by incorporating the expected value of all possible actions instead of a single sampled action, improving convergence speed in environments like Cliff Walking. Moreover, Silver et al. [13] examined TD learning in online planning, and Konda & Tsitsiklis [21] introduced Actor-Critic algorithms, integrating TD learning into policy optimization.

This study builds on previous research by comparing TD(0) and MC methods, analyzing the performance of SARSA and Expected SARSA in various learning scenarios, and investigating how different hyperparameter settings impact TD-based methods. The findings aim to provide deeper insights into optimizing RL for dynamic and complex environments.

3. Materials and Method

3.1. Environment

RL is a machine learning approach that enables an agent to interact with its environment and learn optimal decision-making strategies. In this study, Q-learning, SARSA, and Expected SARSA algorithms were employed to analyze agent performance under varying environmental conditions. The Sliding Block and Windy Gridworld environments were designed to evaluate the exploration-exploitation trade-off and learning dynamics. During the training process, the ε-greedy exploration strategy was utilized to examine how agents balance exploration and exploitation, and the algorithms were compared under different learning rates (α). This framework provides a robust foundation for assessing the adaptability of RL algorithms to dynamic environments and analyzing their learning efficiency.

3.2. SARSA (State Action Reward State Action)

SARSA is an on-policy Temporal Difference (TD) learning method that enables agents to interact with their environment and learn optimal action strategies. While updating Q-values, it also considers future rewards. With the **ε-greedy** exploration strategy, the agent selects random actions with a certain probability for exploration while choosing the best-known actions for exploitation. Unlike off-policy methods such as **Q-learning**, SARSA updates based on the current policy, making it effective in dynamic environments with changing reward structures. Discounting future rewards and managing the exploration-exploitation balance are critical factors that enhance the efficiency of the algorithm.

The SARSA algorithm updates the Q-values using the Temporal Difference (TD) learning update:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (1)$$

Where:

- $Q(s_t, a_t)$ → Q-value for state action pair (s_t, a_t)
- α → Learning rate
- r_{t+1} → Reward received after taking action a_t
- γ → Discount factor for future rewards
- $Q(s_{t+1}, a_{t+1})$ → Q value of the next state action pair (s_{t+1}, a_{t+1})

Since SARSA is an on-policy algorithm, the next action a_{t+1} is selected based on the updated policy.

SARSA is typically used with the ε-greedy exploration stategy, where the agent balances exploration and exploitation:

$$a = \begin{cases} argmax_a Q(s,a), & (probability: 1-\epsilon)\ exploitation\ selected\ best\ action \\ Random\ action, (probability: \epsilon) exploration\ selecting\ a\ random\ action \end{cases} \quad (2)$$

Where:

- ϵ → Exploration rate
- With probability $1 - \epsilon$, theagent chooses the action with the highest known reward
- With probability ϵ, the agent selects a random action for exploration

SARSA accounts for future rewards by discounting then over time. This process is modeled by the discount factor (γ):

$$V(s) = \sum_{t=0}^{\infty} \gamma^t r_t \quad (3)$$

Where:

- $V(s)$ → Expected long term reward for state s
- γ →Discount factor, controlling how much future rewards influence current decisions.
- - $\gamma = 0$ → Only immediate rewards matter.
- - $\gamma \approx 1$→ Future rewards are given more importance

3.3. Expected SARSA

Expected SARSA is an improved version of standard SARSA and an on-policy Temporal Difference (TD) learning method. Unlike traditional SARSA, it updates the Q-value by computing the expected value of the next action rather than using a single sampled action. This approach

enhan**ces** the exploration-exploitation balance by considering the expected reward of all possible actions and directly integrates the effect of the **ε**-greedy exploration strategy, resulting in smoother updates and a more stable learning process. The key advantage of Expected SARSA is that it stabilizes Q-value updates, making learning more efficient, especially in environments with high learning rates or dynamic reward structures.

Instead of using the next selected action directly, Expected SARSA updates Q- values based on the expected value of the next action:

$$Q(s_t, a_t)\ ) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \mathbb{E}Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (4)$$

Where:

- $Q(s_t, a_t) \rightarrow$ Q-value for state action pair (s_t, a_t)
- $\alpha \rightarrow$ Learning rate
- $r_{t+1} \rightarrow$ Reward received after taking action a_t
- $\gamma \rightarrow$ Discount factor for future rewards
- $\mathbb{E}[Q(s_{t+1}, a)] \rightarrow$ Expected Q-values computed as the weighted average of all possible actions.

Expected SARSA does not use a single next action but instead calcultes the expected Q-value under the ε-greedy policy:

$$\mathbb{E}[Q(s_{t+1}, a)] = (1 - \epsilon)Q(s_{t+1}, a^*) + \epsilon \sum_a \frac{Q(s_{t+1}, a)}{|A|} \quad (5)$$

Where:

- $a^* \rightarrow$ Besy known action $(argmax_a Q(s_{t+1}, a))$
- $\epsilon \rightarrow$ Exploration rate
- $|A| \rightarrow$ Total number of actions
- $(1 - \epsilon) \rightarrow$ Probability of selecting the best known action
- $\epsilon \sum_a \frac{Q(s_{t+1}, a)}{|A|} \rightarrow$ Average Q-value of randomly selected actions

Expected SARSA follows the ε-greedy strategy for exploration and exploitation:

$$a = \begin{cases} argmax_a Q(s, a), & (probability: 1 - \epsilon)\ \ (explotation) \\ Random\ action, & (probability: \epsilon) \quad (exploration) \end{cases} \quad (6)$$

This approach allows the agent to primarily select the best known action while occasionally exploring alternatives to improve long-term learning.

3.4. Q-Learning

Q-learning is an off-policy TD learning algorithm and one of the most widely used methods in RL. Unlike on-policy methods like SARSA, it learns by considering the highest possible future reward rather than following the current policy. The agent updates Q-values based on the best future

reward, regardless of the next action taken, allowing it to learn the optimal long-term strategy. Due to its off-policy nature, it enables more extensive exploration and faster learning, but high exploration rates (ε) can lead to instability in the learning process.

Q learning updates Q-values based on the maximum future reward:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (7)$$

Where:

- $(Q(s_t, a_t) \rightarrow$ Q-value fort he state-action pair (s_t, a_t)
- $\alpha \rightarrow$ Learning rate
- $r_{t+1} \rightarrow$ Reward received after taking action a_t
- $\gamma \rightarrow$ Discount factor for future rewards
- $max_a Q(s_{t+1}, a) \rightarrow$ Maximum Q-value of the next state

Since Q-learning selects the best possible action (maxQ) in the update step, it is considered an off-policy method.

Q-learning follows the ϵ-greedy strategy to balance exploration and exploitation:

$$a = \begin{cases} argmax_a Q(s, a), & (probability: 1 - \epsilon) \ \ (exploitation) \\ Random\ action, & (probability: \epsilon) \quad (exploration) \end{cases} \quad (8)$$

Where:

- 1-$\epsilon \rightarrow$ Probability of selecting the best-know action
- $\epsilon \rightarrow$ Probability of taking a random action for exploration

Since Q-learning learns by selectingthe best possible action rather than following a fixed policy, it accelerates the learning process but may lead to instability in highly dynamic environments.

Future rewards are modeled using the discount factor (γ):

$$V(s) = \sum_{t=0}^{\infty} \gamma^t r_t \quad (9)$$

Where:

- $V(s) \rightarrow$ Expected long-term reward for state s
- $\gamma \rightarrow$ Discount factor
  - $\gamma = 0 \rightarrow$ Only immadiate rewards matter
  - $\gamma \approx 1 \rightarrow$ Future rewards are given more importance.

4. Result and Finding

This section presents the results of experiments with Q-learning, SARSA, and Expected SARSA, evaluating learning efficiency and performance stability.The results are structured into an overview of learning performance, graphical analysis, and a discussion of key findings.

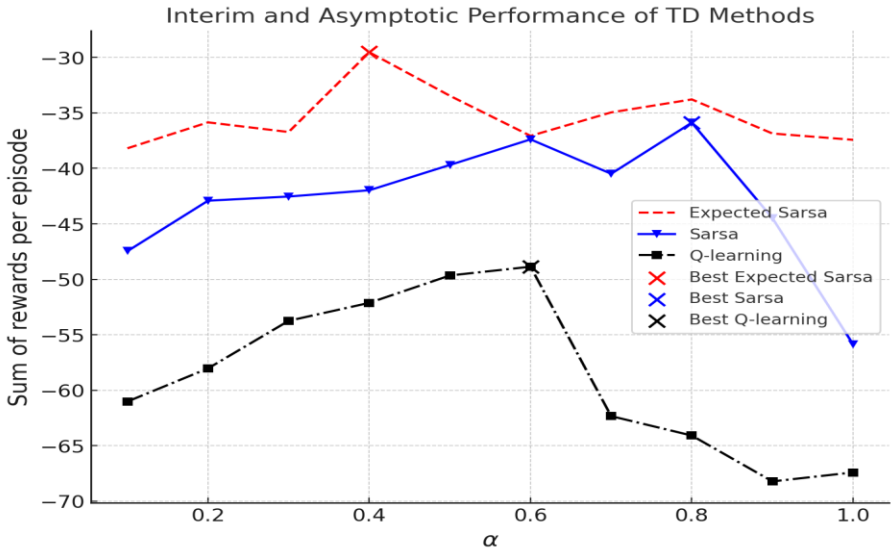

Figure 1. Interim and Asymptotic Performance of TD Methods

This study analyzes the performance of Q-learning, SARSA, and Expected SARSA in solving the **sliding block** problem. The goal is to determine the **optimal learning rate (α)** that enables the block to reach its target efficiently. **Figure 1** compares the reward performance of these algorithms under different **learning rates (α).** The **interim performance** is measured as the average reward over the first 100 episodes, while the **asymptotic performance** is based on the average reward over 100,000 episodes. **Expected SARSA (red dashed line)** achieved the highest reward at $\alpha \approx 0.6$, while **SARSA (blue line)** performed best at $\alpha \approx 0.7$ but showed fluctuations at higher α values. **Q-learning (black dashed line)** improved as α increased but experienced a significant decline beyond $\alpha > 0.6$.

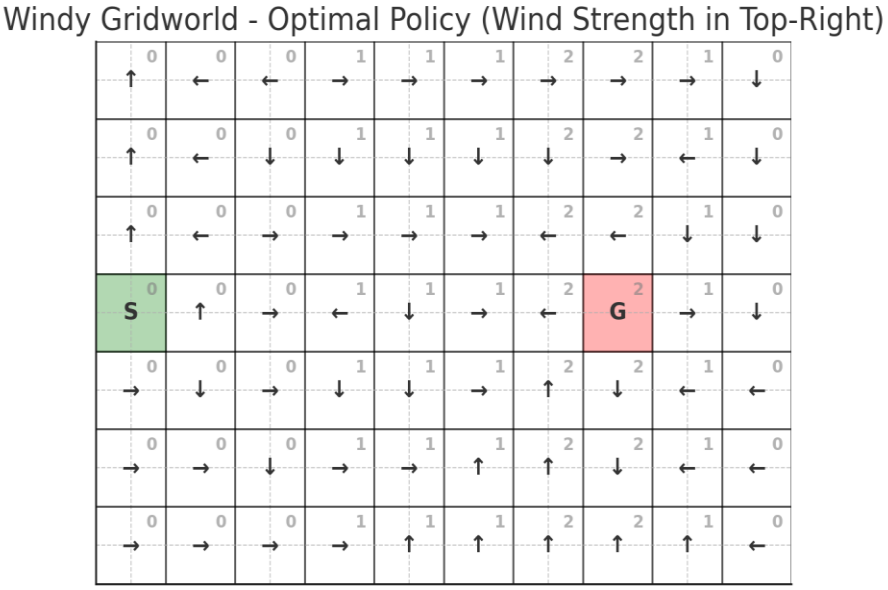

Figure 2. Windy Gridworld

The **Windy Gridworld** problem is a RL task where an agent must navigate from a start position (green) to a goal (red) while dealing with wind forces that push it upward in certain columns. Unlike standard gridworld environments, the wind alters movement, requiring the agent to learn an efficient path despite these disturbances. The training was conducted using the **SARSA algorithm**, with parameters **α = 0.5**, **ϵ = 0.1**, and initial **Q-values set to zero.** Over **200 episodes and 8,500 steps**, the agent gradually improved its policy, initially moving randomly but eventually learning the optimal actions. **Figure 2** visualizes the learned policy, where arrows indicate the best action in each state, and numbers in the top-right corners represent wind strength. The results show that the agent successfully adapts to the wind's influence, finding the shortest and most efficient path to the goal.

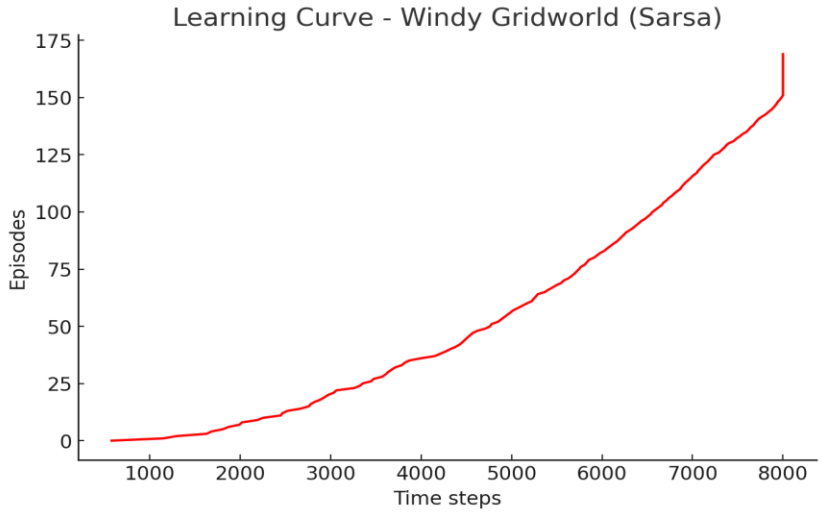

Figure 3. Learning Curve-Windy Gridworld(SARSA)

Figure 3 illustrates the agent's learning process over time, showing how episode lengths vary as the agent gains experience. The x-axis represents time steps, while the y-axis indicates the number of episodes completed. Initially, since the agent moves randomly, episodes take longer to finish. However, as the agent learns from its interactions, it discovers more efficient paths, leading to shorter episode durations. The increasing steepness of the curve highlights the improvement in learning efficiency. After **8,500 time steps**, the agent successfully learns the optimal policy, completing episodes in significantly fewer steps, demonstrating the effectiveness of the **SARSA algorithm** in optimizing movement within the **Windy Gridworld** environment.

Overall, the results demonstrate that **SARSA** successfully learns the optimal policy in the **Windy Gridworld** environment, allowing the agent to adapt to wind conditions and move more efficiently over time. **Figure 3** illustrates that as the learning process progresses, the agent reaches the goal in fewer steps, indicating an accelerated exploration phase. **Figure 2** confirms that the agent effectively adopts the optimal movements, while **Figure 1** provides a comparative analysis of the algorithms. Although **SARSA** exhibits some fluctuations at higher learning rates, it generally performs well. However, **Expected SARSA** remains the most stable and efficient learning algorithm, making it the most reliable choice for scenarios requiring long-term adaptation.

Conclusions

This study evaluated the performance of Q-learning, SARSA, and Expected SARSA in RL tasks, focusing on their adaptability and stability. The results showed that Expected SARSA provided the most stable learning performance, making it the best choice for long-term adaptation, while SARSA effectively learned the optimal policy in Windy Gridworld despite fluctuations at higher learning rates. Future research can explore adaptive exploration strategies and deep RL methods to enhance learning efficiency and stability. Additionally, investigating how these algorithms perform in more complex, dynamic environments could further improve their real-world applicability.

Acknowledgements

Not applicable.

References

[1] Sutton RS, Barto AG. (2018). Reinforcement Learning: An Introduction. MIT Press. Cambridge: Massachusetts; 2018.
[2] Bertsekas DP. Reinforcement Learning and Optimal Control. Athena Scientific. Belmont: Massachusetts; 2019.
[3] Yu C, Liu J, Nemati S, Yin G. Reinforcement learning in healthcare: a survey. ACM Computing Surveys 2021; 55(1) :1-36.
[4] Arakawa R and Shiba S. Exploration of reinforcement learning for event camera using car-like robots. 2020.
[5] Luong N, Hoang D, Gong S, Niyato D, Wang P, Liang Y et al. Applications of deep reinforcement learning in communications and networking: a survey. IEEE Communications Surveys & Tutorials 2019; 21(4): 3133-3174.
[6] Lei L, Tan Y, Zheng K, Liu S, Zhang K, Shen X. Deep reinforcement learning for autonomous internet of things: model, applications and challenges. IEEE Communications Surveys & Tutorials 2020; 22(3): 1722-1760.
[7] Sutton, RS. Learning to predict by the methods of temporal differences. Machine Learning 1988; 3(1): 9-44.
[8] Tesauro G. Temporal difference learning and TD-Gammon. *Communications of the ACM 1995;38(3): 58-68.
[9] Tsitsiklis JN, Van Roy B. An analysis of temporal-difference learning with function approximation. IEEE Transactions on Automatic Control 1997; 42(5): 674-690.
[10] Szepesvári C. Algorithms for Reinforcement Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning 2010; 4(1): 1-103.
[11] Amiranashvili A, Dosovitskiy A, Koltun V, Brox T. Td or not td: analyzing the role of temporal differencing in deep reinforcement learning. 2018.
[12] Asis, KD and Sutton, RS. Per-decision multi-step temporal difference learning with control variates.2018. https://doi.org/10.48550/arxiv.1807.01830

[13] Silver D, Sutton RS, Müller M. Sample-based learning methods for online planning. Journal of Machine Learning Research 2008; 9: 1937-1959.
[14] Van Seijen H, Van Hasselt H, Whiteson S, Wiering M. A theoretical and empirical analysis of Expected SARSA. IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning 2009; 177-184.
[15] Ganger M, Duryea E, Hu W. Double sarsa and double expected sarsa with shallow and deep learning. Journal of Data Analysis and Information Processing 2016; 04(04): 159-176.
[16] Jiang H, Gui R, Chen Z, Wu L, Dang J, & Zhou J. An improved sarsa(λ) reinforcement learning algorithm for wireless communication systems. IEEE Access 2019; 7: 115418-115427.
[17] Lin B, Han L, Xiang C, Liu H, Ma T. A real-time energy management strategy for off-road hybrid electric vehicles based on the expected sarsa. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 2022; 237(2-3): 362-380.
[18] Asis K, Hernandez-Garcia J, Holland G, Sutton R. Multi-step reinforcement learning: a unifying algorithm. 2017.
[19] MoradiMaryamnegari H, Frego M, Peer A. Model predictive control-based reinforcement learning using expected sarsa. IEEE Access 2022; 10: 81177-81191.
[20] Doya, K. Temporal difference learning in continuous time and space. Neural Computation 2000; 12(1): 219-245.
[21] Konda VR, Tsitsiklis JN. Actor-critic algorithms. Advances in Neural Information Processing Systems 2000; 12: 1008-1014.