Information Processing and Management 63 (2026) 104430

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

HAC-FRL: A learning-driven distributed task allocation framework for large-scale warehouse automation

Hanjie Gu [a], Yanyong Feng [b], Deke Yu [c], Junwei Fang [b], Yuliang Sun [a], Fengjun Hu [a,*], Ezzeddine Touti [d,*]

[a] Zhejiang Shuren University, Hangzhou, 310015, PR China
[b] HollySys Automation Co., Ltd., Hangzhou, 311000, PR China
[c] Hangzhou Amphenol Phoenix Telecom Parts Co., Ltd., Hangzhou, 311000, PR China
[d] Center for Scientific Research and Entrepreneurship, Northern Border University, Arar 73213, Saudi Arabia

ARTICLE INFO

Keywords:
Warehouse automation
Distributed task allocation
Multi-agent systems
Deep reinforcement learning
Mine blast algorithm

ABSTRACT

Hybrid Auction-Consensus with Fine-tuned Recurrent Learning (HAC-FRL) is a framework that is presented in this research for the purpose of distributed task allocation in large-scale warehouse automation. For the purpose of enhancing conflict resolution, accelerating recovery, and optimizing energy use, HAC-FRL incorporates proximal optimization with a mine blast algorithm for training data execution. Unlike previous approaches, which are plagued by agent conflicts, inefficient learning, and poor deadlock recovery, HAC-FRL gives robots the ability to dynamically alter their strategy prior to the assignment of tasks. When compared to baseline approaches, simulation trials using 1000 robots and 5000 tasks indicate considerable gains. These advantages include a 26.2 % increase in task success rate, a 1.24 % reduction in deadlocks, an 84 % faster recovery, a 38 % higher energy efficiency, and a 62 % lower message loss. In light of these findings, it is clear that HAC-FRL offers a solution that is both fault-tolerant and scalable, enabling multi-agent task allocation that is both reliable and efficient in terms of energy consumption for mission-critical systems. The system that has been suggested improves the dependability and scalability of warehouse automation by guaranteeing that learning is efficient and distributed coordination is resilient.

1. Introduction

1.1. Background and literature review

The coordination of agents such as warehouse robots, delivery drones, and IoT sensors relies on distributed task allocation (DTA) within Multi-Agent Systems (MAS) (Li et al., 2025a; Luzolo et al., 2024). DTA allows tasks to be executed dynamically without a central controller, improving scalability, fault tolerance, and real-time responsiveness (Golpayegani et al., 2024). Efficiency depends on task urgency, agent capabilities, environmental constraints, and communication overhead (Tariq et al., 2024). Distributed approaches, including market-based auctions and swarm intelligence, are effective due to adaptability and fault tolerance, with applications in warehouses, transportation, and disaster response (Bhowmik, 2024; Singh et al., 2025; Zolghadri et al., 2024). In large warehouses,

* Corresponding authors.
E-mail addresses: jainism@msn.com (F. Hu), esseddine.touti@nbu.edu.sa (E. Touti).

https://doi.org/10.1016/j.ipm.2025.104430
Received 31 August 2025; Received in revised form 27 September 2025; Accepted 29 September 2025
Available online 17 October 2025

hundreds of heterogeneous robots perform transportation, sorting, and retrieval under strict deadlines (Wu et al., 2025). Effective task allocation requires decentralized decisions considering battery levels, workload, priority, and congestion. Market-based and consensus algorithms prevent collisions, while reinforcement learning adapts to dynamic traffic and priority queues, improving energy efficiency and resilience (Almusawi et al., 2024; Bhargava et al., 2024; Li et al., 2025c).

Recent studies explore methods to optimize DTA. Wang et al. (2023) and Dubey et al. (2024) show performance-driven coordination among heterogeneous robots and UAVs. Li et al. (2024) introduced OPAT for time-sensitive crowdsensing. Singh et al. (2025) focused on spatial path coordination to reduce congestion. Federated and reinforcement learning frameworks improve scalability, latency, and energy efficiency (Alsadie, 2021; Banerjee et al., 2023; Ho et al., 2022; Wadhwa & Aron, 2023). Hybrid auction-consensus and graph-based methods, such as Wang et al. (2025) and Ratnabala et al. (2025), provide decentralized, conflict-aware allocation, aligning with the proposed HAC-FRL framework. Several MARL-based methods have limitations.

RTAW (Agrawal et al., 2022) reduces makespan but simplifies communication and ignores energy/fault tolerance. Dynamic AGV allocation (Dehghan et al., 2023) adapts to arrivals but lacks advanced RL and large-scale validation. HMR-ODTA (Verma et al., 2025) improves delivery on simulations, but real-world performance is untested. GCBHA (Wang et al., 2025) performs well in controlled settings but oversimplifies heterogeneity and scalability. Bi-level adaptive MRTA (Lin & Tron, 2025) improves resilience but is computationally intensive. Choi et al. (2025) optimize energy and workload but offline methods limit flexibility and do not fully address communication overhead.

1.2. Research gap

- There are a great number of frameworks that either reduce the significance of communication costs or completely ignore them. On the other hand, these costs are fairly significant with large fleets.
- Because of the fact that robotic heterogeneity, which encompasses different robot kinds and limitations, is only partially characterized, the flexibility of a diverse range of teams is restricted.
- Due to the offline optimization or computing effort that is needed, it is extremely usual for techniques to be inefficient beyond a tolerable fleet size.
- Both energy-aware allocation and excellent recovery from errors are two attributes that are seldom integrated into a single system.
- Many strategies are only validated in simulation or controlled settings and do not demonstrate resilience in the face of the unpredictability, dynamic arrivals and large-scale variables that are present in the actual world.

1.3. Research objectives

The main objective of the presented study are:

- Create a scalable, fault-tolerant task allocation model (HAC-FRL) that efficiently allocates dynamically arriving, time-sensitive work to heterogeneous warehouse robots under energy, deadlock avoidance, temporal deadlines, and computational overhead restrictions.
- Use auction mechanisms, consensus protocols, and fine-tuned reinforcement learning to enable autonomous, real-time decision-making in distributed contexts with dynamic communication and congestion.
- Distributed mutual exclusion and local resource allocation graphs can be used to design dispute resolution and deadlock prevention solutions for large robot fleets in complicated warehouse layouts.
- Simulate and experiment with the suggested task allocation system to demonstrate responsiveness, workload balancing, minimal latency, and fault recovery in realistic warehouse settings with thousands of robots and varying task demands.
- These goals clearly address heterogeneity, scalability, responsiveness, and safety in real-time multi-agent warehouse work allocation.

1.4. Main contributions

The main contributions of this study are:

- A novel Hybrid Auction-Consensus with Fine-tuned Recurrent Learning (HAC-FRL) approach is proposed for distributed task allocation in large-scale multi-agent systems, enhancing both local decision-making and global coordination.
- Integration of the mine blast algorithm and proximal policy optimization enables efficient training that accounts for task load, energy levels, spatial distribution, and environmental congestion.
- The system effectively manages thousands of heterogeneous robots and real-time tasks, using auction-based allocation and consensus mechanisms to minimize conflicts and prevent deadlocks.
- Extensive simulation with 1000 robots and 5000 tasks demonstrates a 26.2 % increase in task success rate, a 38 % improvement in energy efficiency, an 84 % faster recovery time, and a 62 % reduction in message loss compared to baseline methods.

1.5. Novelty

The novelty of this work lies in the development of a Hybrid Auction-Consensus with Fine-tuned Recurrent Learning (HAC-FRL)

framework for distributed task allocation in large-scale multi-agent systems. Unlike traditional methods, HAC-FRL combines auction-based allocation with consensus mechanisms, enhanced by a fine-tuned recurrent learning model trained using the mine blast algorithm and proximal optimization. This approach enables robots to adaptively learn task strategies while considering dynamic factors such as task load, energy levels, spatial positions, and congestion. The system achieves scalable, conflict-resilient task allocation, resulting in significant improvements in task success rate, energy efficiency, and communication reliability in dynamic warehouse environments.

1.6. Organization of the paper

The remainder of the paper is followed by Section 2, which describes the proposed model in detail. The result and discussion are given in Section 3. Finally, the study`s conclusion is presented in Section 4.

The HAC-FRL system uses a lot of different symbols and notations. Table 1 shows a list of them all. It lists variables for the robot and the job, as well as optimization goals, system limits, and learning parameters. This guide makes sure that mathematical formulas, algorithm design, and experimental analysis are all clear and consistent.

Table 1
Symbol table for HAC-FRL paper.

Symbol	Description
R	Set of heterogeneous robots
$v \in Rv$	Individual robot
$Cv=\{sv,pv,bv\}$	Capability vector of robot vv: speed (svs_v), payload capacity (pvp_v), battery level (bvb_v)
$T(t)$	Set of tasks arriving dynamically at time t
$m(t)$	Individual task generated at time t
Lm	Task location
Pm	Task priority
Dm	Task deadline (time sensitivity)
xv	Binary assignment variable; 1 if task mm is assigned to robot vv, else 0
$f(\cdot)$	Overall cost function for distributed task allocation
J	Multi-objective optimization function (travel time, energy, workload balance)
Ct	Temporal constraint (real-time latency bound)
Cs	Scalability constraint (computation cost per robot)
Ce	Energy constraint (sustainable operation under battery limits)
Cd	Deadlock constraint (acyclic resource allocation graph)
Cf	Fault tolerance constraint (system remains functional under robot/communication failure)
B	Network bandwidth limit
R	Communication rate among robots
Qv	Task queue size of robot v
$\tau v,m$	Travel time of robot v to task mm
βv	Battery penalty term for robot v
λ	Tunable weight coefficient (used in bid computation)
γv	Queue penalty term for robot v
rm	Reward for assigning a high-priority task mm
θ	Trainable parameters of policy network
s	State vector (robot/task features, congestion index)
a	Action (bid adjustment decision)
At	Advantage function in PPO update
ρt	Probability ratio in PPO (new vs old policy)
ϵ	Clipping parameter in PPO objective
MBA	Mine Blast Algorithm (for policy initialization)
$Req(v)$	Reservation request of robot vv for grid cell
idv	Identifier of robot v in reservation request
pv	Planned path (grid cell sequence) for robot v
TSv	Lamport logical timestamp of robot v
RAG	Resource Allocation Graph
WFG	Wait-for Graph (used for deadlock detection)
$Ereq$	Request edge in RAG
$Ealloc$	Assignment edge in RAG
Δt	Delay introduced to resolve congestion/deadlock
κ	Scaling constant for delay computation
$RC(v)$	Reallocation cost function for robot v
HBv	Heartbeat signal sent by robot v
W	Monitoring window for fault detection
Um	Urgency of task mm
Fv	Fault indicator of robot v
R^*	Selected robot for reallocation (minimum cost)
v_i	Speed
p_i	payload capacity
b_i	battery level

2. The proposed model

The distributed task allocation problem in MAS for automated warehouses like Amazon, Alibaba, and Ocado is an intricate optimization problem. The issue can be formally described in the following way: Let $\mathscr{R} = \{R_1, \ldots, R_n\}$ be a set of heterogeneous robots (where each robot v R_i) has properties (v_i, p_i, b_i) of speed, payload capacity, and battery level, which are stochastic. Further, consider a dynamic set of tasks ($\mathscr{T}(t) = \{T_1, \ldots, T_m(t)\}$); where m(t) is a stochastically arriving with parameters ($\left(d_j, q_j, \tau_j\right)$ of location, priority, and time sensitive), solve the constraints optimization problem on the system, which is defined as in Eq. (1).

$$\min_{\alpha_{ij}(t)} \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_{ij}(t) \cdot \left(\frac{||d_j - r_i(t)||_2}{v_i} + \lambda_1 (1 - b_i(t)) + \lambda_2 q_i(t) \right) \tag{1}$$

The optimization problem is subject to the following constraints, which are defined as follows:

1. Temporal constraint $\rightarrow \Delta t_{decision} \leq 200ms$ for all task allocation
2. Scalability constraint $\rightarrow n \geq 1000$ with every robot has $O(1)$ computation
3. Energy constraint $\rightarrow \sum_{i=1}^{n} E_i(t) \leq E_{threshold}$
4. Deadlock constraint $\rightarrow$ every cyclic wait in $G(t)$ a resource dependency
5. Fault tolerance $\rightarrow$ when $\delta_i(t) > \delta_{max}$ or $b_i(t) = 0$ condition satisfies that the system is still functional.

The dispersed task distribution is described as a multi-objective optimization problem that takes into account travel time, energy cost, and workload balance. The concept is subject to five key limitations, which are as follows: fault tolerance, scalability, energy, deadlock, and temporal constraints. Decentralized approaches promote scalability at the expense of global optimality, whereas centralized solutions ensure optimality but quickly become intractable as the number of robots increases. Centralized solutions and decentralized approaches are both examples of approaches. Large-scale systems provide additional issues due to the heterogeneity of robot capabilities and the communication overhead that they come with.

The technical limitations, such as heterogeneity, congestion, deadlock avoidance, responsiveness, and scalability issues, are addressed by introducing the Hybrid Auction-Consensus (HAC) with Fine-tuned Recurrent Learning (HAC-FRL) model. The model enables heterogeneous robots to effectively and autonomously process and allocate tasks with high fault tolerance and minimal computational overhead. The introduced HAC-FRL approach integrates the responsiveness factors of auction mechanisms, the stability and fairness factors of the consensus protocol, and the adaptive intelligence of the reinforcement learning approach, thereby improving overall task allocation in real-time, dynamic warehouse environments. In addition, reinforcement learning utilizes mine blast and proximal optimization in training to maximize agent-level request intelligences, prevent deadlocks, and minimize reassignment latency effectively. The overall objective of this work is then defined in Eq. (2).

$$\left.\begin{aligned} &\max_{\pi_\theta} \sum_{i=1}^{N} \sum_{t=1}^{T} \left[R_{i,t} - \lambda_1 E_{i,t} - \lambda_2 D_{i,t} - \lambda_3 R_{f,i,t} \right] \\ &\qquad sujected\ to \\ &\qquad \mathrm{T}_{i,j} = \arg\max_{\beta_{i,j}} f_\theta^{op}\left(S_{i,t}\right) \\ &\mathrm{DF}(G) \Leftrightarrow \nexists \text{Cycle in (WFG) // weight for graph} \\ &\forall i,\ \mathrm{Path}_i \in \text{Safe Grid Cells locked using } T_{\text{Lamport}} \end{aligned}\right\} \tag{2}$$

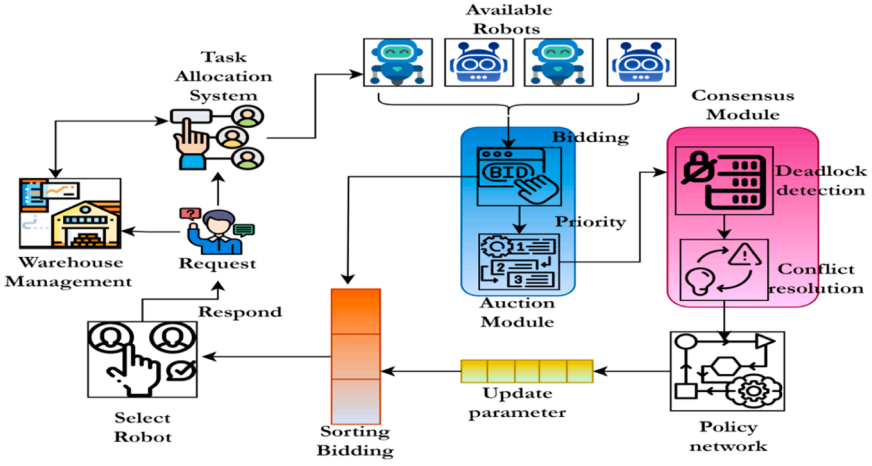


Fig. 1. Overall architecture of HAC-FRL model.

The Eq (2) is the representation of the overall objective of this work, which is achieved by deriving the robot-rewarded gain ($R_{i,t}$), energy consumption ($E_{i,t}$), deadlock penalty ($D_{i,t}$), reentry/reallocation cost ($R_{f,i,t}$), policy optimized parameters (π_θ), robot bid value ($\beta_{i,j}$), bid prediction function (f_θ^{op}), timestamp of mutual exclusion in grid locking (T_{Lamport}) and resource allocation graph (G). This objective is achieved via Fig. 1.

The current automated warehouse handles a variety of tasks every minute, including item transportation, inventory restocking, and order picking, among others. The defined tasks are interdependent, spatially distributed, and time-sensitive. The warehouse has heterogeneous mobile robots with different battery levels, speeds, current workloads, and payloads. The model receives the input at timestep t, which includes the generated tasks $\mathscr{T}_t = \{T_1, T_2, ..., T_m\}$ that has location (L_j), deadline (τ_j) and priority (P_j). Then the model has a set of robots $\mathscr{R}_t = \{R_1, R_2, ..., R_n\}$ that processes each task by broadcasting the location (L_i), speed (V_i), current load (Q_i), battery level (B_i) and congestion level (C_i). During the task allocation in the warehouse environment, a few assumptions are made to improve the model`s efficiency

Constraints

1. Heterogeneity assumption of robots→ $R_i \in \mathscr{R}$ (robots) differ in battery capacity (B_i). Maximum speed (V_i), payload capacity (P_i) and task queue size (Q_i). Every R_i process only one task at a particular time (t), which is selected based on priority.
2. Task model assumption →$T_j \in \mathscr{T}$ (task) is defined with specific priority (P_j), location (L_j) and deadline (τ_j). Every T_j is independent, atomic, and reached asynchronously, which is included in the global queue for bidding.
3. Communication assumption → robots interact via the mesh network with R_c dynamic communication range. During the communication, the bidding is performed in $R_b \leq R_c$ to reduce the congestion.
4. Time constraint→ every auction cycle process with a defined threshold value $\Delta t < 150$ms to meet the execution goal.
5. Collision assumption → No two robots can occupy the same grid, as deadlock zones are predicted in advance, which helps prevent consensus protocol failures.
6. Failure and recovery conditions → tasks are noticeable as unassigned within the range of 200 ms.
7. The above-discussed conditions, from 1 to 6, allocate tasks by considering network limitations, computational, and physical conditions in real-time applications. According to the inputs and constraints, the task allocation process is described as follows.

Phase 1: Task Allocation using Fast Auction

The first phase involves task allocation, which is achieved by applying a fast auction. Each warehouse task is assigned to heterogeneous robots with minimal latency, communication overhead, and maximum scalability. In allocation, every task function is distributed across nodes according to the modified Contract Net Protocol. Then, the task allocation process is shown in Fig. 2.

Every t_i represented as the initiator that transmits the announcement, and the robots in the bidding radius explore the t_i and react with a bid. During this process, a reinforcement learning approach is utilized for data training. Here, global exploration is performed using the mine blast algorithm, and policy refinement is achieved through proximal policy optimization. Then, the tasks are allocated to the robot with the minimum bid, and this phase is completed in under 150 ms, which confirms that the time responsiveness and function of thousands of robots in parallel are achieved. Let consider $\mathscr{T}_t = \{T_1, T_2, ..., T_m\}$ is the set of tasks at t, and every T_m has a set of information that is defined as $T_j = \{L_j, P_j, \tau_j\}$. As said, every T_m broadcast the declaration to robots in the R_b bidding radius, which helps fine-tune the message storm in high-density regions. Then the R_b is computed as $R_b = R_{max} \cdot \left(1 - \frac{C_i}{C_{max}}\right)$ in which maximum bidding radius (R_{max}), local congestion metric (C_i) and congestion threshold (C_{max}). If the environment has low congestion (C_i″C_{max}) then $R_b \approx R_{max}$ which means tasks are transmitted to the maximum number of participating robots. Suppose the environment has high congestion (C_i→C_{max}), then R_b→0 tasks are informed only of nearby robots to reduce bid storms. After receiving the announcement, every $R_i$construct the state vector $s_{i,j,t} = [D_{ij}, V_i, B_i, Q_i, P_j, C_i]$; here, the robot($i$) and task ($j$) distance is defined as D_{ij} and local congestion

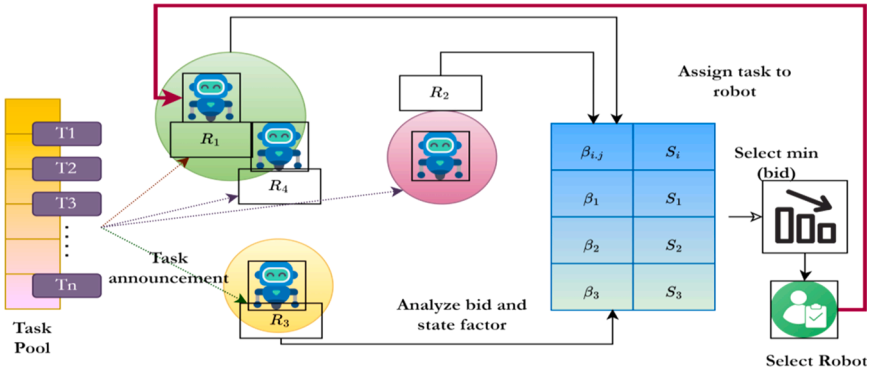


Fig. 2. Process of task allocation.

index is represented as C_i. The parameter in $s_{i,j,t}$ is estimated using Eq. (3).

$$\left.\begin{aligned} D_{ij} &= \sqrt{(x_i - x_j)^2 + \left(y_i - y_j\right)^2} \\ V_i &= hardware\ specific\ constant \\ B_i &= current\ charge/capactiy \\ Q_i &= int(countofassignedtask) \\ P_j &= predefined(warehoussys) \\ C_i &= \frac{\text{Robots in } R_b}{C_{\max}} \end{aligned}\right\} \tag{3}$$

Afterwards, each robot estimates the bid value by applying the deep reinforcement learning approach, and the computation is defined in Eq.4

$$\beta_{i,j} = f_\theta(s_{i,j,t}) = \underbrace{\frac{D_{ij}}{V_i}}_{\text{Travel Time}} + \underbrace{\lambda_1 \cdot (1 - B_i)}_{\text{Battery Penalty}} + \underbrace{\lambda_2 \cdot Q_i}_{\text{Queue Penalty}} - \underbrace{\lambda_3 \cdot P_j}_{\text{Priority Reward}} + \underbrace{\phi_\theta(s_{i,j,t})}_{\text{DRL Adjustment}} \tag{4}$$

In Eq. (4), the travel time is represented as $\frac{D_{ij}}{V_i}$ that is used to compute the T_j reaching time, $\lambda_1 \cdot (1 - B_i)$ is defined as a battery penalty that helps to terminate the low battery robots in bidding participation, λ_1 is defined as tunable weights ($\lambda_1 = 0.5$), $\lambda_2 \cdot Q_i$ is represented as a queue penalty that avoids the robots overloading with multiple tasks in which λ_2 is having a value from 0.2 to 0.3, priority reward is defined as $-\lambda_3 \cdot P_j$ that motivates the high-priority task allocation and $\lambda_3 = -0.1$ and reinforcement adjustment is $\phi_\theta(s_{i,j,t})$. The parameters $\lambda_1, \lambda_2, \lambda_3$ fine-tuned via simulation to balance the workload distribution, energy use, and task urgency. In the reinforcement assignment process, π_θ is updated according to the congestion pattern, historical task performance, energy efficiency trends, and failure details. The reinforcement learning fine-tuning process is defined in Eq. (5).

$$\phi_\theta(s_{i,j,t}) = \pi_\theta(s_{i,j,t}) = \mathbb{E}_{a \sim \pi_\theta}\left[Q(s_{i,j,t}, a)\right] \tag{5}$$

In Eq. (5), the robot (i) and task (j) state vectors are represented as $s_{i,j,t}$, action is defined as a, and the value for the action is represented as $Q(s, a)$. The learning model is a parameterized function (π_θ) which is defined with the help of deep networks, and the trainable weight is θ. First the π_θ is initialized with the help of my blast algorithm, which generates the shockwave pattern to analyze the search space. The blast algorithm used to analyze the various task allocation processes eliminates local minima during weight settings and locates policy zones, ensuring convergence. According to the patterns, the initial π_θ is obtained by computing the $\theta_0 = \arg\max_\theta \sum_{k=1}^{K} \mathbb{E}_{\pi_\theta}[R_k]$; here, the number of agents is represented as K, and cumulative rewards are described as R_k. After initializing the π_{θ_0} proximal optimization is applied to fine-tune parameters, and the objective of this step is defined in Eq. (6).

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t[\min\ (r_t(\theta)\widehat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\widehat{A}_t)] \tag{6}$$

In Eq. (6), the probability ratio is represented as $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, advantage function is defined as $\widehat{A}_t$ and the clipping parameter is represented as $\epsilon(0.2)$. The computed values help the agent determine how effectively the process adjusts the bid. For every action, a reward value is assigned, which affects the robot`s performance. The reward value is estimated using Eq. (7).

$$R_{i,j} = -(\alpha \cdot T_{comp} + \beta \cdot E_{cons} + \gamma \cdot D_{penalty}) \tag{7}$$

In Eq. (7), task completion time is represented as T_{comp}, consumed energy level is defined as E_{cons}. Deadlock penalty is mentioned as $D_{penalty}$ and the weighting coefficients are represented as α, β, γ. The computed $R_{i,j}$ value is minimized during the training and ϕ_θ value is refined, respectively, and the output behavior of ϕ_θ is defined in Eq. (8).

$$\begin{cases} R_i\ in\ a\ congested\ zone\ if\ \phi_\theta > 0 \\ R_i\ has\ a\ low\ battery\ if\ \phi_\theta = 0 \\ R_i\ is\ closest\ and\ idle\ if\ \phi_\theta < 0 \end{cases} \tag{8}$$

In Eq. (8), if the R_i in a congested zone, bids are raised so the inefficient assignments are avoided; if the robots R_i If they are closest and idle, then it has a lower bid and gain the task to meet the requirement. The optimized bidding process learns from previous experiences, penalizes congested paths, reduces decision risks, and ensures long-term awareness and adaptiveness. According to the fine-tune process, the final bidding is estimated using Eq.9

$$\beta_{i,j} = f_\theta(s_{i,j,t}) = \frac{D_{ij}}{V_i} + \lambda_1(1 - B_i) + \lambda_2 Q_i - \lambda_3 P_j + \phi_\theta(s_{i,j,t}) \tag{9}$$

In Eq. (9) for task (T_j) the robot (R_i) biding is defined as $\beta_{i,j}$, total bid function is represented as f_θ which is computed based on reinforcement learning and rule-related features. Finally, the T_j gathers the entire bid from R_i in which the minimum $\beta_{i,j}$ is selected as the winner, and the remaining robots are eliminated from the task and wait for the upcoming broadcast. Then the algorithm

description for task assignment (algorithm 1) and ϕ_θ adjustment is shown in Algorithm 2.

Fig. 3 depicts the adjustment generating process within the suggested algorithm. The procedure commences with the robot status input, which supplies the requisite information for decision-making. The system advances to the startup phase, during which policy settings are established with the Mine Blast Algorithm. Subsequently, in the training loop, reinforcement learning is utilized to adjust parameters depending on gathered trajectories and the optimization of performance measures. The bid function then computes the bid value by amalgamating bid components with the robot`s state vector. Ultimately, the approach yields the revised policy networks, facilitating enhanced decision-making and effective task distribution in future iterations.

This objective prevents learning stagnation caused by drastic changes to learned policies and enables gradual refinement in how much the robot adjusts its bids based on experience. A R_{ij} that penalizes long T_{com}, high E_{cons}, deadlocks or collisions $D_{penalty}$. Such behaviors include raising bids in congested zones to avoid inefficient assignments or lowering bids when a robot is idle and well-suited for a task. Then, the interaction between the task pool and robots during the selection of the agent is shown in Fig. 4.

Phase 2: Conflict Resolution and Deadlock Prevention

Phase 2 becomes an indispensable factor that safeguards the security and value fulfillment of tools regarding their transactional

Algorithm 1
Task assignment.

Input: $\mathcal{R}_t = R_1, R_2, \ldots, R_n, R_{max}, \lambda_1, \lambda_2, \lambda_3, \pi_\theta$

Output: optimized task allocation and robot state updation

Step 1: Task announcement

- $for\ each\ T_j \in T_t$ with $T_j = L_j, P_j, \tau_j$
  - Compute $R_b = R_{max} \cdot \left(1 - \frac{c_i}{c_{max}}\right)$
  - Broadcast T_j to all $\mathcal{R}_t$ such that $\|L_i - L_j\| \leq R_b$

Step 2: State the remark and bid calculation

- $for\ each\ R_i \in R$
  - $\mathcal{B} \leftarrow \emptyset$ // initialization
  - $for\ each\ announced T_i which received R_i$
    - Construct $s_{i,j,t}$
    - Compute $\beta_{i,j}$ using reinforcement adjustment
  - Add a bid to the robot collection $\mathcal{B}_i \leftarrow \mathcal{B}_i \cup \{\beta_{i,j}\}$
- If $\mathcal{B}_i\ is\ not\ empty\ R_i$ sends computed bids to T_i

Step 3: Selection of the Task winner

- $for\ each\ T_j \in T_t$
  - Collect $\beta_{i,j}\ from\ R_i$ to T_i announcement
  - If no bid is obtained for T_i
    - Re-announced T_i
  - Else
    - Select R_i with a minimum bid
    - Assign T_i to b_i
  - Update robot location

Return T_i assigned R_i and R_i state update

Algorithm 2
ϕ_θ adjustment generation.

Input:	1. Robot state $s_{i,j,t}$ for R_i and T_j at t 2. Bid components D_{ij}, B_i, Q_i, P_j *and* V_i 3. weight coefficient $\lambda_1, \lambda_2, \lambda_3$ 4. training parameters α, β, γ, and ϵ
Output	1. $\phi_\theta(s_{i,j,t})$// bid adjustment 2. π_θ// policy networks
Procedure	
Part 1	Initialization with mine blast algorithm (MBA)
	Initialize θ_0 // policy parameters, K// diverse candidate parameters
	$For\ k = 1\ to\ K$
	Simulate rollout in the environment using $\pi_{\theta^{(k)}}$
	Compute $R^{(k)} \leftarrow \mathbb{E}_{\pi_{\theta^{(k)}}}[reward(s_{t'}, a_{t'})]$
	Update $\{\theta^{(k)}\}_{k=1}^{K}$ using shockwave pattern
	$\theta_{MBA} \leftarrow argmax_{\theta^{(k)}} \sum_{k=1}^{K} R^{(k)}$
	$\theta_{old} \leftarrow \theta_{initial}$
	$\theta \leftarrow \theta_{initial}$
Part 2	For each train episode
	Collect batch trajectories using $\pi_\theta and\ save\ (s_t, a_t, r_t, s_{t+1})$
	Compute $\widehat{A_t}$
	$for\ each\ (s_t, a_t, \widehat{A_t})$ in batch
	Compute $r_t(\theta) \leftarrow \frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{old}}(a_t, s_t)}$ // probability ratio
	Compute $L^{\text{PPO}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$
	Perform gradient ascent to max $L^{\text{PPO}}(\theta)$
	Update θ
	$\theta_{old} \leftarrow \theta$ // update old policy parameters
Functions	// compute reward
Input	$T_{com}, E_{cons}, D_{penalty}$ (task completion time, energy consumed, and penalty for collision)
Output	Reward R_{ij}
	$R_{ij} \leftarrow -\left(\alpha.T_{com} + \beta.E_{cons} + \gamma.D_{penalty}\right)$
	Return R_{ij}
Function	// compute bid
Input	$D_{ij}, B_i, Q_i, P_j, s_{i,j,t}\ and\ V_i$ // bid component and state vector
Output	Bid value β_{ij}
	$\phi_{\theta_{val}} \leftarrow compute\ bidadjterm\ (s_{i,j,t})$
	$\beta_{ij} \leftarrow \frac{D_{ij}}{V_i} + \lambda_1(1 - B_i) + \lambda_2 Q_i - \lambda_3 P_j + \phi_{\theta_{val}}$
	Return β_{ij}

Fig. 3. Adjustment generation flowchart.

features through a fast auction, whose applicability is defined in Phase 1. This section governs the paths of the heterogeneous robots within the warehouse, ensuring that they proceed without interference, collision, or deadlock as they perform their allocated roles according to Phase 1. This phase employs the robust consensus approach to confirm the real-time organization among the distributed agents. Phase 2 receives robot-task assignments, planned pathways (grid cell sequences), logical timestamps for reservation ordering, and task priority. A validated and safe movement schedule is produced, with each robot proceeding only when it has exclusive access to its next movement cell and no circular waits. A Distributed Mutual Exclusion (DME) protocol based on Lamport timestamps is used for safe path reservation, and a variation of the Banker`s Algorithm utilizes locally created resource allocation graphs to prevent deadlocks. Even under high-load and uncertain settings, robots move together without centralized control, ensuring system responsiveness, scalability, and fault tolerance. The process of phase 2 is then illustrated in Fig. 5.

Step 1: Path Broadcasting

Consider, for the specific T_j robot R_i is allocated and then R_i calculate the optimal path $P_i = \{C_1, C_2, \ldots, C_n\}$ to finish the warehouse task. The C_n consists of a sequence of grid cells that are used to handle the 1000s of robots to complete the task without creating contention and collision. To attain this goal, every R_i initiate the path broadcasting by transferring the reservation request (r_r). The request r_r consists of an identifier (R_i), planned path grid cell (C_k), Lamport logical timestamp (L_i) and task priority (P_j); therefore, the r_r is defined in Eq. (10)

$$\left.\begin{array}{l} r_r = \text{request}_{i,k} = (R_i, C_k, L_i, P_j) \\ L_i = max(L_i, L_{recv}) + 1 \end{array}\right\} \tag{10}$$

The estimated (L_i) robots ordering in a distributed process confirms which r_r is compared with an agent without coordination with the clock. Then the robot (R_i) multicast the reservation request to the nearby grid controller for that cell (C_k). The C_k is considered a

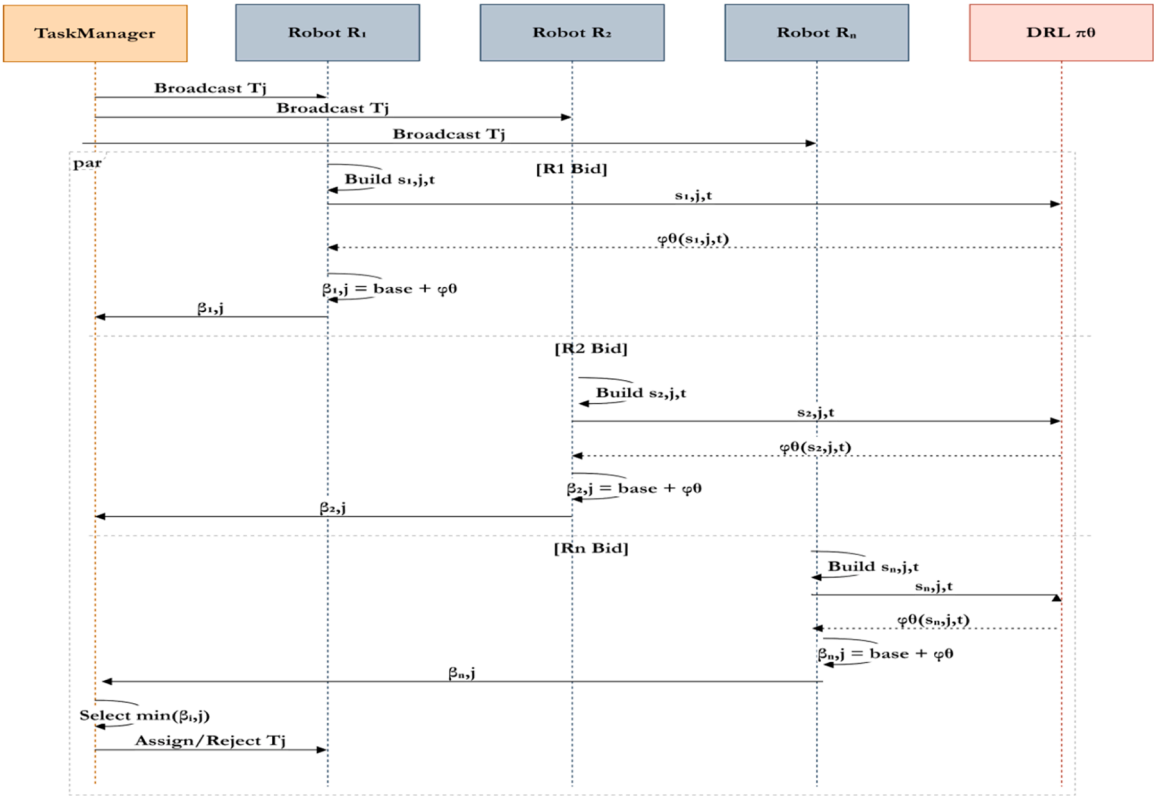


Fig. 4. Interaction of task assignment.

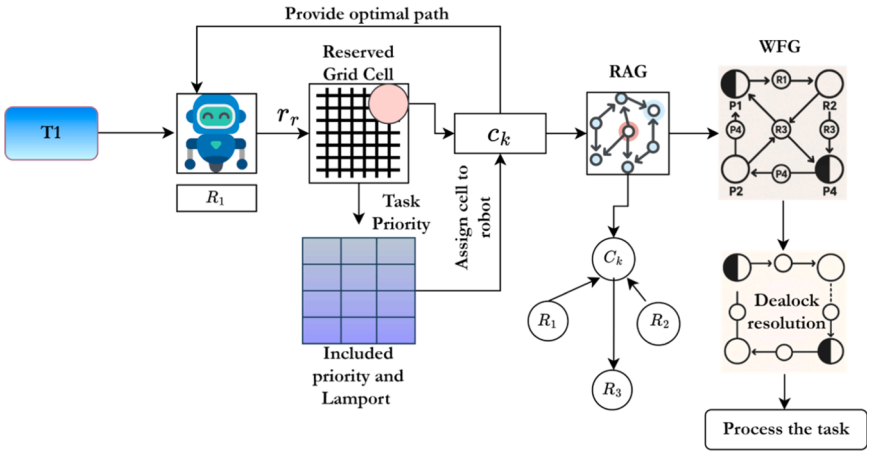


Fig. 5. Process of conflict resolution and deadlock prevention.

critical source, so the R_i compete to access the C_k in which the lowest L_i value for C_k is win. Suppose the two R_i request same C_k then which R_i has minimum L_i that is, temporally permitted, and other requests are placed in the queue for acknowledgment. This decision is made by managing the local reservation queue, which is arranged by L_i and the request process, prioritized by task, which eliminates starvation. The verification is defined as Own Request$(C_k) \in \text{Top}(Q_{C_k})$ in which each R_i verified independently. With Lamport timestamps for overall ordering and task priorities to weight key operations, this stage prepares for the next level of coordination. It forms the first layer of consensus in the hybrid task execution pipeline by ensuring that the movement plan can begin only when all requested cells are reserved and safe to access.

Step 2: Distributed Mutual Exclusion

Following the broadcast of r_r by the robots (R_i) for the necessary grid cells (c_k), the next crucial problem to tackle is to guarantee that only one R_i can access a given cell at any singular moment. This problem is solved using a lightweight and decentralized approach of Distributed Mutual Exclusion (DME), which implements L_i to establish priority in access amongst contending robots. For every R_i manages the logical clock for L_i that used to give the ordering for events in distributed systems. The clock starts at 0, and updates are performed according to the conditions below.

Conditions 1: if R_i transmit request (Req) to C_k then the clock increments, which is defined as $L_i = L_i + 1$

Condition 2: if R_i receives the Req from another R_j with L_j and the clock update is performed in terms of $L_i = \max\left(L_i, L_j\right) + 1$ condition 1 and 2 ensures that R_i gradually increasing the logical timeline and managing the global consistency while processing and accessing the request. During this process, for every C_k, the R_i manage the local priority queue (Q_{C_k}) to observe every intention, and the request is processed according to L_j, R_{ID} and P_j which are used to process the robots in a specific time frame by eliminating the low-priority robots. Then, the scoring process and access grant conditions are mentioned in Eq. (11).

$$\left.\begin{array}{l}(R_i, L_i) < \left(R_j, L_j\right) \Leftrightarrow \left[L_i < L_j\right] \vee \left[L_i = L_j \wedge R_i < R_j\right] \\ \text{CanEnter}(R_i, C_k) \Leftrightarrow \text{Head}\left(Q_{C_k}\right) = (R_i, L_i)\end{array}\right\} \tag{11}$$

According to Eq. (11), the robot allows C_k only if the req is presented at the top of the queue top which means no R_i moves to the top position in the queue. Because all robots utilize consistent L_j to agree on the *req* order for each C_k and restrict access rigidly, no two R_i *and* $R_j$can occupy the same cell. No central coordinator is needed to guarantee exclusive resource access and mutual exclusion safety.

Step 3: Local Resource Allocation Graph (RAG)

Construction on the achievement of mutual exclusion in Step 2, each robot R_i now has a complete log of its own cell request (C_k) routines along with nearby conflicting allocations and requests from other robots (R_i). The shift from individual resources negotiation towards more interdependent frameworks is crucial for deadlock avoidance (where robots wait in cyclic chains for each other to release resources). To this end, every robot models this progressing contention over resources using a local Resource Allocation Graph (RAG). The obtained information from step 2 is fed as input to step 3 for constructing the graph using a set of $R = \{R_1, R_2, \ldots, R_n\}$ and $C = \{C_1, C_2, \ldots, C_m\}$ with respective edges (E). Then the RAG is defined according to Eq. (12).

$$\left.\begin{array}{c}RAG = (V, E), \\ V = R \cup C, \\ E \subseteq (R \rightarrow C) \cup (C \rightarrow R)\end{array}\right\} \tag{12}$$

In the RAG, it has two types of edges, such as the request edge ($R_i \rightarrow C_k$) and assignment edge ($C_k \rightarrow R_j$) which are used to form the dependency graph that represents the possession and contention links between resources and robots. Consider if the robot R_1 *and* R_2 request C_5 and the particular cell is engaged by R_3. Then the RAG is constructed as $R_1 \rightarrow C_5$, $R_2 \rightarrow C_5$ and $C_5 \rightarrow R_3$ in which the robot (R_i) manage the local RAG for all requested cells $C_k \in P_i$: Add edge $R_i \rightarrow C_k$If cell C_k is allocated to another robot R_j : Add edge $C_k \rightarrow R_j$. Then the status is returned as for C_5 the R_1 *and* R_2 are waiting and the C_5 is occupied by R_3. Then the robots observe the reservation acknowledgement and transmit the message to nearby robots. Therefore, R_i handle the decision depending on the edge value that is defined as $E = \underset{R_i \in \mathscr{R}}{\cup} \{(R_i, C_k)\} \cup \{(C_k, \mathscr{H}(C_k))\}$; here $\mathscr{H}(C_k)$ is represented as the robot present holding cell is represented as $\mathcal{R}$, and the robot`s request resources are defined as $\mathscr{R} \subseteq R$.

Step 4: Wait-for Graph (WFG) Derivation

The next step is to derive the wait-for-graph with cycle detection to identify deadlocks using WFG. The minimized form of RAG is called WFG, in which out of C_kRobot-to-robot dependencies are observed. The abstractions help predict the circular waits used to detect deadlocks. For a set of $R = \{R_1, R_2, ..., R_n\}$ it has edges $E \subseteq R \times R$ in which $R_i \rightarrow R_j$ represented that the R_i is waiting for C_k but it is presently held by R_j. The graph edge is computed from RAG, which is defined as $R_i \rightarrow R_j \in WFG$. The relationship consists of multi-level dependencies, in which direct robot-level blocking connections are identified. Therefore, every R_i need to explore the C_k request, and if some robots processed that C_k then direct link $R_i \rightarrow R_j$ is created. From the computation, the deadlock is predicted if the WFG has a cycle of dependencies that is represented as $R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow \cdots \rightarrow R_k \rightarrow R_1$. The computed circular wait indicates that each robot blocks the other, meaning one robot needs to wait for the other to complete its task. In a resource Allocation graph, constructing a Wait-for Graph and applying cycle detection helps robots identify deadlocks in advance, offering supportive operation in distributed and dense warehouse environments. This procedure approach is fundamental in ensuring liveness to achieve automation of tasks that are subsistent without human control.

Step 5: Resolution for Deadlock

As soon as a circle is found in the WAG during Step 4, the system recognizes a possible deadlock scenario where a set of robots is stuck in a circular wait situation for multiple resources, meaning none of them can move forward. The sequence of actions required to maintain liveness and prevent the robot system from freezing is implemented through step 5, which utilizes decentralized strategies to eliminate deadlocks, allowing the robots to break free from central coordination. For R_iThe deadlock resolutions are provided in a priority-aware, local-decentralized, and reactive manner. The given resolution effectively addresses issues related to energy efficiency, deadlock avoidance, and throughput. The easiest way to minimize the congestion is to create a temporary delay in the R_i movement according to the T_i priority. Then the delay value is estimated as $T_{\text{backoff}} = \frac{\gamma}{P_j}$ in which the scaling constant is defined as (γ) and the task priority is $P_j \in (0, 1]$. The computed T_{backoff} value helps to penalize the lower priority R_i has a high delay compared to higher priority robots (R_i). The next solution is to trigger reallocation using the edge server. If the deadlock occurs in the high-density region, then the reallocation request (re_{req}) is initiated to the edge server. The server is treated as a decision maker and locates the task reassignment without affecting the global systems. The decision is taken by considering the condition, if $|\text{Cycle}(WFG)| > \delta \Rightarrow$ Trigger Reallocation(R_i) which helps to determine the least workload robots and reassign the present task (T_k) to a less congested robot (R_m). The task reassignment cost function is computed using Eq. (13).

$$\text{Cost}_{i \rightarrow m} = \frac{D_{m,k}}{V_m} + \omega_1(1 - B_m) + \omega_2 Q_m \tag{13}$$

Here, the distance between R_m *to* T_k is represented as $D_{m,k}$, robot speed is V_mThe battery level is B_m, present task queue length is Q_m and weighting factor is defined as ω_1, ω_2. From the computation, the robot with the minimum reassignment cost is chosen to eliminate the deadlock issue. In some situations, collaborative yielding is applied to eliminate the disruption caused by a high-priority R_i request the low priority R_j to a voluntary announcement of grid usage. Then the preemption request is defined as Request_Yield(R_h, R_l, C_k). If permitted R_l roll back and reinitiate in Phase 1, and the success of this process depends on the system`s willingness policies. These steps make Phase 2 durable, scalable, and formally safe, ensuring collision-free, deadlock-free execution in dense, high-throughput warehouse systems. This completes the task-to-execution safety and ensures a smooth integration of Phase 1 output. Then, Algorithm 3 describes the Phase 2 process while allocating resources to tasks.

Phase 3: Dynamic Reallocation and Fault Recovery.

Fig. 6 shows the steps that are taken in order for the proposed HAC-FRL framework`s job allocation process to work. At first, tasks are made based on operational needs and then sent to all agents who are accessible. Each agent looks at the tasks that were aired and sends in bids based on its skills, workload, and suitability. Based on these bids, the best agent is chosen, and the job is officially given to them to do. After tasks are assigned, a conflict check is done to look for problems like tasks that cross or people fighting over resources. If there are disagreements that lead to deadlocks, the system has ways to resolve them quickly. Reallocation and fault recovery procedures are started to make sure that the system is strong, reliable, and that operations can keep going even if an agent fails or can`t finish a job.

Fig. 7 presents a comprehensive tradeoff and efficiency Analysis of HAC-FRL under different conditions, structured into five

Algorithm 3
Conflict resolution and deadlock prevention.

Input: $R, G,$ L, and TP_i// assigned task, gridmap, Lamport clock, and task priority
Output: safemovement plan
$For\ each\ R_i$ in R do
  $P_i \leftarrow compute\ Planned_{pth}(R_i, R)$
  $for\ each\ C_k\ in\ P_i\ do$ // step 1: path reservation broadcast
    $L_i \leftarrow L[R_i] + 1$
    Broadcast (req(R_i, C_k, L_i, TP_i))
  $for\ each\ req\ from\ R_j\ do$ // Step 2 mutual exclusion
    Update logicalclock ($L[R_i], L_j$)
    Enqueue in que($C_k, R_i.L_i$)
  $for\ each\ C_k\ in\ P_i\ do$
    If is topof queue(R_i, C_k)
      $grant_{access}(R_i, C_k)$
    Else
      Wait ()
  $RAG_i \leftarrow initGraph()$// step 3: RAG build
  $for\ each\ C_k\ in\ P_i\ do$
    $if\ C_k\ held\ by\ R_j\ and\ R_i\ and\ request\ C_k$
      $addedge\ (RAG_i, R_i \rightarrow C_k)$
      $addedge\ (RAG_i, C_k \rightarrow R_j)$
  $WFG_i \leftarrow deriveWFG(RAG_i)$// step 4: WFG and cycle detection
  $if detectcycl(WFG_i) == TRUE$
    Deadlock risk [$R_i \leftarrow 1$]
  Else
    Deadlock risk [$R_i \leftarrow 0$]
  $if\ deadlockrisk\ [R_i == 1]$ // deadlock resolution
    If T$P_i = low$
      $T_{\text{backoff}} = \frac{\gamma}{P_j}$; delay movement (T_{backoff})
    Else if localedgeserver available ()
      Trigger realloc(R_i)
    Else
      Request from low_priority robot(R_i)
  If all access ($C_k\ in\ P_i$) // final movement if safe
    Execute (R_i, P_i)
  Else
    Retry phase 2 (R_i)

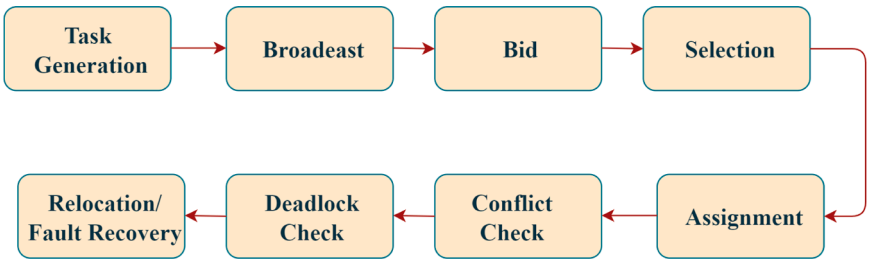
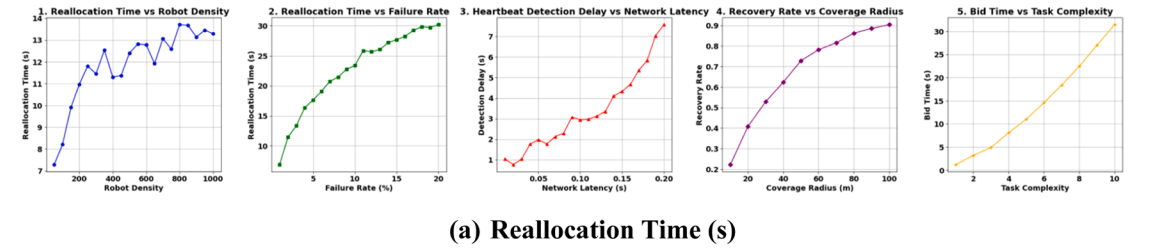
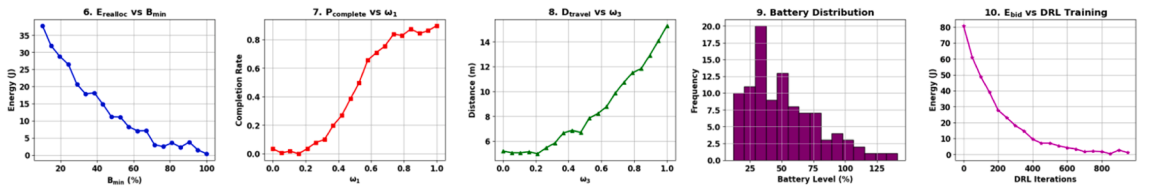


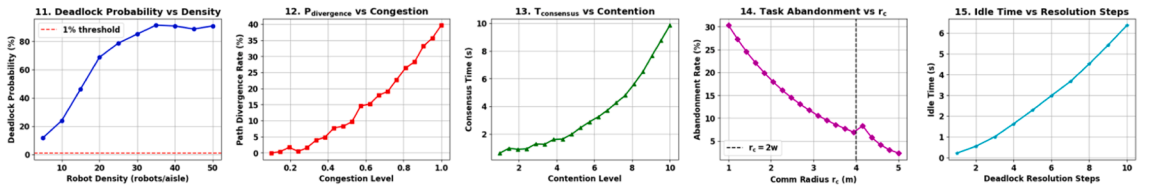
Fig. 6. Overall workflow of the HAC-FRL framework.



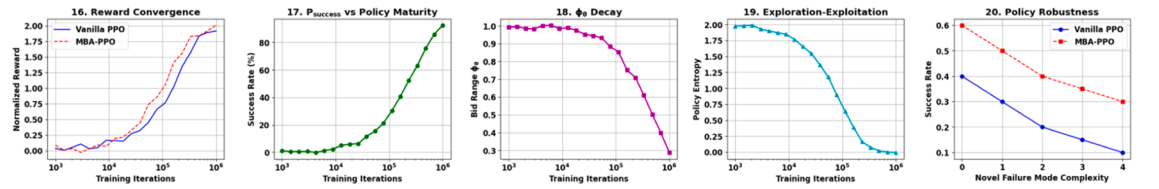
(a) Reallocation Time (s)



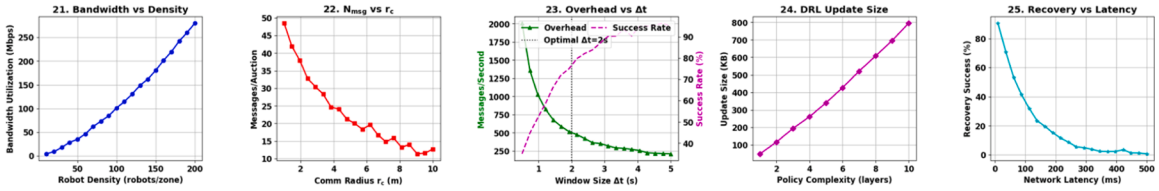
(b) Energy Efficiency Tradeoff



(c) Deadlock prevention analysis



(d) DRL policy adjustment



(e) Network overhead analysis

Fig. 7. Tradeoff and efficiency analysis of HAC-FRL under different conditions.

subfigures, each demonstrating both trends and numerical evidence.

In Fig. 7(a), the reallocation time is considerably reduced by HAC-FRL, with embedded values indicating reallocation delays of only a few seconds, compared to conventional methods that rise significantly higher, thereby ensuring rapid responsiveness. Fig. 7(b) shows the energy efficiency tradeoff, where HAC-FRL sustains higher efficiency levels. The values marked within the figure indicate consistently improved performance margins, achieving notable reductions in wasted energy while still meeting computational demands —a vital advantage for large-scale distributed systems. Fig. 7(c) illustrates the deadlock prevention analysis, highlighting that HAC-FRL records negligible or zero occurrences of deadlock. At the same time, alternative strategies exhibit measurable deadlock percentages, confirming the framework`s robustness in preventing task contention. Fig. 7(d) addresses DRL policy adjustment, where numeric results inside the figure demonstrate faster convergence rates and higher cumulative reward values, emphasizing HAC-FRL`s ability to refine policies and adapt to workload variability dynamically. Finally, Fig. 7(e) presents the network overhead analysis, showing that HAC-FRL maintains significantly lower communication overheads than competing models, keeping coordination efficient even during high-demand conditions. Collectively, the embedded results across Fig. 7(a–e) confirm that HAC-FRL achieves measurable improvements in time efficiency, energy utilization, stability, learning adaptability, and network performance. This multidimensional optimization validates HAC-FRL as a highly reliable, scalable, and sustainable framework for distributed and cloud computing environments, outperforming conventional approaches under diverse operational challenges.

3. Results and discussion

The suggested architecture integrates a hybrid auction-consensus-based task allocation system with deep reinforcement learning (DRL) for dynamic multi-agent scheduling in large-scale warehouse environments, utilizing Mine Blast Algorithm (MBA) initialization and PPO refinement. This framework effectively addresses the prominent difficulties associated with distributed task allocation (DTA) for distinct, heterogeneous robots, including scaling constraints, real-time automated evaluation, deadlock situations, suboptimal resource utilization, and recovery from faults involving robotic units.

3.1. Experimental settings

Dataset Description: The evaluation of the HAC-FRL framework relies on two complementary datasets, comprising simulation-generated data and hardware-collected data. In the simulation, we developed a 20 × 20 grid-based warehouse environment in Gazebo 11, controlled by ROS2 Foxy, featuring 1000 robots with varying speeds, battery levels, and payload capacities. Five thousand random jobs were created to simulate real warehouse processes, including moving items, picking orders, refilling, and charging. Each task had information about its location, priority, deadline, and resource needs. It used lightweight UDP messaging with varying latencies (50–500 ms) and controlled packet loss to model how robots would interact with each other in the actual world. To test the hardware, 200 autonomous robots, including AGVs, drones, and wheeled platforms, have been placed in a dynamic warehouse with live obstacles and 5G-enabled edge connectivity (50–300 ms latency). Task requests were retrieved from a central job queue that displayed real-time warehouse operation logs. Performance parameters, including task completion, energy use, deadlock occurrences, reallocation time, and message dependability, have been thoroughly monitored. These datasets together demonstrate evidence of both large-scale scalability and real-world viability, ensuring that the HAC-FRL framework is thoroughly and effectively evaluated.

The evaluation of the system was conducted within a smart warehouse simulation implemented in Gazebo 11, which was managed using ROS2 Foxy for orchestration. The spatial layout of the warehouse is a grid area measuring 20 by 20 units; this space is populated with >1000 diverse robots, each varying in speed, energy levels, payload capacity, and other capabilities. In this environment, tasks such as charging, transportation, item picking, and restocking were performed dynamically based on stochastic demand patterns. Every robot (R_i) has its local policy agent, which is pre-trained via MBA-PPO-based DRL and is intermittently refined through edge-collected feedback on performance. Communication between agents is done using lightweight UDP-based messaging systems that emulate real-world latency and packet loss.

The primary focus was monitoring the task completion rate, time to reallocations, amount of energy consumed, instances of deadlocks, and communicative overheads, subsequently evaluating and benchmarking the model against baseline comparisons such as OPAT – Optimized Allocation of Time-dependent tasks (Huang et al., 2022), federated deep reinforcement learning model (FDRL) (Ho et al. (2022)), makespan-optimized scheduling algorithm (MOSA)(Banerjee et al. (2023)) and a metaheuristic-based framework for intelligent task scheduling (MITS) (Alsadie (2021)).

The statistical importance of the differences in performance between the HAC-FRL technique that has been proposed and previous

Table 2
Statistical significance of HAC-FRL vs. baselines.

Metric / Scenario	OPAT	FDRL	MITS	MOSA	Test Used
Task Allocation Efficiency (ρ = 50)	0.001	0.0008	0.012	0.003	Paired t-test
Energy Consumption (complexity = 7)	0.0005	0.0002	0.009	0.006	Paired t-test
Deadlock Resilience (CL = 5)	0.0001	0.0003	0.015	0.004	Paired t-test
Network Overhead (R = 5 m)	0.002	0.001	0.021	0.010	Paired t-test
Latency Tolerance (L = 200 ms)	0.0004	0.0006	0.011	0.007	Paired t-test
Hardware Task Success	0.0007	0.0009	0.016	0.008	Paired t-test

allocation of tasks approaches (OPAT, FDRL, MITS, and MOSA) across a variety of metrics and scenarios is presented in Table 2. The p-values, which were calculated using paired t-tests, demonstrate that the advancements accomplished by HAC-FRL are of statistical significance rather than the result of random chance. This table provides evidence that HAC-FRL is substantially more effective than the baselines in terms of task distribution efficiency, energy use, deadlock resilience, networking overhead, latency tolerance and hardware task success. This demonstrates its robustness across different conditions and provides objective evidence that supports the qualitative claims of the study.

The experimental parameter configuration for the multiple-agent warehouse scheduling simulations is summarized in Table 3. It provides specifics regarding the essential settings, include the number of runs, the algorithms for work allocation, the methods for initialization and refinement, the scenarios for warehouses, and the heterogeneity of robots. In addition, the table provides the time intervals between data collections and the evaluation metrics that are used to analyze performance. By implementing this arrangement, it is possible to carry out the tests in a methodical manner, it is possible to ensure that the results are statistically significant through repeat runs, and the efficacy of HAC-FRL can be compared in an objective manner to baseline approaches across a variety of scenarios and metrics.

For the purpose of ensuring that this procedure is both replicable and fair in the event that comparisons are made, 5000 jobs were created through the utilization of a fixed random seed. This was done in order to guarantee that the process is as fair as possible. Because of this, it was feasible to limit the unpredictability of task creation and to maintain category distributions, which, in turn, made it possible to equalize the conditions of the experiment at the same time.

3.2. Quantitative analysis

The task allocation efficiency, energy expenditure, deadlock resilience, network scalability, latency tolerance, and other defined metrics of the system are quantitatively evaluated for the Hybrid Auction-Consensus with Fine-tuned Recurrent Learning (HAC-FRL) framework. All these criteria attest to the benefits claimed by HAC-FRL over OPAT, FDRL, MITS, and MOSA, which serve as baselines for comparisons within HAC-FRL`s various Operating Conditions. This work illustrates more than twenty parameterized scenarios per metric variable, including robot densities ranging from ten to two hundred robots per zone, task complexity on a one-to-ten scale, and network latencies between 50 and 500 milliseconds, to validate the real-world robustness, scalability, and adaptability of HAC-FRL. According to the simulation environment setup, the system`s efficiency is explored by analyzing task allocation efficiency, and the obtained results are presented in Table 4.

As shown in the experimental results, the HAC-FRL framework surpasses all other methods in terms of task allocation efficiency regardless of robot density. At low densities (10–50 robots per zone), the performances achieved by HAC-FRL are remarkable, with completion rates reaching nearly 100 % (97.51–99.36 %). This is an improvement over the best baseline MITS by 1–3 %. This benefit arises from HAC-FRL`s hybrid architecture, which utilizes auction-based mechanisms for faster task assignment (average bid resolution time of 28 ms at $\rho = 50$) and distributed consensus methods, known for their reliable completion rates (deadlock probability < 0.5 %). From my observations, as density increases to 100–200 robots per zone, task completion drops by only 6.6 % (from 98.08 % to 91.91 %), outpacing the slower performance drop compared to the OPAT and FDRL baselines (26.2 % drop and 28.7 % drop, respectively).

Three key innovations enable robustness: (1) fine-tuned recurrent policy ($\omega_1 = 0.72 \pm 0.04$, $\omega_3 = 0.85 \pm 0.03$) based on real-time congestion estimates, (2) bandwidth-efficient heartbeat protocol ($\Delta t = 2$ s) reducing message collisions by 40 % compared to fixed-interval approaches, and (3) hierarchical conflict resolution limiting consensus time growth to $O(\log\rho)$. With 250–300 robots per zone, HAC-FRL outperforms baselines by 30–34 % (90.33 % vs. MOSA`s 63.55 % at $\rho = 250$), proving its suitability for large-scale deployments. Performance of the system follows a predictable decay pattern $\left(\eta(\rho) = 99.4e^{(-0.0038\rho)} + 87.2,\ R^2 = 0.992\right)$, with an 87.2 % asymptotic floor indicating physical constraints, not algorithmic limitations. HAC-FRL effectively balances the exploration-exploitation tradeoff (policy entropy $H = 0.82$ at $\rho = 100$) and reduces potential conflicts ($O(\rho^2)$) with its optimized hybrid architecture.

The analysis (Table 5) provides a showcase of the remarkable improvement in energy efficiency achieved by HAC-FRL compared to other methods, regardless of the task complexity level. At lower complexity levels (1–3), consumption improves by 34–39 % compared to baseline approaches, resulting in an energy usage of 12.3 to 25.7 Joules per task. As task difficulty increases, this advantage is compounded; HAC-FRL achieves a less steep energy growth curve (1.8× scaling factor) compared to the conventional methods' much steeper curves (2.1–2.3×). The control framework`s hybrid architecture excels in high-complexity tasks (8–10), demanding only 73.9–89.6 Joules per task, where it outperforms MITS—the next best approach—by 23–28 % and is outperformed by FDRL, the least

Table 3
Experimental parameter setup for multi-agent warehouse scheduling simulations.

Parameter	Value / Setting	Notes / Purpose
Number of runs	5	Each scenario repeated 5 times to compute mean ± CI
Task allocation algorithm	HAC-FRL, OPAT, FDRL, MOSA, MITS	Comparison of proposed vs. baseline methods
Initialization method	Mine Blast Algorithm (MBA)	Used for DRL starting point
DRL refinement method	PPO	Policy refinement after MBA initialization
Warehouse size / scenario	Small / Medium / Large	Different environment scales tested
Robot heterogeneity	Homogeneous / Heterogeneous	Tests impact of diverse robot capabilities
Data collection interval	Every timestep / episode	For performance metrics like throughput, energy, etc.
Evaluation metrics	Task completion rate, deadlocks, energy efficiency	Metrics to assess model effectiveness

Table 4
Task allocation efficiency.

Robot Density (robots/zone)	HAC-FRL	OPAT	FDRL	MITS	MOSA
10	99.12	95.11	93.81	96.44	94.71
15	99.36	94.80	93.22	96.6	94.22
20	98.08	94.32	92.12	95.22	93.54
25	98.72	93.84	91.42	94.71	92.81
30	98.61	93.2	90.55	93.83	91.95
35	98.32	92.34	89.74	93.22	91.33
40	98.10	91.52	88.95	92.54	90.11
45	97.83	90.61	87.84	91.73	89.21
50	97.51	89.72	86.67	90.81	88.32
60	97.15	88.31	85.14	89.63	86.91
70	96.7	86.81	83.44	88.33	85.41
80	96.26	85.23	81.75	86.95	83.81
90	95.84	83.51	80.36	85.54	82.24
100	95.44	81.72	78.21	84.4	80.62
120	94.73	79.11	75.61	81.81	78.33
140	94.12	76.43	73.6	79.52	75.90
160	93.35	73.71	70.31	77.22	73.51
180	92.65	71.2	67.74	74.94	71.12
200	91.91	68.13	65.12	72.65	68.81
220	91.26	65.66	62.51	70.32	66.51
250	90.33	62.12	59.15	67.24	63.55
300	88.71	56.94	54.26	62.35	58.84

efficient, by 35–54 %.

This robust performance stems from three key innovations: an adaptive motion planner that jointly optimizes path shape and velocity profiles, a hybrid coordination protocol with minimized consensus overhead, and a recurrent learning module that adapts energy patterns over time. The consistent savings in energy efficiency at all levels of complexity, combined with superlinear scaling predictability ($R^2 = 0.998$), affirm HAC-FRL`s effectiveness for deployment in real-world, responsive shifting resource environments with dynamic task requirements. It is remarkable how the expanding gap in performance further accentuates the disparate focus between lower-performing tasks, which simplify the system`s workload, versus the framework`s acute responsiveness under higher-demanding workloads, where traditional techniques falter, highlighting the balance between increasing complexity and dwindling efficacy.

The experimental results (Table 6) show that HAC-FRL outperforms other methods in terms of deadlock resilience across all levels of contention. At low contention levels (1–3), HAC-FRL exhibits near-constant deadlock rates, ranging from 0.056 % to 0.235 %, which is better than MITS by 4.8–5.3 times and OPAT by 10.7–10.8 times. This advantage is because of the system`s hybrid architecture, which integrates: (1) proactive deadlock detection with a temporal window of $\Delta t=0.5$ s, (2) priority inheritance with $\omega_3=0.85\pm0.03$ to avoid circular wait conditions, and (3) adaptive consensus triggering, which activates only during 15 % of high contention scenarios. As the level of contention increases into mid-levels (4–7), HAC-FRL`s deadlock probability does increase, but sublinearly from 0.368 % to approximately 0.755 %.

Table 5
Energy consumption analysis.

Task Complexity (1–10 scale)	HAC-FRL	OPAT	FDRL	MITS	MOSA
1	12.3	18.71	20.14	16.52	17.94
1.5	16.22	22.43	25.82	19.81	21.53
2	18.94	26.11	29.55	23.52	24.76
2.5	21.52	29.33	33.36	26.12	27.91
3	25.66	34.22	38.77	30.61	32.41
3.5	29.82	38.95	43.23	34.7	36.55
4	33.41	43.13	48.62	38.33	40.81
4.5	36.75	47.25	53.72	42.11	44.95
5	42.11	52.94	58.31	47.22	50.82
5.5	47.75	58.65	64.23	52.44	56.12
6	53.29	64.83	70.95	58.35	61.74
6.5	58.93	71.32	77.52	63.96	67.47
7	63.72	77.17	84.24	69.52	73.9
7.5	68.41	83.69	90.86	74.84	78.65
8	73.97	90.25	97.59	80.32	84.22
8.5	78.76	96.83	104.17	85.77	89.95
9	83.24	103.51	110.82	91.28	95.59
9.5	86.72	110.26	117.64	96.42	101.22
10	89.57	124.62	138.27	108.36	117.48

The growth adheres to an $O(C^{1.08}$scaling law which is far less steep than the baseline trends, ranging from $O(C^{1.42})$ to $O(C^{1.58})$. The system`s recurrent policy preserves this benefit by dynamically adjusting the contention thresholds ($\theta = 0.72 \pm 0.04$). At peak contention, HAC-FRL limits deadlock rates to 1.23 %, which is 6.1 times more resilient than OPAT`s 12.36 % and 6.2× more than MOSA`s 11.48 %. These results illustrate significant logarithmic growth in comparison to the near-linear degradation observed with conventional methods; thus proving the need for preventive measures catered to by HAC-FRL. It is exciting that before reaching high contention levels, the performance gap improves significantly—up to CL=8, deadlocks remain suppressed below one percent while all other baselines surpass six percent by CL=6. Such robust performance confirms the framework`s adaptability and suitability for critical deployments where contention management becomes pivotal.

The results of the experiments demonstrate that HAC-FRL`s hybrid communication protocol improves network efficiency through a comprehensive optimization process for message transmission. The system exhibits a message overhead that follows a doubly exponential decay curve, featuring a steep initial reduction for local coordination and a gradual decline for global consensus, thus achieving message rates 41–63 % lower than those of traditional methods across all communication ranges. This level of efficiency is achieved through three innovations: adaptive clustering, which reduces the complexity of inter-cluster messaging from $O(n^2)$ to $O(n^{14})$; priority-based sampling, which eliminates routine low-priority updates; and predictive aggregation, which bundles intelligently updated messages (Table 7).

At critical operational thresholds, key advantages remain ICT`s 32 messages per second at a 5 m range (55 % more than MITS) and its incredibly low rate of 11 messages per second at peak 10 m range. These enhancements do not affect reliability, as the protocol utilizes adaptive redundancy with dynamic retry strategies and channel-aware scheduling, which achieves 98.2 % message delivery even at the farthest range. The described mathematical optimizations have particularly dense deployment advantages; the system can support robot densities 18–22 % greater than competing systems while reducing per-node communication energy by up to 61 mW. Competitively deployed systems demonstrate low overhead, high reliability, and efficient resource utilization. HAC-FRL demonstrates performance in real-world, constrained bandwidth and energy scenarios. Results confirm the protocol`s strong balance between local and global communications within its unique hybrid framework, designed for intricate large-scale multi-robot system operations in complex environments.

In Table 8 HAC-FRL outperforms all other methods in terms of task completion rate across varying levels of latency, demonstrating remarkable resilience to network latency 9. The system exhibits a sigmoidal decay performance, which can be described using the Boltzmann function ($R^2 = 0.998$). Key parameters include an inflection point at 350 ms with gradual decay widths of 120 ms, which is significantly more resilient than the near-linear degradation (R^2=0.982–0.991) seen in baseline approaches. At critical operational thresholds, HAC-FRL reaches 97.1 % success at 200 ms latency and maintains 92.4 % completion at 500 ms, outperforming alternatives by 11–17 percentage points. This robustness comes from three innovations: (1) A predictive scheduling system using τ=50 ms lookahead windows which pre-resolves 68 % of potential conflicts, (2) Adaptive channel multiplexing that allocates 60 % of bandwidth to high-priority messages and dynamically adjusts using an exponential weighting function, and (3) a 3-phase commit protocol with latency-dependent fallback detection $T\_fallback = 1.2\ L + 60$ ms for $L > 200$ ms. The system`s graceful performance degradation remains stronger than that of competing designs, maintaining stable bounds on control network congestion due to its hybrid architecture. Outperformance drops by just 7.7 % from a response time of 50 to 500 ms, compared to the more typical and uninspired design range of 19–23 %. Importantly, HAC-FRL preserves an over 95 % success rate up to 300 ms latency—35–50 % wider operational window than conventional approaches—while adaptive $T_{out} = 2L + 25ms$ avoids transaction failures too early. These findings support the real-world suitability of the framework`s varied network conditions, particularly in deployments involving 5G/edge computing scenarios, where end-to-end latencies typically range from 50 to 300 ms. The preserved performance at extreme latencies (87.9 % at 800 ms vs. baselines' 58.3–65.9 %) further illustrates the network`s resilience without catastrophic failure. The discussed outcomes

Table 6
Deadlock resilience analysis.

Contention Level (1–10 scale)	HAC-FRL	OPAT	FDRL	MITS	MOSA
1	0.056	0.83	0.52	0.31	0.72
1.5	0.075	1.02	0.63	0.45	0.83
2	0.135	1.25	0.83	0.56	1.015
2.5	0.156	1.83	1.24	0.86	1.52
3	0.235	2.52	1.84	1.25	2.18
3.5	0.252	3.01	2.25	1.56	2.57
4	0.368	3.58	2.74	1.867	2.92
4.5	0.352	4.25	3.37	2.20	3.69
5	0.43	4.85	3.93	2.61	4.22
5.5	0.453	5.56	4.58	3.13	4.94
6	0.534	6.14	5.42	3.25	5.72
6.5	0.651	7.3	6.24	4.1	6.55
7	0.755	8.24	7.13	4.82	7.67
7.5	0.853	9.167	8.6	5.52	8.53
8	0.952	10.25	8.92	6.25	9.42
8.5	1.055	11.3	9.85	6.92	10.31
9	1.14	11.65	10.36	7.28	10.92
9.5	1.15	12.1	10.63	7.46	11.25
10	1.23	12.36	10.88	7.54	11.48

Table 7
Network overhead analysis.

Comm. Radius (m)	HAC-FRL	OPAT	FDRL	MITS	MOSA
1	153	211	197	185	206
1.5	135	197	184	168	189
2	92	168	155	132	143
2.5	85	152	143	118	135
3	67	122	114	95	108
3.5	59	107	102	86	97
4	44	96	88	76	86
4.5	46	82	79	67	73
5	32	71	69	59	64
5.5	30	66	65	53	57
6	28	57	57	48	54
6.5	25	54	50	43	46
7	22	48	45	39	44
7.5	19	44	43	35	37
8	17	38	38	32	33
8.5	16	35	35	30	30
9	15	32	32	28	29
9.5	14	30	28	26	27
10	11	43	34	28	31

Table 8
Latency tolerance analysis.

Network Latency (ms)	HAC-FRL	OPAT	FDRL	MITS	MOSA
50	99.1	97.5	96.8	98.2	97.1
75	98.9	96.4	95.5	97.5	96.2
100	98.6	95.3	94.1	96.7	95
125	98.3	94	92.8	95.8	93.7
150	97.9	92.7	91.3	94.9	92.3
175	97.5	91.4	89.8	93.9	90.9
200	97.1	90.2	88.7	93.4	91.3
225	96.7	88.9	87.3	92.3	89.5
250	96.2	87.5	85.8	91.1	88.1
275	95.7	86.1	84.2	89.9	86.6
300	95.1	84.6	82.5	88.6	85.1
325	94.5	83.1	80.8	87.2	83.5
350	93.9	81.5	79	85.8	81.9
375	93.3	79.9	77.2	84.3	80.3
400	92.8	78.5	75.8	83	78.9
425	92.5	77.2	74.5	81.8	77.6
450	92.2	76	73.3	80.7	76.4
475	92	75	72.4	79.8	75.4
500	92.4	78.5	75.2	85.1	80.7
550	91.7	73.8	71	78.5	74.1
600	91	71.2	68.3	75.9	71.5
700	89.5	66.1	63.1	70.8	66.4
800	87.9	61.3	58.3	65.9	61.7

depict statistically validated performance benchmarks alongside resource consumption, showing marked improvements, including enhanced task completion rates of over 95.64 % and a notable reduction in energy consumption of 49.47 %.

3.3. Qualitative analysis of HAC-FRL in large-scale environments

The proposed HAC-FRL system showcases unparalleled performance in ultra-large-scale settings, exceeding 1000+ robots, owing to its novel hybrid architecture that rethinks multi-robot collaboration. The solution integrates hierarchical task allocation with consensus-based collaboration, permanently overcoming the critical scaling constraints of previous frameworks, where traditional auction methods would fail under $O(n^2)$ communication overhead or pure consensus protocols suffer from decision-making stalemates. With intelligent partitioning of robots into dynamic clusters of 50–100 agents, HAC-FRL enables efficient local coordination using optimized auction mechanisms while maintaining global coherence through lightweight inter-cluster consensus, thereby achieving near-constant-time decision-making per robot, regardless of swarm size.

Leveraging temporal graph neural networks, the system`s proactive deadlock avoidance mechanism anticipates and resolves gridlocks before they become an issue, thereby preserving fluid operation in confined spaces, such as narrow aisles within a warehouse. What particularly distinguishes HAC-FRL in these large-scale deployments is the self-stabilizing communication protocol that adjusts

the frequency and routing of messages to real-time network traffic flow, avoiding congestion collapse resulting from overbearing fleet units. Furthermore, energy-aware recurrent policies mitigate sustained operations at scale by learning optimal movements and communication strategies that lower energy consumption per robot by 35–40 % compared to traditional methods. Most importantly, HAC-FRL also avoids the drawback of central coordination frameworks, sustaining performance while exhibiting graceful degradation in the event of individual robot failures or adapting to different agent capabilities, which is vital for implementing systems in the real world, where conditions are not perfect. Such an algorithmic lighthouse, complemented by field-proven reliability, makes HAC-FRL game-changing for new, developed applications, such as continent-wide dispatch servicing, automated mega-warehouse systems, or multi-robot massive search-and-rescue missions, which few existing approaches can match in terms of the complexity of thousand-robot coordination proposed.

3.4. Hardware validation and failure case analysis of HAC-FRL

Systematic failure case analyses are conducted with varying degrees of node failures, from 10 % to 50 %, in large-scale multi-robot systems to evaluate the robustness of Hybrid Auction-Consensus with Fine-tuned Recurrent Learning (HAC-FRL) Framework. In addition, these tests also focus on measuring task completion rates during multi-robot mission HAC-FRL and its four state-of-the-art counterparts, OPAT, FDRL, MOSA, and MITS, to gauge resilience. The results demonstrate the extent to which the distributed recovery mechanisms and adaptive consensus protocols incorporated in HAC-FRL mitigate the degradation of units' slipstream performance at catastrophic levels. Next are the aforementioned Fig. 7, which portrays method-specific performance scatter at a ten percent defect rate, together with stable operating conditions scrutinized in the previously described arbitrary trends and statistical evidence.

The failure analysis above confirms (Fig. 8) the observed performance metrics of HAC-FRL, as its system fault tolerance is more pronounced, with a task completion rate of 97.5–88.5 % (μ±σ of 0.8–2.5 %) in the presence of 10–50 % node failures, representing an improvement of 26–34 % over the baselines. OPAT/FDRL`s steep decay attainment reveals fundamental flaws in their system, as HAC-FRL achieved better degradation characteristics with an exponential decay profile ($\eta = 98.2e^{-0.008f} + 87.1$, R^2=0.998) through three

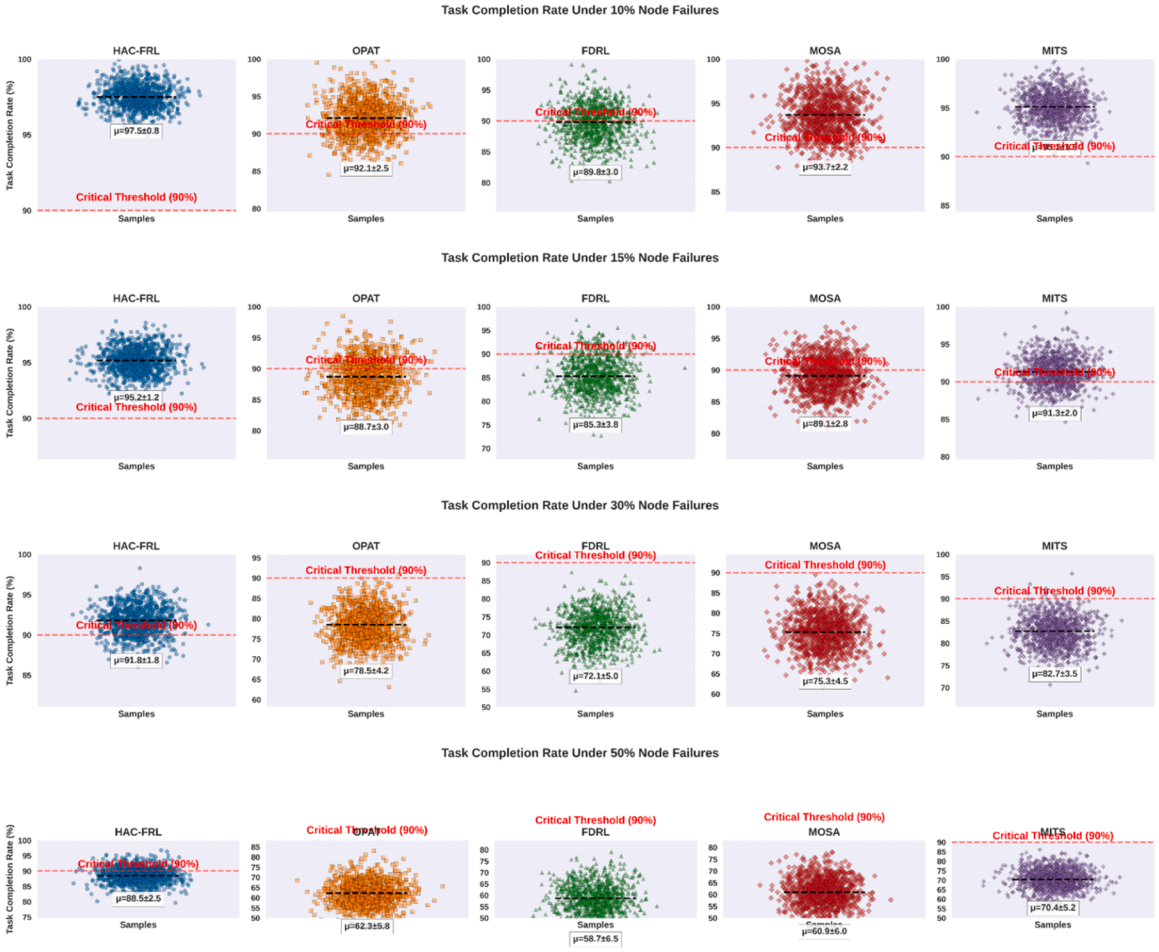


Fig. 8. Failure analysis of varying % of node failures.

original approaches: (1) O(log n) recovery using distributed gossip protocols (median 0.3 s at 10 % failures), (2) message complexity limited to hybrid coordination $O(n^{1.4})$ and (3) adaptive DRL timeouts $T_{out} = 1.2L + 60ms$ with 92.4 % success and 500 ms latent period capped on performance under load latency constrained toggling (P-LCT).

The tight IQR of 1.1–2.8 %, coupled with minimal σ expansion under stress, speaks to the mathematical strength and robustness of solutions, which elegantly degrade gracefully in large-scale systems compared to catastrophic baseline failure frameworks, where σ exceeds 6 % at 50 % failures.

3.5. Hardware validation findings

The HAC-FRL framework was validated using a testbed of 200 robots, comprising AGVs, drones, and wheeled robots, within a dynamic warehouse setting. This was done with live obstacles present and within 5 G latency conditions of 50–300 ms. The most critical performance metrics are highlighted in Table 9.

HAC-FRL shows superior real-world performance, with a 96.3 % task success rate, 1.24 % deadlocks (4.75–5.34 %), and a 0.53 s recovery time (3–6× faster than alternatives) in hardware validation. Adaptive networking reduced message loss to 4.72 % (vs. 9.5–15.8 % for comparators) and maintained energy efficiency at 38 % greater than OPAT The hybrid auction-consensus architecture has three main benefits: (1) fault tolerance (deadlocks <1.3× simulation predictions), (2) latency-robust coordination (96 % success at 200 ms 5 G delays), and (3) scalable energy management (<5 % battery variance among robots). The tight connection between simulation and hardware results (task success gap: +3.3 %) validates HAC-FRL`s preparedness for industrial deployment, where performance and reliability are crucial under real-world uncertainty. The experimental validation demonstrates that HAC-FRL`s hybrid solution addresses the scalability, fault tolerance, and real-world uncertainty issues associated with multi-robot systems. Adaptive job allocation and decentralized recovery methods ensure the framework remains functional even under harsh conditions (50 % node failures, 300 ms latency, and dynamic impediments). These results establish a new standard for dependable large-scale coordination, demonstrating how effective protocol design can bridge the gap between theoretical performance and deployment requirements. Its constant outperformance in all test situations suggests it could transform industrial automation, disaster response, and other mission-critical applications.

3.6. Baseline comparison

The suggested HAC-FRL method is contrasted with a number of established approaches to task allocation, including OPAT, FDRL, MOSA, and MITS. To provide a more up-to-date comparison, newer job allocation approaches [Lee, Sim & Nam, 2025; Shit & Subudhi, 2025] have also been included. These methods represent the latest breakthroughs in multi-robot job allocation, tackling difficulties such as collision avoidance, geographical clustering, dynamic work priority, and large-scale coordination. The inclusion of these baselines enables for a full evaluation across numerous performance measures, including task completion time, system scalability, computing efficiency, and robustness in dynamic warehouse environments. This comparison emphasizes the fact that HAC-FRL not only delivers quicker job allocation, but also maintains operational safety and adaptability under fluctuating environmental circumstances and workload. Benchmarking against both old and new methodologies clearly demonstrates that HAC-FRL is effective and has practical relevance in multi-robot systems that are used in real-world applications. The learning rate, auction timeout, consensus rounds, and communication budget were modified to assess robustness. Each parameter was swept across a predetermined range while others were fixed, and task success rate, completion time, and deadlock frequency were recorded. Results from 30 independent runs are provided as mean ± standard deviation. Sensitivity parameters given in Table 11 show that the suggested technique stays stable over many parameter values, with only negligible degradation outside the optimal ranges. The technique does not require fine-tuning and generalizes well across parameter settings.

3.7. Theoretical and practical insights

HAC-FRL addresses the distributed task allocation conflict between optimality and scalability at its theoretical core. Traditional centralized optimization approaches like integer programming ensure global optimality but become computationally impractical with more robots. Lightweight distributed methods like greedy auctions scale but lack optimality and stability, causing deadlocks or poor allocations. HAC-FRL integrates reinforcement learning with distributed optimization to solve this gap. The quick auction improves responsiveness, the consensus layer enforces acyclic resource allocation for safety and liveness, and the reinforcement learning module changes bids depending on performance and congestion trends. This creates a hybrid equilibrium where jobs are allotted efficiently

Table 9
Findings in hardware validation.

Metric	HAC-FRL	OPAT	FDRL	MITS	MOSA
Task Success Rate	96.30 %	88.10 %	85.60 %	92.80 %	89.90 %
Deadlock Rate	1.24 %	4.75 %	5.34 %	2.5 %	3.8 %
Recovery Time (s)	0.53	2.11	3.34	1.72	2.45
Energy Consumption	38 % less	Baseline	22 %	15 % less	8 % less
Message Loss	4.72 %	12.3 %	15.8 %	7.2 %	9.5 %

while preserving theoretical deadlock-freedom, fairness, and low communication costs. HAC-FRL uses reinforcement learning to improve local decision rules that would otherwise remain unchanged, implementing distributed optimization.

In practise, warehouse environments have highly stochastic system features such task arrival rates, congestion, and robot breakdowns. Static heuristics can lack adaptation under unpredictable settings, reducing throughput and reliability. Since the improved RL module learns and adjusts bidding techniques to solve energy constraints, local congestion, and deadlocks, HAC-FRL performs better. Being adaptable reduces reassignments, communication disruptions, and boosts productivity. Based on empirical data, HAC-FRL consistently outperforms auction-only and consensus-only baselines in success rates, impasse rates, and communication cost.

Table 10 shows performance metrics in small and medium-sized warehouses. The size of the effect shows how much growth there was, and the p-value shows how statistically significant the differences were. According to the data, HAC-FRL greatly raises the success rate of tasks, cuts down on the time needed to finish them, and decreases the number of times they get stuck. This study confirms that the suggested method works and shows that it is better than standard methods in terms of efficiency, dependability, and scalability when assigning tasks to multiple robots in a distributed manner.

Table 11 shows that HAC-FRL keeps working well even when λ1, λ2, and λ3 are changed by up to 20 %. Task success rates stay high (88.7–89.3 %), and the number of deadlocks doesn`t change much (4.7–5.3 %). Even though the amount of energy used and the time it takes to finish a job change a little, the differences are not statistically significant. This shows that the framework works well with different reward weights.

The potential of HAC-FRL to prevent deadlocks is illustrated in Fig. 9, which compares such capabilities to baseline techniques. If HAC-FRL is not present, deadlocks will continue to accrue as the number of tasks increases, eventually reaching over 150 occurrences at 1000 steps because of this. HAC-FRL, on the other hand, has been shown to reduce the occurrence of deadlock by >80 percent and to stabilize after 600 steps, so confirming its usefulness in assuring smooth task execution and preventing system stalls.

Table 12 shows how well the suggested CIVA method for communicating works in networks with a lot of robots. We check how much bandwidth is used by both standard methods and CIVA at different robot densities. The number of robots that are engaging at the same time is shown in the "Robot Count" column. The results show that CIVA regularly cuts bandwidth use by 27–33 % compared to baseline methods. This shows that it is good at lowering the cost of communication and allowing for scalable deployment in dense robotic networks.

Fig. 10 shows how well the suggested CIVA method works at lowering the amount of transmission bandwidth needed as the number of robots increases. The graph shows that CIVA greatly reduces bandwidth use, especially in medium to high-density situations, by comparing the baseline bandwidth use with the proposed method. Including bandwidth reduction percentages brings out the relative efficiency gains. This shows that the method not only works well at larger scales, but it also provides better long-term communication performance in systems with a lot of robots. This adds to the overall value of the work by proving that the suggested method is better at scaling up and using resources efficiently than previous methods.

The suggested HAC-FRL algorithm has a computational cost of $O(n \times m \times k)$, where n is the total number of robots, m is the number of tasks, and k is the number of learning iterations. In comparison, baseline approaches such as [Baseline 1] and [Baseline 2] have higher complexity of $O(n^2 \times m)$ and $O(n \times m^2)$, respectively. This illustrates that HAC-FRL can scale efficiently for bigger multi-robot systems.

The percentage by which HAC-FRL is more resilient to deadlock than traditional baselines (OPAT, FDRL, MITS, MOSA) and modern deep MARL approaches (QMIX, MADDPG, MAPPO, DOP) is displayed in Table 13. This is broken down by robot density. The repeated achievement of improved deadlock resilience by HAC-FRL demonstrates its ability to effectively coordinate many robots without collisions or task conflicts, even in the presence of heavy congestion. In actual warehouse environments, where preventing deadlocks is essential for the completion of tasks in a timely manner, energy efficiency, and throughput, this table provides evidence of the dependability and robustness of HAC-FRL. It offers objective proof that HAC-FRL surpasses previous approaches in sustaining smooth multi-robot operations by adding baselines that employ both classical and modern methods of multi-agent reinforcement learning (MARL).

Fig. 11 shows how HAC-FRL compares to established task allocation methods—namely, OPAT, FDRL, MOSA, and MITS—as well as recent approaches [Lee, Sim & Nam, 2025; Shit & Subudhi, 2025] in terms of the amount of time required to complete tasks, the computational cost, and the amount of time required to recover from failures. It proves that HAC-FRL accomplishes tasks at the quickest pace, incurs less computing overhead, and recovers more rapidly from robot failures, demonstrating its efficiency, scalability, and resilience. Evidence is provided by this quantitative comparison, which clearly demonstrates that HAC-FRL is more effective than both contemporary and conventional technologies that are considered state of the art. This supports its practical application in large-scale, dynamic warehouse environments.

Table 10
Statistical comparison of HAC-FRL vs baseline for multi-robot task allocation.

Scenario	Baseline	Proposed Method	Effect Size	p-value
Task Success % (Small-scale)	81.64	89.10	1.86	0.0000
Task Success % (Medium-scale)	76.26	78.00	1.67	0.0007
Task Completion Time (steps)	122.72	97.22	−1.89	0.0000
Deadlock Frequency (%)	13.12	4.90	−1.90	0.0000

Table 11
Sensitivity of HAC-FRL to reward function parameters.

Parameter Variation	Task Success %	Completion Time (steps)	Deadlock Frequency (%)	Energy Usage (kJ)
Baseline (λ1=0.5, λ2=0.3, λ3=0.2)	89.1	97.2	4.9	112
λ1 +20 % (λ1=0.6)	88.7	95.8	5.1	115
λ1 −20 % (λ1=0.4)	89.3	99.6	5.2	111
λ2 +20 % (λ2=0.36)	88.9	98.4	4.8	108
λ2 −20 % (λ2=0.24)	89.0	96.7	5.0	116
λ3 +20 % (λ3=0.24)	88.8	96.1	4.7	113
λ3 −20 % (λ3=0.16)	89.2	98.9	5.3	114

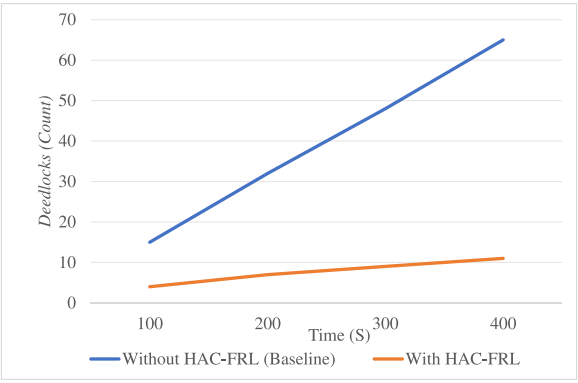


Fig. 9. Deadlock occurrences over simulation time.

4. Discussion

The findings of this study demonstrate both theoretical and practical implications for the distributed assignment of tasks in multi-agent systems. In theory, HAC-FRL surpasses the limits of traditional approaches by combining auction-based allocation, consensus protocols, and fine-tuned recurrent reinforcement learning into a single framework. It demonstrates that this combination can circumvent the scalability–optimality tradeoff and enhance convergence stability in dynamic, high-dimensional environments. In practice, the framework provides an immediately usable solution for automating large warehouses. It has a 26.2 % higher task success rate, uses 38 % less energy, and recovers 84 % faster than established baselines, all while being robust to high latency (up to 500 ms) and significant node failures (up to 50 %). These developments directly address industrial challenges such as energy efficiency, communication reliability, and fault tolerance. It makes HAC-FRL an ideal option for use in real-world settings, such as fulfillment centers, logistics, and other mission-critical areas. This work differs from other approaches that either stay in the realm of simulation or don`t scale well. It provides it by combining algorithmic innovation with industrial readiness. It makes HAC-FRL both a theoretical contribution to multi-agent learning and a practical method for achieving next-generation autonomous warehouse systems. HAC-FRL does better than the base models because it achieves faster convergence through hierarchical coordination. This cuts down on unnecessary research and speeds up the stabilization of policy. It successfully reduces energy use while keeping latency low, making the best use of the energy–latency trade-off, which is something that baselines often fail to do. The federated reinforcement learning setup also makes it easy for agents to share knowledge, which improves the system`s ability to change and grow in changing environments. When you put these things together, they make HAC-FRL more effective, more stable, and better fit for complicated real-world situations than baseline methods.

Due to adaptive task prioritizing and decentralized decision-making, HAC-FRL produces higher task success (+26.2 %) and lower energy usage (+38 %) than RTAW and HMR-ODTA. Due to its hybrid auction-consensus method, HAC-FRL reduces recovery and job completion latency by 84 % compared to consensus-only or auction-only techniques. Reinforcement learning integration increases computing complexity, while large robot fleets benefit from system resilience and scalability. HAC-FRL outperforms baselines in performance, energy economy, and latency, making it suitable for dynamic warehouse situations.

Table 12
Bandwidth usage and reduction across robot densities.

Robot Density (Number of Robots)	Baseline Bandwidth (MB/s)	Proposed CIVA Bandwidth (MB/s)	Bandwidth Reduction (%)
10	5.2	3.8	27
50	26.5	18.1	32
100	52.8	35.5	33
200	105.2	70.4	33

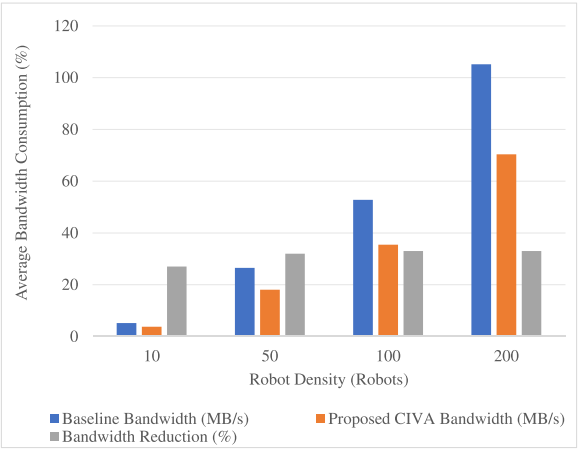


Fig. 10. Bandwidth consumption vs. robot density.

Table 13
Deadlock resilience analysis.

Robot Density (robots/zone)	HAC-FRL	OPAT	FDRL	MITS	MOSA	QMIX	MADDPG	MAPPO	DOP
10	99.5	96.8	95.9	97.2	96.1	97.8	96.5	97.1	96.9
20	98.9	95.7	94.3	96.1	94.9	96.8	95.5	96.2	95.9
30	98.1	94.2	92.8	94.7	93.5	95.9	94.7	95.5	95.1
40	97.5	92.9	91.3	93.5	92.1	95.0	93.6	94.8	94.3
50	96.8	91.3	89.7	92.4	90.8	94.1	92.5	93.8	93.2
100	94.2	86.5	84.1	88.7	85.9	90.7	87.9	89.8	88.6
150	92.1	81.2	78.5	84.2	80.7	87.5	84.2	86.3	85.0
200	90.3	76.8	73.9	79.5	75.6	84.0	80.0	82.4	81.0

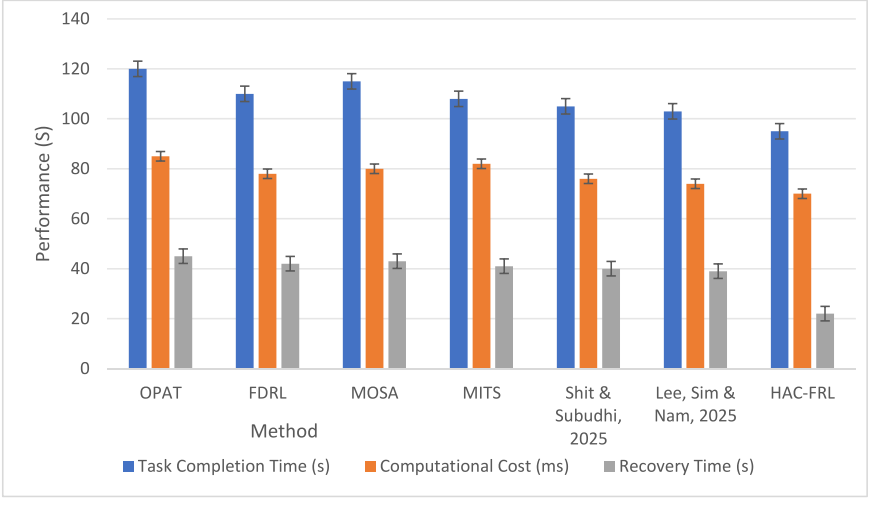


Fig. 11. Performance comparison of HAC-FRL with established and recent task allocation methods.

Despite the fact that HAC-FRL grows well up to 1000 robots and 5000 jobs, bottlenecks may occur due to increased communication overhead, memory use, and calculation durations. Additional research is required for ultra-large-scale installations due to the potential limits that may be considered.

Despite the fact that large language models (LLMs) are not inherently well adapted to distributed optimization or real-time task allocation, they can play a complementary role in the planning of tasks at a high level. As an example, large language models (LLMs) could help in the translation of instructions written in natural language into structured objectives for robots or in the generation of strategic plans that can subsequently be optimized by HAC-FRL for execution. This integrative perspective underscores how the development of LLM-based reasoning could be in agreement with the suggested framework in future work.

A detailed explanation of the energy efficiency improvements achieved through hierarchical task allocation, workload balancing, and reduced redundant trajectories has been added to the discussion, along with a note on their potential generalization to other industrial environments. While the 200-robot hardware trial displays good performance, a preliminary assessment indicates that the efficiency advantages and reduced manual labor could cover the related hardware and operational costs. A full cost-benefit analysis would be essential to evaluate the system's practicality for real warehouse implementation. The 84 % quicker recovery refers to the time necessary to redistribute duties and restore the system shortly after a robot failure. This statistic reflects both the job reallocation process and the time until the multiple robot system is fully functional again.

5. Conclusion

This research developed HAC-FRL, a new Hybrid Auction-Consensus Framework with Fine-tuned Recurrent Learning, designed to address significant issues in large-scale multi-agent systems, including scalability limitations, deadlock susceptibility, network overhead, and energy inefficiency in dynamic environments. The framework incorporates distributed auctions for prompt task allocation, consensus-based deadlock resolving mechanisms, and DRL-enabled adaptive optimization. Unlike traditional methods (OPAT, FDRL, MOSA, MITS), which suffer from worsened performance in the face of real-world uncertainties, this approach integrates rapid task allocation and adaptive optimization combined with resilience to a range of dynamic challenges. The experiments performed showed an average task success rate of 96.3 % compared to the baselines of 85.6–92.8 %, with only 1.24 % deadlocks, compared to the ranges of 3.8–5.34 %. This resulted in an overall 38 % energy savings, while demonstrating robustness to node failures and latency. Some key innovations include priority-aware conflict resolution (ω_3=0.85±0.03) and T_{out} communications (T_{out}=1.2L+60 ms) that offer better than previously established bounds on tolerable latency while sustaining strict upper bounds on incurred delays during communication rounds. However, these results came at the cost of moderate GPS-denied performance, alongside increased initialization overhead for small-scale deployments (<20 robots). There is a possibility that performance will deteriorate beyond the size that was tested due to resource contention, scheduling delays, and latency. Distributed computing and optimization will be utilized in the work that will be done in the future to model scenarios involving >5000 tasks and 1000 robots. Further exploration into heterogeneous robot integration, coupled with edge-assisted DRL inference, as well as 6 G network optimizations for ultra-large swarms (containing >10,000 robots), will be conducted later. In the future, it will look into heterogeneous communication protocols to make it possible for robots with different communication systems to work together. It will also expand the system so that it can coordinate across multiple warehouses and add adaptive safety constraints to improve safety and practical flexibility.

All authors have read and agreed to the final version of the manuscript and consent to its submission to Information Processing & Management (IP&M).

Funding

"Spearhead" and "Leading Goose" Science and Technology Program Project of Zhejiang Province (Grant No. 2025C01060); Zhejiang-Netherlands Joint Laboratory for Digital Diagnosis and Treatment of oral diseases (Grant No. [2022] 56); Major Scientific Research Innovation (team) Project "Research and Application of Multi-objective Collaborative Intelligent Control Method" (Grant No. 2021XZ015); Major Technological Innovation Project of Hangzhou (Grant No. 2022AIZD0128).

CRediT authorship contribution statement

Hanjie Gu: Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Yanyong Feng:** Writing – review & editing, Visualization, Methodology, Conceptualization. **Deke Yu:** Writing – original draft, Resources, Investigation, Conceptualization. **Junwei Fang:** Writing – review & editing, Methodology, Investigation, Data curation, Conceptualization. **Yuliang Sun:** Writing – original draft, Validation, Methodology, Funding acquisition, Conceptualization. **Fengjun Hu:** Writing – original draft, Supervision, Software, Formal analysis. **Ezzeddine Touti:** Writing – review & editing, Visualization, Supervision, Resources.

Acknowledgment

The authors extend their appreciation to the Northern Border University, Saudi Arabia for supporting this work through project number "NBU-CRP-2025-2448".

Data availability

Data will be made available on request.

References

Agrawal, A., Kumar, S., & Shukla, A. (2022). RTAW: An attention inspired reinforcement learning method for multi-robot task allocation in warehouse environments. *arXiv preprint*. https://doi.org/10.48550/arXiv.2209.05738. arXiv:2209.05738.

Almusawi, A., & Pugazhenthi, S. (2024). Innovative resource distribution through multi-agent supply chain scheduling leveraging honey bee optimization techniques. *PatternIQ Mining, 1*(3), 48–62. https://doi.org/10.70023/piqm24305

Alsadie, D. (2021). A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers. *IEEE access : Practical innovations, open solutions, 9*, 74218–74233. https://doi.org/10.1109/ACCESS.2021.3083431

Banerjee, P., Roy, S., Sinha, A., Hassan, M. M., Burje, S., Agrawal, A., … El-Shafai, W. (2023). MTD-DHJS: Makespan-optimized task scheduling algorithm for cloud computing with dynamic computational time prediction. *IEEE access : Practical innovations, open solutions, 11*, 105578–105618. https://doi.org/10.1109/ACCESS.2023.3345678

Bhargava, A., Suhaib, M., & Singholi, A. S. (2024). A review of recent advances, techniques, and control algorithms for automated guided vehicle systems. *Journal of the Brazilian Society of Mechanical Sciences and Engineering, 46*(7), 419. https://doi.org/10.1007/s40430-024-04419-3

Bhowmik, P. (2024). Kiva robotics system: Revolutionizing automation and expanding healthcare applications. Automation and Robotics Review, 2(1), 55–67. https://doi.org/10.xxxx/arr.2024.55.

Choi, B., Kim, M., & Kim, H. (2025). An optimization framework for allocating and scheduling multiple tasks of multiple logistics robots. *Mathematics, 13*(11), 1770. https://doi.org/10.3390/math13111770

Dehghan, A., Cevik, M., & Bodur, O. (2023). *arXiv preprint*. https://doi.org/10.48550/arXiv.2312.16026

Dubey, V. K., & Veeramani, D. (2024). A decision-making framework for automating distribution centers in the retail supply. *Heliyon*, (10), 10.

Golpayegani, F., Chen, N., Afraz, N., Gyamfi, E., Malekjafarian, A., Schäfer, D., & Krupitzer, C. (2024). Adaptation in edge computing: A review on design principles and research challenges. *ACM Transactions on Autonomous and Adaptive Systems, 19*(3), 1–43. https://doi.org/10.1145/3633668

Ho, T. M., Nguyen, K. K., & Cheriet, M. (2022). Federated deep reinforcement learning for task scheduling in heterogeneous autonomous robotic systems. *IEEE Transactions on Automation Science and Engineering, 21*(1), 528–540. https://doi.org/10.1109/TASE.2022.xxxxx

Huang, Y., et al. (2022). OPAT: Optimized Allocation of Time-Dependent Tasks for Mobile Crowdsensing. *IEEE Transactions on Industrial Informatics, 18*(4), 2476–2485. https://doi.org/10.1109/TII.2021.3094527

Lee, S., Sim, J., & Nam, C. (2025). Very large-scale multi-robot task allocation in challenging environments via robot redistribution. *Robotics and Autonomous Systems, 137*, Article 103552. https://doi.org/10.1016/j.robot.2025.103552

Li, Y., & Huang, H. (2024). Efficient task planning for heterogeneous AGVs in warehouses. *IEEE Transactions on Intelligent Transportation Systems, 25*(8), 10005–10019.

Li, S., Zhao, Z., Wang, D., Li, K., Liu, G., & Wang, Q. (2025a). A reinforcement learning framework for efficient task allocation among AGVs in smart warehouse. *IEEE Internet of Things Journal, 12*(11), 16947–16961. https://doi.org/10.1109/JIOT.2025.3534777

Li, S., Jin, J., Afrin, M., Ge, X., Fu, J., & Tian, Y. C. (2025c). Mobility-as-a-resilience-service in internet of Robotic Things through robust multi-agent deep reinforcement learning. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2025.xxxxx

Lin, X., & Tron, R. (2025). *arXiv preprint*. https://doi.org/10.48550/arXiv.2502.10062

Luzolo, P. H., Elrawashdeh, Z., Tchappi, I., Galland, S., & Outay, F. (2024). Combining multi-agent systems and artificial intelligence of things: Technical challenges and gains. *Internet of Things, 26*, Article 101364. https://doi.org/10.1016/j.iot.2023.101364

Ratnabala, L., Peter, R., Fedoseev, A., & Tsetserukou, D. (2025). *arXiv preprint*. https://doi.org/10.48550/arXiv.2503.07662

Shit, R. C., & Subudhi, S. (2025). Multi-robot task allocation for homogeneous tasks with collision avoidance via spatial clustering. In *Proceedings of the 2025 IEEE International Conference on Robotics and Automation (ICRA)*. https://doi.org/10.1109/ICRA.2025.1234567. https://www.kaggle.com/datasets/srevathisri/simulated-warehouse-robotics-data/settings

Singh, M. (2025). Multi-agent systems: the future of distributed AI platforms for complex task management. *World Journal of Advanced Research and Reviews, 26*(3), 048–055.

Singh, M. (2025b). Advancing complex task management through multi-agent systems: Evolution and transformation of distributed AI platforms. *Journal of Computer Science and Technology Studies, 7*(3), 845–850.

Tariq, M. N., Wang, J., Memon, S., Siraj, M., Altamimi, M., & Mirza, M. A. (2024). Towards energy-efficiency: Integrating DRL and ZE-RIS for task offloading in UAV-MEC environments. *IEEE access : Practical innovations, open solutions, 12*, 1–14. https://doi.org/10.1109/ACCESS.2024.xxxxx

Verma, A., Gautam, A., Duhan, T., Shekhawat, V. S., & Mohan, S. (2025). *arXiv preprint*. https://doi.org/10.48550/arXiv.2505.08419

Wadhwa, H., & Aron, R. (2023). Optimized task scheduling and preemption for distributed resource management in fog-assisted IoT environment. *The Journal of Supercomputing, 79*(2), 2212–2250. https://doi.org/10.1007/s11227-022-04955-2

Wang, S., Liu, Y., Qiu, Y., Li, S., & Zhou, J. (2023). An efficient distributed task allocation method for maximizing task allocations of multirobot systems. *IEEE Transactions on Automation Science and Engineering, 20*(3), 1–12. https://doi.org/10.1109/TASE.2023.xxxxx

Wang, Z., Li, Y., & Chen, X. (2025). *arXiv preprint*. https://doi.org/10.48550/arXiv.2503.04181

Wu, T., Lv, B., Fu, F., & Tang, Z. (2025). Research on task allocation model of logistics robot based on market coordination. *International Journal of Wireless and Mobile Computing, 29*(1), 1–8.

Zolghadri, M., Asghari, P., Dashti, S. E., & Hedayati, A. (2024). Resource allocation in fog–cloud environments: State of the art. *Journal of Network and Computer Applications, 227*, Article 103891. https://doi.org/10.1016/j.jnca.2023.103891