

Flexible material handling system for multi-load autonomous mobile robots in manufacturing environments: a hierarchical reinforcement learning approach

Keonwoo Park, Seongbae Jo, Youngchul Shin & Ilkyeong Moon

To cite this article: Keonwoo Park, Seongbae Jo, Youngchul Shin & Ilkyeong Moon (2025) Flexible material handling system for multi-load autonomous mobile robots in manufacturing environments: a hierarchical reinforcement learning approach, International Journal of Production Research, 63:15, 5671-5691, DOI: [10.1080/00207543.2025.2461131](https://doi.org/10.1080/00207543.2025.2461131)

To link to this article: <https://doi.org/10.1080/00207543.2025.2461131>



Published online: 06 Feb 2025.



Submit your article to this journal [↗](#)



Article views: 359




View related articles [↗](#)



View Crossmark data [↗](#)



Flexible material handling system for multi-load autonomous mobile robots in manufacturing environments: a hierarchical reinforcement learning approach

Keonwoo Park^a, Seongbae Jo^a, Youngchul Shin^b and Ilkyeong Moon ^{a,c}

^aDepartment of Industrial Engineering, Seoul National University, Seoul, Republic of Korea; ^bDepartment of Industrial Engineering, Ajou University, Suwon-si, Gyeonggi-do, Republic of Korea; ^cInstitute of Engineering Research, Seoul National University, Seoul, Republic of Korea

ABSTRACT

The increasing complexity of customer demands has led to the implementation of flexible job-shop scheduling and automated material handling systems across manufacturing sectors. In particular, advances in robotic technology have made autonomous mobile robots (AMRs) essential for material handling tasks within these sectors. The capability of free movement, path planning, and loading multiple work-in-processes (WIPs) can significantly enhance the efficiency of material handling operations. However, the full flexibility of AMRs cannot be utilised when their decisions regarding the sequence of loading and unloading multiple WIPs are made by specific rule-based operations, resulting in inefficiencies in the throughput of WIPs in manufacturing environments. To address this inefficiency, we introduce a hierarchical reinforcement learning algorithm to optimise material handling with AMRs, thereby maximising the throughput of WIPs. In this approach, a graph attention network (GAT) serves as an encoder for the hierarchical reinforcement learning (HRL) input, effectively capturing the complex relationships between different nodes. Computational experiments demonstrate that our approach enhances the efficiency of the material handling system more effectively than existing rule-based methods.

ARTICLE HISTORY

Received 6 August 2024
Accepted 18 January 2025

KEYWORDS

Autonomous mobile robot; automated material handling system; hierarchical reinforcement learning; graph attention network; smart logistics

SUSTAINABLE DEVELOPMENT GOALS

SDG 9; Industry; innovation and infrastructure

1. Introduction

Autonomous mobile robots (AMRs) are emerging as pivotal factors in driving significant changes across various industries, especially in the realm of automation. AMRs proficiently handle and transport work-in-processes (WIPs) between different production and storage areas. This capability substantially reduces the manual labour and time associated with material handling, thereby promoting operational continuity. The ability of AMRs to autonomously navigate makes them highly adaptable to various modern manufacturing environments. Their flexibility is increasingly crucial as customer demands grow more diverse and complex, leading to the adoption of flexible job-shop scheduling systems in many manufacturing environments. These systems organise groups of machines or workstations for similar production processes, offering benefits like easy modifications and expanded capacity. However, the flexible job-shop scheduling system complicates production planning and material handling management. In such environments, the versatility of AMRs in handling WIPs is vital for

enhancing production efficiency. They enable just-in-time movement of WIPs, ensuring adherence to production schedules. Consequently, AMRs are increasingly becoming a critical component in modern factories that manufacture a diverse range of products in limited quantities, emphasising their significant role in contemporary manufacturing practices.

Unlike automated guided vehicles (AGVs) or overhead hoist transport (OHT) vehicles that move on fixed tracks or predetermined paths, marked by elements like radio frequency identification (RFID) tags or ceiling-mounted rails, AMRs navigate autonomously through their environment, offering significant flexibility. This means that AMRs provide a more advanced solution for a material handling system compared to traditional systems, as they can freely move without being confined to specific routes. Equipped with sensors, 3D cameras, and advanced laser scanning technologies, AMRs are highly efficient in complex environments. They are designed to handle unexpected obstacles smoothly by employing smart navigation strategies. For instance, when they

come across something in their way, they can slow down, pause, or find another path, using collision avoidance techniques. This ability gives AMR greater operational flexibility compared to AGVs or OHTs, which are limited to their fixed paths.

Another key advantage of AMRs is the ease they offer in scaling up operations. Increasing the number of AMRs in a facility is relatively straightforward, as they do not require the installation of specialised travel routes or depend on predefined paths. This scalability is a major benefit, especially in dynamic and evolving manufacturing settings. To summarise, the benefits of AMRs include their autonomous navigation capabilities, their adaptability to complex environments, their effective obstacle management, and the ease they offer in expanding their numbers in a facility. These advantages make AMRs a valuable asset in modern manufacturing and logistics operations, offering a significant improvement over more traditional material handling systems like AGVs and OHT vehicles. The benefits of AMRs can be summarised as follows:

- AMRs' ability to navigate and dynamically reroute makes them versatile and adaptable to changes in the work environment.
- The number of AMRs can be easily scaled up or down to align with the fluctuating material handling demands of various periods.
- AMRs offer easy material handling redeployment, regardless of changes in production lines or work spaces.

While AMRs provide superior operational flexibility compared to AGVs or OHT vehicles, they introduce higher complexity when optimising a system to operate efficiently in terms of throughput maximisation. Therefore, in practical applications, material handling with AMRs is often operated by rule-based heuristics. For instance, when considering an AMR with a load capacity of three, the following operational logic could be considered:

- (1) If the task of transporting WIP from specific equipment to other equipment arises, an idle AMR is assigned, and the AMR moves to load the WIP.
- (2) The AMR loads additional WIPs while moving to the nearest production equipment until it reaches its maximum load capacity of three.
- (3) Once the AMR has accumulated three WIPs, it transports them to their respective destinations. The transportation order of these WIPs can be determined based on either the longest queue time or the minimum travel distance.

The above operation method, while intuitive and having low computational requirements, has a significant limitation: It fails to fully leverage the flexibility of AMRs, particularly in establishing flexible movement routes and in loading multiple WIPs simultaneously. To fully utilise the capability of AMRs, decisions regarding the handling of WIPs should be made adaptively, rather than by relying on simplistic rule-based heuristics. This is the case because, multiple decision-making options arise at each juncture, such as the decision about whether to add additional loads immediately or to wait until current WIPs are transported. While segmenting the logic into more specific rules could be considered, such an approach would likely be ad-hoc and unable to adapt to the system's dynamically changing conditions.

To overcome the limitations of such operation methods, the focus is on enhancing the efficiency of AMR operations within manufacturing environments. The goal is to maximise the material handling performance of AMRs through the application of hierarchical reinforcement learning (HRL). Unlike simplistic, rule-based heuristics, our proposed HRL enables AMRs to analyze and learn from their environment, making real-time decisions that adapt to changing conditions and demands from the data. This not only improves efficiency but also increases the overall effectiveness of AMRs in diverse and dynamic manufacturing settings.

In this study, the production schedule is determined by a higher control system and then passed on to the next level of the control system for implementation. That is, our focus is on optimising the movement of WIP through the AMR control system. For a detailed explanation, please refer to Figure 1, which illustrates how AMRs are operated within a centralised system in a manufacturing environment. Once the manufacturing execution system (MES) establishes the production schedule, it is relayed to the material control system (MCS). The mobile robot control system (MRCS) then receives movement orders for WIPs from the MCS. At this point, the MRCS decides which tasks will be allocated to which AMRs and determines the path each AMR will take. Specifically, our study focuses on how the MRCS can efficiently manage material handling within a manufacturing setting. The main contributions of this study can be summarised as follows:

- The problem definition was extended beyond the dynamic pickup and delivery problem to a novel form that considers the dynamic nature of automated material handling systems. A Markov decision process was formulated to address such dynamic scenarios and to make sequential decisions efficiently. This approach

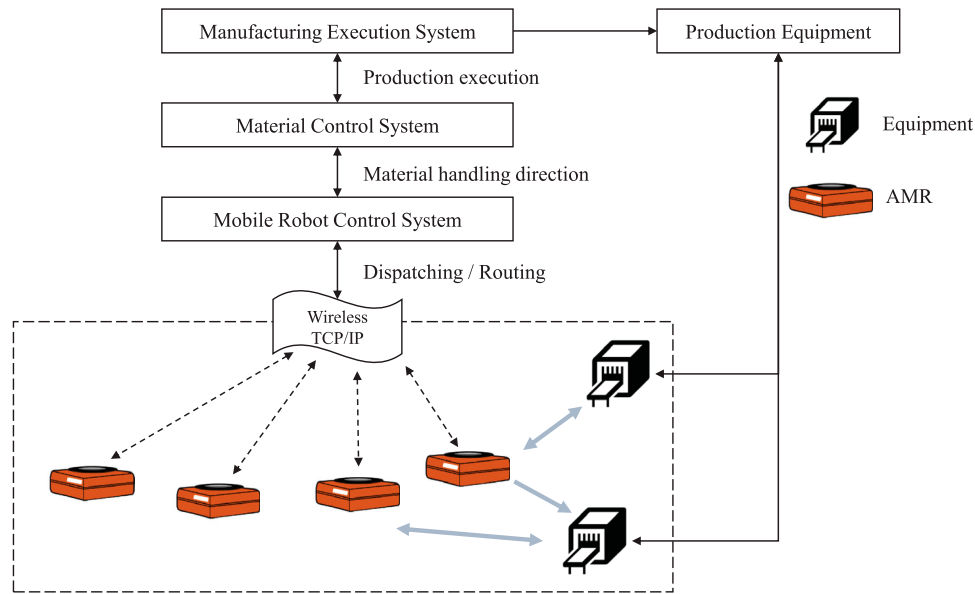


Figure 1. Schematic illustration of the system hierarchy for the material handling system of autonomous mobile robots.

enables us to allocate various AMRs while planning rational paths in real-time.

- A hierarchical deep reinforcement learning framework has been introduced as a solution for the material handling system. Utilising a hierarchical framework, two agents were introduced: a higher-level agent responsible for AMR allocation and a lower-level agent in charge of planning the paths for these allocated AMRs. Through this framework, we enable the accomplishment of the ultimate goal, which is maximising the throughput of AMRs.
- Through numerical experiments, the practical applicability of the proposed method was demonstrated. Furthermore, by comparing it with well-established techniques, we showed that our approach outperforms existing methods in terms of effectiveness and efficiency.

The remainder of this paper is structured as follows: Section 2 provides an overview of relevant literature. Section 3 details the problem description and the reinforcement framework employed in this study. Section 4 presents the results of our computational experiments. Finally, Section 5 concludes the paper by summarising the key findings of this research.

2. Literature review

In this section, two problems related to the study are introduced: dynamic flexible job-shop scheduling problems (DFJSPs) and dynamic pickup and delivery problems (DPDPs). We present a suitable problem

formulation and corresponding solution method for the AMR control system and highlight relevant studies.

Flexible job-shop scheduling problems (FJSPs) allow a job or operation to be processed by any machine in a given set of machines (Chaudhry and Khan 2016; Thörnblad et al. 2015). FJSPs can be divided into two subproblems, which are the assignment of operations to machines and the scheduling of the machines (Chaudhry and Khan 2016; Rossi and Dini 2007). During the assignment problem, each operation is assigned to one of the available machines. During the scheduling problem, the sequence of the operations assigned to each machine has to be determined. In dynamic FJSPs (DFJSPs), dynamic events such as random job arrivals and machine breakdowns can occur. Shen and Yao (2015) considered both dynamic job arrivals and machine breakdowns, and Shahgholi Zadeh, Katebi, and Doniavi (2019) considered the DFJSP with variable processing times. Baykasoğlu, Madenoğlu, and Hamzadayı (2020) compared the event-driven and the periodical rescheduling strategies under the dynamic environment with job cancellations, appearance of urgent jobs, and sequence-dependent setup times. In addition, various dynamic rescheduling frameworks have been proposed to deal with the dynamic events (Chien and Lan 2021; Fang et al. 2019; Fekih et al. 2020; Liu, Piplani, and Toro 2022).

The AMR control system can be formulated as a DFJSP because WIPs dynamically arrive, and they have to be assigned to AMRs in real time. Improvements of the system can be made by proposing a DFJSP incorporating transportation time and capacity of AMRs, as in Ren et al. (2022). However, such approaches can not fully capture the advantage of AMRs, which lie in their

ability to have flexible and unrestricted paths compared to AGVs and OHT vehicles. Therefore, in the next paragraph, we introduce another framework (i.e. DPDPs) that is considered more appropriate for this study. Additional papers that investigated AMR control systems from a scheduling perspective can be found in Fracapane et al. (2021).

In pickup and delivery problems (PDPs), transportation requests are satisfied using a fleet of vehicles that traverse routes constructed by a decision maker (Savelsbergh and Sol 1995). Given a set of pickup and delivery locations, every transportation request with a single origin and destination is also given before the construction of vehicle routes in static PDPs. However, in DPDPs, transportation requests can dynamically arrive (Berbeglia, Cordeau, and Laporte 2010). That is, while vehicles are handling previously generated transportation requests on predetermined routes, new transportation requests can be generated. Therefore, the decision maker has to dynamically design a pickup and delivery plan based on the observed transportation requests. Extensive literature exists that studies DPDPs in various types of dynamic events. Several papers introduced the problem in which requests are served by crowd-sourced drivers (Arslan et al. 2019; Behrendt, Savelsbergh, and Wang 2023; Simoni and Winkenbach 2023; Tao, Zhuo, and Lai 2023). In these studies, not only are delivery requests dynamically observed, but crowd-sourced drivers dynamically announce their availability. In addition, most of them considered time windows for the requests or the drivers. Some studies have explored DPDPs in the transportation services industry, as it can be characterised by a range of dynamic events. In Ferrucci and Bock (2014), dynamic traffic congestion and vehicle slowdowns are considered. Ulmer et al. (2021) studied the dynamic problem in which the time that the pre-ordered food can begin its delivery process is also dynamically realised. Their approach incorporates probabilistic information, which means that the probability distributions of time and locations for transportation requests are known in advance. There are also several studies that applied PDPs to manufacturing environments (Jun, Lee, and Yih 2021; Zou, Pan, and Wang 2021).

Transportation requests for WIPs are dynamically observed in the AMR control system. Although the number of AMRs is predetermined and remains constant, the sequence of assigned transportation requests of each AMR can be adjusted flexibly. Notably, AMRs have the capability to take over tasks already assigned to other AMRs and also to pass on their own tasks to different AMRs. By analyzing these characteristics of the system and by exploring the two aforementioned groups of previous studies, we concluded that DPDPs offer a

more suitable framework for addressing the complexities inherent in the AMR control system. Distinctive characteristics of the problem proposed in this study, as compared to the previous studies, can be summarised as follows:

- (1) Transportation requests with a single pickup or delivery node can be generated repetitively.
- (2) Pickup and delivery node sets are not distinctly defined (they are the same).
- (3) Except for the currently processing transportation requests, the real-time reordering of requests already assigned to each vehicle is allowed (both inner and outer).
- (4) There are five states of vehicles (AMRs): idle, retrieving, loading, transporting, and unloading.

Addressing the complexities and dynamic nature of pickup and delivery environments, numerous studies have leveraged reinforcement learning (RL) algorithms for DFJSPs and DPDPs (Chien and Lan 2021; Li et al. 2022; Liu, Piplani, and Toro 2022; Luo 2020; Zong et al. 2022). Additionally, various studies have actively explored the use of encoder and decoder structures, including the use of self-attention mechanisms, to solve complex combinatorial optimisation problems such as routing (Kool, Van Hoof, and Welling 2018; Xin et al. 2020). Further, research has been conducted on solving these problems by representing them as graphs. Veličković et al. (2017), for instance, introduced the graph attention network (GAT), which combines self-attention with graph structures and demonstrated superior performance. However, in the AMR control systems, the management of job assignment and AMR movements does not immediately yield observable results. This means that the rewards for RL agents are not given right after they execute actions. For instance, when the objective is to minimise overall tardiness, the tardiness resulting from a job assignment becomes evident only upon the completion of the assigned job. To deal with the delayed reward issue in RL, Han et al. (2022) proposed an off-policy RL framework with a newly designed Q-function. By Kulkarni et al. (2016), it has been recognised that the hierarchical RL (HRL) framework can effectively address the issues from delayed and sparse reward in large-scale problems. In their framework, a top-level module and a lower-level module work at different time scales. The top-level agent provides a goal for the lower-level agent, and then the lower-level agent selects actions to reach the given goal. Our proposed approach is also developed within the context of the HRL framework, aligning with the many studies applying HRL to DPDPs or to DFJSPs. Although our focus

is on DPDPs, we introduce studies that employed HRL algorithms for both DPDPs and DFJSPs, contributing to a comprehensive understanding of HRL in the relevant domains. Ma et al. (2021) proposed an HRL framework to solve large-scale DPDPs. Their upper-level agent determines whether to release a subproblem with accumulated transportation requests in each period. And a lower-level agent assigns transportation requests to vehicles and arranges the route of each vehicle for the given subproblems, which are static versions of DPDPs. Liu, Piplani, and Toro (2022) proposed hierarchical and distributed architecture for DFJSPs. Their routing agent dynamically assigns jobs to any one of available machines. Then a sequencing agent corresponding to each machine decides the order in which jobs will be processed. This does not constitute a multi-agent setting, because the sequencing agents share network parameters. Lei et al. (2023) also employed HRL to solve large-scale DFJSPs. As in Ma et al. (2021), their higher-level agent decides whether to reschedule or not when a new job arrives. They constructed their lower-level agent by integrating the routing and sequencing agents as in Liu, Piplani, and Toro (2022). After reviewing the studies above, we concluded that the HRL framework is suitable for sequentially making two critical decisions, which are the assignment and the routing, in the large-scale AMR control system with a delayed reward.

In summary, the AMR control problem in the manufacturing environment can be formulated as DFJSPs or DPDPs. Extensive research considers various dynamic events and constraints in the DFJSPs and DPDPs literature. Because DPDPs can incorporate the flexibility and unrestricted paths of AMRs while DFJSPs can not, we focus on DPDPs. To deal with large-scale and complex environments with the delayed reward issue, our solution approach is based on the HRL framework. Our work contributes to the existing literature by proposing a novel DPDP formulation and a carefully designed solution approach to address the complex characteristics of the AMR control system.

3. Problem formulation

3.1. Problem description

In this section, we describe the problem associated with using AMRs for transporting WIPs in manufacturing systems. The aim of this study is to develop the operational logic for AMRs within a centralised control system. This system is designed to dispatch WIPs to AMRs for transportation between production equipment in a job-shop manufacturing environment. With the production

schedule established by a higher control system, the focus of the MRCS is on maximising the throughput of WIP transportation through real-time movement commands. Our scope is, therefore, confined to managing these real-time WIP transfer commands and to directly controlling the sequences of AMRs.

We design our system around a decision-making cycle measured in seconds, meaning the control system updates its operational decisions within a few seconds using real-time data to determine the next actions of AMRs. Furthermore, we assume that AMRs operate on a grid-based map. This approach contrasts with the predefined movement paths of AGVs and OHT vehicle systems, which typically use a graph consisting of nodes and arcs. The choice of grid-based operations was made to better exploit the capabilities of AMRs in generating flexible paths, thereby providing a more adaptable and dynamic solution for navigating the complexities of a job-shop manufacturing environment. That is, our study aims to demonstrate how such a system can be significantly enhanced by our proposed HRL approach in real-world manufacturing settings.

To fully utilise the flexibility of AMRs, we consider operations involving multiple sources and destinations. This means that AMRs can load and unload WIPs at various locations, without necessarily reaching their maximum load capacity. For example, an AMR can pick up new WIPs without needing to unload all previously acquired ones. An AMR with a loading capacity of one WIP adheres to the following sequence for its status updates. Upon receiving a new WIP transfer command, the MRCS assigns the task to a suitable idle AMR, selecting it based on factors such as proximity and estimated travel time. The selected AMR's status then changes to *retrieving*, and it heads to the starting point of the WIP. Upon arrival, the status switches to *loading*, and the loading process begins. Once loaded, the status updates to *transporting*, and the AMR transports the WIP to its intended destination. There, the status changes to *unloading* for the unloading process. After completing these stages, the AMR's status reverts to *idle*.

Let us consider the case where an AMR's load capacity is two. Assume that the AMR has been assigned to *Job 1*, which arrived first, and that loading has been completed. When the new *Job 2* arrives, the AMR may have several options available. The AMR can either load *Job 2* after completing the transport of the currently loaded WIP to the destination, or it may load *Job 2* immediately, interrupting the transport of the currently loaded WIP. Figure 2 illustrates how the AMR operates in the latter case. The situation will be more complex if the loading capacity of the AMR exceeds two and the decision

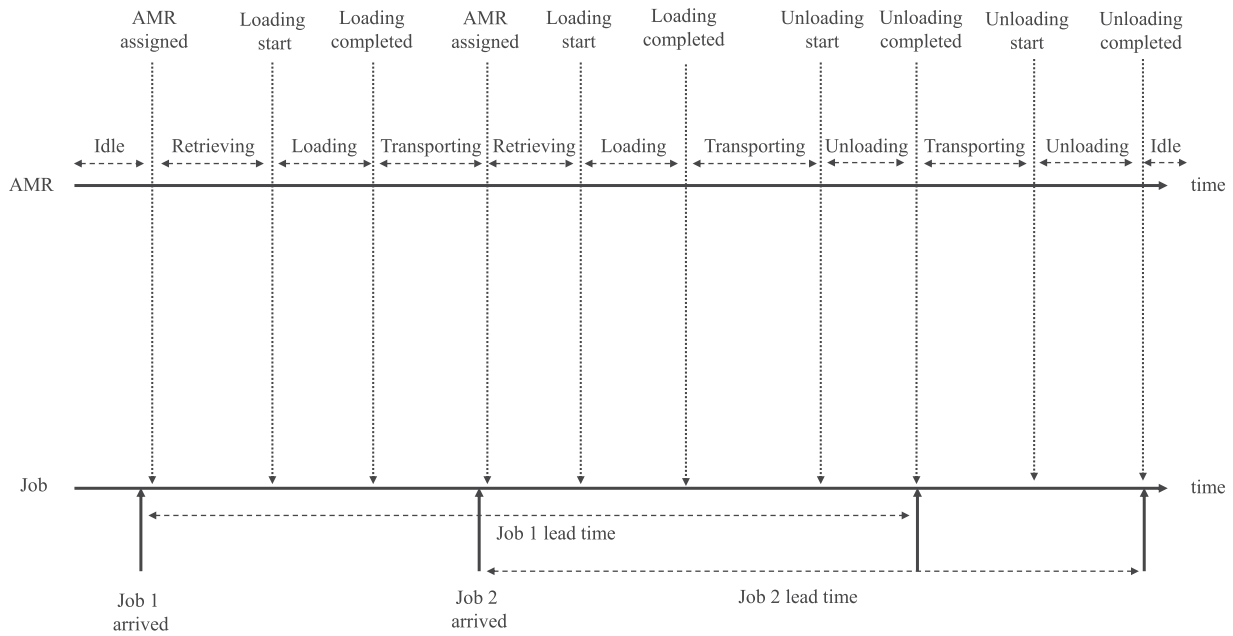


Figure 2. AMR status changes in response to job arrivals during operations when the load capacity is two.

is made as part of a system involving multiple AMRs operating simultaneously.

Let us consider a scenario in which an AMR arrives at the designated equipment. If all WIPs from this equipment are destined for the same location, loading the AMR to its full capacity before transportation might be more efficient. However, if the WIPs from the same equipment have different destinations, the *traveling salesman problem* can be used to determine the most efficient sequence of stops, thereby optimising the AMR's loading capacity. A potential drawback of this approach is the possibility of extended idle periods for an AMR while waiting for tasks to be completed on a single piece of equipment, which could diminish overall efficiency, especially when multiple AMRs are involved. In situations in which an AMR is tasked with loading from multiple equipment sources and with delivering to various destinations, the scenario resembles a *pickup and delivery problem*. However, this approach might not exploit the full flexibility of AMRs. Moreover, some WIPs could lead to situations in which they remain undelivered for an extended period. To overcome these challenges, we propose a data-driven real-time decision-making approach that aims to maximise the AMR's flexibility. This approach involves intelligently deciding when to load or unload additional WIPs, even if the AMR's capacity is not fully utilised. We illustrate the job assignment process for AMRs in a manufacturing environment in the Appendix 2.

3.2. Hierarchical framework

In this paper, we present a decision-making framework that dispatches the most suitable AMR at the moment a task is issued and conducts path planning for each robot based on this dispatch. Decision-making on a per-second basis is facilitated by conceptualising a decision horizon, denoted as T . At each decision point within this horizon, either the dispatching of robots or the determination of path planning for a robot is executed. To address these challenges, we propose a hierarchical reinforcement learning framework, comprising higher-level and lower-level agents, as depicted in Figure 3. As mentioned in Figure 1, MRCS, which receives instructions for delivery tasks from MCS, efficiently manages the tasks of each AMR. MRCS communicates wirelessly with all AMRs and can adjust plans in real-time. In this framework, the higher-level agent determines which AMR to dispatch by analyzing information from both the WIP buffer and the MRCS. If the higher-level agent determines that it is not optimal to dispatch any AMR, or if all AMRs have reached their full capacity, the WIP will not be assigned and will remain in the WIP buffer, waiting for one timestep. Once dispatched, the AMR proceeds toward its assigned task, during which it communicates its status to the MRCS. The MRCS then relays updated information back to the AMR, which interacts with the lower-level agent to adjust its path planning dynamically. Through this interaction, centralised management of path planning for all AMRs is effectively achieved. The lower-level agent, informed by the dispatch status of

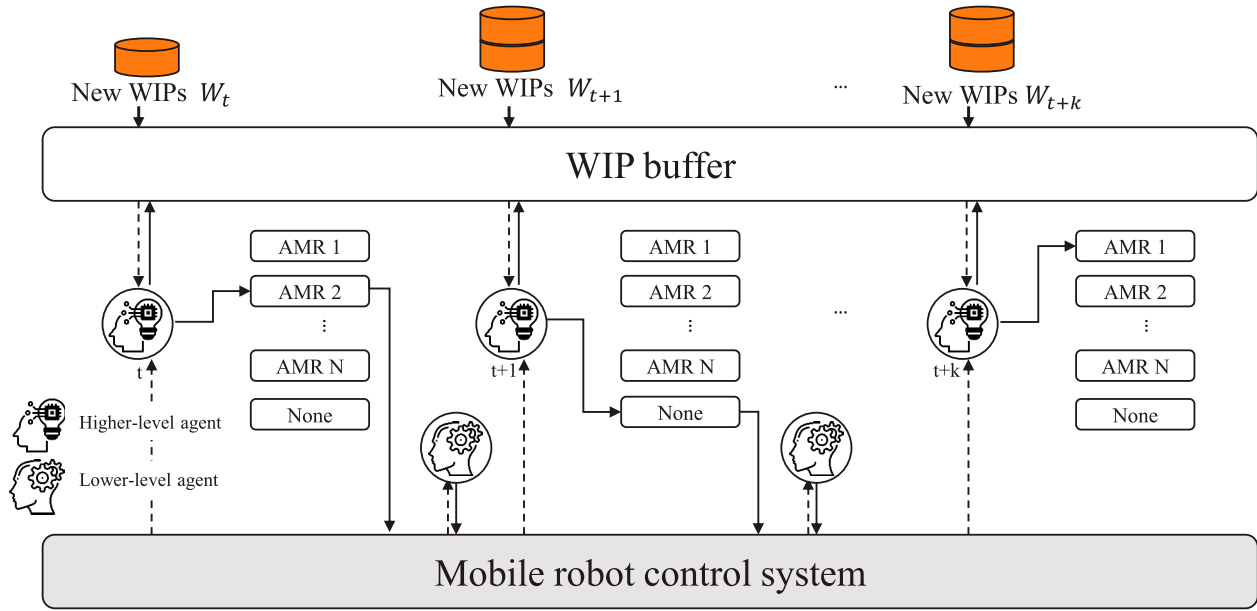


Figure 3. Illustration of hierarchical reinforcement learning framework.

WIPs for the robots from the higher-level agent, decides on dynamic and flexible path planning in a pickup and delivery environment at every moment.

In utilising AMRs, two of the most crucial challenges are dispatching and path planning. By considering not only the immediate operational status but also the overall context of the system, our proposed hierarchical framework enables decision-making that is aimed at maximising overall return. This separation of responsibilities allows the overall system return to be maximised by focussing on long-term task assignment through the higher-level agent and immediate operational efficiency through the lower-level agent. The higher-level agent ensures that AMRs are assigned in a way that minimises overall workload imbalance and analyzes the current situation, thereby preventing likely points of congestion before they occur. Meanwhile, the lower-level agent's real-time path planning ensures that the tasks are executed as efficiently as possible, considering immediate constraints such as AMR availability, current workload, and proximity to WIPs. This dual approach not only reduces idle time but also enables an efficient distribution of workload across all AMRs, ultimately maximising the throughput of WIPs and minimising the total delivery time.

The efficacy of this hierarchical framework is further enhanced by the integration of real-time data analytics, which play a crucial role in both the decision-making processes of the higher-level and lower-level agents. This integration allows for a more responsive and adaptive system, capable of adjusting to changes in the manufacturing environment, such as variations in

WIP demand, equipment capacity, and congestion in vehicle operations. The higher-level agent continuously analyzes trends and patterns from historical and real-time data, enabling it to make more informed decisions about robot dispatch. The higher-level agent continuously monitors real-time data from the MRCS and the WIP buffer, analyzing the current situation to prevent congestion and enhance efficiency. Meanwhile, the lower-level agent focuses on the immediate execution of tasks, utilising the latest data to navigate and adapt to the dynamic environment. It employs advanced algorithms for path optimisation, considering factors such as the planned route and remaining capacity of each robot. This level of detailed management enables each robot to operate with enhanced efficiency, thereby contributing to a more streamlined and agile production process.

3.3. State encoding based on graph attention network

In this section, we introduce the GAT for encoding inputs used in HRL. Given the graph-structured nature of the problem, the approach combines raw state features with those processed by the GAT, as depicted in Figure 4. This concatenated representation serves as input for the HRL framework. To effectively train the GAT with dimensions suitable for its structure, we expand the dimensions to higher-level features using Equation (1):

$$h_t^k = \text{MLP}(v_t^k) = \mathbf{W}v_t^k + b \quad (1)$$

where v_t^k represents the state information of each robot k at each step t , $v_t^k \in \mathbb{R}^l$ and $h_t^k \in \mathbb{R}^p$ with $p > l$. Here, \mathbf{W}

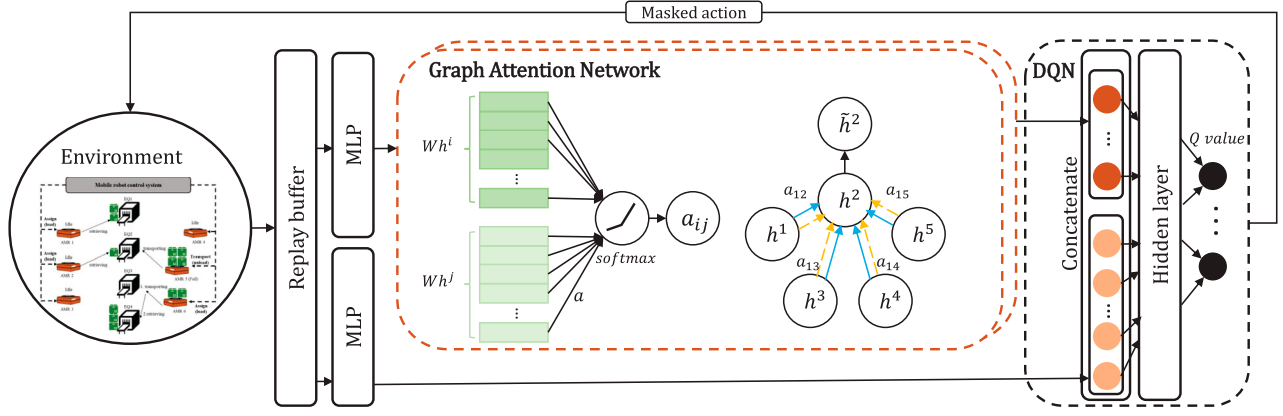


Figure 4. Illustration of graph attention network based hierarchical reinforcement learning framework.

represents the weight matrix and \mathbf{b} represents the bias vector. This transformation enables the network to map the input features to a higher-dimensional space, thereby capturing more intricate representations and interactions among the nodes.

The influence each node receives from its adjacent nodes is represented by the attention coefficient e_{kj} . Based on Veličković et al. (2017) and Shao et al. (2021) studies, the GAT structure is illustrated in Figure 4. The attention coefficient e_{kj} is given by Equation (2):

$$e_{kj} = \text{att}(\mathbf{W}h_t^k, \mathbf{W}h_t^j) \quad (2)$$

where the self-attention mechanism att is introduced. To ensure that the scalar value derived from this equation is influenced only by adjacent nodes, we use the softmax function as shown in Equation (3):

$$a_{kj} = \text{softmax}(e_{kj}) = \frac{\exp(e_{kj})}{\sum_{j \in \mathcal{L}_k} \exp(e_{kj})} \quad (3)$$

where \mathcal{L}_k denotes the set of neighbouring nodes accessible from node k . The softmax function normalises the attention coefficients across all neighbouring nodes. This normalisation step is crucial for stabilising the training process and for ensuring that the attention mechanism is focussed on the most relevant neighbours.

Next, for normalisation during training, we implement Equation (4):

$$\tilde{h}_t^k = \text{ReLU} \left(\sum_{j \in \mathcal{L}_k} a_{kj} \mathbf{W}h_t^j \right) \quad (4)$$

To ensure stability during the training process, we extend h_t^k using a multi-head attention mechanism. By parallelly concatenating M independent attention mechanisms in parallel, Equation (5) is derived.

$$\tilde{h}_t^k = \parallel_{m=1}^M \text{ReLU} \left(\sum_{j \in \mathcal{L}_k} \alpha_{kj}^m \mathbf{W}^m h_t^j \right) \quad (5)$$

Through this approach, we obtain stabilised results for input values in higher dimensions. In this experiment, we used two convolutional layers with four attention heads each.

3.4. Higher-level agent

As introduced in Figure 1, the MRCS receives WIP information from the higher control system. The timestep in a finite horizon is set to be t ($t \in \{1, \dots, T\}$), with each interval set to one second. As depicted in Figure 3, upon receiving WIP data from the MCS, the higher-level agent decides which AMR to assign the WIP to, based on the outcomes of its training. At each timestep, it is assumed that a maximum of one WIP can be assigned to a robot. At timestep t , the higher-level agent selects the most suitable AMR for the given WIP. This information is then transmitted to the mobile robot control system's environment, where it interacts with the lower-level agent, and the interaction results are conveyed to the next timestep, $t + 1$. At timestep $t + 1$, if all robots are in *loading* or *unloading*, or exceed their maximum capacity C , an action of *None* is inevitably enforced by action masking, even if there is a WIP to be assigned. Additionally, the agent may also choose not to take any action, despite the availability of AMRs to assign, in order to preserve the solution for finding a better outcome. A Markov decision process (MDP) for implementing the dispatching procedure is formulated as follows.

State: In the MDP formulation for the higher-level agent, each state is represented as U_t . At each timestep, the state, representing the status of the robot, is expressed as $U_t = \{v_t^1, v_t^2, \dots, v_t^k\} = \{(c_t^1, p_t^1, d_t^1, o_t^1), (c_t^2, p_t^2, d_t^2, o_t^2), \dots, (c_t^k, p_t^k, d_t^k, o_t^k)\}$ where k represents the number of

robots, with k being an element of the set $\{1, \dots, K\}$. This state includes information such as the current load, c_t , the type of current task, p_t , the total distance of the route that the robot belongs to, d_t , and the categorised distance to the earliest destination, o_t , for each robot, v^k at each step t . All these features are concatenated with the output features of the GAT, resulting in $s_t = (\tilde{h}_t, U_t)$

Action: The action a_t consists of assigning a WIP to a mobile robot or taking no action at all. Therefore, the number of actions a_t is $|K| + 1$, and in each timestep, only one robot is selected.

Reward: While the overall objective of the agents, as previously mentioned, is to maximise the throughput of WIPs, for the effective learning of each agent, we propose a new immediate reward structure. To achieve the higher-level agent's goal of efficient AMR assignment, we have set the reward as follows:

$$r_{H_t} = -\lambda_t \cdot \alpha - \psi_t \cdot \beta + \delta_t \cdot \eta$$

Where λ_t is the standard deviation of the number of robot queues at timestep t , ψ_t is the standard deviation of the estimated distance of robots at timestep t , and δ_t is the number of assigned WIPs at timestep t . The parameters α , β , and η were utilised in this study, with their optimal values being established through extensive experimental work with respect to the environment.

The Q-learning technique, a prominent method in reinforcement learning, updates the Q-value for a specific state-action pair, as shown in Equation (6).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)] \quad (6)$$

Nonetheless, there are challenges in updating the Q-table, especially when dealing with high-dimensional state inputs or action spaces. As noted in Mnih et al. (2015), deep Q-networks are efficient in addressing problems with complex state and action spaces. By approximating the Q function as $Q(s, a, \theta) \approx Q_\pi(s, a)$ with the parameter θ , the estimation of the value function becomes more efficient.

$$L(\theta) = \mathbb{E} \left[\left(R_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta') - Q(s_t, a_t; \theta) \right)^2 \right] \quad (7)$$

Through stochastic gradient descent, it is feasible to optimise the parameter θ to minimise the loss function in the equation.

3.5. Lower-level agent

The lower-level agent, based on the AMR assignment results received from the preceding higher-level agent,

conveys decisions on how to conduct path planning for each AMR within the mobile robot control system. According to Ahamed et al. (2021), computing path planning for the total number of AMRs, denoted as $|K|$, significantly increases computational complexity, rendering it impractical. Therefore, in this study, path planning consists of a total of five actions: four path improvement actions considering the environment, and one action of inaction. The MDP formulation for the lower-level agent is as follows.

State: The state of the lower-level agent, denoted as $l_t = (U_t, E_t)$, is composed of two types of state information. The first state, U_t , is inherited from the state of the higher-level agent and consists of information about the robot. The second state, E_t , combines information about the robot and the state of the equipment, which are the destinations for transporting WIPs, expressed as $E_t = \{e_t^1, e_t^2, \dots, e_t^m\} = \{(b_t^1, i_t^1), (b_t^2, i_t^2), \dots, (b_t^m, i_t^m)\}$. This is concatenated with the coordinates of the equipment, denoted as b_t , and the number of robots designated to visit each equipment, represented as i_t for each equipment e^m at each step t . All these features are concatenated.

Action: Considering the dynamic and potentially expanding number of robots, it becomes evident that optimising the next destination node for each robot's visit poses a significant computational challenge. Given this context, in this paper we introduce a heuristic-based methodology for action selection. This approach not only maintains computational efficiency but also effectively enhances path planning strategies. The action framework includes five distinct types: *inner relocate*, *inner exchange*, *inter exchange*, *inter relocate*, and the option of selecting no action.

In the application of these actions aimed at refining solutions, it's essential to follow the specific constraints and characteristics inherent in the dynamic pickup and delivery problem. Adhering to these constraints, as outlined below, is vital for maintaining the integrity and feasibility of the proposed solutions.

- (1) The pickup sequence must always precede the delivery sequence for the same task.
- (2) The updated delivery task cannot be processed before the currently ongoing task.
- (3) If a task has already been picked up, it cannot be transferred to another robot.
- (4) Tasks cannot be transferred in a way that exceeds the robot's capacity.

In contrast to the higher-level agent, where the assignment of an AMR as an action directly impacts the environment, the lower-level agent is tasked with determining the appropriate path improvement methodology. Consequently, the implementation of the selected action

in the lower-level agent's context varies based on the specifics of the four distinct action types: inner relocate, inner exchange, inter relocate, and inter exchange. These actions provide the lower-level agent with different strategies to either relocate or exchange tasks within or between robots, ensuring adaptive and efficient task planning. Further explanations of each action type can be found in Appendix 1.

Reward: The ultimate goal of the lower-level agent is to efficiently plan the paths of robots, thereby minimizing the total path length required for AMR deliveries. To minimise the AMR paths while enabling the agent to effectively learn from the immediate outcomes of actions, we have constructed the reward as the difference between the current timestep t and the previous timestep $t-1$, defined as $r_{L_t} = Z_{t-1} - Z_t$, where $Z_t = \sum_{k \in K} d_t^k$.

Based on the hierarchical framework and MDP setting previously presented, we have illustrated the HRL algorithm for the material handling system in Algorithm 1. We have introduced both a higher-level agent (A_H) and a lower-level agent (A_L) that operate based on the necessary parameters, with both agents employing the Deep Q-Network (DQN) algorithm in their functioning. Note that at each moment of action selection, there exists an action mask procedure for preventing the selection of infeasible actions, which is implemented individually for each agent in lines 10 and 14. Each agent undergoes iterative updates, and the operational outcomes of the higher-level agent are employed as input observations for the lower-level agent.

4. Computational experiments

4.1. Experiment settings

The experimental environment for testing the problem structure described in the previous Section 3 is as follows. In each episode, the experiment randomly generates the initial positions of the AMRs on the grid, and the occurrence of WIPs is uniformly distributed across all equipment. To reflect realistic conditions at the equipment, if more than ten WIPs occur at a specific equipment at the current system time, they are uniformly generated at other equipment. Experiments on the environment were performed using the dataset listed in Table 1. The experiment was conducted on a Python 3.8 with an AMD Ryzen 5 5600G processor with 16GB RAM. The parameters of the model and their respective values are presented in Table 2.

4.2. Training performance and comparative analysis

4.2.1. Training convergence

The convergence of our proposed HRL algorithm is demonstrated through the training results on a 10 ×

Algorithm 1 Hierarchical Reinforcement Learning Algorithm for Material Handling System

Input: Initial parameters for agents (θ_1, θ_2), experience replay buffers (D_1, D_2), environment details, action masks.

Output: Learned policies (A_H, A_L), final parameters (θ_1, θ_2), cumulative rewards.

- 1: Initialize the higher-level agent A_H with experience replay memory D_1 and parameters θ_1 .
 - 2: Initialize the lower-level agent A_L with experience replay memory D_2 and parameters θ_2 .
 - 3: **for** $i = 1$ to N_{episodes} **do**
 - 4: Reset the material handling environment and obtain initial observation o .
 - 5:
 - 6: **for** $t = 1$ to T **do**
 - 7: Map the input to high dimensions and pass it through the GAT network to obtain a new input feature h_t .
 - 8:
 - 9: Concatenate h_t and U_t to form a new state s_t .
 - 10: Select an AMR selection action a_t^H using A_H based on observation s_t and an action mask considering the state of AMR.
 - 11:
 - 12: Apply action a_t^H , observe new state s_{t+1} , reward r_t^H .
 - 13:
 - 14: Store transition $(s_t, a_t^H, r_t^H, s_{t+1})$ in replay buffer D_1 .
 - 15:
 - 16: Set observation for lower-level agent l_t based on s_{t+1} .
 - 17:
 - 18: Select a path planning action a_t^L using A_L based on l_t and the action mask.
 - 19:
 - 20: Apply action a_t^L , observe the new state l_{t+1} , and reward r_t^L .
 - 21:
 - 22: Store transition $(l_t, a_t^L, r_t^L, l_{t+1})$ in replay buffer D_2 .
 - 23:
 - 24: Update observation for higher-level agent.
 - 25: Update A_H and A_L using Adam optimiser with parameters θ_1 and θ_2 , respectively.
 - 26:
 - 27: **end for**
 - 28: **if** $i \% 10 == 0$ **then**
 - 29: Update target networks of A_H and A_L .
 - 30: **end if**
 - 31: **end for**
-

10 grid with ten AMRs and four pieces of equipment. Figure 5(a,b) show the training curves for the higher-level and lower-level agents, respectively, while Figure 5(c) displays the results graph for the completed WIPs, which represents the ultimate goal of our system. Figure 5(d) illustrates the average travel distance of the robots.

Table 1. Definition of parameters.

Parameter settings for the simulator	Value
Number of equipment (m)	4, 10, 30
Number of AMRs (k)	8, 20, 30, 50
Maximum capacity of AMR (C)	4
Maximum robot queue	10
Maximum WIP queue	10
Size of grid (height equals width) ($size$)	10, 20, 30

Table 2. Parameter settings for the training.

Parameter	Value
Number of training episode	10,000
Number of timesteps in one episode	200, 300
Replay memory size	1,000,000
Minibatch size	256
Learning rate	0.0001
Discount factor	0.8
Optimiser	Adam

Next, a sensitivity analysis of the experimental results was conducted. Figure 6 shows the boxplots of the number of completed WIPs and the average travel distance of robots under different WIP arrival distributions. The mean values are indicated by circles. Experiments were conducted for scenarios following Uniform, Poisson, and Normal distributions. It was observed that as the probability of WIP generation or the mean value of the Poisson and Normal distribution increased, both the number of completed WIPs and the travel distance increased. However, it is noteworthy that the increase in WIP generation probability did not result in a proportional increase in these values due to the occurrence of congestion.

4.2.2. Comparative analysis with existing benchmarks

In this paper, we have developed various benchmark algorithms for comparative experiments, referencing and adapting from previous studies. For the purpose of comparison with our model, we present the following dispatching rules based on reinforcement learning algorithms and heuristic methods:

(a) IH rule

First, the heuristic denoted as IH is based on the work of Ahamed et al. (2021), and it operates as follows:

Step 1: Calculate the distance from the pickup equipment of the registered WIP in the system to all AMRs. For assigned robots, the total path of the robot is considered as the *distance*. For AMRs without assigned WIPs, the *distance* is calculated as the distance between the AMR and the equipment.

Step 2: For the robot with the smallest *distance*, if it already has an assignment, the new WIP is added at

the end of its current sequence. If not, a new WIP is assigned to it.

Step 3: Perform the intra-relocate action for the sequence of the assigned AMR, as proposed in this paper.

Step 4: If the robot with the smallest *distance* is not in a state to be assigned, assign the WIP to the robot with the next smallest distance.

(b) Dispatching rules

To demonstrate the effectiveness and practical applicability of our proposed HRL algorithm, we conducted a comparison experiment with existing dispatching rules developed for solving combinatorial optimisation problems similar to the one addressed in our study. These dispatching rules include FIFO, 2-OPT heuristic, closest selection (CS), least remaining processing time (LRT), and nearest neighbour (NN).

(c) Dispatching rule selection with reinforcement learning (DSRL)

The DSRL method combines various dispatching approaches to derive rational solutions. This method has been applied not only in recent reinforcement learning literature but also in various combinatorial optimisation fields. This paper is based on several prior studies, including the studies by Luo (2020) and by Lei et al. (2023). The MDP for DSRL is defined as follows:

State: The state information must encompass all details relevant to the assignment and path planning of AMRs. Therefore, it is identical to the state l_t of the lower-level agent, which includes information about both the AMRs and the equipment.

Action: The action consists of the top three performing dispatching rules, namely IH, CS+2-OPT, and CS+NN.

Reward: The reward is a concatenation of the rewards from the higher-level and lower-level agents, composed of the sum of r_t^H and r_t^L .

The benchmark algorithms mentioned above underwent a thorough comparative assessment with our proposed model, as presented in Table 3. All values listed in the table represent the average of 20 experiments, conducted to reflect the generality across various environments with instances randomly generated. In this experiment, four different cases are presented along with the respective experimental results. First, S1 presents an environment with a 10x10 grid where there are eight AMRs and four equipment. Next, S2 features a 20x20 grid with 20 AMRs and ten equipment, while S3 introduces a larger 30x30 grid with 30 AMRs and 30 equipment. For S4, an even larger-scale environment is

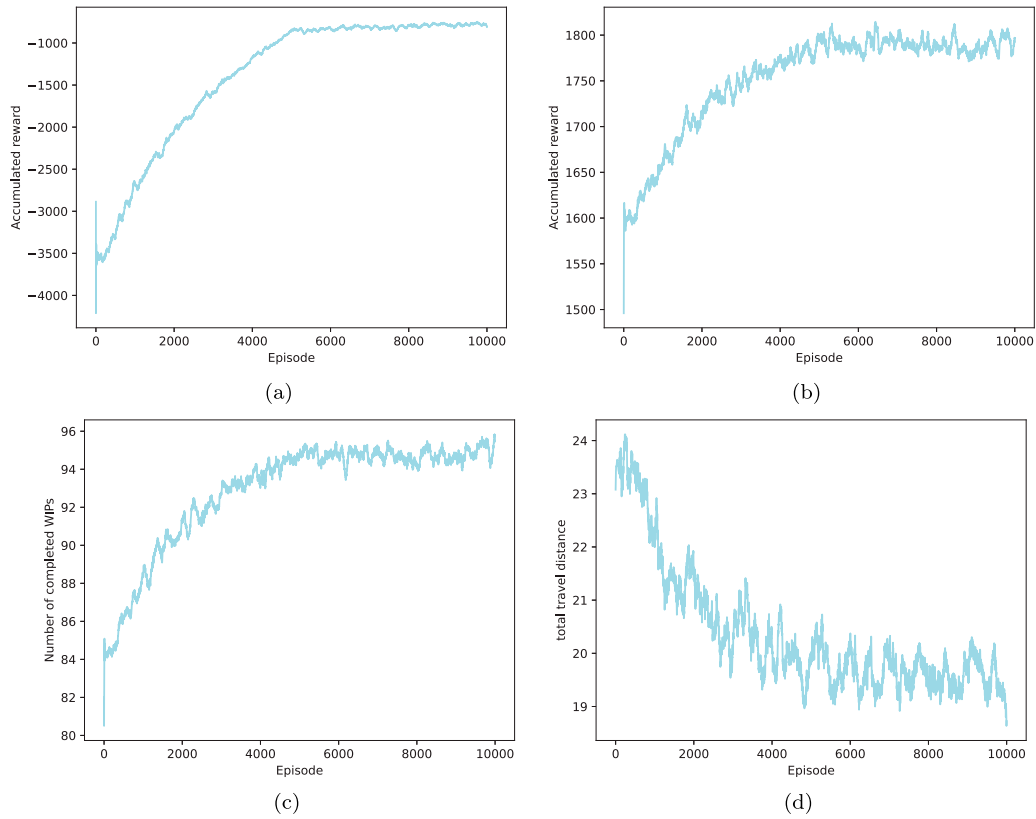


Figure 5. Training curves of the HRL algorithm. (a) Cumulative rewards of the higher-level agent during the training phase. (b) Cumulative rewards of the lower-level agent during the training phase. (c) Number of completed WIPs throughout the training phase. (d) Average travel distance of the robots.

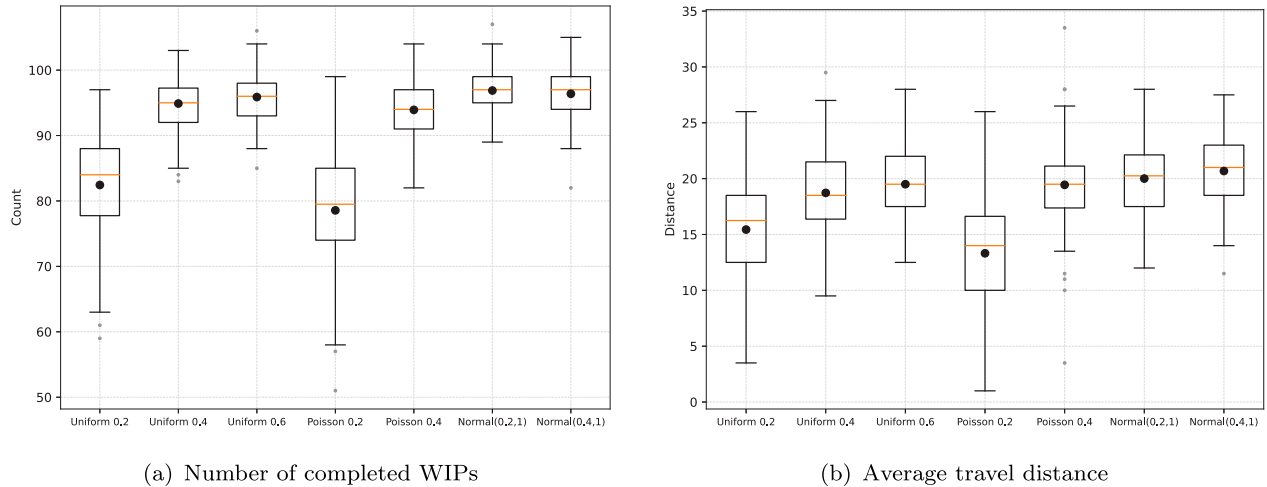


Figure 6. Comparison of completed WIPs and travel distances under different arrival probabilities. (a) Number of completed WIPs and (b) Average travel distance.

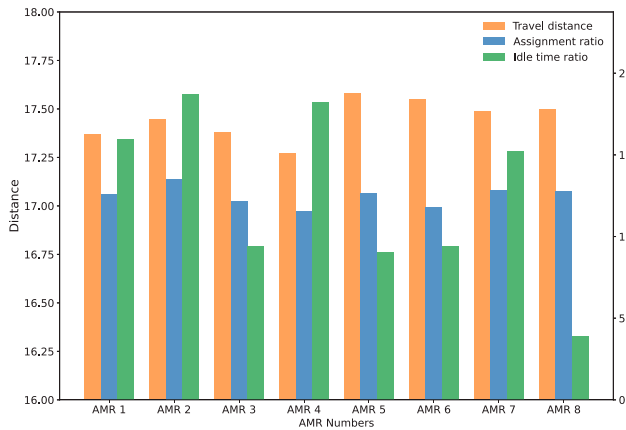
considered with a 30x30 grid, 50 AMRs, and 30 equipment. Note that, to ensure clarity in the experimental results, the timestep in S3 and S4 was extended to 300.

As demonstrated in Table 3, our proposed HRL algorithm outperforms all known algorithms and those developed from previous studies. Given that our research

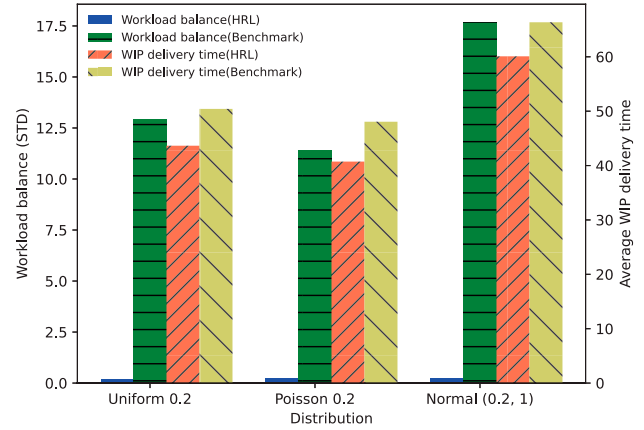
required rapid decision-making within a short time frame, we aimed for decision-making within approximately 10 seconds in smaller problem sizes like S1 and S2. Regardless of the problem size, the performance of our proposed algorithm not only surpassed other algorithms in all cases but also showed a significantly larger gap compared to existing heuristics as the problem size

Table 3. Comparison of HRL algorithm with baseline approaches.

Case		HRL	DSRL	IH	CS+FIFO	CS+2-OPT	CS+NN	LRT+2-OPT	LRT+NN
S1	Obj.	94.5	79.0	79.5	65.2	81.0	77.4	75.2	57.2
	Gap	0.0%	16.4%	15.8%	30.9%	14.3%	18.0%	20.4%	39.4%
	Time (s)	1.23	1.03	1.08	1.04	0.88	0.93	0.98	0.94
S2	Obj.	111.4	96.7	90.2	78.5	94.3	89.5	92.5	88.4
	Gap	0.0%	13.2%	19.0%	29.5%	15.4%	19.7%	17.0%	20.6%
	Time (s)	10.67	9.87	1.83	1.76	1.57	1.67	1.43	1.56
S3	Obj.	182.3	153.3	151.0	114.8	164.8	161.0	163.7	151.8
	Gap	0.0%	15.9%	17.2%	37.0%	9.6%	11.7%	10.2%	16.7%
	Time (s)	22.42	14.09	10.83	9.84	10.72	10.56	11.64	10.13
S4	Obj.	221.3	161.5	156.6	123.1	175.8	145.4	174.4	149.7
	Gap	0.0%	27.0%	29.2%	44.4%	20.5%	34.3%	21.2%	32.4%
	Time (s)	32.47	28.08	23.56	24.13	24.61	23.82	25.34	21.73



(a) AMR task allocation and performance



(b) Impact of WIP arrival distributions

Figure 7. Effectiveness of HRL algorithm on workload and delivery management. (a) AMR task allocation and performance and (b) Impact of WIP arrival distributions.

increased, while addressing the tasks within a practically feasible time frame.

The effectiveness of the HRL algorithm was explored by conducting additional experiments that evaluated performance in terms of workload distribution and WIP delivery time. Figure 7(a) illustrates how our HRL-based policy influenced task allocation across all AMRs, highlighting that while the proportion of idle tasks varied, the assignment of tasks and travel distances were evenly distributed across the robots. This balanced assignment is particularly important in minimising bottlenecks and in ensuring that no single AMR is overburdened.

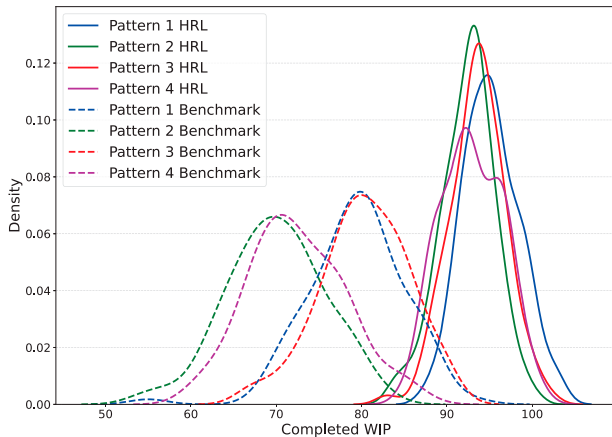
Figure 7(b), depicts the results under different WIP arrival distributions: Uniform, Poisson, and Normal, providing insights into both the workload balance and the average delivery time. Notably, our HRL algorithm consistently maintained an even workload distribution across these varied settings and reduced the average delivery time for WIPs compared to the best-performing heuristic (CS+2-OPT). These results emphasise the robustness and adaptability of the proposed HRL approach, highlighting its superior capability in

efficiently managing AMR resources even in dynamically changing environments.

Confirming the generality of our HRL algorithm in different environments, we varied the positions of the equipment generating WIPs, as depicted in Figure 8. The configurations labelled as Pattern 1 to 4 represent different setups: equipment arranged along the diagonal and opposite diagonal of the grid, as well as in square and diamond formations. In the figure, solid lines and dotted lines illustrate the distribution of the completed number of WIPs for the HRL and benchmark methods, respectively. The results show that the HRL's outcomes exhibit consistent distributions, indicating that it derived stable policy solutions while also demonstrating superior performance, whereas the benchmark showed varying distributions. These results demonstrate that our HRL algorithm consistently provided effective policy solutions, regardless of changes in equipment layout. However, questions may arise regarding the individual utility of each agent. To validate this query, the following subsection will outline our numerical experiment conducted to determine the effectiveness of each agent by fixing one and varying the other.

Table 4. Effectiveness of the higher-level agent.

Method	S1			S2			S3		
	Aver.	Gap	Time(s)	Aver.	Gap	Time(s)	Aver.	Gap	Time(s)
CS + Random insertion	59.3	37.3%	0.98	67.4	38.9%	1.47	96.2	45.4%	9.84
CS + Lower-level agent	79.1	16.3%	1.26	99.7	9.67%	9.21	130.6	25.9%	14.26
Our (Higher + Lower-level agent)	94.5	0.0%	1.13	111.4	0.0%	10.67	182.3	0.0%	22.42

**Figure 8.** Generality of HRL algorithm across different equipment layout patterns.

4.2.3. Evaluating the effectiveness of the higher-level agent

To demonstrate the performance of the higher-level agent, we compared our proposed model, which utilises both higher-level and lower-level agents, with two other algorithms that do not. For comparison, the CS heuristic was used, as it was identified as the best-performing robot selection rule in the previous comparative analysis. We compared this with two algorithms: one that randomly updates the path after selecting AMRs using the CS heuristic and another that utilises the results of the lower-level agent. As evident in Table 4, although our proposed model took the longest time, it showed the best performance, indicating the effectiveness of the higher-level agent in efficiently assigning robots. This result demonstrates that our model is effectively learning through the assignment of robots by the higher-level agent.

4.2.4. Assessing the impact of the lower-level agent

In assessing the impact of the lower-level agent, our approach involved conducting comparisons with two alternative algorithms, excluding the use of the lower-level agent. The 2-OPT heuristic, which showed superior performance in our previous path planning experiment, was implemented in these comparison algorithms. As shown in Table 5, the lower-level agent also exhibited outstanding performance. While the performance

gap was not as significant as that observed with the higher-level agent, the lower-level agent nonetheless contributed notably to path improvement, highlighting its effectiveness in enhancing the overall system's efficiency.

4.3. Managerial insights

We recommend the following instructions for practitioners in logistics automation and warehouse management who are seeking to apply the HRL-based algorithm effectively. These insights are derived from extensive numerical experiments conducted to validate the proposed model:

- The proposed HRL algorithm demonstrated remarkable scalability, making it applicable across a range of environments, from small warehouses to large distribution centres. Whether managing 10 AMRs or 50, our model consistently performed well, showcasing its ability to efficiently manage resources regardless of the scale. This adaptability makes the HRL approach suitable for industries aiming to automate processes without facing scalability issues. It assures warehouse managers that the model can grow along with their operational needs, providing consistent optimisation as more AMRs and equipment are introduced.
- The hierarchical decision-making framework in the HRL algorithm effectively addresses the complexities of task assignment and workload balancing. The experimental results indicated that the model can allocate tasks in a way that balances workloads across all AMRs, can reduce idle times, and can ensure efficient use of resources. This leads to smoother operations and increased throughput, particularly in high-demand situations. Managers can use these insights to avoid resource bottlenecks and to ensure more streamlined and efficient operations, which is especially critical in environments with frequent task requests and high movement levels.
- One of the key strengths of the HRL algorithm is its ability to adapt to varying WIP arrival patterns and different layout configurations. Whether dealing with Uniform, Poisson, or Normal WIP arrival distributions, the model maintained consistent performance, effectively balancing workloads and minimising travel

Table 5. Effectiveness of the lower-level agent.

Method	S1			S2			S3		
	Aver.	Gap	Time(s)	Aver.	Gap	Time(s)	Aver.	Gap	Time(s)
Random selection + 2-OPT	71.7	24.1%	0.93	86.3	21.8%	1.56	119.7	32.1%	9.88
Higher-level agent + 2-OPT	88.7	6.1%	1.07	103.3	6.4%	8.87	152.8	13.3%	12.06
Our (Higher + Lower-level agent)	94.5	0.0%	1.13	111.4	0.0%	10.67	182.3	0.0%	22.42

distances. This flexibility makes it highly practical for dynamic environments where operational conditions, such as WIP demand or equipment layout, change frequently. Managers benefit from a system that requires minimal reconfiguration while adapting smoothly to operational changes, reducing downtime and ensuring continuous performance.

5. Conclusions

The integration of AMRs in manufacturing systems greatly improves material handling efficiency by dynamically assigning tasks and enabling flexible path planning, along with the loading of multiple WIPs. In this paper, we have proposed a novel hierarchical reinforcement learning framework for the material handling system in a dynamic manufacturing environment. We enhanced learning efficiency and performance by utilising a graph attention network as an encoder for the input states of our agent. To address the complexities of this system, our approach introduces a hierarchical structure: a higher-level agent responsible for assigning robots and a lower-level agent dedicated to planning the paths of these assigned robots. In response to the evolved challenges from the dynamic pickup and delivery problem, a suitable MDP was formulated for each agent. Through numerical experiments, we have demonstrated that the interaction between the higher-level and lower-level agents in this hierarchical structure enables us to achieve our goal of maximising the throughput of WIPs in the system.

Our proposed HRL algorithm is particularly well-suited to environments in which an equipment serves as both a pickup and a delivery node, and in which WIPs are dynamically transported via AMRs. Such an environment, as presented in this study, represents the challenges faced in modern manufacturing settings, particularly in smart logistics and other highly flexible, dynamic logistics industries. We anticipate that our model will be particularly effective in real-time decision-making scenarios within these industries, offering a significant advancement in the way material handling tasks are approached and executed. Through this paper, we demonstrate not only the feasibility but also the effectiveness of applying the advanced reinforcement learning method to complex, real-world problems in the manufacturing environment.

The successful implementation of our HRL framework sets a new benchmark in operational efficiency, paving the way for future innovations in intelligent and adaptive manufacturing systems.

Acknowledgments

The authors are grateful for the valuable comments from the associate editor and anonymous reviewers.

Data availability statement

The data used in this study are available from the first author upon reasonable request.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT) (No. RS-2024-00337285 and No. RS-2023-00218913).

Notes on contributors



Keonwoo Park received the B.S. degree in industrial engineering from Pusan National University, Korea, in 2020 and the M.S degree in industrial engineering from Seoul National University, Korea, in 2022. He is currently working toward the Ph.D. degree in industrial engineering from Seoul National University, Korea.

His research interests include reinforcement learning and optimisation with applications in supply chain management, logistics, and method of planning and scheduling.



Seongbae Jo received the B.S. and M.S. degrees in industrial engineering from Seoul National University, Korea, in 2021 and 2023, respectively. He is currently working toward the Ph.D. degree in industrial engineering from Seoul National University, Korea. His research interests include reinforcement learning and optimisation with applications in maritime logistics, supply chain management, and real-time decision-making problems.



Youngchul Shin received the B.S. degree in industrial engineering from Hanyang University, Korea, in 2015 and the M.S. and Ph.D. degrees in industrial engineering from Seoul National University, Korea, in 2017 and 2020, respectively. He is currently an assistant professor of industrial engineering with Ajou University, Korea. His research interests include the application of the optimisation methodologies to the production planning and inventory control, designing supply chain network, and transportation network and logistics.



Ilkyeong Moon received the B.S. and M.S. degrees in industrial engineering from Seoul National University, Korea, in 1984 and 1986, respectively, and the Ph.D. degree in operations research from Columbia University, USA, in 1991. He is currently a professor of industrial engineering with Seoul National University. He published 180 papers in international journals. His research interests include supply chain management, logistics, and inventory management. He has been an editor-in-chief for European Journal of Industrial Engineering. He was the president of the Korean Institute of Industrial Engineers (KIIE) from 2019 to 2020. He was selected as a regular member of the Korean Academy of Science and Technology in 2023 which is the most prestigious engineering and science society in Korea.

ORCID

Ilkyeong Moon  <http://orcid.org/0000-0002-7072-1351>

References

- Ahamed, Tanvir, Bo Zou, Nahid Parvez Farazi, and Theja Tulabandhula. 2021. "Deep Reinforcement Learning for Crowdsourced Urban Delivery." *Transportation Research Part B: Methodological* 152:227–257. <https://doi.org/10.1016/j.trb.2021.08.015>.
- Arslan, Alp M., Niels Agatz, Leo Kroon, and Rob Zuidwijk. 2019. "Crowdsourced Delivery—A Dynamic Pickup and Delivery Problem with Ad Hoc Drivers." *Transportation Science* 53 (1): 222–235. <https://doi.org/10.1287/trsc.2017.0803>.
- Baykasoğlu, Adil, Fatma S. Madenoğlu, and Alper Hamzadayı. 2020. "Greedy Randomized Adaptive Search for Dynamic Flexible Job-shop Scheduling." *Journal of Manufacturing Systems* 56:425–451. <https://doi.org/10.1016/j.jmsy.2020.06.005>.
- Behrendt, Adam, Martin Savelsbergh, and He Wang. 2023. "A Prescriptive Machine Learning Method for Courier Scheduling on Crowdsourced Delivery Platforms." *Transportation Science* 57 (4): 889–907. <https://doi.org/10.1287/trsc.2022.1152>.
- Berbeglia, Gerardo, Jean-François Cordeau, and Gilbert Laporte. 2010. "Dynamic Pickup and Delivery Problems." *European Journal of Operational Research* 202 (1): 8–15. <https://doi.org/10.1016/j.ejor.2009.04.024>.
- Chaudhry, Imran Ali, and Abid Ali Khan. 2016. "A Research Survey: Review of Flexible Job Shop Scheduling Techniques." *International Transactions in Operational Research* 23 (3): 551–591. <https://doi.org/10.1111/itor.2016.23.issue-3>.
- Chien, Chen-Fu, and Yu-Bin Lan. 2021. "Agent-Based Approach Integrating Deep Reinforcement Learning and Hybrid Genetic Algorithm for Dynamic Scheduling for Industry 3.5 Smart Production." *Computers & Industrial Engineering* 162:107782. <https://doi.org/10.1016/j.cie.2021.107782>.
- Fang, Yilin, Chao Peng, Ping Lou, Zude Zhou, Jianmin Hu, and Junwei Yan. 2019. "Digital-Twin-Based Job Shop Scheduling toward Smart Manufacturing." *IEEE Transactions on Industrial Informatics* 15 (12): 6425–6435. <https://doi.org/10.1109/TII.9424>.
- Fekih, Asma, Hatem Hadda, Imed Kacem, and Atidel B. Hadj-Alouane. 2020. "A Hybrid Genetic Tabu Search Algorithm for Minimising Total Completion Time in a Flexible Job-shop Scheduling Problem." *European Journal of Industrial Engineering* 14 (6): 763–781. <https://doi.org/10.1504/EJIE.2020.112479>.
- Ferrucci, Francesco, and Stefan Bock. 2014. "Real-Time Control of Express Pickup and Delivery Processes in a Dynamic Environment." *Transportation Research Part B: Methodological* 63:1–14. <https://doi.org/10.1016/j.trb.2014.02.001>.
- Fragapane, Giuseppe, Rene De Koster, Fabio Sgarbossa, and Jan Ola Strandhagen. 2021. "Planning and Control of Autonomous Mobile Robots for Intralogistics: Literature Review and Research Agenda." *European Journal of Operational Research* 294 (2): 405–426. <https://doi.org/10.1016/j.ejor.2021.01.019>.
- Han, Beining, Zhizhou Ren, Zuofan Wu, Yuan Zhou, and Jian Peng. 2022. "Off-Policy Reinforcement Learning with Delayed Rewards." In *International Conference on Machine Learning*, 8280–8303. PMLR.
- Jun, Sungbum, Seokcheon Lee, and Yuehwern Yih. 2021. "Pickup and Delivery Problem with Recharging for Material Handling Systems Utilising Autonomous Mobile Robots." *European Journal of Operational Research* 289 (3): 1153–1168. <https://doi.org/10.1016/j.ejor.2020.07.049>.
- Kool, Wouter, Herke Van Hoof, and Max Welling. 2018. "Attention, Learn to Solve Routing Problems!" arXiv preprint arXiv:https://arxiv.org/abs/1803.08475.
- Kulkarni, Tejas D., Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. "Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation." *Advances in Neural Information Processing Systems* 29: 3675–3683.
- Lei, Kun, Peng Guo, Yi Wang, Jian Zhang, Xiangyin Meng, and Linmao Qian. 2023. "Large-Scale Dynamic Scheduling for Flexible Job-Shop With Random Arrivals of New Jobs by Hierarchical Reinforcement Learning." *IEEE Transactions on Industrial Informatics* 20 (1): 1007–1018. <https://doi.org/10.1109/TII.2023.3272661>.
- Li, Yuxin, Wenbin Gu, Minghai Yuan, and Yaming Tang. 2022. "Real-Time Data-Driven Dynamic Scheduling for Flexible Job Shop with Insufficient Transportation Resources Using Hybrid Deep Q Network." *Robotics and Computer-Integrated Manufacturing* 74:102283. <https://doi.org/10.1016/j.rcim.2021.102283>.
- Liu, Renke, Rajesh Piplani, and Carlos Toro. 2022. "Deep Reinforcement Learning for Dynamic Scheduling of a Flexible Job Shop." *International Journal of Production Research* 60 (13): 4049–4069. <https://doi.org/10.1080/00207543.2022.2058432>.

- Luo, Shu. 2020. "Dynamic Scheduling for Flexible Job Shop with New Job Insertions by Deep Reinforcement Learning." *Applied Soft Computing* 91:106208. <https://doi.org/10.1016/j.asoc.2020.106208>.
- Ma, Yi, Xiaotian Hao, Jianye Hao, Jiawen Lu, Xing Liu, Tong Xialiang, Mingxuan Yuan, Zhigang Li, Jie Tang, and Zhaopeng Meng. 2021. "A Hierarchical Reinforcement Learning Based Optimization Framework for Large-Scale Dynamic Pickup and Delivery Problems." *Advances in Neural Information Processing Systems* 34:23609–23620.
- Miyamoto, Toshiyuki, and Kensuke Inoue. 2016. "Local and Random Searches for Dispatch and Conflict-Free Routing Problem of Capacitated AGV Systems." *Computers & Industrial Engineering* 91:1–9. <https://doi.org/10.1016/j.cie.2015.10.017>.
- Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves. 2015. "Human-Level Control through Deep Reinforcement Learning." *Nature* 518 (7540): 529–533. <https://doi.org/10.1038/nature14236>.
- Ren, Weibo, Yan Yan, Yaoguang Hu, and Yu Guan. 2022. "Joint Optimisation for Dynamic Flexible Job-shop Scheduling Problem with Transportation Time and Resource Constraints." *International Journal of Production Research* 60 (18): 5675–5696. <https://doi.org/10.1080/00207543.2021.1968526>.
- Rossi, Andrea, and Gino Dini. 2007. "Flexible Job-shop Scheduling with Routing Flexibility and Separable Setup times Using Ant Colony Optimisation Method." *Robotics and Computer-Integrated Manufacturing* 23 (5): 503–516. <https://doi.org/10.1016/j.rcim.2006.06.004>.
- Savelsbergh, Martin W. P., and Marc Sol. 1995. "The General Pickup and Delivery Problem." *Transportation Science* 29 (1): 17–29. <https://doi.org/10.1287/trsc.29.1.17>.
- Shahgholi Zadeh, Melissa, Yalda Katebi, and Ali Doniavi. 2019. "A Heuristic Model for Dynamic Flexible Job Shop Scheduling Problem considering Variable Processing times." *International Journal of Production Research* 57 (10): 3020–3035. <https://doi.org/10.1080/00207543.2018.1524165>.
- Shao, Yan, Rongpeng Li, Bing Hu, Yingxiao Wu, Zhifeng Zhao, and Honggang Zhang. 2021. "Graph Attention Network-Based Multi-Agent Reinforcement Learning for Slicing Resource Management in Dense Cellular Network." *IEEE Transactions on Vehicular Technology* 70 (10): 10792–10803. <https://doi.org/10.1109/TVT.2021.3103416>.
- Shen, Xiao-Ning, and Xin Yao. 2015. "Mathematical Modeling and Multi-Objective Evolutionary Algorithms Applied to Dynamic Flexible Job Shop Scheduling Problems." *Information Sciences* 298:198–224. <https://doi.org/10.1016/j.ins.2014.11.036>.
- Simoni, Michele D., and Matthias Winkenbach. 2023. "Crowd-sourced On-demand Food Delivery: An Order Batching and Assignment Algorithm." *Transportation Research Part C: Emerging Technologies* 149:104055. <https://doi.org/10.1016/j.trc.2023.104055>.
- Tao, Yi, Haibing Zhuo, and Xiaofan Lai. 2023. "The Pickup and Delivery Problem with Multiple Depots and Dynamic Occasional Drivers in Crowdshipping Delivery." *Computers & Industrial Engineering* 182:109440. <https://doi.org/10.1016/j.cie.2023.109440>.
- Thörnblad, Karin, Ann-Brith Strömberg, Michael Patriksson, and Torgny Almgren. 2015. "Scheduling Optimisation of a Real Flexible Job Shop including Fixture Availability and Preventive Maintenance." *European Journal of Industrial Engineering* 9 (1): 126–145. <https://doi.org/10.1504/EJIE.2015.067451>.
- Ulmer, Marlin W., Barrett W. Thomas, Ann Melissa Campbell, and Nicholas Woyak. 2021. "The Restaurant Meal Delivery Problem: Dynamic Pickup and Delivery with Deadlines and Random Ready times." *Transportation Science* 55 (1): 75–100. <https://doi.org/10.1287/trsc.2020.1000>.
- Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. "Graph Attention Networks." arXiv preprint arXiv:<https://arxiv.org/abs/1710.10903>.
- Xin, Liang, Wen Song, Zhiguang Cao, and Jie Zhang. 2020. "Step-Wise Deep Learning Models for Solving Routing Problems." *IEEE Transactions on Industrial Informatics* 17 (7): 4861–4871. <https://doi.org/10.1109/TII.2020.3031409>.
- Zong, Zefang, Meng Zheng, Yong Li, and Depeng Jin. 2022. "Mapdp: Cooperative Multi-Agent Reinforcement Learning to Solve Pickup and Delivery Problems." In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, 9980–9988.
- Zou, Wen-Qiang, Quan-Ke Pan, and Ling Wang. 2021. "An Effective Multi-Objective Evolutionary Algorithm for Solving the AGV Scheduling Problem with Pickup and Delivery." *Knowledge-Based Systems* 218:106881. <https://doi.org/10.1016/j.knosys.2021.106881>.

Appendices

Appendix 1. Description of actions for the Lower-Level Agent

Inner Relocate Algorithm

- (1) *Robot Selection*: Select the robot with the largest total sequence length.
- (2) *Sequence Rearrangement*: Relocate one of the sequences that the robot is scheduled to visit to a later order in the remaining sequence.
- (3) *Path Length Evaluation*: After each relocation, check for a decrease in the original path length and ensure compliance with the precedence conditions.
- (4) *Path Confirmation*: If conditions are met, establish the new path after relocation. Otherwise, revert to the original path and continue the rearrangement process until the end of the robot's sequence.

The Inner Relocate Algorithm is designed to optimise the path of an individual robot by reordering the tasks assigned to it. This action begins by selecting the robot with the largest total sequence length, as this robot is likely to have the greatest opportunity for optimisation. The process then involves relocating one of the sequences currently assigned to that robot to a different, later position within the remaining sequence. This relocation is evaluated to check whether it reduces the overall path length while ensuring compliance with the precedence conditions, that is, while maintaining the correct order of pickup and delivery tasks for WIPs. If these conditions are satisfied, the new path is adopted; otherwise, the original sequence is restored, and the rearrangement process continues.

This action is particularly valuable for smoothing intra-robot and for operations and reducing travel redundancy.

Inner Exchange Algorithm

- (1) *Robot Selection*: Select the robot with the largest total sequence length.
- (2) *Sequence Exchange*: Exchange one of the sequences that the robot is scheduled to visit with another sequence in the remaining list.
- (3) *Path Length Evaluation*: After each exchange, check whether there is a decrease in the original path length and ensure that previously mentioned precedence conditions are met.
- (4) *Path Confirmation*: If the conditions are satisfied, fix the result of this exchange as the new path. If not, revert to the original path and repeat the exchange process until the end of the robot's sequence.

The Inner Exchange Algorithm targets further optimisation within a robot's assigned sequence. Similar to the Inner Relocate Algorithm, it begins by selecting the robot with the largest total sequence length. Instead of relocating a sequence, this action involves exchanging two sequences within the same robot's path. Each exchange is tested for its effectiveness in reducing the total path length and is confirmed only if it also maintains precedence conditions. By attempting to exchange tasks rather than just relocating them, this algorithm introduces a more dynamic level of reorganisation, potentially allowing for shorter paths while also reducing overall task clustering that may lead to inefficiencies.

Inter Relocate Algorithm

- (1) *Robot Selection*: Select Robot1 as the robot with the largest total sequence length and Robot2 as the robot with the smallest total sequence length from the state information.
- (2) *Sequence Rearrangement*: Identify the last WIP in Robot1's sequence that is not currently being carried and append it to the end of Robot2's sequence.
- (3) *Optimisation Process*: Apply Inner relocate algorithm to Robot2. Compare the total route length with the original route.
- (4) *Iterative Process*: Continue the process, moving backward from the end of Robot1's sequence, until a reduction in the total route is achieved.

The Inter Relocate Algorithm focuses on balancing workloads between multiple robots. This algorithm begins by selecting two robots: Robot1, which has the largest sequence length, and Robot2, which has the smallest. The goal is to move some tasks from the heavily loaded Robot1 to Robot2, thereby reducing overall imbalance. The last WIP in Robot1's sequence, which is not yet being carried, is appended to the end of Robot2's sequence. Following this, the Inner Relocate Algorithm is applied to Robot2 to optimise the new sequence configuration. This iterative process continues, working backward through Robot1's sequence until a reduction in the total route length is achieved. The purpose of this action is to ensure more evenly distributed workloads among the robots, thereby reducing the chance of any single robot being overwhelmed while others remain underutilised.

Inter Exchange Algorithm

- (1) *Robot Selection*: Select Robot1 with the longest total sequence length and Robot2 with the second longest, based on state information.
- (2) *Sequence Rearrangement*: Identify the last non-carrying WIP in Robot1's sequence and move it to the end of Robot2's sequence.
- (3) *Sequence Rearrangement2*: Similarly, find the last non-carrying WIP in Robot2's sequence and move it to the end of Robot1's sequence.
- (4) *Optimisation Process*: Apply Inner relocate algorithm to both robots. Compare the total route length with the original route.
- (5) *Iterative Process*: Iterate this process for both Robot1 and Robot2, starting from the end of their sequences to the first sequence, until a reduction in the overall route is observed.

The Inter Exchange Algorithm also addresses workload balancing between robots but with a different approach compared to the Inter Relocate Algorithm. It selects Robot1 and Robot2, both having relatively long sequence lengths based on current state information. The algorithm identifies the last non-carrying WIP in each of the robots' sequences and swaps them, moving the WIP from Robot1's to Robot2's sequence and vice versa. After this swap, the Inner Relocate Algorithm is applied to both robots to attempt further optimisation of each robot's new sequence. This process continues iteratively, swapping sequences from the end to the beginning, until there is a noticeable improvement in the overall path length. This action aims to not only balance workloads between robots but also to enhance the overall efficiency of their routes, preventing any robot from being overburdened while also ensuring effective use of the available AMRs.

Appendix 2. Description of job assignment process for AMRs

A.1 Virtual environment

The aim of our study is to develop a decision support system for an automated material handling system using hierarchical reinforcement learning. This system is specifically designed for a manufacturing process in which AMRs transport WIPs from one equipment source to the next. The environment was programmed in Python, and it is based on the premise that AMRs navigate using a grid map. Although AMRs are capable of moving a network of nodes and arcs, we adopted a grid map to facilitate their movement. This choice was made to more accurately represent the AMRs' unique characteristics, including their directional flexibility, which is a notable advantage over other material handling systems like OHT vehicles or AGVs. In this grid map, certain equipment is positioned as either a *from* point, where WIP is generated, or a *to* point, where WIP is received.

A.2 Dynamic job assignment for AMRs

We illustrate the job assignment process for AMRs in a manufacturing environment using Figure A9, which represents a decision-making snapshot. In this scenario, AMRs 1 to 4 are

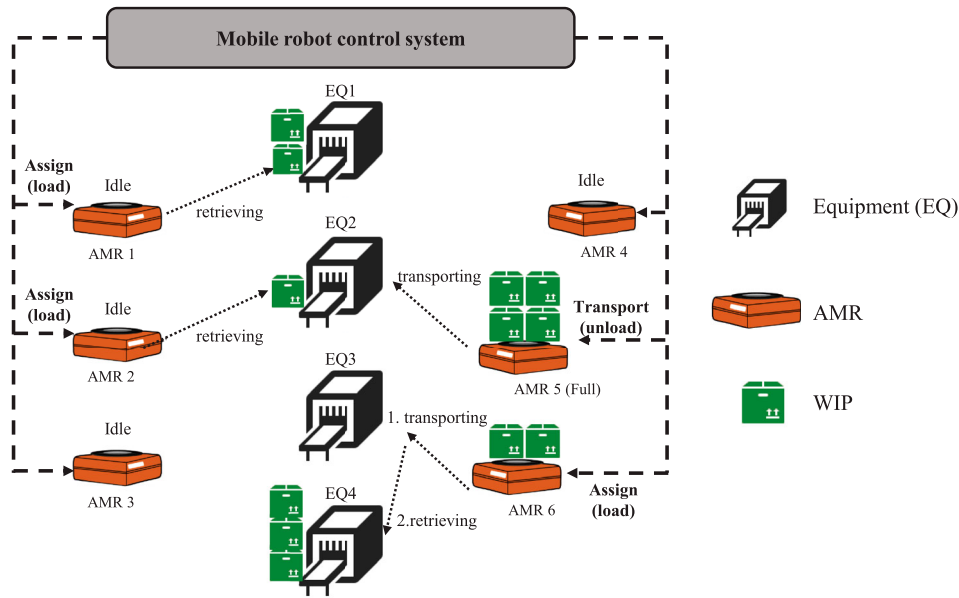


Figure A9. Illustration of a scenario for job assignment with AMRs.

idle and without any loaded WIPs, making them available for new assignments. With WIPs waiting at equipment (EQ) 1 and 2, these AMRs are needed for loading tasks. Here, AMRs 1 and 2 are selected for assignment. Meanwhile, AMR 6, although already carrying two WIPs, has a maximum capacity of four and can therefore take on additional loads. In contrast, AMR 5 is at full capacity and cannot load more WIPs. Despite being in transit, AMR 6 is identified by the control system as the best choice for loading additional WIPs from EQ 4, leading to its assignment for further loading. Therefore, this dynamically shifts AMR 6 from its existing transporting action to a retrieving action. This example shows the versatility of the AMR system; not only idle AMRs but also those engaged in retrieval, loading, transporting, or unloading can be considered for new tasks.

A.3 Interpreting learned rules from RL agents

To better understand the behaviour of the RL agents, we analysed their converged policies through a Gantt chart illustrating the deliveries executed by the AMRs, as shown in Figure A10. Specific segments of these charts were selected to highlight notable patterns observed in the experimental results.

As depicted in the Gantt chart, the typical delivery sequence, such as those seen at the initial stages for AMR 5 and AMR 6, follows a predictable flow: Retrieval → Loading → Transporting → Unloading. However, there are instances where the decision-making process becomes more adaptive and responsive to changes in real time. For example, AMR 8 modifies its retrieval process at an early stage, and AMR 3 shifts from retrieval to transportation in the middle of the process.

These dynamic behaviours are attributable to the lower-level agent's ability to evaluate real-time environmental factors, such as the proximity of the subsequent delivery task, current AMR capacity, and relative distance to other entities in the environment. This continuous assessment enables the agent to dynamically implement an optimised strategy, thereby contributing to increased system efficiency.

Appendix 3. Integer programming for dispatching and path planning of AMRs

The integer programming model for dispatching and path planning of AMR considers a dynamic pickup and delivery problem with simultaneous pickup and delivery nodes. We present the following IP model, adapted from the methodology described in Miyamoto and Inoue (2016).

In the referenced study, constants were classified into the physical domain and the task domain for clarity and organisation. Following this approach, we define the physical domain constants in our study as follows: E represents the set of equipment, K represents the set of robots, S denotes a set of nodes, A represents the set of edges, and A_s is the set of adjacent nodes of $s \in S$. For the task domain constants, P is the set of all points, P_A is the set of pickup points for already picked-up tasks, P_E is the set of points for equipment, and P_K is the set of points for robots. Additionally, P_N represents the set of pickup-and-delivery points excluding P_A , P_{DEL} is the set of delivery points, and P_{PICK} is the set of pickup points. The current load of vehicle k and the buffer of machine e after a pickup or delivery at point $i \in P_N$ are denoted as c_i^k and c_i^e , respectively, with initial values c_{0k}^k and c_{0e}^e . The functions f and g map the initial positions of robots and equipment to nodes and associate PD points with equipment, respectively. TE represents the set of machines in the task domain ($q_e \in TE$), and TR represents the set of robots in the task domain, where $\tau_k \in TR$. Furthermore, n^p represents a pickup point, and the robot assigned to handle n^p is denoted as k^n . Similarly, n^d represents a delivery point, while w_i denotes the quantity of material picked up or delivered at point i . For a detailed description of the mathematical formulation, readers may refer to the aforementioned paper.

The decision variables are defined as follows: The binary variable x_{ijk} is equal to 1 if robot k directly visits PD point j after PD point i , otherwise 0. The binary variable y_{ije} equals to 1 if the loading or unloading operation at PD point j occurs immediately after that of PD point i at equipment e , otherwise 0. The binary variable z_{skt} is equal to 1 if robot k is positioned at node

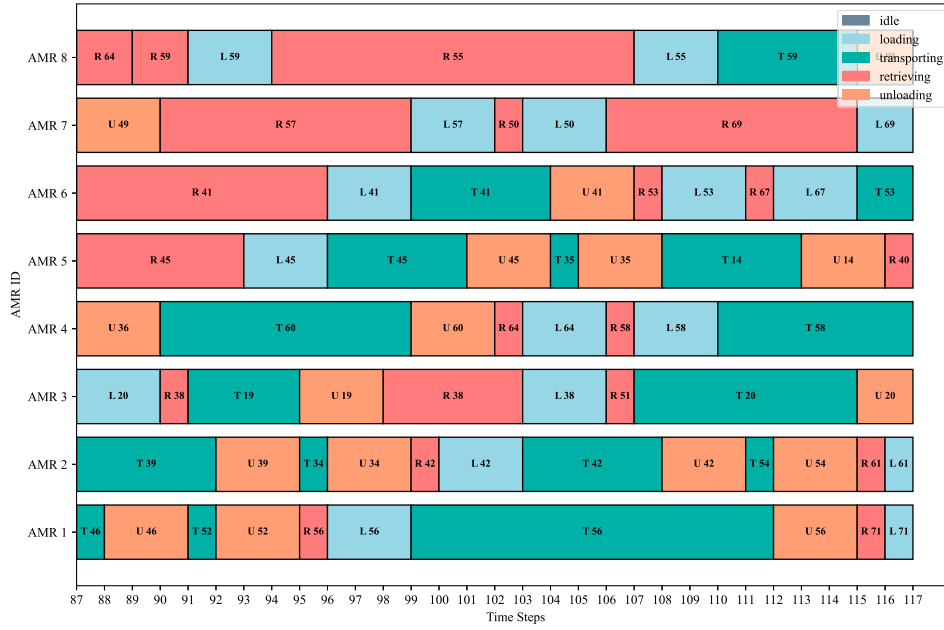


Figure A10. AMR task schedule with status.

s at time t , otherwise 0. The continuous variable arr_i represents the time at which a robot arrives at PD point i .

$$\min \sum_{s \in S, k \in K, t \in T} |z_{skt} - z_{sk(t+1)}| \quad (\text{A1})$$

$$\text{s.t.} \sum_{j \in P_N} x_{\tau_k j k} = 1 \quad (k \in K) \quad (\text{A2})$$

$$\sum_{i \in P_K} x_{ihk} = \sum_{j \in P_K} x_{hjk} = 0 \quad (h \in P_N, k \in K) \quad (\text{A3})$$

$$\sum_{j \in P_K, k \in K} x_{ijk} = \sum_{j \in P_K, k \in K} x_{jik} = 1 \quad (i \in P_K) \quad (\text{A4})$$

$$\sum_{j \in P_K} x_{n^p j k} = \sum_{j \in P_K} x_{j n^d k} \quad (n^p \in P_{\text{PICK}} \setminus P_A, k \in K) \quad (\text{A5})$$

$$\sum_{i \in P_K} x_{i n^d k n} = 1 \quad (n^p \in P_A) \quad (\text{A6})$$

$$c_{\tau_k}^k = c_{0k}^k \wedge c_{q_e}^e = c_{0e}^e \quad (k \in K, e \in E) \quad (\text{A7})$$

$$x_{ijk} = 1 \implies c_i^k + w_j = c_j^k \quad (i \in P_K, j \in P_N, k \in K) \quad (\text{A8})$$

$$y_{ije} = 1 \implies c_i^e - w_j = c_j^e \quad (i \in P_E, j \in P_N, e \in E) \quad (\text{A9})$$

$$\sum_{i \in P_K} x_{ijk} = 1 \implies c_j^k \leq C \quad (j \in P_N, k \in K) \quad (\text{A10})$$

$$\sum_{j \in P_E} y_{q_e j e} = \sum_{i \in P_E} y_{i q_e e} = 1 \quad (e \in E, q_e \in TE) \quad (\text{A11})$$

$$\sum_{i \in P_E} y_{i h e} - \sum_{j \in P_E} y_{h j e} = 0 \quad (h \in P_N, e \in E) \quad (\text{A12})$$

$$\sum_{j \in P_E, e \in E} y_{j i e} = \sum_{j \in P_E, e \in E} y_{i j e} = 1 \quad (i \in P_E) \quad (\text{A13})$$

$$\sum_{j \in P_E} y_{i j f_i} = \sum_{j \in P_E} y_{j i f_i} = 1 \quad (i \in P_N) \quad (\text{A14})$$

$$\text{arr}_{\tau_k} = 0 \wedge \text{arr}_{q_e} = 0 \quad (k \in K, e \in E) \quad (\text{A15})$$

$$x_{ijk} = 1 \implies \text{arr}_i < \text{arr}_j \quad (i \in P_K, j \in P_N, k \in K) \quad (\text{A16})$$

$$y_{ije} = 1 \implies \text{arr}_i < \text{arr}_j \quad (i \in P_E, j \in P_N, e \in E) \quad (\text{A17})$$

$$f_i = f_j \implies \text{arr}_i \neq \text{arr}_j \quad (i, j \in P_N, i \neq j) \quad (\text{A18})$$

$$\sum_{s \in S} z_{skt} = 1 \quad (k \in K, t \in T) \quad (\text{A19})$$

$$\sum_{k \in K} z_{skt} \leq 1 \quad (s \in S, t \in T) \quad (\text{A20})$$

$$z_{g_k k 0} = 1 \quad (s \in S, k \in K) \quad (\text{A21})$$

$$z_{skt} \leq \sum_{s_2 \in A_s} z_{s_2 k(t+1)} \quad (s \in S, k \in K, t \in T \setminus T_{\max}) \quad (\text{A22})$$

$$z_{s_1 k_1 t} z_{s_2 k_2 t} + z_{s_1 k_2(t+1)} z_{s_2 k_1(t+1)} \leq 1 \\ (s_1 \in S, s_2 \in A_{s_1}, k_1, k_2 \in K, k_1 \neq k_2, t \in T \setminus T_{\max}) \quad (\text{A23})$$

$$\text{arr}_j = t \implies \sum_{i \in P_K} x_{ijk} \leq z_{g_j k t} \quad (j \in P_N, k \in K, t \in T) \quad (\text{A24})$$

Constraints (A2)–(A6) address the path planning of robots, ensuring that the variable x_{ijk} directs each robot k to start at its initial point τ_k , visit all assigned points, and allocate all the

tasks. Constraints (A5) and (A6) guarantee that tasks are transported by the same robot. Constraints (A7)–(A10) enforce that the capacities of robots and equipment, as defined by the environment, are satisfied. Constraints (A11)–(A14) ensure that the variable y_{ije} is determined to define the sequence of loading and unloading operations occurring at the equipment, ensuring all

tasks are processed. Constraints (A15)–(A18) impose time conditions that must be satisfied by arr, ensuring proper sequencing and scheduling of tasks. Constraints (A19)–(A24) pertain to collision avoidance in a bidirectional graph, ensuring that no more than one robot occupies a single node at any given time.