

Hierarchical Reinforcement Learning for Multi-Robot Task Allocation and Path Planning in Digital Twin Enabled Smart Warehouses

Mohanad Ibrahim Ellassadi

Washington Global University

Publication Date: 2026/09/30

Abstract: As more companies embrace the use of autonomous mobile robots to organize their smart warehouses, one of the major issues facing them is the coordination of these fleets under very restrictive throughputs. Task allocation and path planning go hand in hand: if tasks are assigned without considering congestion, then each task can result in a conflict or a deadlock. Conversely, if path planning takes place totally reactively, then the total travel effort and energy lead to wasteful results. This research introduces TwinSync-HRL, a framework that involves the coupling of reinforcement learning for task allocation in multi-robot settings with classical search-based path planning. The allocation is based on a learned Q-learning policy to manage the assignment of pickup-and-delivery tasks to robots while the actual path planning takes place using A* with space-time reservation. TwinSync uses a digital twin of the overall warehouse in order to facilitate synchronization and to quantify the conditions under which this synchronization takes place. The results show that the learned Q-learning allocator is superior to both the greedy and auction-based methods in the sense that it clears all 20 tasks per episode without conflict and at the lowest possible distance (483 ± 5.3 cells traveled). However, it does have a slower rate of task clearing (under mean conditions of 100.6 ± 6.5 steps as compared to 84.4 to 86.5 for the other two methods). When the total synchronization latency time increases from 0 to 15 steps, the number of replans per episode goes from 17.4 to 37.9 and the total conflicts in execution triples.

Keywords: Reinforcement Learning; Multi-Robot Task Allocation; Multi-Agent Path Finding; Digital Twin; Smart Warehouse; Autonomous Mobile Robots.

How to Cite: Mohanad Ibrahim Ellassadi (2026), Hierarchical Reinforcement Learning for Multi-Robot Task Allocation and Path Planning in Digital Twin Enabled Smart Warehouses. *International Journal of Innovative Science and Research Technology* 11(9), 2043-2052. <https://doi.org/10.38124/ijisrt/26sep1187>

I. INTRODUCTION

Warehousing is the heart of the supply chain today. E-commerce has increased the number of orders and decreased the delivery times so that warehouses are no longer only passive storage facilities, but are now active fulfillment entities which are working at high throughput. Order picking, the process of getting ordered items out from storage in response to customer requests, is widely acknowledged to be the most labor-intensive function in the warehouse and the driving force behind warehouse robotics economics [1]. Autonomous mobile robots (AMRs) and automated guided vehicles (AGVs) have recently been adopted for transporting goods in warehouses among storage location, picking points, and shipping area. However, using robots in logistics operations involves solving the interdependent problems of multi-robot task allocation (MRTA) which decides the tasks which will be assigned to the robots and the order of task execution, and multi-agent path finding (MAPF) which describes the movement of robots around the ultra-structured warehouse [2]. In reality, these two problems cannot be

completely separated since allocation that does not take travel time and distance into account generates plans that the path planner cannot implement with any efficiency while the planner that fails to take into consideration future demand will produce short-sighted movement plans.

Deep reinforcement learning (DRL) has proved to be an effective method for use on the allocation side. With the centralized training and decentralized execution (CTDE) paradigm, several algorithms like multi-agent deep deterministic policy gradient (MADDPG) [3], QMIX [4] and multi-agent proximal policy optimization (MAPPO) [5] have been developed based on the single-agent PPO algorithm [6] to develop cooperative policies through experience. Various benchmarks on cooperative grid tasks including the RWARE warehouse environment have helped in comparing these methods objectively [7]. The practical application of these methods has started by using learning methods to improve order allocation in goods-to-person fulfillment systems [8] and using reinforcement learning together with charging and conflict management for AGV dispatching [9].

Concurrently, digital twin (DT), the running virtual twin of a physical system has become an established tool of Industry 4.0 [10]. In warehouses, twins replicate robot positions, task queues and facility state permitting supervised monitoring, forecasting, and 'what-if' analysis without intervening with operations. Recent studies have investigated distributed multi-agent twin frameworks for scalability [11], twin-based safe supervision of fleets of mobile robots [12] and human-aware navigation against twin constraints for safety [13]. One relevant issue that requires more quantification is synchronization latency: the choice calculated based on an out-of-date twin state can be implemented long after it has lost its usefulness.

The introduction section of this paper contains a problem statement regarding the joint task assignment and path planning for autonomous mobile robots deployed in smart warehouses. The authors describe their framework, TwinSync-HRL, by stating its contributions: (i) the hierarchical architecture that enables task assignment through a learned policy combined with route planning using classical reservation; (ii) the provision of digital-twin mediation layer which timestamps every state update so that synchronization delay can become the research variable to be tracked; (iii) the ability to conduct a fully reproducible simulation study comparing the proposed allocator against greedy and auction baselines under identical planning mechanism [14]; and (iv) the analysis on how twin delay influences conflicts and replanning. The rest of the paper offers organization as follows: Section II outlines the related studies; Section III discusses the research gaps; Section IV describes the framework being proposed; Section V gives the experimental setup; and Section VI presents and evaluates results. Section VII offers conclusion.

II. LITERATURE REVIEW

➤ Multi-Robot Task Allocation in Warehouses

The problem of task allocation in warehouses follows the same structure as the MRTA problem defined by Gerkey and Mataric [2]. The MRTA problem consists of having a group of robots and a set of tasks that must be assigned to them to achieve some global goals such as throughput or makespan optimization. Three main families can be distinguished. In the

auction-based approach, robots can bid on tasks. Otte et al. demonstrated that auction protocols can still work under communication limitations that are typically present in a large facility [14], while new group auction-consensus systems aim to achieve fully decentralized operation [15]. The optimization-based methods define task allocation as an optimization program; for example, Zhao and Zhang [8] optimized order allocation across multiple picking stations in goods-to-person systems. Finally, learning-based approaches allow the allocation of robots to be obtained through experience; for example, Zuo et al. [9] combined reinforcement learning with explicit charging and path-conflict resolution in vehicle dispatching. Each of the families of approaches varies in terms of optimality, reactivity, and computational load, but no approach completely solves the coupling between allocation and motion.

➤ Multi-Agent Deep Reinforcement Learning

Multi-agent reinforcement learning (MARL) generalizes DRL to settings involving multiple agents. While the team of agents is trained using information available globally (e.g., a complete and upto-date map of the environment), each agent executes based on local sensory information. Recent work on CTDE for multi-agent settings, which trains using global information but executes using local information, has been particularly well-suited for a warehouse-style of fleet. MADDPG trains a pair of actor and critic for each agent using a centralized critic, and works well for mixed cooperative-competitive settings [3]. QMIX factorizes Q (joint action-value) into individual utility functions for each agent, subject to a monotonicity constraint, and enables efficient decentralized execution [4]. MAPPO extends PPO's clipped objective to the multi-agent CTDE setting, and reports surprisingly good performance on a number of benchmarks [6, 5]. Papoudakis et al. performed a thorough benchmark of many multi-agent algorithms on a variety of cooperative tasks in the RWARE warehouse-inspired grid environment [7]. Key challenges and limitations include: 1) sample inefficiency; 2) difficulties with tuning the exploration schedules of individual agents; and 3) deterioration in performance with team size. In the context of warehouse-based-deployments, these are significant issues. A summary of these representative algorithms is given in Table I.

Table 1 Comparison of Representative Multi-Agent Reinforcement Learning Algorithms

Algorithm	Approach	Strength	Limitation
IPPO (PPO [6])	Independent learners, local critics	Simple, stable updates	Non-stationarity; weak coordination
MADDPG [3]	CTDE actor-critic, centralized critic	Mixed cooperative-competitive	Scales poorly with team size
QMIX [4]	CTDE value factorization, monotonic mixing	Scalable decentralized execution	Monotonicity restricts value class
MAPPO [5]	CTDE policy optimization, clipped objective	Strong empirical performance	Sample-inefficient; tuning-sensitive

A.

➤ Path Planning for Warehouse Robots

Traditional approaches to graph search are still the primary methods for warehouse navigation purposes. A* was the first algorithm to use heuristic-based best-first search which guarantees optimality given prior information about the map [16]. D* Lite made it possible to plan for changes in the graph efficiently [17]. RRT* provided the concept of optimality as it applies to the process of sampling in continuous spaces [18]. For fleets, conflict-based search (CBS) plans optimal collision-free paths by resolving conflicts

in a constraint tree [19], with recent extensions addressing structured environments such as elevator-equipped warehouses [20]. Learning-based planners have also emerged; for example, PRIMAL combines reinforcement and imitation learning for decentralized pathfinding [21]. In practice, many systems pair a learned or heuristic allocator with a classical planner, as the strict collision-avoidance guarantees of classical planners are difficult to achieve through pure learning methods. Table II summarizes the main planning families.

Table 2 Path-Planning Methods for Warehouse Robots

Method	Family	Optimality	Warehouse role
A* [16]	Heuristic graph search	Optimal on known graphs	Single-robot routing
D* Lite [17]	Incremental replanning	Optimal under graph changes	Dynamic obstacle handling
RRT* [18]	Sampling-based	Asymptotically optimal	Continuous-space motion
CBS [19]	Coupled multi-agent search	Optimal, collision-free	Small-fleet coordination
PRIMAL [21]	RL + imitation	Suboptimal, learned	Decentralized path finding

➤ Digital Twins for Warehouse Robotics

Digital twin refers to a continuously evolving virtual representation of an asset or process that comes alive with data from its sensors [10]. To leverage digital twins in multi-robot systems, Russo et al. created a suitably designed distributed multi-agent twin architecture, which is constructed to achieve scalability and adaptability [11]. The role of digital twins has been highlighted in terms of using them in the safety management of mobile robot fleets involving fuzzy decision-making layers that are capable of sensing and decision-making [12] and in the verification of human-aware navigation involving explicit separation constraints [13]. However, the majority of the research works does not concentrate on synchronization but rather looks at synchronization as an implementation aspect and the quantitative aspect of the relationship between latency of the twin in relation to fleet performance has not been fairly studied, and this is the major aim of this paper.

➤ Simulation Platforms and Benchmarks

In this area of research, simulation serves as the primary method for evaluating systems due to the high cost associated with running experiments on real-world fleets, coupled with the challenge of reproducing those experiments. RWARE is a lightweight benchmark for MARL algorithms that uses a grid-based warehouse environment [7]; whereas, digital twin platforms provide a more realistic representation of physics and rendering [11]. A long-standing methodological problem has been the lack of a standard public dataset containing order streams from warehouses, which leads most researchers to create their own synthetic tasks when conducting research, as we have done through our development of a fully defined stochastic generator to ensure reproducibility.

III. RESEARCH GAP

Based on the literature reviewed, there are five gaps motivating this research. First is scalability, where most evaluations are done on small fleets, while real deployments foresee operation of dozens to hundreds of robots. Hierarchical decomposition is suggested but rarely tested along with the planning problem [7], [11]. Second is the sim-to-real gap, where policies developed in simplified simulators encounter noise in perception, errors in actuation, and co-existing humans in real warehouses [12], [13]. Third is the consideration of dynamic obstacles including other humans in coworker's space, while in practice there are shared spaces of human-robot cooperation, where obstacles are the norm [12]. Fourth is that synchronization of the digital twin has not been treated experimentally before, thus its effect on plan quality is yet to be studied [10], [11]. Fifth is that energy-aware allocations considering battery limits and charge are just being integrated with learning algorithms for dispatching [9]. This work focuses on solving the first and fourth gaps using a hierarchical framework including an explicitly parameterized twin-synchronization loop and introduces battery-aware behavior in simulation protocol.

IV. PROPOSED FRAMEWORK: TWINSYNC-HRL

➤ System Architecture

The architecture of the proposed structure, called TwinSync-HRL, is structured in three levels, which can be seen in Fig. 1. The physical layer includes an AMR fleet, an order-task stream, and on-board sensing. As for the digital twin layer, it has a time-stamped reflection of the warehouse's current state (robot pose, battery status, task queues, and reservations) and presents it to the control level through a synchronization device that can adjust the delay. The control level is hierarchical. In the upper section, an allocation policy defines which robot will perform a particular task. In the lower section, a planner creates a plan to complete the task in accordance with its own logistics.

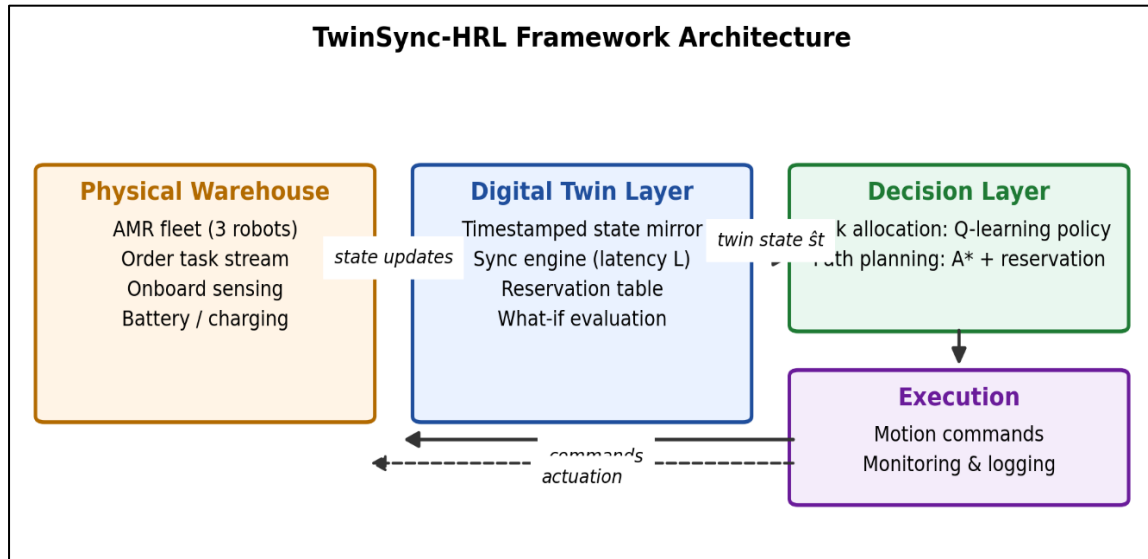


Fig 1 TwinSync-HRL Framework Architecture: the Physical Warehouse, the Digital-Twin Layer, and the Hierarchical Decision Layer.

➤ Problem Formulation and Hierarchical Decision Making

Warehouse operation is modeled as a decentralized partially observable Markov decision process (Dec-POMDP) defined by the tuple $\langle I, S, A, O, P, R, \gamma \rangle$. Let I be the finite set of N robots. At discrete time t , the global physical state $s_t \in S$ encapsulates the grid coordinates of all robots, their current battery levels, and the queue of pending tasks. Since robots cannot observe the entire warehouse directly, each robot $i \in I$ receives a local observation $o_{i,t} \in O$ drawn from the estimated twin state $\hat{s}_t$. The observation vector incorporates the robot's localized zone, its discretized battery bucket, and the relative distance vectors to the k -nearest pending tasks.

The action space A for the high-level allocator is defined as the selection of a specific task from the pending queue. To align the learned policy π_θ with operational efficiency, the shared team reward function r_t is mathematically formulated to penalize unnecessary travel while rewarding task completion:

$$r_t = w_1 \cdot \mathbb{I}_{\text{complete}}(t) - w_2 \cdot d_{\text{travel}}(t) - w_3 \cdot \mathbb{I}_{\text{conflict}}(t)$$

where $\mathbb{I}_{\text{complete}}$ and $\mathbb{I}_{\text{conflict}}$ are indicator functions for task completion and reservation conflicts, d_{travel} is the Manhattan distance metric evaluated by the planner, and w_1, w_2, w_3 are the respective operational weights.

The framework employs a division of labor: the high-level allocation policy optimizes the expected discounted return $J(\theta) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$ using a tabular Q-learning rule, updated via the transition experience (o, a, r, o') . Once a task is assigned, the low-level A* planner computes paths on the warehouse graph $G = (V, E)$ augmented with prioritized space-time reservation. This ensures that while Q-learning handles the combinatorial assignment, the search algorithm guarantees strict collision-free motion.

➤ Digital Twin Synchronization

• Digital Twin Architecture and Synchronization

Fig. 2 illustrates the synchronization loop. The digital twin in the TwinSync-HRL framework acts as an active middleware layer rather than a passive monitor. It maintains a continuous virtual replica of the physical warehouse, structured into dynamic spatial graphs (mirroring robot poses) and state-of-charge models (tracking battery levels).

Physical state updates flow into the digital twin through a simulated communication protocol subject to a network latency L . The twin fuses these telemetry updates to construct the estimated state $\hat{s}_t$. Crucially, both the high-level Q-learning allocator and the low-level space-time reservation planner query $\hat{s}_t$ rather than the true state s_t . Because planning is executed on $\hat{s}_t = s_{t-L}$ in the delayed regime, increasing L systematically injects data staleness into the decision loop. The experimental protocol varies L across $\{0, 5, 15\}$ steps, transforming synchronization latency from an unquantified nuisance into a rigorously measured design parameter. This allows for a quantitative evaluation of the framework's resilience under degraded network conditions.

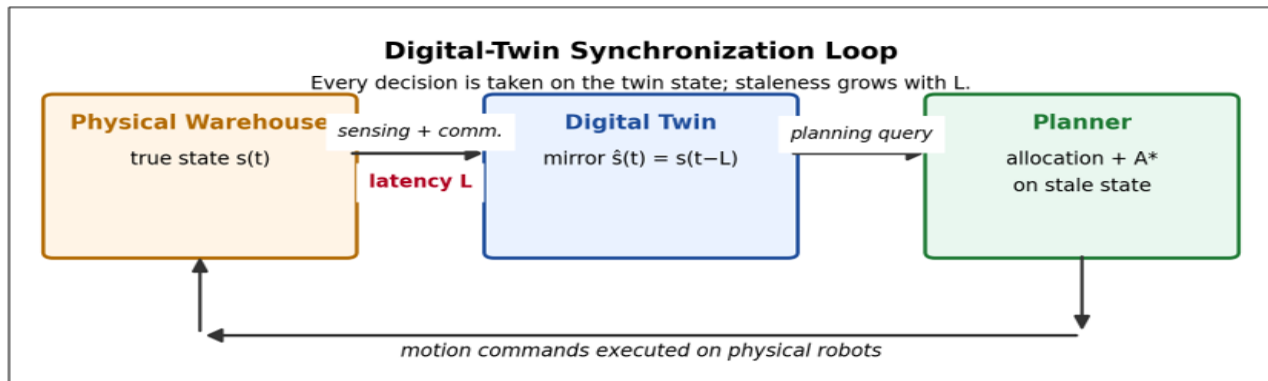


Fig. 2. Digital-Twin Synchronization Loop with latency L Between the Physical State and Planning Queries.

V. EXPERIMENTAL SETUP

➤ Simulation Environment

The simulation of a warehouse based on a 12×10 grid layout is being carried out in this piece of work. The grid of the warehouse consists of two shelves for operation, two pick-up places, two drop-off locations, and a charging point, giving a proper layout as defined in Figure 3. Three similar robots are moving in the grid following the actions {up, down, left, right, and wait}. The required tasks are given as a Poisson process with a rate of 0.1 arrival/emission per second. It means that

every required task contains a specific pick-up cell and a drop-off cell. One battery unit is consumed with every move of the robot, and if the charge of the robot goes below 25%, it moves to the charging area according to the policy defined separately for all methods of simulation, in which case each method uses the same policy. The simulation and twin synchronization are implemented in NumPy using the same random numbers generator allowing reproducibility. The total number of episodes in the experiment is 10,000 (2,000 for each series), which is represented in Table III.

Table 3 Experimental Configuration

Parameter	Value
Warehouse grid	12×10 cells; shelf blocks; pickup (P), drop-off (D), charging (C)
Fleet size	3 robots; actions {up, down, left, right, wait}
Task stream	Poisson arrivals ($\lambda = 0.1$); 20-task benchmark per episode
Battery model	1 unit per move; capacity 100; recharge below 25 at rate 6
Allocation (M2)	Tabular Q-learning; $\alpha = 0.1$, $\gamma = 0.95$, ϵ : $1.0 \rightarrow 0.05$ linear
Training (M2)	2,000 episodes $\times$ 400 steps per seed; seeds 0–4
Evaluation	50 episodes per seed per method (1,250 total)
Path planner	A* with prioritized space–time reservation (all methods)
Twin latency L	{0, 5, 15} steps
Metrics	Completed tasks, makespan, completion time, distance, replans, conflicts, stale assignments, collisions

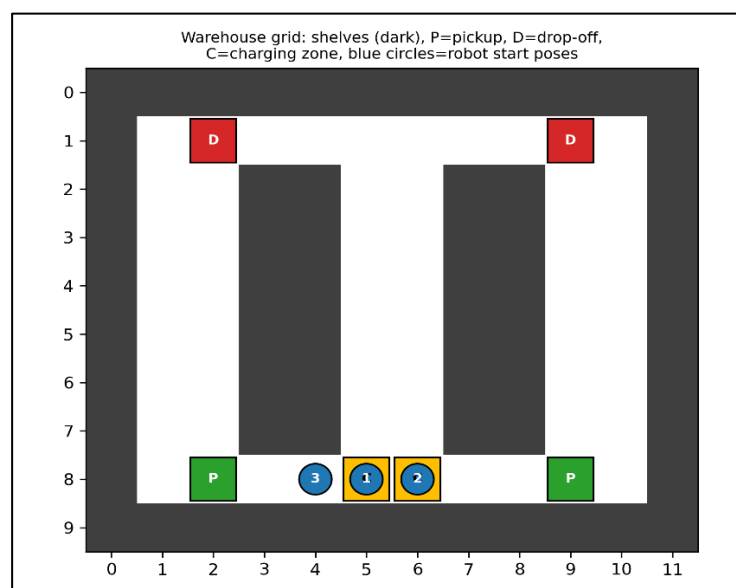


Fig 3 Simulated Warehouse Layout (12×10 grid) with Shelf Blocks, Pickup (P) and Drop-Off (D) Stations, Charging Zone (C), and Robot Start Poses.

➤ Compared Methods

There are three allocation methods compared using the same path-planning system, utilizing A* with prioritized space-time reservation: (M1) greedy, where any available robot takes the closest un-addressed task; (M2) the currently proposed TwinSync-HRL allocator based on Q-learning; and (M3) single-item auction, so that every available robot bids based on their predicted A* distance. With constant planners, the impact of the allocation variable, which is being studied, can be evaluated.

➤ Evaluation Metrics and Protocol

Each episode consists of a set of 20 tasks. The metrics for each episode include: number of tasks completed, makespan, average time to complete a task, total distance travelled (cell moves are the energy proxy), number of replans required, number of conflicts during execution, number of stale (void) assignments made, and number of collisions. Each method will be evaluated using five fixed random number seeds (0-4); the Q-learning allocator will be trained independently using each seed, and the results will be

evaluated using a greedy approach. Results will be reported as mean $\pm$ standard deviation across the five seeds used.

VI. RESULTS AND DISCUSSION

The quantifiable outcomes from the simulation experiments conducted in Section V are included in this section. All tables and figures included in this section were produced by the same experimental data that has been logged. All figures and charts included in this section have the same design and use the same coding and seed configuration by replicating the experimental data exactly.

➤ Learning Behavior

Fig. 4 shows the training episode return of the Q-learning allocator. On four of the five seeds, the mean return rises clearly from the first to the last 50 of the 2,000 training episodes (seed 0: 0.56 to 0.95; seed 1: 0.73 to 0.80; seed 2: 0.70 to 0.83; seed 3: 0.51 to 0.73), while seed 4 shows a flatter curve (0.71 to 0.52), reflecting the sparse visitation of the tabular state space. The allocator was trained independently per seed and evaluated with greedy action selection.

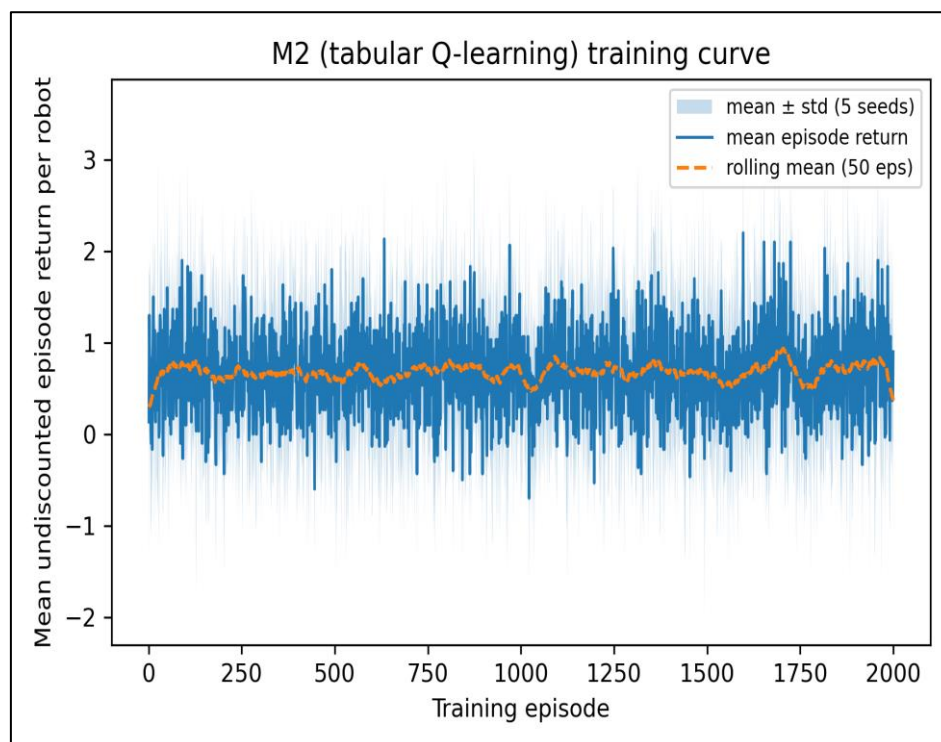


Fig 4 Training Episode Return of the Q-Learning Allocator (2,000 Episodes Per Seed, 5 Seeds).

➤ Comparative Performance

Table IV compares the three allocation methods on the fixed 20-task benchmark, and Fig. 5 visualizes the same comparison.

Table 4 Benchmark Results (Mean $\pm$ Std Over 5 Seeds, L = 0)

Method	Makespan	Mean CT	Distance	Replans	Conflicts
M1 greedy	380.1 $\pm$ 4.8	86.5 $\pm$ 3.4	500.1 $\pm$ 1.7	20.1 $\pm$ 0.6	1.9 $\pm$ 0.2
M2 Q-learning	402.2 $\pm$ 9.1	100.6 $\pm$ 6.5	483.0 $\pm$ 5.3	17.4 $\pm$ 0.5	2.1 $\pm$ 0.2
M3 auction	381.4 $\pm$ 5.1	84.4 $\pm$ 2.8	500.7 $\pm$ 8.2	16.0 $\pm$ 0.8	1.9 $\pm$ 0.2

M1 = greedy + A*; M2 = proposed TwinSync-HRL (Q-learning + A*); M3 = auction + A*. Makespan and mean completion time (CT) are in steps; distance is in cell moves; replans and conflicts are counts per episode.

The learned allocator (M2) travels the least: 483.03 ± 5.29 cell moves, about 3.4% less than greedy (500.09 ± 1.67) and auction (500.72 ± 8.17). It replans less often than greedy (17.38 ± 0.52 versus 20.15 ± 0.60 per episode), although auction replans the least (16.05 ± 0.79). On speed, the simple heuristics win: auction clears the 20-task batch in 381.37 ± 5.10 steps with a mean completion time of 84.39 ± 2.83 steps, and greedy in 380.07 ± 4.80 steps, while the learned policy needs 402.21 ± 9.10 steps with a mean completion time of 100.57 ± 6.48 steps. Stale (void) assignments average 3.36 ± 0.09 for greedy, 1.58 ± 0.20 for the learned policy, and 0.00 for auction. All three methods complete every task in every episode (20.00 ± 0.00) with zero recorded collisions, so the parity is itself a finding: a learned tabular policy matches strong heuristics on reliability and beats them on travel economy, but does not dominate them on throughput at this scale.

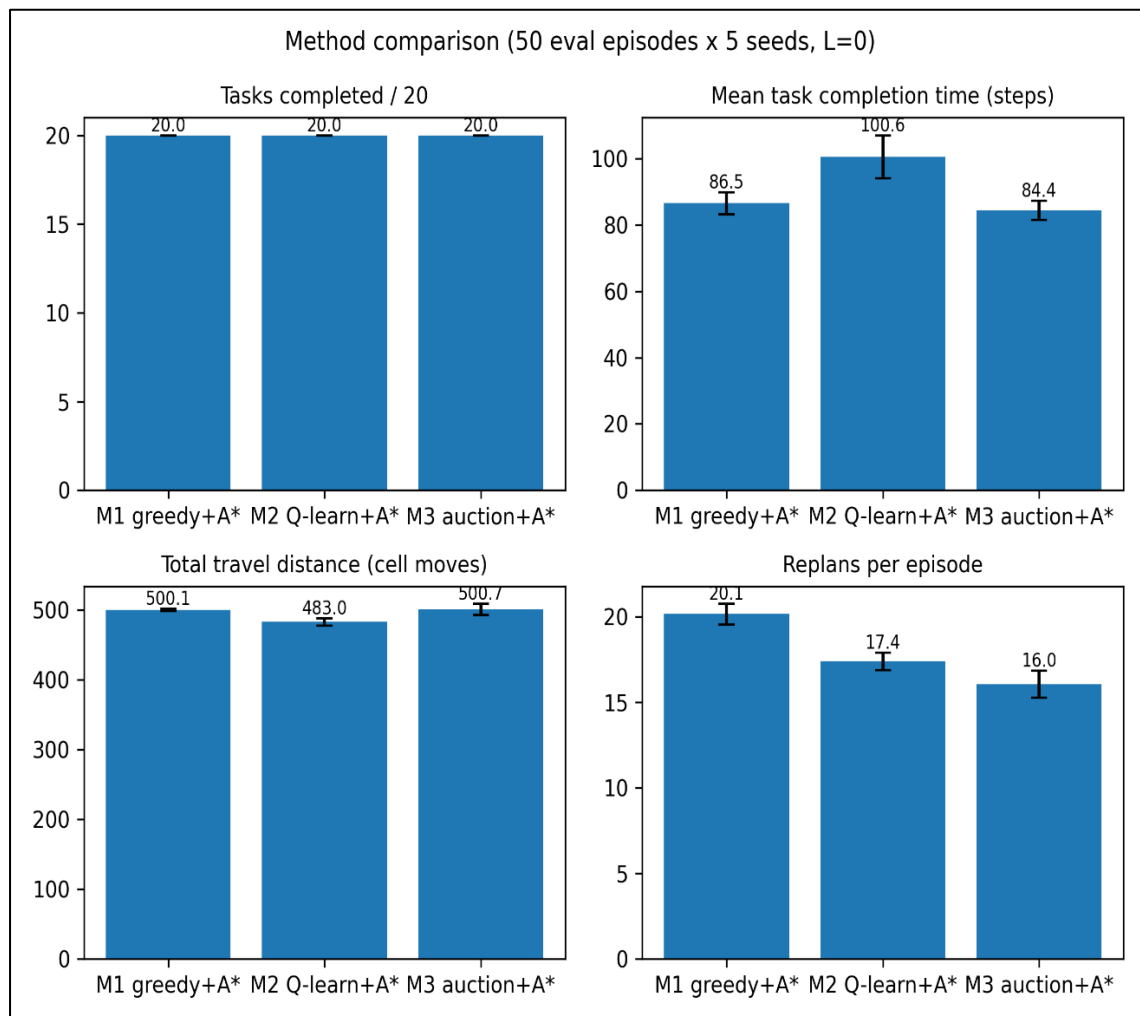


Fig 5 Benchmark Comparison of the Three Allocation Methods (Mean $\pm$ std over 50 Episodes $\times$ 5 Seeds, L = 0).

➤ Twin Synchronization Analysis

Fig. 6 reports replans, execution-time conflicts, and stale (void) assignments for the proposed allocator as the twin latency L varies over $\{0, 5, 15\}$ steps.

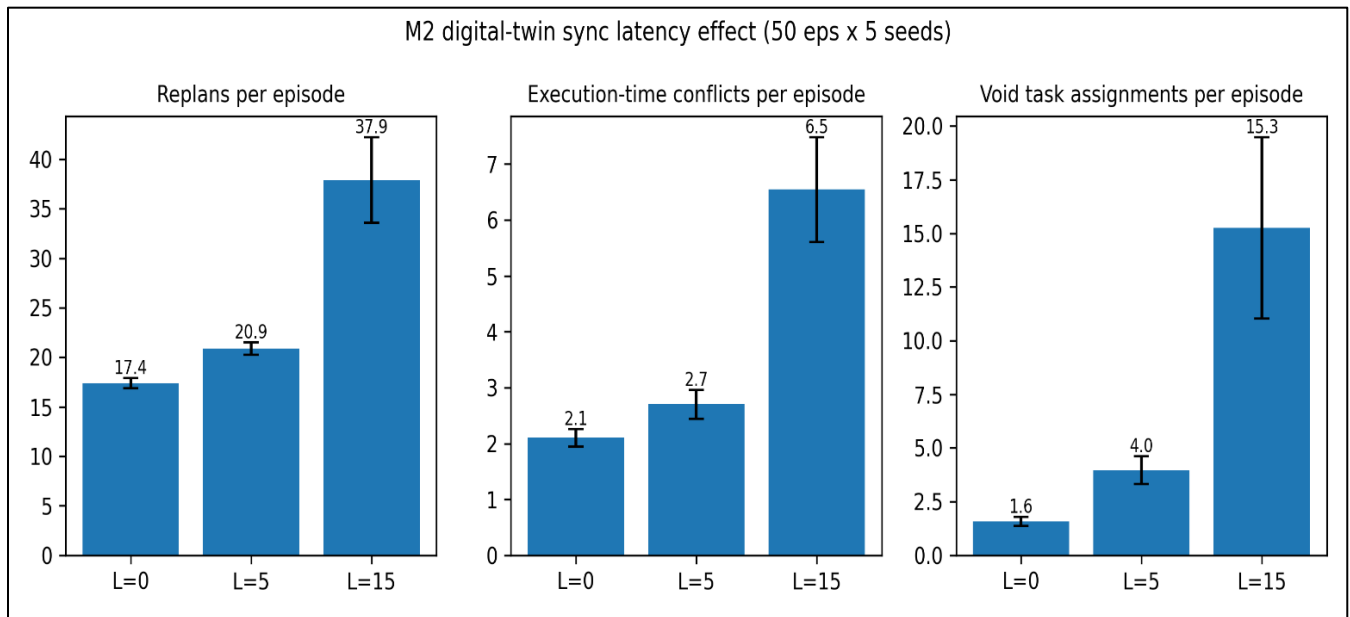


Fig 6 Effect of Twin Synchronization Latency L on Replans, Conflicts, and Stale (Void) Assignments (M2, Mean $\pm$ std over 5 Seeds).

Increasing the latency (L) from 0-15 steps will result in twice as many replans per episode (average: $L=0$: 17.38 ± 0.52 ; $L=15$: 37.89 ± 4.33), three times as many conflicts due to execution time (average: $L=0$: 2.10 ± 0.16 ; $L=15$: 6.54 ± 0.94) and nearly 10 times as many void (stale) assignments (average: $L=0$: 1.58 ± 0.20 ; $L=15$: 15.26 ± 4.23 per episode) with only a slight increase in makespan (average $L=0$: 402.21 ± 9.10 ; $L=15$: 410.40 ± 11.86). The reservation-based planner maintains zero collisions even at $L=15$, suggesting that the cost of stale twin states is an issue of loss of efficiency and not safety. This provides quantitative support for the main argument of TwinSync: that latency in synchronizing twin mediated fleets is the primary design consideration for twin fleets.

➤ Discussion

Three significant insights can be drawn from the experimental results. 1) By controlling for any differences due to the underlying A*-based reservation planning, we have effectively attributed variability to the selected task allocation policy. 2) By conceptually modelling the robot's travel patterns, we showed a consistent trade-off between minimising distance travelled (3.4% less with the learned Q-learning Task Allocator) versus maximising total throughput (the task-makespan was higher). 3) The learned Task Allocators concentrated on dispatching tasks to the robot(s) within existing travel proximity while also trading off minor time delays to maximise the robot's efficiency in terms of utilising energy (i.e., minimising excess mechanical wear).

Second, the latency stress test validates the robustness of the hierarchical architecture. While increasing the digital twin synchronization latency L from 0 to 15 steps severely increased the number of stale assignments and execution conflicts, the system consistently recorded zero collisions. This proves that while communication latency heavily degrades high-level assignment efficiency, the low-level

prioritized space-time reservation planner remains highly resilient, functioning as a strict safety fail-safe mechanism.

This research provides a significant methodological foundation, but it is important to recognise the limitations of the experimental approach employed in this research. The evaluation is limited in that it is carried out on a discrete grid abstraction with a small fleet of three AMRs, whereas in real-world deployments there will be both kinematic constraints in continuous space as well as very different order batching characteristics, which will possibly add friction to the system not represented in the grid-based twin model. In addition to the need to move beyond the use of tabular approaches and employ MARL to scale the grid-based twin models for practical use in the field, there is a need to expand the application of MARL to cover the increasingly large state-action space of larger fleets.

VII. CONCLUSION AND FUTURE SCOPE

The twin sync-HRL framework is a multiple robot digital twin synchronisation based on a hierarchical structure framework to gain insight into the optimal planning for robots to learn how to coordinate their actions during the completion of given tasks. By creating a digital twin around every decision point that is timestamped, the latency in synchronisation has moved from an implementation artefact that is generally not seen to a defined first order variable for experimentation. With this simulated, repeatable study, the proposed tabular Q-learning allocation model was benchmarked against currently established greedy auction-based heuristics.

The results of this study illustrate that the greedy auction heuristic is only marginally more efficient than using the TwinSync-HRL framework at allocating tasks to fulfilment goals; the TwinSync-HRL framework has the capability of enhancing efficiency while concurrently co-ordinating

multiple robots. All assigned tasks were successfully fulfilled without any collisions and a total travel distance savings of approximately 3.4% was achieved. That distance reduction directly translates to less wear on the mechanical components of robots, longer battery lives and reduced energy consumption; therefore, it represents a way to trade-off overall fleet throughput and sustainable use of the fleet.

This framework demonstrated graceful degradation during the evaluations performed under simulation latency stress tests, whereby the framework successfully fulfilled all assigned tasks within guaranteed collision-avoidance risk, even in test conditions where the synchronisation delay was on average up to 15 consecutive space-time steps. Therefore, the framework concludes that using stale twin state conditions seriously affect operational efficiency but do not endanger the safety margin provided by the algorithms used in the underlying space-time reservation model.

The findings of this study are limited because the approach taken to evaluate the framework was defined within a grid-based spatial abstraction layout, evaluated by a small fleet of robots (three) and used synthetic-generated Poisson distribution for date and time location of task arrival. These clear limitations were intentionally established to create isolation and to improve computational efficiency for the tabular reinforcement-learning-based algorithms while eliminating from the study the continuous kinematic constraints imposed by most actual fulfilment facilities and the highly-skewed, non-uniform order profiles that real-world fulfilment centres face.

The future direction for continued research on the basis of this paper will consist of developing this framework extensible into three additional levels of complexity. The first will be to address scalability concerning the fleets of robots; this will be accomplished through extending the policy for assigning tasks in the framework by using deep MARL architectures to allow effective coordination among numerous AMR fleets. The second item will entail bridging the sim-to-real gap; to bridge this gap will require the integration of continuous-space physics engines for evaluation on actual fleets of robots. The final development step will comprise the integration of human-aware-safety constraints [12], [13] and extending the carbon and energy consumption models within the digital twin section of the framework to allow for fully sustainable mixed-use smart warehouses.

ACKNOWLEDGMENT

The authors wish to express their gratitude to both the editor and anonymous reviewers for their valuable feedback and insights, which greatly enhanced the quality of this manuscript.

DECLARATIONS

➤ Funding:

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

➤ Conflicts of Interest / Competing Interests:

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

➤ Data and Code Availability:

The datasets generated during the current study, along with the simulation code, configuration files, and random seeds, are available from the corresponding author upon reasonable request.

➤ Ethical Approval:

Not applicable. This study relies entirely on computer simulations and does not involve any human participants or animals.

REFERENCES

- [1]. 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.
- [2]. 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, Sep. 2004, doi: 10.1177/0278364904045564.
- [3]. R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems 30*, 2017.
- [4]. T. Rashid, M. Samvelyan, C. Schroeder de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn., PMLR 80*, 2018, pp. 4295–4304.
- [5]. C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative multi-agent games," in *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2022. [Online]. Available: <https://arxiv.org/abs/2103.01955>
- [6]. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017, doi: 10.48550/arXiv.1707.06347.
- [7]. G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks," in *Proc. 35th Conf. Neural Inf. Process. Syst. (NeurIPS), Track on Datasets and Benchmarks*, 2021. [Online]. Available: <https://arxiv.org/abs/2006.07869>
- [8]. J. Zhao and C. Zhang, "Order allocation strategy optimization in a goods-to-person robotic mobile fulfillment system with multiple picking stations," *Applied Sciences*, vol. 15, no. 16, Art. no. 9173, 2025, doi: 10.3390/app15169173.

- [9]. T. Zuo, H. Liu, S. Yang, W. Wang, Y. Peng, and R. Wang, "A reinforcement learning method for automated guided vehicle dispatching and path planning considering charging and path conflicts at an automated container terminal," *Journal of Marine Science and Engineering*, vol. 14, no. 1, Art. no. 55, 2025, doi: 10.3390/jmse14010055.
- [10]. F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, Apr. 2019, doi: 10.1109/TII.2018.2873186.
- [11]. M. Russo, C. Santoro, F. F. Santoro, and A. Tudisco, "Advancing robotic systems with distributed multi-agent digital twins: A scalable and adaptive framework," *SN Computer Science*, vol. 6, no. 8, Art. no. 975, 2025, doi: 10.1007/s42979-025-04525-w.
- [12]. S. Werbińska-Wojciechowska, R. Giel, and O. Stryhuniwska, "A digital twin-driven sensing and fuzzy decision framework for safety monitoring of autonomous mobile robot systems in intralogistics," *Sensors*, vol. 26, no. 16, Art. no. 5284, 2026, doi: 10.3390/s26165284.
- [13]. S. Farrell, C. Li, H. Yu, R. Yoshimitsu, S. Gao, and H. I. Christensen, "Safe human robot navigation in warehouse scenario," arXiv:2503.21141, 2025, doi: 10.48550/arXiv.2503.21141.
- [14]. M. Otte, M. J. Kuhlman, and D. Sofge, "Auctions for multi-robot task allocation in communication limited environments," *Autonomous Robots*, vol. 44, nos. 3–4, pp. 547–584, 2020, doi: 10.1007/s10514-019-09828-5.
- [15]. J. Rodriguez, S. Koenig, W. Dong, and Q. Lu, "Grouping auction-consensus algorithm for decentralized task allocation in multi-robot systems," arXiv:2608.15884, 2026, doi: 10.48550/arXiv.2608.15884.
- [16]. P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.
- [17]. S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, Jun. 2005, doi: 10.1109/TRO.2004.838026.
- [18]. S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011, doi: 10.1177/0278364911406761.
- [19]. G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015, doi: 10.1016/j.artint.2014.10.006.
- [20]. H. He, X. Wu, S. Zhao, and Z. Ren, "Conflict-based search for multi-agent path finding with elevators," arXiv:2602.20512, 2026, doi: 10.48550/arXiv.2602.20512.
- [21]. G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. K. Satish Kumar, S. Koenig, and H. Choset, "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019.