ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240705

Optimizing Multi-Robot Task Allocation in Dynamic Environments via Heuristic-Guided Reinforcement Learning

Aritra Pal[a], Anandsingh Chauhan[a], Mayank Baranwal[a, b,*] and Ankush Ojha[c]

[a]Tata Consultancy Services Research, Mumbai
[b]Indian Institute of Technology, Bombay
[c]Ernst & Young LLP, Gurugram

Abstract. In modern warehousing environments, efficient task allocation among multiple robots is crucial for optimizing productivity and meeting the ever-increasing demands of online order fulfillment. In this paper, we address the challenging problem of real-time multi-robot task allocation (MRTA) in a warehouse setting, where tasks appear dynamically with corresponding start and end locations. The objective is to minimize both the total travel distance of robots and the delay in task execution while considering practical charging/discharging constraints and collision-free navigation. To tackle this combinatorially hard problem, we propose a heuristic guided reinforcement learning (RL) agent, *HeuRAL-MATE*, which learns to prioritize prompt task execution while optimizing the assignment of tasks to robots. Our proposed approach outperforms standard practices like First-In-First-Out (FIFO), as well as a brute-force optimal approach in terms of efficiency and performance. The results on multiple synthetic datasets exhibit an average cost reduction of approximately **8.58%** and **10.74%** in total expenses when compared with brute-force optimal approach and FIFO, respectively.

1 Introduction

Multi-robot systems in a cooperative environment find applications in several domains, including but not limited to search and rescue operations [22], precision agriculture [6], warehouse automation [13, 10], construction [24], environmental monitoring [18], and transportation and logistics [7]. Such systems are characterized by availability of multiple robots collaborating to achieve common objectives, and offer numerous benefits, such as scalability, efficiency, and fault-tolerance.

Combinatorial complexity in warehouse environments Automating warehouse operations through the utilization of multi-robot systems engenders a unique spectrum of challenges rooted in the complexities of the spatial layout, the diverse range of tasks, the heterogeneity of robot capabilities, and the imperative for safe robotic navigation [4, 28]. From a broader perspective, these challenges can be categorized under three primary objectives: (a) *Path planning*, (b) *Real-time robot assignment*, and (c) *Task allocation*. While these objectives are interdependent, each gives rise to a distinct array of subproblems that necessitate resolution, subsequently influencing the comprehensive optimization of warehouse operations. For instance, the requisites of effective path planning encompass the generation of collision-free trajectories, encompassing both static obstacles and the trajectories of other in-motion robots. Moreover, the physical constraints of robots, such as their state of charge (SOC), must also be factored in. In parallel, within the dynamic milieu of a warehouse, real-time task generation becomes imperative, precluding a priori knowledge. This injects complexities into the planning process, requiring the prioritization of tasks based on their generation time, anticipated completion timeframe while considering the concurrent execution of pre-assigned tasks, and the minimization of collective robot travel distances.

Furthermore, the immediate allocation of robots to tasks requires a seamless and continuous distribution of assignments, regardless of whether robots are available or already occupied. This requirement highlights the need for instant coordination across multiple goals, all while upholding a variety of distinct physical and task-related limitations. In essence, the intricate interplay of these challenges emphasizes the vital importance of a refined and flexible multi-robot framework. Such a framework must possess the capacity to systematically handle the nuances of path planning, real-time robot assignment, and task allocation within the dynamic landscape of an automated warehouse environment. Neglecting the interconnectedness of diverse subtasks leads to suboptimal outcomes.

Motivating example This study is driven by the goal of optimizing end-to-end operations within a typical sorting center (Figure 1). Incoming packages must be efficiently directed from conveyor belts to designated pickup trucks situated across the warehouse. These packages are stored in bins at conveyor ends, and as bins approach full capacity, robots are tasked with emptying them near assigned pickup points. Additionally, robots can be assigned to move empty bins closer to conveyor belts. Amidst these tasks, robots must navigate without collisions, maintain adequate energy levels, and dock for recharging if needed. Real-time package arrivals render offline optimization of robot assignments and task selection challenging when confronted with multiple pending tasks. Furthermore, addressing charging constraints is often disregarded in existing warehouse management literature. In real-time decision-making, brute-force or simulation-based approaches evaluating all assignment scenarios prove impractical and inefficient for such systems.

Why RL? The complex interplay among diverse subtasks, combined with the real-time influx of tasks and the varying state of charge

* Corresponding Author. Email: baranwal.mayank@tcs.com.

(SOC) of robots, accentuates the limitations of existing warehouse management strategies. The typical warehouse environment is characterized by its dynamic nature, necessitating a real-time, sequential decision-making approach for effective optimization. In this context, RL emerges as the optimal choice, given its ability for handling such scenarios. Furthermore, RL can adeptly learn to optimize its task selection procedure in conjunction with other heuristic policies, such as space-time A^* [26], alongside a greedy approach for robot selection and charging. Hence, the optimization of task selection occurs in tandem with existing operational policies, rather than in isolation.

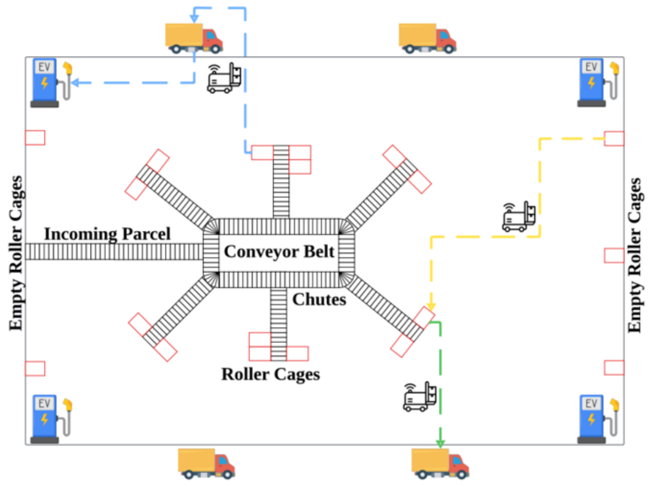

Figure 1. Schematic of a typical warehouse

Statement of Contributions In this study, we present an innovative approach called *HeuRAL-MATE*, which combines heuristic guidance with RL to address the multi-robot task allocation challenge in warehouse environments. HeuRAL-MATE effectively manages real-time task selection, the allocation of tasks to robots, and the secure navigation of robots, all while considering constraints related to robot charging and specific task requirements. Notably, it boasts scalability, seamlessly accommodating any quantity of robots and task frequencies. Below, we outline the primary contributions.

1. Real-time warehouse management under practical constraints Contemporary warehouse management research frequently relies on impractical assumptions, like assuming robots never collide, are always instantly available for tasks, or never discharge. Our study presents a holistic framework that eliminates these assumptions. We optimize task selection to improve decision-making throughout all process stages, employing efficient heuristics to address practical constraints. The task selection module, based on RL, is trained to act optimally in tandem with these heuristics, facilitating cooperative learning under realistic warehouse conditions. Our framework, HeuRAL-MATE, thus ensures that the task selection carried out by the RL agent is suitably influenced by the performances of concurrently employed heuristic policies. To the best of our knowledge, this represents the first such effort that concurrently addresses all the above constraints.

2. SOTA performance on diverse datasets We evaluate HeuRAL-MATE on a variety of datasets comprising of dynamic task generation in a real sorting center[1], and benchmark our approach against the commonly employed FIFO-policy (aimed to minimize delay in task execution post generation), and a brute-force optimal policy (a brute-force approach aimed to minimize the total travel distance over short horizon). The datasets are generated under several practical scenarios including distributional shift in task generation, variable number of robots, and SOC considerations. HeuRAL-MATE significantly outperforms standard policies in all these scenarios while maintaining the least execution time.

3. Post-hoc analysis and insights We demonstrate that the optimal policy acquired through HeuRAL-MATE strikes a balance between two advantageous aspects. It gives priority to minimizing task execution delays, while simultaneously developing the capability to reduce the overall travel distance.

[1] name withheld due to NDA

2 Preliminaries

2.1 States, Actions and Rewards

The problem of MRTA in dynamic warehouse environment settings can be modeled as a Markov Decision Process (MDP) [21]. An MDP is denoted by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}_{\mathcal{A}}, r \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ represent the finite set of states and actions, respectively. For each $s, s' \in \mathcal{S}$, the transition probability from $s \rightarrow s'$ under the effect of an action $a \in \mathcal{A}$, is denoted by $p_a(s, s') \in \mathcal{P}_{\mathcal{A}}$. Finally, the step-reward associated with each state-action pair (s, a) is depicted by $r(s, a)$. Below, we summarize the set of all possible states, actions and reward in the context of warehouse management.

States At each time step, the comprehensive state of the dynamic warehouse environment necessitates the inclusion of information pertaining to tasks and robots. Incoming tasks are immediately stored in a buffer, creating a limited-size look-ahead (LA) queue in a FIFO fashion. The RL agent is trained to optimally select tasks from this queue based on the current environment state. Designated as $s_t \in \mathcal{S}$ at time-step t, the environment's state encompasses task attributes including their source and destination coordinates, navigation duration between them obtained via the standard A^*-type algorithm, and the time of task appearance in the environment's LA. Furthermore, robot features entail their positions, the time at which an engaged robot becomes free for executing subsequent tasks, and their charge levels.

Actions The planner's main objective is to enhance task assignment, thereby minimizing total operational time. Accordingly, the planner's actions encompass systematic task selection from the queue. This sequential selection reduces both the overall operational expenses and the task completion time.

Rewards The step reward attributed to a task-action pair encompasses three distinct components. The initial component is calculated based on the time it takes for the robot to travel from its current position to the task's starting point, termed as *travel time to origin* (TRTO). The second facet is the time gap between task arrival in the LA and the robot's initiation of execution, denoted as the *total time gap for the task* (TTGT). Additionally, the step reward incorporates a third element that enforces penalties for tasks lingering in the queue for extended periods. (see (1) for detailed mathematical formulation). This reflects the real-world principle that tasks ideally should not remain unattended for prolonged duration.

3 Related Works

Efficient MRTA and path planning within unmanned warehouses lead to optimized order fulfillment, resource management, and obstacle-free navigation, ultimately elevating warehouse productivity. Consequently, MRTA has garnered substantial interest, spanning from heuristic-driven approaches to contemporary learning-based methods in both centralized and distributed communication scenarios over the last two decades [14]. One of the earlier works in [9] examines a comparative analysis of SOTA multi-robot coordination strategies within a domain-specific context. Current MRTA research predominantly centers on two pivotal elements: (a) Model-driven optimization, as exemplified in [27], and (b) Communication-efficient decentralized algorithms, illustrated by [5, 1].

The problem under consideration can be mapped to multi-agent pickup and delivery (MAPD) problems addressed using both distributed approaches and centralized ones [17, 16, 23, 29]. But, Most existing literature focuses on offline MAPD. Our choice of learning-based methods for online task allocation arises from the need for reliable solutions in dynamic environments where solving optimization problems every time can be computationally prohibitive.

Driven by the recent achievements of deep RL in solving intricate dynamic challenges, a current trend is emerging to embrace learning-based approaches to manage the complexities of warehouses [30, 1, 2]. These learning strategies target various facets of end-to-end warehouse management. For instance, in [30], a Q-learning framework emphasizes generating collision-free, secure paths for multi-robot systems. Conversely, the RL framework in [1, 2] focuses on optimal task selection, neglecting task-to-robot assignment under the assumption of constant robot availability post-selection. Additionally, the authors leverage A^* coupled with optimal reciprocal collision avoidance (ORCA) [3] for collision-free navigation at the low-level path planning stage.

However, as previously discussed, most SOTA learning-based warehouse management approaches, including the mentioned references, overlook a key benefit of RL-namely, the possibility of integrating of multiple echelons of warehouse management during the training phase of RL. For instance, the learned policy of the RL-agent in [1, 2] is limited in its applicability to realistic warehouse scenarios due to the neglect of constraints related to robot availability and state of charge (SOC). Apart from that, they aim for a seamless sequential flow of tasks without considering their generation times. Likewise, the authors in [20] utilize a cooperative multi-agent RL framework under the assumption of robots never colliding. In our study, we integrate task arrival times to ensure the timely execution of tasks by amalgamating practical considerations, such as robot availability, SOC and generating collision-free paths, performance under distribution shifts and variable-sized fleet. Moreover, within our framework, the reward structure is meticulously crafted to achieve an optimal balance between prompt task allocation and the shortest possible execution duration for the allocated tasks. This is achieved by harmonizing efficient heuristics with the RL agent. The approach also ensures competitive runtime during the deployment phase, catering to real-time task selection and allocation via HeuRAL-MATE.

4 Our approach: HeuRAL-MATE

In this section, we present HeuRAL-MATE (see Algorithm 1), a novel heuristic guided RL framework designed for optimizing MRTA within dynamic warehouse settings. The generation of tasks occurs in real-time, initially populating a task buffer. The planner (RL agent) has access to a small LA queue of tasks. When a task from the LA, visible to the RL agent, is assigned to a robot for execution, a new task from the buffer enters the LA, making it available for selection by the RL agent. Upon task allocation from the LA queue, if no new tasks are available in the buffer for inclusion in the LA, the task with the longest duration of stay in the LA is duplicated to ensure a consistent LA length. This increases the probability of its selection by the RL agent in subsequent steps. In an exceptional scenario where both the LA and the task buffer are empty, HeuRAL-MATE waits for a task to appear in the environment. This process of action selection, revolves around task choice, unfolds across three hierarchical levels:
(a) At each instant t, the RL agent picks a task from the LA for assignment to one of the robots, guided by the prevailing environmental state. Importantly, the RL agent doesn't wait for robots to become available before engaging in task selection, as the state information includes time markers denoting when robots will be free.
(b) Upon task selection, a heuristic-guided framework comes into play, facilitating the identification of the most suitable robot for task execution. This determination considers the positions of all robots post completion of their ongoing assignments, the times they become available, and their SOCs.
(c) To direct the robot's movement, a lower-level navigation algorithm, such as Space-Time A^* [26] is utilized to guide the robot from its current position to the task's starting point and subsequently to the destination, ensuring collision avoidance with both stationary and moving obstacles (and other robots).

Our study tackles both (a) non-charging and (b) charging scenarios in dynamic warehouse settings. In the charging scenario, we account for the unique behavior of robots that do not discharge while stationary but discharge at a steady rate during motion. When a robot's charge falls below a predefined threshold (charging threshold), it navigates to the nearest available charging dock for recharging. The threshold is carefully calibrated to ensure that it enables the robot to reach the nearest charging dock from any location on the grid. During task assignment, we compare the robot's current battery level with the anticipated charge required to complete the task against this threshold. This guarantees that the robot will always fulfill its assigned tasks without facing any issues related to insufficient battery levels. In the simplified non-charging scenario, we assume that robots consistently maintain charge levels above the charging threshold.

4.1 Proximal Policy Optimization (PPO)-guided Reinforcement Learning framework

In order to learn to perform optimal sequential decision making, we train an on-policy PPO [25] agent with a prioritized replay buffer defined over the set of environment states. The state of the agent consists of the features related to tasks in the LA (denoted by $\mathcal{P}$) as well as set of robots ($\mathcal{R}$). The agent's state, denoted as $s_t \in \mathcal{S}$ at time-step t, encompasses the following components: (a) origin coordinates of tasks $\{o_i\}$, (b) destination coordinates of tasks $\{d_i\}$, (c) distance information between task origin and destination $\{k_i\}$, computed using the standard A^* method, (d) timestamp of task appearance in the LA queue $\{l_i\}$, (e) robot coordinates $\{p_j\}$, (f) robot availability and the anticipated time for ongoing task completion $\{r_j\}$, (g) robot charge percentage $\{c_j\}$. Features (a)-(d) are task-specific features and collectively have a dimensionality of 6 for each task. Conversely, features (e)-(g) are linked to robot-specific attributes, constituting a dimensionality of 3 for setups without charging considerations and 4 for the ones with charging constraints.

Let (x_{o_i}, y_{o_i}) and (x_{d_i}, y_{d_i}) represent the origin and destination coordinates of the ith-indexed task. Similarly, (x_{r_j}, y_{r_j}) indicates the current position of robot j, which can vary based on whether the robot is idle, performing tasks, or at a charging location for recharging if needed. Additionally, we introduce t_{stamp_i}, t_{exec_i}, and t_{wait_i} to denote the time when task i appears in the task allocation (referred to as LA), the time at which its execution begins, and the waiting time for a robot to start executing the task after completing other preassigned tasks, respectively. In each decision-making step, we use the variable allotT to signify the index of the selected task which then

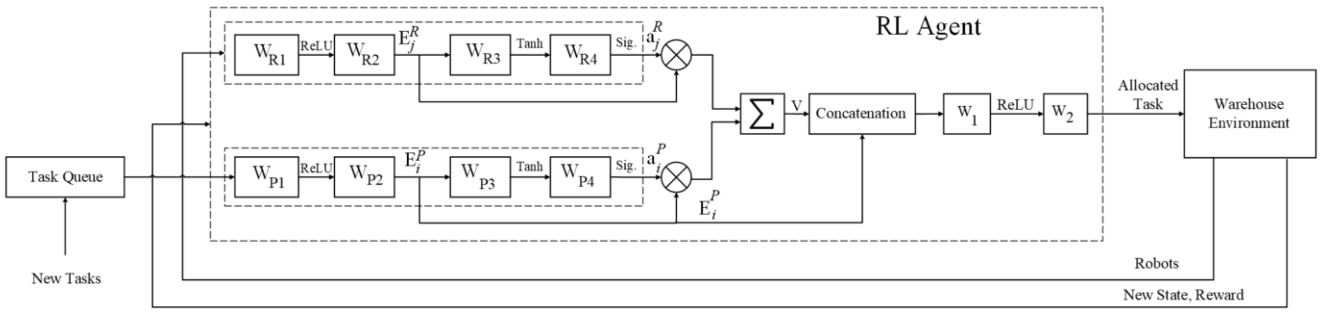

Figure 2. HeuRAL-MATE Architecture with trainable weights denoted by $\{W_k\}$

gets assigned to a robot selR. Furthermore, we utilize the function $d[(x_a, y_a), (x_b, y_b)]$ to calculate the A^* distance between two points (x_a, y_a) and (x_b, y_b) within our system. The step reward for training the PPO agent is:

$$\begin{aligned} R_{\text{step}} \quad = \quad & -d[(x_{r_{\text{selR}}}, y_{r_{\text{selR}}}), (x_{o_{\text{allotT}}}, y_{o_{\text{allotT}}})] \\ & -\alpha * (t_{\text{exec}_{\text{allotT}}} - t_{\text{stamp}_{\text{allotT}}}) - \beta * \text{Penalty}, \end{aligned} \qquad (1)$$

$$\text{where,} \qquad \text{Penalty} = \begin{cases} p, & \text{if } t_{\text{wait}_{\text{allotT}}} > t_{\text{threshold}} \\ 0, & \text{otherwise} \end{cases},$$

imposes a penalty of p in cases where a task indexed as allotT remains in the LA for a duration exceeding a specified threshold $t_{\text{threshold}}$. The coefficients α and β represent positive constants. The first term in (1) correspond to TRTO, while the second term is associated with TTGT, (see Section 2.1).

4.2 *Robot Selection Heuristic*

The decision at the second level entails the selection of a robot to execute designated tasks once the robot becomes free. To achieve this, we have adopted a simple heuristic based approach where we calculate the execution time for all the existing robots. The robot selected for a particular task is the one capable of completing the task earliest. There is a possibility that a robot, even if immediately available but located farther away, might not be able to promptly finish the task. As a result, we employ a comprehensive heuristic to ascertain the most suitable robot for task execution (see `selectRobot(·)` in Algorithm 1).

4.3 *Navigation Algorithm : Space-Time A**

In a real-world warehouse environment, a substantial fleet of robots is deployed. The primary operational challenge emerges when calculating collision-free paths. In our work, we integrate space-time A^* [26] for dynamic collision avoidance.

5 Experiments

5.1 *Datasets and Experimental Setup*

The dynamic warehouse environment comprises several critical parameters that govern its efficient operation. To evaluate our RL-based model, we utilize synthetic data due to the absence of publicly accessible real-world datasets for comparable problem instances. Each episode encompasses some τ time units, and the synthetic datasets employed for training and evaluation are structured as square-shaped 2D grids within the range of [0, 64] × [0, 64]. The task origin and destination, as well as robot locations, are within the above range. In our experiments, the concentration of tasks is varied to capture specific time intervals during the day when majority of the tasks tend to accumulate. Accordingly, we generate datasets using Gaussian distributions with varying means and standard deviations. The task's starting and ending coordinates, along with task generation times, are disclosed to the planner using a limited-size LA. The dataset also includes initial charge levels and locations of charging docks.

In this work, we investigate four unique combinations:
(1) Non-charging + Normally distributed task arrival times
(2) Charging + Normally distributed task arrival times
(3) Non-charging + Uniformally distributed task arrival times
(4) Charging + Uniformally distributed task arrival times

Within each of the four configurations, datasets containing approximately 500 tasks per episode are generated following the corresponding distributions. As an illustration, in the case of normally distributed tasks, task generation times adhere to $\mathcal{N}(600, 50)$, with 600 denoting the mean task generation time. We first consider a fleet of 10 robots, and the LA length is fixed at 5. The robots' permissible charging threshold is established at 30%, signifying that robots with a SOC below 30% must dock for recharging before resuming task execution. The steady charging rate for the robots is calibrated to be 16 times the discharging rate.

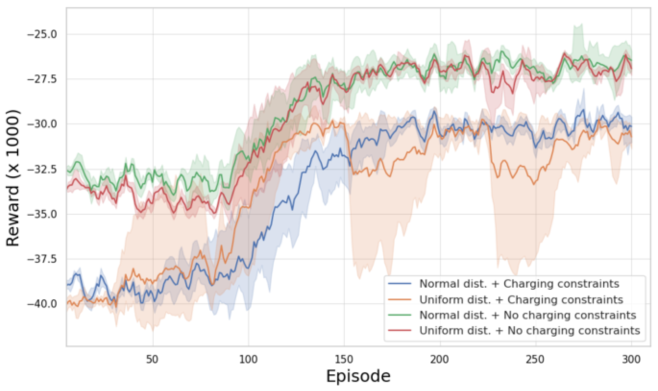

Figure 3. Learning curves of the HeuRAL-MATE

5.2 *Neural Network Architecture and Training*

The RL agent employs a novel PPO-based framework to facilitate online task selection. This model architecture inspired from [2] is distinctly structured into three core segments (see Figure 2):
• **Feature Extraction Layers** This initial component involves the procedure of feature extraction, particularly focusing on attributes related to robots and tasks. The embedding for robot attributes is created using a sequence of four linear layers of dimensions [Input, 16,

Algorithm 1: HeuRAL-MATE

```
Initialize policy parameters $\theta_0$, value function parameters $\phi_0$
for episodes $k = 0, 1, 2, \cdots K$ do
    Step t=0
    State $s_t := \{(o_i, d_i, k_i, l_i)_{\forall i \in \mathcal{P}}, (p_j, r_j, c_j)_{\forall j \in \mathcal{R}}\}$
    Each robot $j \in \mathcal{R}$ is executing a task $(o_j, d_j)$
    while True do
        Update $p_j \; \forall j \in \mathcal{R}$ using space-time $A^*$
        if $p_j == d_j$ then
            if $c_j <$ charging threshold for some $j \in \mathcal{R}$ then
                Robot $j$ navigates to closest available
                  charging dock for recharging
            end if
            if model is in training mode then
                run policy $\pi_k = \pi(\theta_k)$ in the environment to
                  select a task $i \in \mathcal{P}$ as action
            else
                RL-policy takes state $s_t$ as input and selects a
                  task $i \in \mathcal{P}$ as action
            end if
            selR $= selectRobot(o_i, d_i)$
        end if
        $t \leftarrow t + 1$
    end while
    if model is in training mode then
        Collect set of trajectories $\mathcal{D}_k$
        Compute rewards-to-go $\hat{R}_t$
        Compute advantage estimates $\hat{A}_t$ based on the current
          value function $V_{\phi_k}$
        Update policy by maximizing PPO-Clip objective:
          $\theta_{k+1} =$
          $\arg\max_\theta \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} min\left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\right.$
          $\left. g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right)$
          via stochastic gradient ascent with Adam
        Fit value function by regression on MSE:
          $\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left( V_\phi(s_t) - \hat{R}_t \right)^2$
          via gradient descent
    end if
end for
function selectRobot$(o_i, d_i)$:
    Calculate estimated completion time of task $i$ by each
      robot $j \in \mathcal{R}$, store them in array $A_i$
    selR $= argmin_{j \in \mathcal{R}} A_i$
    return selR
end function
```

16, 1]. Here, the Input is 3 for scenarios excluding charging considerations, and 4 for scenarios involving charging constraints. Concurrently, embeddings related to task attributes are generated using a similar sequence of four linear layers of dimensions [6, 16, 16, 1].

Let F_j^R and F_i^P represent the feature vectors for the robot $j \in \mathcal{R}$ and the ith-task, then the associated embeddings are defined as:

$$\begin{aligned} E_j^R &= W_{R2} * \text{ReLU}(W_{R1} * F_j^R) \\ E_i^P &= W_{P2} * \text{ReLU}(W_{P1} * F_i^P) \end{aligned}$$

• **Concatenation and Transformation** The second module transforms the extracted embeddings by concatenating the embeddings of robots and tasks. Subsequently, the concatenated feature is channelled through a linear layer boasting 48 input neurons and 8 output neurons with ReLU activation. Mathematically we have

$$\begin{aligned} a_j^R &= \text{Sigmoid}(W_{R4} * \text{Tanh}(W_{R3} * E_j^R)) \\ a_i^P &= \text{Sigmoid}(W_{P4} * \text{Tanh}(W_{P3} * E_i^P)) \\ \text{Output} &= \left(\sum E_j^R * a_j^R, \sum E_i^P * a_i^P, E_i^P \right) \end{aligned}$$

• **Final Layer** Comprising a linear layer, this element operates with 8 input neurons and 1 output neuron.

$$Task^{Allocated} = Categorial(W_2 * \text{ReLU}(W_1 * \text{Output}))$$

The model is implemented using the PyTorch library in Python 3.8, with an Adam optimizer [15], a discount factor at 0.99, lambda value of 0.95, learning rate of 0.0003, entropy coefficient of 0.001, value function coefficient of 0.0002, and a batch size of 32. The policy network is trained employing the cross-entropy loss function, whereas the value network is fine-tuned via an MSE loss metric.

5.3 *Baselines*

It is crucial to re-emphasize that, to the best of our knowledge, there is no existing work in the literature that concurrently addresses multiple aspects of MRTA. In the absence of any known approaches, we present two strong baselines:

• **Brute-force optimal** where all task-robot pairs (within the LA) undergo a brute-force evaluation of time duration required for task execution by the robots, determined using the standard A^* navigation algorithm. Subsequently, the algorithm selects the robot-action pair that minimizes this time duration. Consequently, while brute-force optimal approach represents a locally optimal solution, the brute-force evaluation significantly amplifies the run-time, posing practical challenges. The brute-force optimal baseline is frequently adopted in literature as a reference point for evaluating decoupled task allocation and navigation methodologies [19, 1].

• **FIFO** The FIFO baseline employs a dual-tiered decision framework. Initial allocation entails selecting the task that joined the LA queue earliest. The goal is to reduce pending tasks within the LA queue and minimize the time gap between task arrival and robot-initiated execution (TTGT) [12, 11]. Due to its simplicity, the FIFO approach requires the least execution time among all the considered approaches.

5.4 *Experiments and Results*

In this section, we present a comprehensive comparative analysis of our proposed learning-based framework, HeuRAL-MATE, in comparison with baseline methods, specifically the brute-force optimal approach and the FIFO strategy. The fundamental logic for robot selection remains consistent across both the learning-based policy and the baseline strategies. Our experiments encompass scenarios with and without charging considerations, effectively addressing real-world operational challenges. The results consistently highlight the superiority of HeuRAL-MATE over the baseline methods.

Figure 3 illustrates the average training curve for the RL approach. This average is computed from four distinct runs utilizing random initial seeds. The RL model undergoes 300 training episodes, each

Table 1. Cost $(\times 10^3)$ evaluation on various test datasets with 5 tasks in LA, 10 robots & 505 tasks per episode (the lower the better)

	Test distibution	Charging			Non-Charging		
		HeuRAL-MATE	Brute-force optimal	FIFO	HeuRAL-MATE	Brute-force optimal	FIFO
Training with Gaussian dist. data	Similar	**28.81 $\pm$ 0.08**	29.18	32.69	**24.36 $\pm$ 0.29**	27.01	27.63
		28.98 $\pm$ 0.24	30.91	31.40	**25.83 $\pm$ 0.14**	26.83	27.88
		28.61 $\pm$ 0.35	30.92	31.36	**26.41 $\pm$ 0.22**	28.38	28.77
		28.32 $\pm$ 0.25	29.21	31.69	**24.80 $\pm$ 0.44**	26.70	27.92
		28.62 $\pm$ 0.23	29.82	31.69	**26.73 $\pm$ 0.59**	28.03	28.48
	Moderately Different	**28.44 $\pm$ 0.62**	29.22	31.19	**25.29 $\pm$ 0.09**	27.29	27.36
		29.99 $\pm$ 0.82	31.20	31.74	**25.81 $\pm$ 1.01**	26.39	28.38
		30.32 $\pm$ 0.57	31.37	31.99	**26.79 $\pm$ 0.50**	27.55	29.51
		28.79 $\pm$ 0.48	31.11	31.61	**24.79 $\pm$ 0.05**	26.43	26.98
		29.57 $\pm$ 0.18	32.40	33.67	**26.50 $\pm$ 0.17**	27.28	29.27
	Totally Different	**29.01 $\pm$ 0.45**	30.34	31.74	**25.78 $\pm$ 0.20**	27.79	28.30
		29.04 $\pm$ 0.46	30.87	32.75	**26.65 $\pm$ 0.11**	27.75	28.63
		31.02 $\pm$ 2.02	31.72	33.56	**27.31 $\pm$ 0.05**	28.81	29.93
		29.18 $\pm$ 0.93	31.48	33.02	**25.84 $\pm$ 0.32**	26.27	27.89
		29.02 $\pm$ 2.04	**28.95**	33.30	**27.57 $\pm$ 0.34**	29.02	29.17
Training with uniform data	Similar	**29.90 $\pm$ 1.18**	30.29	33.21	**27.41 $\pm$ 0.07**	28.81	29.93
		28.81 $\pm$ 0.76	31.55	31.80	**25.62 $\pm$ 0.33**	26.27	27.89
		29.70 $\pm$ 1.08	32.12	33.19	**26.73 $\pm$ 0.03**	29.02	29.17
		29.57 $\pm$ 0.83	31.34	33.19	**26.53 $\pm$ 0.75**	28.65	27.88
		28.07 $\pm$ 0.91	30.40	31.97	**26.53 $\pm$ 0.01**	27.50	27.58
	Different	**28.87 $\pm$ 0.14**	27.96	31.77	**25.11 $\pm$ 0.14**	27.01	27.63
		30.13 $\pm$ 0.85	30.70	31.61	**25.98 $\pm$ 0.17**	26.83	27.88
		28.65 $\pm$ 0.15	31.68	32.06	**24.84 $\pm$ 2.20**	28.38	28.77
		28.02 $\pm$ 0.28	30.91	31.40	**25.46 $\pm$ 0.20**	26.70	27.92
		30.26 $\pm$ 0.79	31.39	33.23	26.50 $\pm$ 0.16	**25.63**	27.88
	Avg run-time (s/task)	0.17	0.88	0.15	0.14	0.79	0.13

Table 2. Cost $(\times 10^3)$ comparison on Gaussian distributed data for charging scenario with 5 tasks in LA, 10 robots & 505 tasks/episode

Avg TRTO			Avg TTGT + Delay Penalty		
HeuRAL-MATE	Brute-force optimal	FIFO	HeuRAL-MATE	Brute-force optimal	FIFO
12.30	**10.80**	12.25	**16.47**	18.38	20.43
12.05	**10.28**	12.51	**16.93**	20.63	18.89
12.34	**10.98**	12.22	**16.27**	20.00	19.15
12.43	**11.15**	12.31	**15.89**	18.06	19.37
12.51	**10.31**	12.34	**16.11**	19.51	19.35

Table 3. Cost $(\times 10^3)$ evaluation on Gaussian distributed dataset (charging) with 5 tasks in LA, 20 robots & 505 episodic tasks

	HeuRAL-MATE	Brute-force optimal	FIFO
Gaussian dist. data	**20.60**	22.13	22.35
	21.73	21.84	22.61
	20.95	21.65	21.99
	20.61	22.01	22.10
	21.60	22.79	23.38

Table 4. Cost $(\times 10^3)$ evaluation on Gaussian distributed dataset (charging) with 5 tasks in LA, 15 robots & 505 episodic tasks

	HeuRAL-MATE	Brute-force optimal	FIFO
Gaussian dist. data	**24.05**	25.20	26.19
	24.01	24.53	26
	23.44	24.95	26.44
	24.78	25.61	26.79
	24.32	25.32	26.17

Table 5. Cost $(\times 10^3)$ evaluation on Gaussian distributed dataset (charging) with 5 tasks in LA, 10 robots & 255 episodic tasks

	HeuRAL-MATE	Brute-force optimal	FIFO
Gaussian dist. data	**14.73 $\pm$ 0.15**	14.96	15.81
	14.45 $\pm$ 0.45	14.71	15.67
	14.86 $\pm$ 0.90	14.88	15.75
	14.45 $\pm$ 0.21	15.33	15.99
	14.07 $\pm$ 0.56	14.53	14.97

encompassing 505 tasks with 10 robots in the environment and LA length 5. This training employs the PPO-based RL algorithm within the `PFRL` [8] framework. The training procedure utilizes randomly generated task lists sampled from Gaussian distribution to mirror real-world task profiles, alongside uniformly sampled task lists. Separate models are trained for both robot charging and non-charging scenarios using both datasets. The reward plots illustrate the stable convergence of HeuRAL-MATE towards a favorable optimal policy. Enhancements in rewards compared to initial values indicate proficient task selection and allocation to robots, leading to decreased waiting times for tasks within the look-ahead. It is important to note that the RL agent is trained using a random task list periodically, preventing it from merely memorizing performance on a particular task list. Instead, it learns to adeptly adapt to various scenarios. This explains the minor fluctuations in rewards across episodes, even as the agent develops a steady policy.

Following the training phase, the model's performance is evaluated across various test datasets. For the model trained on task lists adhering to a specific Gaussian distribution, the assessment encompasses four distinct datasets: (a) instances analogous to the training data with the same mean and variance, (b) datasets displaying $\pm 30\%$ variations, termed *moderate variation*, and (c) testing on an entirely distinct uniformly generated dataset. This evaluation aims to measure HeuRAL-MATE's performance in handling distributional shifts. Meanwhile, the model trained on instances generated from a uniform

Table 6. Cost ($\times 10^3$) evaluation on Gaussian distributed dataset (charging) with 5 tasks in LA, 10 robots & 1005 episodic tasks

	HeuRAL-MATE	Brute-force optimal	FIFO
Gaussian dist. data	**58.57 ± 0.37**	62.15	65.89
	56.47 ± 0.17	61.39	64.78
	58.54 ± 0.32	60.41	66.49
	57.69 ± 0.44	63.44	67.23
	56.70 ± 0.10	60.17	63.64

distribution is evaluated using two separate test datasets: (a) dataset sampled from same distribution and (b) dataset generated using a Gaussian distribution, providing insights into its capabilities under distributional shift.

Table 1 presents a comprehensive comparison of the test results. It is noteworthy that the total number of tasks in the entire episode, the number of deployed robots, and the length of the LA queue remain consistent with the training phase, specifically 505 tasks, 10 robots, and an LA length of 5. The results underscore HeuRAL-MATE's consistent outperformance over brute-force optimal and FIFO-based methods across nearly all test scenarios. Only in a couple of instances does the brute-force approach marginally surpass HeuRAL-MATE. These cases involve uniformly distributed task arrival times throughout the episode, where the brute-force optimal approach, with brute-force evaluations at any point, proves to be a potent heuristic. Additionally, Table 1 provides execution time details, further confirming the efficiency of HeuRAL-MATE in handling large order volumes within notably brief timeframes. This highlights HeuRAL-MATE's adaptability to dynamic variations in task management and its effective handling of substantial volumes of online orders, exhibiting a total cost improvement of 8.58% and 10.74% compared to the brute-force optimal approach and FIFO heuristics, respectively.

To comprehend the superiority of HeuRAL-MATE over other baselines, we scrutinize the distinct reward components of all approaches as outlined in Table 2 for the first five test scenarios from Table 1. The total cost (reward) comprises two primary elements: (a) TRTO, which focuses on reducing robot travel distance, and (b) TTGT, which prioritizes minimizing task execution delays. As evident from Table 2, the brute-force optimal approach excels (lower value) in the TRTO component compared to HeuRAL-MATE, as the brute-force optimal baseline employs brute-force calculations to choose robot-task pairs that minimize total travel distance for executing LA tasks. However, due to its lack of emphasis on minimizing delays for existing LA tasks, the brute-force optimal approach has a notably higher TTGT component. Conversely, FIFO, despite its sequential task assignment, might still not effectively reduce the TTGT component. This is because assigned task endpoints could be distant from the starting locations of subsequent tasks in the LA, potentially leading to inherent execution delays. Notably, the TRTO component is bound to be larger than the brute-force optimal approach.

Variable number of robots: To evaluate the generalizability of HeuRAL-MATE, we conduct experiments involving varying numbers of robots and tasks during inference. Initially, our model is trained for a fleet comprising 20 robots, and we present the results of this training in Table 3. Subsequently, we assess the performance of the trained RL agent in scenarios where only 15 robots are available without having to retrain the model. This is readily achieved by artificially assigning extremely large values to the parameter t_{wait} for the additional 5 robots, effectively preventing any task allocation to these 5 robots. This approach enables us to generalize our framework to accommodate various robot quantities without the need for model retraining. The results for the 15-robot scenarios are presented in Table 4. As anticipated, executing the same task set with fewer robots incurs additional costs. Nevertheless, HeuRAL-MATE consistently outperforms baseline approaches in all scenarios.

Variable number of tasks: We now evaluate the HeuRAL-MATE agent trained for a fleet of 10 robots and 505 tasks (Table 1) on variable number of tasks - (a) 255 tasks (see Table 5), and (b) 1005 tasks (see Table 6). It can be seen that HeuRAL-MATE consistently outperforms the baselines for variable number of tasks without requiring model retraining.

Despite common intuition, brute-force optimal approach is one of the strongest baselines which, given the causal state of the environment, evaluates all possible task-robot pairs and optimally selects the suitable pair for subsequent execution. As such, this approach is computationally exhaustive and does not scale well with the number of robots or LA length. Moreover, the approach is greedily optimal and it is difficult for most algorithms to outperform it. The reason why HeuRAL-MATE is able to outperform it is due to non-adherence of brute-force optimal in ensuring that tasks do not remain in the LA beyond a threshold which results in a penalty, and the fact that HeuRAL-MATE exploits the underlying distribution defining task generation to plan for tasks to appear in future despite it having access to the same causal information as brute-force optimal.

6 Conclusion, Limitations and Future Work

This study introduces HeuRAL-MATE, a novel heuristic-guided RL framework, aimed at improving multi-robot task allocation in modern warehousing. The approach optimizes operational costs and efficiency, addressing the increasing demands of online orders. Through an optimal sequential task selection process, coupled with a robot selection heuristic and a collision-free navigation algorithm, HeuRAL-MATE significantly outperforms baseline methods across diverse test datasets. In future, we plan to integrate a learning-based cooperative path-planning framework into decision-making.

While our HeuRAL-MATE algorithm presents a promising approach, it is essential to acknowledge certain limitations that warrant attention in future research endeavors:

1. Single-Task Assumption: The algorithm currently assumes that robots are engaged in one task at a time, potentially limiting its applicability in scenarios where multitasking is prevalent.

2. No Contingency for Sudden Breakdowns: The model does not account for the sudden breakdown of a robot during operation. Once a task is allocated, our algorithm lacks the capability to reconsider or revert the decision in the event of an unforeseen robot malfunction.

3. Homogeneous Robots: We have made the simplifying assumption that all robots in the system are homogeneous, sharing identical values of characteristics such as velocity, acceleration, and load capacity. This assumption may not reflect the diversity present in real-world robot fleets.

4. Negligible Load/Unload Time Assumption: The algorithm assumes negligible time for loading/unloading operations after a robot reaches the task origin/destination. In reality, this may not hold true, and accounting for realistic loading and unloading times is a consideration for future enhancements.

5. Lack of Real World Environment Evaluation: Our algorithm's performance has been evaluated exclusively in simulated environments. The absence of validation in real-world settings, considering all physical constraints that robots may encounter, is a notable limitation.

Addressing these limitations in future iterations of our algorithm will contribute to a more robust and applicable solution in diverse operational scenarios.

References

[1] A. Agrawal, S. Hariharan, A. S. Bedi, and D. Manocha. Dc-mrta: Decentralized multi-robot task allocation and navigation in complex environments. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11711–11718. IEEE, 2022.

[2] A. Agrawal, A. S. Bedi, and D. Manocha. Rtaw: An attention inspired reinforcement learning method for multi-robot task allocation in warehouse environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1393–1399. IEEE, 2023.

[3] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Distributed autonomous robotic systems: The 10th international symposium*, pages 203–216. Springer, 2013.

[4] A. Bolu and Ö. Korçak. Adaptive task planning for multi-robot smart warehouse. *IEEE Access*, 9:27346–27358, 2021.

[5] Y. Chen, U. Rosolia, and A. D. Ames. Decentralized task and path planning for multi-robot systems. *IEEE Robotics and Automation Letters*, 6 (3):4337–4344, 2021.

[6] A. Dutta, S. Roy, O. P. Kreidl, and L. Bölöni. Multi-robot information gathering for precision agriculture: Current state, scope, and challenges. *IEEE Access*, 9:161416–161430, 2021.

[7] A. Farinelli, E. Zanotto, E. Pagello, et al. Advanced approaches for multi-robot coordination in logistic scenarios. *Robotics and Autonomous Systems*, 90:34–44, 2017.

[8] Y. Fujita, P. Nagarajan, T. Kataoka, and T. Ishikawa. Chainerrl: A deep reinforcement learning library. *Journal of Machine Learning Research*, 22(77):1–14, 2021. URL http://jmlr.org/papers/v22/20-376.html.

[9] B. P. Gerkey and M. J. Mataric. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *2003 IEEE international conference on robotics and automation (Cat. No. 03CH37422)*, volume 3, pages 3862–3868. IEEE, 2003.

[10] M. Gini. Multi-robot allocation of tasks with temporal and ordering constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[11] R. Hassin and A. Nathaniel. Self-selected task allocation. *Manufacturing & Service Operations Management*, 23(6):1669–1682, 2021.

[12] G. Humbert, M.-T. Pham, X. Brun, M. Guillemot, and D. Noterman. Comparative analysis of pick & place strategies for a multi-robot application. In *2015 IEEE 20th conference on emerging technologies & factory automation (ETFA)*, pages 1–8. IEEE, 2015.

[13] B. Kartal, E. Nunes, J. Godoy, and M. Gini. Monte carlo tree search for multi-robot task allocation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[14] A. Khamis, A. Hussein, and A. Elmogy. Multi-robot task allocation: A review of the state-of-the-art. *Cooperative robots and sensor networks 2015*, pages 31–51, 2015.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] M. Liu, H. Ma, J. Li, and S. Koenig. Task and path planning for multi-agent pickup and delivery. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

[17] H. Ma, J. Li, T. Kumar, and S. Koenig. Lifelong multi-agent path finding for online pickup and delivery tasks. *arXiv preprint arXiv:1705.10868*, 2017.

[18] K.-C. Ma, Z. Ma, L. Liu, and G. S. Sukhatme. Multi-robot informative and adaptive planning for persistent environmental monitoring. In *Distributed Autonomous Robotic Systems: The 13th International Symposium*, pages 285–298. Springer, 2018.

[19] A. R. Mosteo and L. Montano. Comparative experiments on optimization criteria and algorithms for auction based multi-robot task allocation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3345–3350. IEEE, 2007.

[20] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021. URL http://arxiv.org/abs/2006.07869.

[21] M. L. Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[22] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *Ieee Access*, 8:191617–191643, 2020.

[23] O. Salzman and R. Stern. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1711–1715, 2020.

[24] G. Sartoretti, Y. Wu, W. Paivine, T. S. Kumar, S. Koenig, and H. Choset. Distributed reinforcement learning for multi-robot decentralized collective construction. In *Distributed Autonomous Robotic Systems: The 14th International Symposium*, pages 35–49. Springer, 2019.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[26] D. Silver. Cooperative pathfinding. In *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, volume 1, pages 117–122, 2005.

[27] C. Wei, Z. Ji, and B. Cai. Particle swarm optimization for cooperative multi-robot task allocation: a multi-objective approach. *IEEE Robotics and Automation Letters*, 5(2):2530–2537, 2020.

[28] S. Wilson, P. Glotfelter, S. Mayya, G. Notomista, Y. Emam, X. Cai, and M. Egerstedt. The robotarium: Automation of a remotely accessible, multi-robot testbed. *IEEE Robotics and Automation Letters*, 6(2):2922–2929, 2021.

[29] Q. Xu, J. Li, S. Koenig, and H. Ma. Multi-goal multi-agent pickup and delivery. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9964–9971. IEEE, 2022.

[30] Y. Yang, L. Juntao, and P. Lingling. Multi-robot path planning based on a deep reinforcement learning dqn algorithm. *CAAI Transactions on Intelligence Technology*, 5(3):177–183, 2020.