Combining Q-Learning and the Hungarian Algorithm for Multi-Robot Systems in Dynamic Environments

Hoang Mai Nguyen[1]
University of Science and Technology – The University of Danang
Da Nang, Viet Nam
nhmai@dut.udn.vn

Huynh Gia Bao Nguyen[2]
University of Science and Technology – The University of Danang
Da Nang, Viet Nam
nguyenhuynhgiabao06082004@gmail.com

Thi Kim Oanh Nguyen[3]
University of Science and Technology – The University of Danang
Da Nang, Viet Nam
laoanhdo@gmail.com

Ngoc Son Vo[4]
University of Science and Technology – The University of Danang
Da Nang, Viet Nam
ngocsonvo1912@gmail.com

Van Thinh Vo[5]
University of Science and Technology – The University of Danang
Da Nang, Viet Nam
thinhvovan88@gmail.com

***Abstract*—This paper presents a hierarchical reinforcement learning framework for multi-robot systems in dynamic warehouse environments. The proposed approach integrates Q-learning at two levels: motion control and task reassignment. At the motion level, Q-learning adaptively adjusts heuristic coefficients in the Artificial Potential Field (APF) model to improve trajectory smoothness and collision avoidance. At the task level, Q-learning evaluates whether to accept new assignments proposed by the Hungarian algorithm in scenarios with moving targets.**

Simulation results show that the proposed hybrid method reduces travel distance, shortens task completion time, and decreases collision events compared to conventional APF and heuristic approaches. These results demonstrate that combining reinforcement learning with classical optimization improves adaptability and overall system performance in dynamic multi-robot environments.

Keywords: Multi-robot systems; Task allocation; Hungarian algorithm; Reinforcement learning; Q-learning; Artificial Potential Field; Dynamic environments

Nomenclature

Symbol	Description
s_t	System state at time t
a_t	Action taken at time t
$Q(s,a)$	Action-value function in Q-learning
η	Learning rate
λ	Discount factor
ε	Exploration probability in the ε-greedy policy
old_cost	Current task cost
new_cost	Proposed task cost
s_{prog}	Task progress level
s_{diff}	Cost improvement level
d_o	Distance to obstacle
d_r	Distance to a neighboring robot

Abbreviations

APF	Artificial Potential Field
RL	Reinforcement Learning
Q-learning	Q-value based Reinforcement Learning
MRS	Multi-Robot System
GA	Genetic Algorithm
ε-greedy	Exploration – Exploitation Strategy

I. INTRODUCTION

In multi-robot systems for logistics and smart warehouses, the operating environment is typically dynamic, with continuous changes in targets and system states. Various optimization methods have been studied [1], among which the Hungarian algorithm is a widely used approach for task allocation. However, this method mainly provides instantaneous optimal solutions and lacks adaptability in complex dynamic scenarios. To address this limitation, this study proposes a hierarchical reinforcement learning framework, where Q-learning is integrated to adjust control parameters in the Artificial Potential Field (APF) and to assist in task reassignment decisions, while the Hungarian algorithm is still used to ensure optimal task allocation. Simulation results demonstrate that the proposed approach improves operational efficiency and system stability in dynamic multi-robot environments.

II. SYSTEM MODEL AND PROBLEM FORMULATION

A. Multi-Robot System Model in Logistics

Consider a multi-robot system consisting of multiple mobile robots operating in a two-dimensional space that simulates a warehouse floor plan. The robots move independently in an environment containing static obstacles, while packages act as moving targets. During operation, each robot must be assigned an appropriate task and must move safely in a multi-robot environment. Although the assignment problem has been widely studied [6], [11], in which assignment problem models provide an important theoretical foundation for algorithms such as the Hungarian algorithm, the continuous variation of package positions makes the problem dynamic and requires a decision-making mechanism capable of adapting over time.

The objective of the system is to improve operational efficiency by reducing task completion time, shortening travel distance, and limiting collisions among robots and obstacles.

B. Hierarchical Control Architecture

The system is designed according to a hierarchical control architecture consisting of three layers: the Perception Layer, the Task Allocation Layer, and the Motion Control Layer.

This is a common approach in multi-agent robotic systems [6], [10] and is also consistent with multi-agent reinforcement learning frameworks reviewed in several survey studies [8], [11].

The Perception Layer collects position information of robots, packages, and obstacles from the environment. The Task Allocation Layer performs optimal robot-to-package assignment using the Hungarian algorithm [2] based on a cost matrix updated in real time.

The Motion Control Layer controls robot motion based on the Artificial Potential Field (APF) model [5], [9]. In this layer, heuristic rules are used to determine the priority between the target attractive force and the repulsive collision-avoidance forces, while Q-learning [4], one of the fundamental reinforcement learning algorithms described in detail in [12], acts as an adaptive layer to dynamically tune the control coefficients α, β, γ according to the environmental state, thereby improving collision-avoidance capability and navigation efficiency in multi-robot environments [7], [8].

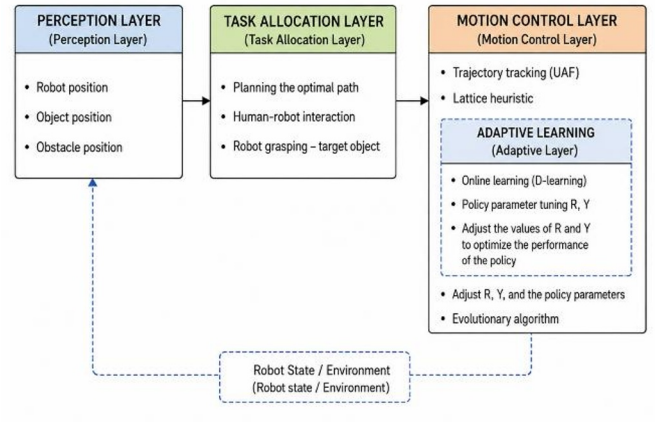

Figure 1. Architecture of the multi-task robot coordination system (Perception – Task Allocation – Motion Control)

III. Control System Design

A. Perception Layer

The Perception Layer is responsible for determining the positions of robots, packages, and obstacles in the warehouse space, providing input data for the task allocation and motion control layers. In this study, position data are assumed to be acquired through a surveillance camera system and processed in a two-dimensional plane.

To transform coordinates from the image coordinate system to the real-world coordinate system of the warehouse, homography is used [14]:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \sim H \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (3.1)$$

where (u, v) denotes the image coordinates, (X, Y) denotes the coordinates in the warehouse plane, and H is the homography matrix. The obtained coordinates are used to track robot states and provide data for the subsequent processing layers.

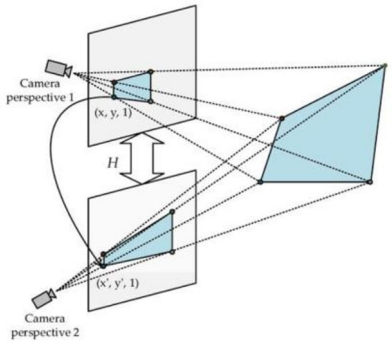

Figure 2. Coordinate transformation using homography

B. Task Allocation Layer

The Task Allocation Layer is responsible for determining the robot-target assignment scheme such that the total operating cost of the system is minimized. In this study, the problem is modeled as a one-to-one assignment problem and solved using the Hungarian algorithm [2], which can find the optimal solution with low computational cost and is suitable for robot systems operating in dynamic environments.

Based on the state information from the Perception Layer, the system constructs the cost matrix C_{cp}, in which each element c_{ij} is determined from the estimated distance and time required for robot i to reach target j. The Hungarian algorithm is applied to this matrix to determine the optimal assignment scheme and is updated periodically with period Δt or when the system state changes.

After assignment, tasks are transmitted to the robots through the MQTT protocol using the publish/subscribe mechanism. This mechanism helps maintain real-time communication with low latency and supports flexible scalability in the number of robots in a distributed multi-robot system.

C. Motion Control Layer

After tasks are assigned using the Hungarian algorithm, the Motion Control Layer is responsible for guiding each robot to approach its target efficiently while ensuring collision avoidance in an environment with obstacles and multiple robots operating simultaneously.

In the APF model, robot motion is determined by the combination of the attractive force toward the target and the repulsive forces for avoiding obstacles and neighboring robots. The total control force is expressed as follows:

$$\vec{F} = \alpha \vec{F}_{goal} + \beta \vec{F}_{obs} + \gamma \vec{F}_{veh} \quad (3.2)$$

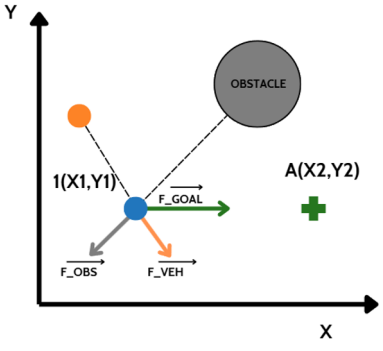

Figure 3. Artificial Potential Field model

Where $\vec{F}_{goal}$ is the target attractive vector, $\vec{F}_{obs}$ is the obstacle-avoidance repulsive vector, and $\vec{F}_{veh}$ is the repulsive vector for avoiding other robots; α, β and γ are weighting control coefficients that reflect the trade-off between task efficiency and operational safety. To enhance system adaptability in dynamic environments, the control coefficients are adjusted based on heuristic rules depending on the distances to obstacles and neighboring robots.

At each control cycle, the robot computes:

- Distance to the nearest obstacle d_{obs}
- Distance to the nearest robot d_{veh}.

When a robot approaches obstacles or neighboring robots, the system prioritizes repulsive forces to reduce collision risk. In clear environments, it emphasizes the attractive force to shorten the travel distance.

The baseline heuristic coefficients are defined as follows:

$$\beta_h = f(d_{obs}) \tag{3.3}$$

$$\gamma_h = f(d_{veh}) \tag{3.4}$$

The target attraction coefficient is determined from the remaining priority:

$$\alpha_h = 1 - \beta_h - \gamma_h \tag{3.5}$$

This heuristic mechanism enables fast real-time response without complex optimization. However, fixed rules cannot exploit operational experience, limiting adaptability in dynamic, high-density robot environments.

D. Kinematic and Steering Control Model

After the guidance vector $\vec{F}(t) = (F_x, F_y)$ is determined, the desired heading angle of the vehicle is calculated as follows:

$$\theta_d(t) = atan\,2(F_y, F_x) \tag{3.6}$$

The control error between the desired heading and the actual heading $\theta(t)$ is:

$$e(t) = \theta_d(t) - \theta(t) \tag{3.7}$$

This error is fed into a PID controller to generate the steering-angle control signal, enabling the vehicle to adjust its heading in a stable and smooth manner. The control signal is then mapped to the servo rotation angle:

$$\theta_{servo}(t) = 90° + u(t) \tag{3.8}$$

and is constrained within the following range:

$$\theta_{\text{servo}} \in [\theta_{\min}, \theta_{\max}]$$

This model allows the vehicle to effectively track the guidance direction in a dynamic environment.

IV. Q-LEARNING FOR HEURISTIC TUNING IN ARTIFICIAL POTENTIAL FIELD-BASED CONTROL

A. Motivation and Approach

In the APF model, the coefficients α, β, γ are usually tuned by heuristic rules, but their adaptability is limited in dynamic environments. Therefore, Q-learning [4] is employed to adjust these coefficients online, improving collision avoidance, trajectory stability, and navigation efficiency in multi-agent robot systems [7]–[9].

B. Heuristic Tuning Mechanism Using Reinforcement Learning

First, the baseline control coefficients are defined using heuristic rules:

$$(\alpha_h, \beta_h, \gamma_h) = \mathcal{H}(d_{obs}, d_{veh}) \tag{4.1}$$

where d_{obs} and d_{veh} are the distances to the static obstacle and the neighboring robot, respectively.

Reinforcement learning is used to tune these coefficients through learned multiplier factors:

$$\alpha' = k_\alpha \alpha_h, \beta' = k_\beta \beta_h, \gamma' = k_\gamma \gamma_h \tag{4.2}$$

Then, the coefficients are normalized to ensure the consistency of the potential field:

$$\alpha = \frac{\alpha'}{\alpha' + \beta' + \gamma'}, \beta = \frac{\beta'}{\alpha' + \beta' + \gamma'}, \gamma = \frac{\gamma'}{\alpha' + \beta' + \gamma'} \tag{4.3}$$

This approach allows the learning algorithm to adjust only the relative priority among the force components, while the physical constraints and control structure of the artificial potential field model are preserved.

C. Reinforcement Learning Problem Modeling

1) State

The state space is constructed from local environmental features and discretized to reduce computational complexity. The state at time t is expressed as:

$$s_t = (s_{obs}, s_{veh}) \tag{4.4}$$

where the degree of obstacle proximity is defined as follows:

$$s_{obs} = \begin{cases} 0, & d_{obs} > R_{safe} \\ 1, & 0.5R_{safe} < d_{obs} \le R_{safe} \\ 2, & d_{obs} \le 0.5R_{safe} \end{cases} \tag{4.5}$$

and the presence of a neighboring robot is defined as follows:

$$s_{veh} = \begin{cases} 0, & d_{veh} > d_{safe} \\ 1, & d_{veh} \le d_{safe} \end{cases} \tag{4.6}$$

State discretization enables the learning algorithm to operate stably in real time without requiring a large state space.

2) Action

The action of the learning agent corresponds to selecting a heuristic-tuning strategy, represented by the multiplier coefficient set:

$$a_i = \left(k_\alpha^{(i)}, k_\beta^{(i)}, k_\gamma^{(i)}\right) \tag{4.7}$$

The action set $A = \{a_0, a_1, a_2, a_3\}$ is predefined to ensure that the control coefficients always remain within the safe domain. The actions correspond to the following strategies: maintaining the heuristic unchanged, prioritizing the target, increasing obstacle avoidance, or increasing avoidance of neighboring robots.

3) Action Selection Strategy

Action selection is performed using the epsilon-greedy strategy:

$$a_t = \begin{cases} random\ a \in A, & with\ probability\ \varepsilon \\ arg\ max_{\,a \in A}\, Q(s_t, a), & with\ probability\ 1 - \varepsilon \end{cases} \tag{4.8}$$

In the simulation, the parameter ε is initially set to 0.1 and gradually decreased over time to increase the stability of the control behavior as the learning process converges.

D. Reward Function and Q-Value Update

1) Reward Function

The reward at time t is defined as follows:

$$r_t = r_{\text{step}} - \lambda_{\text{obs}} \cdot I_{\text{collision\_obs}} - \lambda_{\text{veh}} \cdot I_{\text{collision\_veh}} + \lambda_{\text{reach}} \cdot I_{\text{reach}} + r_{\text{global}} \quad (4.9)$$

where r_{global} is the global penalty term, defined as:

$$r_{global} = \begin{cases} -\lambda_{global}, & if\ a\ collision\ occurs \\ 0, & otherwise \end{cases} \quad (4.10)$$

This design enables a combination of local behavior optimization for each robot and the global safety level of the overall system.

2) Q-Value Update

The Q-value is updated according to the standard Q-learning rule [4], based on the theoretical foundations of reinforcement learning and dynamic programming presented in [3], [12], which are built upon Bellman dynamic programming [3]:

$$Q(s_t, a_t) += \eta \left[r_t + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (4.11)$$

In the simulation, the learning rate is selected as $\eta = 0.1$ and the discount factor is $\lambda = 0.9$.

E. Illustration of the Coefficient Tuning Process

At time t, when the robot operates in state $s_t = (2,1)$, the initial heuristic coefficients are defined as $(\alpha_h, \beta_h, \gamma_h) = (0.40, 0.35, 0.25)$. The learning algorithm selects action $a_3 = (0.8, 0.8, 1.2)$, resulting in the tuned and normalized coefficients $(\alpha,\ \beta\ , \gamma) = (0.36, 0.31, 0.33)$.

The robot moves with this set of coefficients, no collision occurs, and the task has not yet been completed; therefore, it receives the reward $r_t = -1$. The value $Q(s_t, a_3)$ is updated, increasing the probability of selecting this strategy when the system encounters a similar state in subsequent cycles. Over multiple operating cycles, high-performance strategies are gradually prioritized, enabling the system to adaptively tune the APF control coefficients based on experience.

V. Q-LEARNING FOR HUNGARIAN-BASED TASK REASSIGNMENT DECISIONS

A. Limitations of Pure Hungarian Task Allocation in Dynamic Environments

The Hungarian algorithm [2] determines the optimal assignment based on the instantaneous cost matrix. However, in dynamic environments, the optimal assignment may frequently change and the method does not account for task progress or reassignment benefit. Therefore, a reassignment support mechanism is required to improve system stability and adaptability.

B. Reinforcement Learning Model for Task-Switching Decisions

To address the above limitations, this study integrates reinforcement learning to support the decision of whether to accept a task switch proposed by the Hungarian algorithm.

1) State

At time t, the state of the agent is constructed from two main features:

$$s_t = (s_{prog}, s_{diff}) \quad (5.1)$$

where s_{prog} represents the completion level of the current task, and $s_{diff} = old\_cost - new\_cost$ represents the cost improvement obtained when switching to the new task proposed by the Hungarian algorithm. These two features are discretized into a finite number of levels to reflect the trade-off between the cost of canceling the current task and the benefit of reassignment. This construction reduces computational complexity and supports faster convergence of the learning algorithm.

2) Action

The action set of the learning agent is defined as follows:

$$A = \{0,1\} \quad (5.2)$$

where:

- 0: keep the current task;
- 1: switch to the new task proposed by the Hungarian algorithm.

It should be noted that the learning agent does not directly perform assignment; it only decides whether to accept the new assignment result.

3) Action Selection Strategy and Learning Update

The action is selected using an ε-greedy strategy to balance the exploitation of learned experience and the exploration of new decisions. The Q-value is updated according to the standard Q-learning rule given in Equation (4.11).

C. Combination of Q-learning and the Hungarian Algorithm

In the proposed architecture, the Hungarian algorithm determines the optimal assignment at each update cycle, while Q-learning evaluates the system state to decide whether reassignment is necessary. This integration maintains the instantaneous optimality of the Hungarian algorithm, reduces unnecessary task switching, and improves adaptability in dynamic environments. As a result, the robot system achieves better assignment efficiency and operational stability with moving targets.

VI. SIMULATION AND EXPERIMENTAL RESULTS

A. Simulation Environment and Setup

The simulation environment is constructed in a two-dimensional space with dimensions 10×12, corresponding to the warehouse floor plan, with an update period of $\Delta t = 0.1\ s$. The robots move at a constant velocity based on the guidance vector from the artificial potential field. Static obstacles are modeled as circles, while packages may remain stationary or move slowly to simulate a dynamic warehouse environment.

TABLE I. SIMULATION PARAMETERS

Parameter	Value
Simulation periodΔt	0.1 s
Robot speed	5 m/s
Number of packages	20 packages
Obstacle radius	0.5 m
Robot-robot safe distance	0.3 m
ε initial	0.1
ε minimum	0.01

The above parameters were selected based on experiments to ensure system stability and the convergence capability of the reinforcement learning algorithm.

B. Comparison of Motion Trajectories Among Methods

The simulation results in Fig. 4, Fig. 5, and Fig. 6 illustrate the trajectory performance of the three path-planning methods. Fig. 4 shows that the pure APF method generates sharp turns and trajectory intersections in dense areas due to fixed control coefficients, causing limited obstacle avoidance, poor smoothness, and collisions. Fig. 5 indicates that the heuristic method improves trajectory smoothness and obstacle avoidance, but close vehicle movement and trajectory intersections still occur in the warehouse area. Fig. 6 demonstrates that the hybrid method provides more adaptive robot behavior, smoother trajectories, fewer intersections, and no inter-vehicle collisions, confirming improved safety and operational stability.

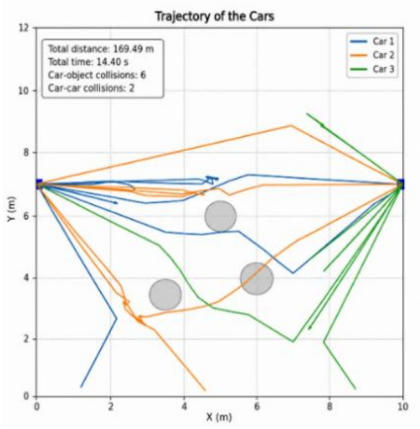

Figure 4. Artificial Potential Field with fixed coefficients (pure APF)

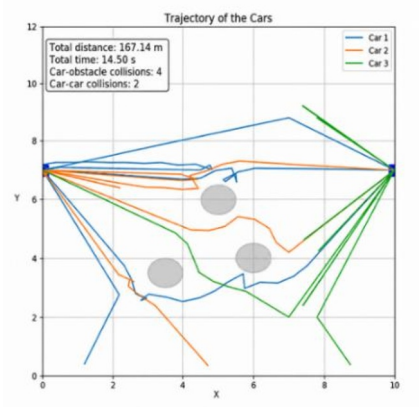

Figure 5. Coefficient-adjustment method based on heuristic rules

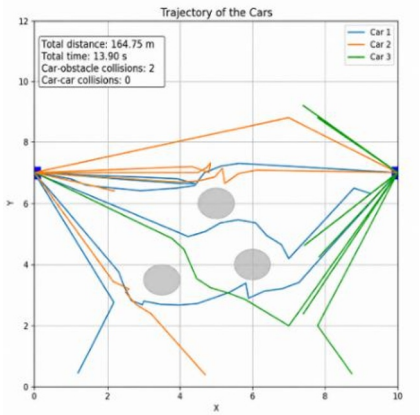

Figure 6. Proposed method combining heuristic rules with reinforcement learning (Hybrid).

TABLE II. COMPARISON OF SIMULATION RESULTS AMONG METHODS

Method	Total distance (m)	Time (s)	Vehicle-obstacle collisions	Vehicle-vehicle collisions
Pure APF	169.49	14.40	6	2
Heuristic	167.14	14.50	4	2
Hybrid	164.75	13.90	2	0

From TABLE II, it can be seen that the proposed method achieves the highest overall performance. Compared with pure APF, the hybrid method reduces the travel distance, shortens task completion time, and decreases the number of obstacle collisions. Compared with the heuristic method, the proposed method improves safety because no vehicle-to-vehicle collisions are recorded in the simulation scenarios, while still maintaining a shorter trajectory and lower execution time. This result indicates that combining heuristic rules with reinforcement learning enables the system to adapt better in a dynamic multi-robot environment.

C. Simulation Results in the Case of Moving Packages

In Fig. 7, vehicle 2 is initially assigned to package E, which is the optimal target according to the Hungarian algorithm based on the instantaneous cost matrix. However, as the environment changes, package E1 later provides a lower total cost by considering both the vehicle-to-package and package-to-warehouse distances, as shown in Fig. 8.

Since vehicle 2 is still in the early or middle stage of approaching E and the cost reduction from switching to E1 is significant, the proposed system performs reassignment. As a result, vehicle 2 smoothly changes direction toward package E1 without collisions with other robots or obstacles.

The simulation results show that, in an environment with moving packages, the robot trajectories have more complex and highly curved paths because the robots must continuously adjust their heading to track the targets. In a typical situation, when the cost of package E1 becomes lower than that of E, the system performs reassignment and vehicle 2 smoothly changes direction toward E1 without causing trajectory oscillation.

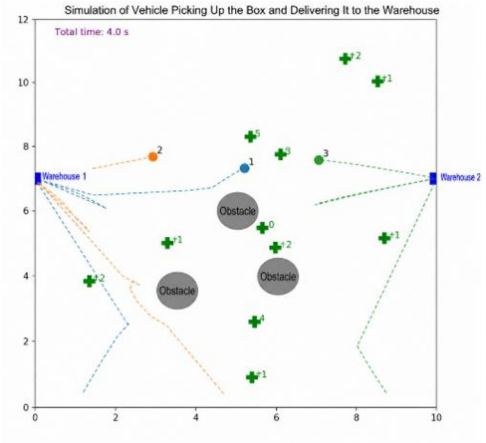

Figure 7. Vehicle 2 moving toward package E

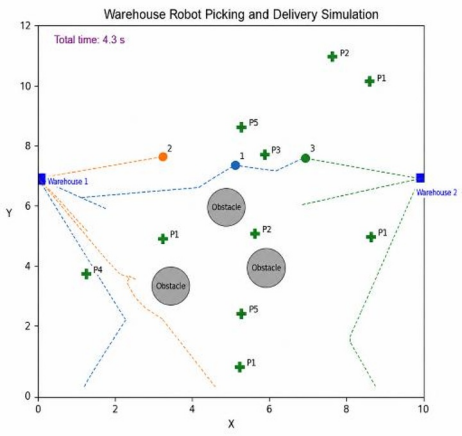

Figure 8. Vehicle 2 switching to package E1

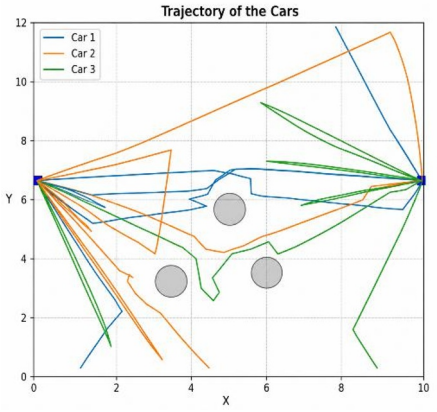

Figure 9. Global result

Although this is only an illustrative case, the combined Hungarian-Q-learning mechanism allows the system to continuously adjust tasks toward beneficial trends, thereby reducing global operating cost and improving the overall performance of the multi-robot system.

D. Average Statistical Results Over Multiple Simulation Runs

To objectively evaluate the effectiveness of the proposed method, each control configuration was executed over 20 independent simulation runs with different random seed values. The three compared methods include: (i) APF with fixed coefficients (Fixed), (ii) non-learning heuristic, and (iii) the proposed method combining heuristic rules with Q-learning (Hybrid).

TABLE III. COMPARISON OF AVERAGE RESULTS AFTER 20 SIMULATION RUNS

Method	Distance (m)	Time (s)	Obstacle collisions	Vehicle-vehicle collisions
Fixed	169.04	14.92	5.30	1.60
Heuristic	172.88	15.01	3.25	1.55
Hybrid	169.28	14.44	1.55	1.25

The results show that the proposed method (Hybrid) achieves the lowest task completion time, averaging 14.44 s, while also significantly reducing the number of obstacle collisions compared with the remaining methods. Although the total travel distance is slightly higher than that of fixed APF, this trade-off improves the safety and operational stability of the system.

The simultaneous reduction in task completion time and number of collisions indicates that the reinforcement learning mechanism helps the system achieve a better balance between navigation efficiency and operational safety. These statistical results show that integrating Q-learning into the motion control and task reassignment processes has the potential to improve the performance of multi-robot systems in dynamic environments.

VII. CONCLUSION

This paper proposed a method that combines Q-learning and the Hungarian algorithm within a hierarchical control architecture for multi-robot systems in dynamic environments. Q-learning is used to adaptively tune motion control and support task-reassignment decisions, while the Hungarian algorithm ensures instantaneous cost optimization.

Simulation results show that the proposed method reduces travel distance, shortens task completion time, and limits collisions, while enhancing the stability and adaptability of the system compared with traditional methods and recent multi-agent reinforcement learning approaches [11], [13]. This confirms the effectiveness of combining reinforcement learning with classical optimization algorithms in multi-agent robotic systems.

VIII. REFERENCES

[1] N. H. Mai and T. V. Dung, “A method for football team model optimization and application of the optimization control,” *Proceedings of the 12th International Scientific and Practical Conference: Environment. Technology. Resources*, vol. II, pp. 84–88, Rezekne, Latvia, 2019.

[2] H. W. Kuhn, “The Hungarian method for the assignment problem,” *Naval Research Logistics Quarterly*, vol. 2, no. 1–2, pp. 83–97, 1955.

[3] R. E. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1957.

[4] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, 1992.

[5] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[6] B.P. Gerkey and M.J. Mataric, “A formal analysis and taxonomy of task allocation in multi-robot systems,” *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[7] Y. Liu and G. Nejat, “Multirobot cooperative learning for semistructured environment exploration,” *IEEE Transactions on Cybernetics*, vol. 44, no. 3, pp. 361–372, 2014.

[8] L. Busoniu, R. Babuska, and B. De Schutter, “A comprehensive survey of multi-agent reinforcement learning,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 38, no. 2, pp. 156–172, 2008.

[9] S. Ge and Y. Cui, “Dynamic motion planning for mobile robots using potential field method,” *Autonomous Robots*, vol. 13, no. 3, pp. 207–222, 2002.

[10] H. Chakraa, F. Guerin, E. Leclercq, and D. Lefebvre, “Optimization techniques for Multi-Robot Task Allocation problems: Review on the state-of-the-art,” *Robotics and Autonomous Systems*, vol. 168, 2023.

[11] R. Burkard, M. Dell’Amico, and S. Martello, *Assignment Problems*, SIAM, 2009.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

[13] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, “Market-based multirobot coordination: A survey and analysis,” *Proceedings of the IEEE*, 2006.

[14] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed., Cambridge University Press, 2004.