Robotic Agents through Scalable Multi-agent Reinforcement Learning for Optimization of Warehouse Logistics

HALA KHANKHOUR[1,*], CHAKIR TAJANI[2], NAJAT RAFALIA[1], JAAFAR ABOUCHABAKA[1]
[1]Department of Computer Science, Faculty of Sciences,
Ibn Tofail University,
Kenitra,
MOROCCO

[2]SMAD Team, Department of Mathematics,
Polydisciplinary Faculty of Larache,
Abdelmalek Essaidi University,
MOROCCO

**Corresponding Author*

Abstract: - Optimizing warehouse logistics is a daunting challenge, especially in today's ever-changing industrial environment. As warehouses become complex labyrinths with robots dedicated to various tasks, heuristic methods become indispensable. Based on experience and intuition, these methods offer shortcuts to solving complex challenges, enabling rapid decisions without resorting to exhaustive research. The recent research, based on pre-established rules and previous experience, has found that heuristic methods are difficult to adapt quickly to frequent changes in logistical fields. The heuristic methods are weak when faced with new situations requiring frequent change. These constraints allow us to migrate to reinforcement learning, which introduces a dynamic and continuous path in logistics environments, unlike static heuristic methods. In this environment, this paper aims to provide agents with intelligent and structured strategies to manage navigation efficiently in such dynamic logistic environments, and to meet the challenges of modern warehouses and their respective targets in real time. To achieve this, we have hybridized the BAT algorithm of Meta heuristics and reinforcement learning algorithms, which will yield remarkable results.

Key-Words: - Multi-agent reinforcement Learning, Warehouse-Logistics, Robotic Agents, artificial intelligence, Bat algorithm, Q-learning Algorithm**,** Sockets.

Received: October 4, 2024. Revised: November 16, 2024. Accepted: December 19, 2024. Published: March 26, 2025.

1 Introduction

Optimizing modern warehouse logistics is proving a formidable challenge, especially in a constantly evolving industrial context. Warehouse management is becoming more efficient, with robots dedicated to different tasks, enabling rapid decisions to be made without the need for exhaustive research [1], like the methods heuristic are limited in the face of frequent change, especially in the field of modern warehouses, this gap has allowed us to converge on reinforcement learning, [2].

However, the field of reinforcement learning (RL) has been evolving rapidly in recent years, showing the performance of its ability to solve a wide variety of problems. In many fields such as telecommunication to natural language processing robotics, energy distribution, finance and traffic control, using an agent-based trial-and-error process to reach the destination in real time and to maximize future rewards in modern logistics environments. In addition, the continued evolution of multi-agent systems (MAS) has become imperative, as MAS have proven their performance and ability to solve a wide variety of problems.

On the other hand, some scientific research has shown that the direct implementation of RL to an SMA presents a number of difficulties, such as the fact that the actions of agents can act on the actions of other agents, and in particular non-stationarity.

To meet this challenge, knowing that scalability and convergence is becoming a difficulty in the field of logistics, we have deployed a new approach to extend single-agent RL algorithms and incorporate multi-agent approaches. In this context, we deployed a new approach to extend single-agent RL algorithms and incorporate multi-agent approaches,

which in turn require an approach capable of handling the complexity and dynamism of multi-agent environments, [3]. Therefore, we took these obstacles into consideration to develop our efficient multi-agent reinforcement learning (MARL) algorithm for real-world applications.

2 Warehouse Logistics

After an in-depth study of the current situation of warehouses. In recent years, researchers have used approaches based on rigid heuristic methods, but these methods have difficulties in handling frequent environments, especially for modern dynamic warehouse.

Heuristic approaches have difficulty in efficiently solving logistics flows, whereas coordination systems are ready to make the optimal division of tasks between optimized route calculation and robots.

A critical analysis of the literature shows the obstacles found in traditional methods in logistics environments, often resulting in additional costs. This rigidity not only leads to delays in reaching the destination in real time, but also limits companies seeking to optimize the logistics of their warehouses and act quickly according to the configuration and interim demand in real time, [4].

Therefore, in this paper, we propose a hybrid solution between heuristic methods in terms of accuracy, and reinforcement learning methods to achieve a significant improvement in logistics management within warehouses, in line with our fundamental objectives.

3 Methodology

3.1 Reinforcement learning Agent

As mentioned earlier, a reinforcement-learning agent engages in sequential decision making through its interactions with the environment. The environment is often represented as an infinite-horizon, discounted Markov decision process (MDP); and commonly referred to as a Markov decision process, [5]. The CDM is used as a standard model to characterise the agent's decision-making process when it has complete knowledge of the state of the system s. In this model, at each time step t, the agent selects an action at in response to the current state of the system, [6].

3.2 Multi-Agent Reinforcement Learning Framework (MARL)

In a similar context, multi-agent reinforcement learning (MARL) tackles the problems associated with sequential decision-making, but with the added complexity of the presence of several agents. In this scenario, both the evolution of the state of the system and the rewards received by each agent are influenced by the collective actions of all the agents. In particular, each agent strives to optimise its own long-term reward, which becomes a function of the policies adopted by all the other agents. This general model has various applications in practical contexts. In essence [7], there are two apparently distinct but closely interdependent theoretical frameworks for LRA: Markovian/stochastic games and extensive-form games.

3.3 Sockets

Sockets are programming interfaces that enable bidirectional communication between two distinct processes, either on the same machine or across a network. First introduced with the BSD distributions in 1984 and widely used in UNIX systems, sockets act as endpoints, associated with port numbers, and facilitate the transfer of software data between applications. These interfaces are also characterised by their association with specific protocols and are fundamental to the establishment of connections and the transmission of data flows, thus contributing to the implementation of efficient communications within a computing environment.

Sockets offer two distinct modes of communication:

- Connected mode, similar to telephone communication, uses the TCP protocol. In this mode, a persistent connection is established between the two processes, eliminating the need to specify the destination address each time data is sent.
- The non-connected mode, which resembles a mail communication, uses the UDP protocol. In this mode, the destination address must be provided for each transmission, and no acknowledgement of receipt is generated, [8].

To simplify handling, sockets are designed to preserve the semantics of system input/output operations, just like file operations (create, open, read, write, close).

When transmitting data between two sockets in connected mode, it is essential to distinguish between the socket used by the program requesting the connection (client) and that used by the program accepting the connections (server), [9].

A server, in this context, is a program that waits for connections via a socket and then handles all

incoming connections. On the other hand, a client is a program that associates with a server using a socket.

3.4 The Bat Algorithm

The Bat algorithm, conceived in 2010, [10], introduces a contemporary metaheuristic approach rooted in swarm intelligence. It derives inspiration from the foraging behavior of microbats, particularly their variable pulse emission rate and loudness. The Bat algorithm's development is guided by three fundamental principles: firstly, bats employ sound wave (ultrasound) reflection for prey detection during flight in darkness; secondly, bats engage in a randomized flying pattern for foraging, influenced by parameters such as velocity v_i at position (x_i), frequency (q_i), and loudness (L_i); and thirdly, loudness varies from a substantial positive value (L_0) to a consistent minimum (L_{min}). To approximate the target location, each bat is assigned a randomly selected frequency (q_i) for emitted pulses, uniformly drawn from the interval [q_{min}, q_{max}], with the ability to automatically adjust the frequency within the same range. The pulse emission rate (r_i) can also be tuned within the interval [0,1], where 0 signifies no pulse emission, and 1 represents the maximum pulse emission rate. For a virtual bat and its position updating strategy in a D-dimensional search space, the new solution (x_i^t), frequency (q_i^t), and velocity (y_i^t),) (for each bat in the population) at generation t are determined by Eq. 1, Eq. 2.and Eq. 3 :

$$q_i^{(t)} = q_{min} + (q_{max} - q_{min})\beta \quad (1)$$

$$v_i^{(t+1)} = v_i^{(t)} + \left(x_i^{(t)} - x_{GBest}\right) q_i^{(t)} \quad (2)$$

$$x_i^{t+1)} = x_i^{(t)} + v_i^{(t+1)} \quad (3)$$

In this context, the parameter β is confined to the interval [0, 1] and is denoted by a randomly distributed value following a uniform distribution. The expression x_{GBest}, represents the current global optimum solution, as determined by the evaluation and comparison of all solutions within the population of n bats. After the update of velocities and positions for the bats, the initiation of the local search component occurs only when a randomly generated number exceeds the pulse emission rate r_i. Subsequently, a solution is selected from the existing best solutions, and a new position for each bat is generated locally through a random walk. This random walk is characterized by a process that involves by Eq. 4:

$$x_{new}^{(t)} = x_{Gbest} + \varepsilon L^{(t)} \quad (4)$$

In this context, the variable ε is constrained within the interval [0, 1] and represents a random number uniformly distributed. The expression $L^t = \{ L_i^t \}$ signifies the average loudness value calculated across the population of n bats at generation t. Regarding the selection process, a newly generated solution is considered acceptable if a uniformly generated random number falls below the current loudness L_i, and the fitness value of the current solution, denoted as $f(x_i)$, surpasses that of the global best solution, f$f(x_{GBest})$.

To strike a balance between exploration and exploitation throughout the search process, adjustments to both the loudness L^t and pulse emission rate r_i occur exclusively when the candidate solution demonstrates improvement with the progression of iterations, [11]. These adjustments unfold through a procedural mechanism involving By Eq. 5:

$$L_i^{(t+1)} = \alpha L_i^{(t)}, r_i^{(t)} = r_i^{(0)}[1 - \exp(-\gamma(t)] \quad Eq.5$$

In this scenario, the constants α and γ are predefined values. Generally, once a bat successfully pinpoints its prey's location, there is a decrease in loudness and an escalation in pulse rate.

The iterative progression of the entire Bat algorithm persists until a predefined stopping criterion is satisfied. The procedural steps of the original Bat algorithm are concisely delineated in the pseudo code [10] depicted in Figure 1.

Population :
$y_i = (y_{i1}, y_{i2} \ldots, y_{id})^t, for\ i = 1,2,..NP)$
Rate r_i,
Frequency $q_i\ at\ x_i$,
Number of generation G_{max} ,
Loudness z_i
Best solution x_{Gbset},

If (t < G_{max}) do new population y_i ,
  If (rand > r_i,) select best solution x_{Gbset}
  End if
Generate a new population randomly
If (rand < z_i and f (x_i) < f(x_{Gbset},))
Add a new solution
Increase population
End if
Update the x_{Gbset}
Increase the generation
End if

Fig. 1: Bat algorithm

3.5 The Q-learning Algorithm

Q-Learning is a reinforcement learning strategy that identifies the optimal next action [12], based on the current state, selecting this action randomly with the aim of maximizing the reward. This model-free and off-policy reinforcement learning approach determines the best action for an agent given its present state, allowing the agent to decide the subsequent action based on its position in the environment. The absence of a predefined policy in Q-learning means that the model can either establish its own rules or operate outside a given policy.

The model's goal is to identify the most favorable action based on its current state, relying on predictions of expected responses from the environment rather than adhering to a predefined policy. An example application of Q-learning is an advertisement recommendation system, where traditional systems rely on past purchases or visited websites to make recommendations, [13]. Key concepts essential to Q-learning include states (representing the agent's current position), actions (the steps taken in a particular state), rewards (positive or negative outcomes for each action), episodes (concluding when the agent reaches a terminal state), Q-values (determining the quality of an action in a specific state), and Temporal Difference (a formula for computing Q-values based on current and previous states and actions).

The Bellman equation plays a crucial role in Q-learning [14], helping determine the value of a particular state and assessing the significance of being in or taking that state. It calculates the agent's next state by considering the current state, associated reward, expected maximum reward, and a discount rate that influences the importance of the current state. The learning rate also affects the model's learning speed, [15].

But, how to create a Q-Table ?. During the execution of our algorithm, we encounter multiple solutions [16] and the agent traverses various paths. The challenge is to identify the optimal path among these alternatives, and this task is accomplished through the creation of a Q-Table. The Q-Table functions as a mechanism to identify the optimal action for each state within the environment. By applying the Bellman equation at each state, we compute the anticipated future state and the corresponding reward, preserving these findings in the Q-Table for subsequent comparison with other states. Let's exemplify the procedure of formulating a Q-Table for an agent assigned with the objectives of learning to run, fetch an item, and sit on command. The stages encompassed in constructing a Q-Table are as follows:

Step 1: Establish an initial Q-Table where all values are set to 0.

At the outset, the initial values assigned to all states and rewards are set to 0. Take into account the Q-Table presented below, which illustrates the learning process of a dog simulator as it undertakes various actions as shown in Table 1.

Table 1. Q-Table 1

Action	Fetching	Sitting	Running
Start	0	0	0
Idle	0	0	0
Wrong Action	0	0	0
Correct Action	0	0	0
End	0	0	0

Step 2: Select an action and implement it. Adjust the values in the table accordingly.

This marks the initial phase where no other actions have been taken yet. Suppose our initial goal is for the agent to sit, which it accomplishes. The subsequent table update unfolds as follows in Table 2.

Table 2. Q-Table 2

Action	Fetching	Sitting	Running
Start	0	1	0
Idle	0	0	0
Wrong Action	0	0	0
Correct Action	0	0	0
End	0	0	0

Step 3: Retrieve the reward value and compute the Q-value using the Bellman equation.

For the action taken, we need to compute the actual reward value and the Q-value Q(S,A), as represented in Table 3.

Table 3. Q-Table 3

Action	Fetching	Sitting	Running
Start	0	1	0
Idle	0	0	0
Wrong Action	0	0	0
Correct Action	0	34	0
End	0	0	0

Step 4: Continue in the same manner until the table is filled or an episode ends.

The agent keeps taking actions, and for each action, the reward and Q-value are computed, and the table is updated as shown in Table 4.

Table 4. Q-Table 4

Action	Fetching	Sitting	Running
Start	5	7	10
Idle	2	5	3
Wrong Action	2	6	1
Correct Action	54	34	17
End	3	1	4

In the implementation of our solution, we strategically employed an innovative approach by separating the reinforcement learning model from the multi-agent system. This judicious decision provides several key advantages for optimizing the overall solution. First and foremost, the clear delineation of responsibilities between the reinforcement learning model and the multi-agent system enhances modularity. This modular design enables independent development, testing, and enhancement of each component, streamlining the maintenance and scalability of the entire system.

This approach significantly improves resource management efficiency by segregating the reinforcement learning model as an external entity, serving as a server, and positioning the multi-agent system as a client. This design allows for effective distribution of processing load, mitigating bottlenecks and optimizing the overall system performance. To gain a comprehensive understanding of the system's architecture in Figure 2, and for a more in-depth understanding of our system description, as shown in Figure 3.

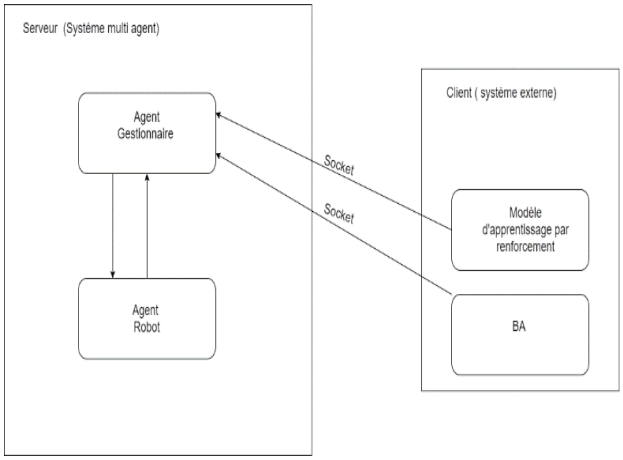

Fig. 2: System's Architecture

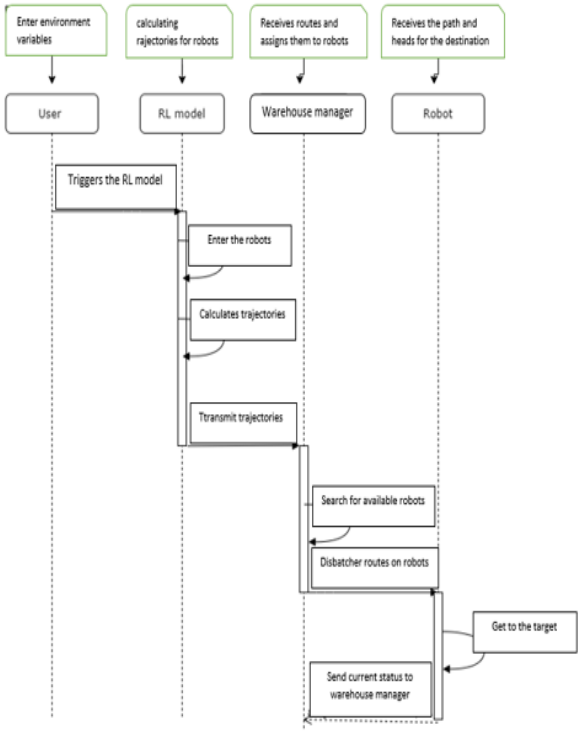

Fig. 3: description of our system

4 Discussion and Results

The environment is intricately characterized by a structured grid, spanning dimensions x * y.

Reward Function:

In this code, the reward function is defined to determine the reward a robot receives as a function of its current state, the action performed, the next state and the target state. The function is structured as follows, here's an explanation of the logic:

- If the next state is an obstacle (border or robot), the robot receives a penalty of -100.
- If the next state is the target state, the robot receives a reward of 100.
- Otherwise, the robot receives a default reward of-1.

Agents (Robots):

The Robot Agent, a key player in the logistics system, plays a central role in carrying out the tasks assigned by the Warehouse Management Agent.

Training Process using Q-learning:

Our iterative learning approach is based on the Q learning algorithm, starting with the definition of actions and states, as well as size and grid parameters, and the creation of robot-specific Q tables. These tables are essential because they

contain values linked to the robots' states and actions so that they can make decisions in the future.

To assess the effectiveness of our solution, we calculate the mean Q-values for each robot, providing insights into their learning progress within the environment. The graph visually represents this performance as shown in Figure 4.

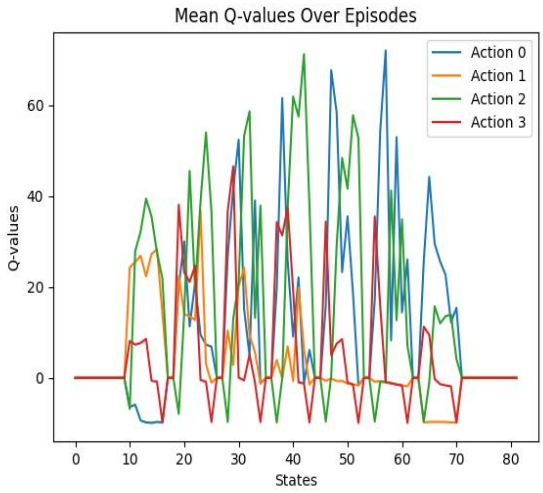

Fig. 4: Calculate the average values of Q

In this graph, the x axis represents the different states and the y axis represents the Q values. The results in Figure 4 show an increase in the average Q values for all robots over the episodes. Each action (Action 3, Action 2, Action 1, Action 0) is represented by a linear plot. Each action (Action 3, Action 2, Action 1, Action 0) is represented by a linear plot, highlighting the average values of Q over the episodes.

Notably, for all actions, we observe a continuous increase in the Q values, which indicates that the robots are executing the actions more efficiently.

To showcase the performance and adaptability of our algorithms, we conducted tests in two distinct environmental dimensions. In the first scenario, three robots navigated a 9x9 grid. The graphical representation illustrates each robot's successful journey from their starting points to their respective targets, demonstrating an avoidance of borders and other robots. Notably, we applied the Bat algorithm to this scenario, yielding identical results as shown in Figure 5.

In the second scenario, as shown in Figure 6, we examined the capabilities of our approach with four robots operating in a larger 14x14 grid. All robots successfully reached their targets without encountering obstacles or colliding with other robots. However, deploying the Bat algorithm for this scenario presented challenges, requiring significant adjustments to hyperparameters for proper adaptation to the altered environment.

This highlights the inherent flexibility of reinforcement learning compared to heuristic methods.

Our findings strongly support the hypothesis that heuristic methods introduce rigidity to the system. In contrast, reinforcement learning exhibits superior adaptability, enabling effective performance across diverse scenarios.

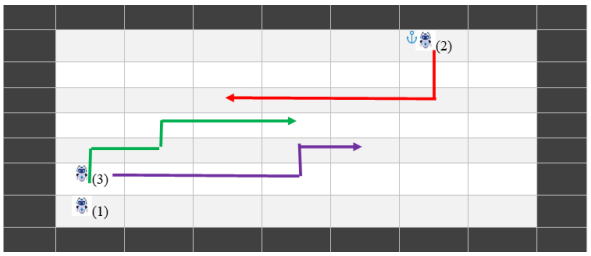

Fig. 5: Applied Bat algorithm to our scenario

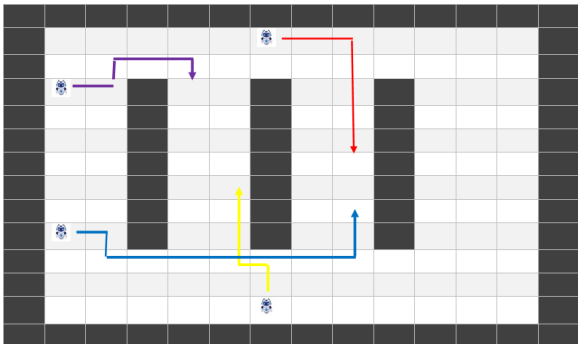

Fig. 6: Applied Bat algorithm to our scenario with four robots

Following a thorough analysis of execution times with different algorithms, it is clear that Q-learning surpasses heuristic methods. The discerned trend underscores Q-learning's superior flexibility, leading to improved performance and optimization in warehouse logistics. This is attributed to our approach of implementing the solution separately; the reinforcement learning model operates independently from the multi-agent system, functioning as microservices. This modular setup allows each component to operate efficiently in tandem as shown in Table 5.

Table 5. Result of simulation

Solution of Q-learning	Solution with BA
1min	2min

5 Conclusion

The main aim of our project was to optimize warehouse logistics using reinforcement learning, and to compare it with heuristic methods. To achieve this, we adopted an innovative approach by separating the reinforcement learning processing from the multi-agent system, thus enabling separate execution of the different tasks. This approach proved to be extremely advantageous, as previously demonstrated when comparing it with other available methods. The main difficulties lay in understanding metaheuristics and reinforcement learning, requiring considerable effort to master the tools and technologies associated with these fields. We are proud to announce that we have fully covered the core functionalities of the solution, alleviating the rigidity of heuristic methods and introducing an innovative approach. In terms of future prospects, we plan to enable the solution to be implemented in real time in response to frequent warehouse changes, introducing the possibility of real-time training to maintain the relevance and effectiveness of our solution over time.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

The authors wrote, reviewed and edited the content as needed and they have not utilised artificial intelligence (AI) tools. The authors take full responsibility for the content of the publication.

References:

[1] C. G. Petersen and R. W. Schmenner, "An evaluation of routing and volume-based storage policies in an order picking operation," *Decision Sciences,* vol. 30, no. 2, pp. 481–501, Spring 1999. DOI: 10.1111/j.1540-5915.1999.tb01619.x.

[2] N. P. Karpova, "Modern warehouse management systems," In: Digital Technologies in the New Socio-Economic Reality. *The 9th International Scientific Conference on Digital Transformation of the Economy: Challenges, Trends and New Opportunities, ISCDTE,* Samara 27 April, vol. 304, spring 2022. pp. 261-267. DOI: 10.1007/978-3-030-83175-2_34.

[3] P. O. Dusadeerungsikul, X. He, M. Sreeram, & S. Y. Nof, "Multi-agent system optimisation in factories of the future: cyber collaborative warehouse study". *International Journal of Production Research,* 2022, vol. 60, no. 20, pp. 6072-6086.

[4] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses," *AI Magazine,* vol. 29, no. 1, pp. 9, 2008. https://doi.org/10.1609/aimag.v29i1.2082.

[5] R. De Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *European Journal of Operational Research,* vol. 182, no. 2, pp. 481–501, 2007. DOI: 10.1016/j.ejor.2006.07.009.

[6] A. Urru, M. Bonini and W. Echelmeyer, "The STIC analysis: A decision support tool for technology related investments in logistics," *2017 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, Bari, Italy, 2017, pp. 33-38, DOI: 10.1109/SOLI.2017.8120965.

[7] J. Zhao, X. Wang, B. Xie, & Z. Zhang, "Human-robot kinematics mapping method based on dynamic equivalent points". *Industrial Robot: the International Journal of Robotics Research and Application,* 2023, vol. 50, no 2, pp. 219-233. DOI: 10.1108/IR-02-2022-0056.

[8] S. Manhas, S. Taterh, Et D. Singh, "Deep Q learning-based mitigation of man in the middle attack over secure sockets layer websites". *Modern Physics Letters B,* vol. 34, no 32, pp. 2050366, 2020. DOI: 10.1142/S0217984920503662.

[9] L. Paternò, M. Ibrahimi, E. Gruppioni, A. Menciassi, and L. Ricotti, "Sockets for limb prostheses: a review of existing technologies and open challenges". *IEEE Transactions on Biomedical Engineering,* vol. 65, no 9, pp. 1996-2010, 2018. DOI: 10.1109/TBME.2017.2775100.

[10] X. S. Yang, X. He. "Bat algorithm: literature review and applications". *International Journal of Bio-inspired computation,* vol. 5, no 3, pp. 141-149, 2013. DOI: 10.1504/IJBIC.2013.055093.

[11] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature,* vol. 588, no. 7836, pp. 77–82, 2020. DOI: 10.1038/s41586-020-2939-8.

[12] V. Ilin, D. Simić, M. Veličković, N. Garunović, & N. Saulić, "Machine Learning in the Last-Mile Delivery: Modified Q-Learning for the TSP". *In: International Conference on Soft Computing Models in*

Industrial and Environmental Applications. Cham: Springer Nature Switzerland, 2024. pp. 108-117. DOI: 10.1007/978-3-031-75013-7_11.

[13] K. Tuyls, K. Verbeeck, and T. Lenaerts. "A selection-mutation model for q-learning in multi-agent systems". *In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, AAMAS 03*, pp. 693-700, 2003. DOI: 10.1145/860575.860687.

[14] J. Kim, & I. Yang, "Hamilton-Jacobi-Bellman equations for Q-learning in continuous time". *In: Learning for Dynamics and Control. PMLR*, Berkeley, pp. 739-748, 2020.

[15] V. K. Saini, R. Kumar, A. S. Al-Sumaiti, A. Sujil, and E. Heydarian-Forushani. "Learning based short term wind speed forecasting models for smart grid applications: An extensive review and case study". *Electric Power Systems Research,* vol. 222, pp. 109502, 2023.

[16] T. T. Nguyen, N. D. Nguyen and S. Nahavandi. "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications". *IEEE Transactions on Cybernetics,* vol. 50, no. 9, pp. 3826-3839, 2020. DOI: 10.1109/TCYB.2020.2977374.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflicts of Interest

The authors have no conflicts of interest to declare.