Robotics and Computer–Integrated Manufacturing 94 (2025) 102959

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

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

Full length article

Dynamic multi-tour order picking in an automotive-part warehouse based on attention-aware deep reinforcement learning

Xiaohan Wang [a,b], Lin Zhang [a,c,*], Lihui Wang [b], Enrique Ruiz Zuñiga [d], Xi Vincent Wang [b,**], Erik Flores-García [b]

[a] School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
[b] Department of Production Engineering, KTH Royal Institute of Technology, Stockholm 10044, Sweden
[c] State Key Laboratory of Intelligent Manufacturing System Technology, Beijing 100854, China
[d] Department of Sustainable Production Development, KTH Royal Institute of Technology, Stockholm 10044, Sweden

ARTICLE INFO

Keywords:
Smart manufacturing system
Industry 5.0
Manual order picking
Deep reinforcement learning
Intelligent decision-making

ABSTRACT

Dynamic order picking has usually demonstrated significant impacts on production efficiency in warehouse management. In the context of an automotive-part warehouse, this paper addresses a dynamic multi-tour order-picking problem based on a novel attention-aware deep reinforcement learning-based (ADRL) method. The multi-tour represents that one order-picking task must be split into multiple tours due to the cart capacity and the operator's workload constraints. First, the multi-tour order-picking problem is formulated as a mathematical model, and then reformulated as a Markov decision process. Second, a novel DRL-based method is proposed to solve it effectively. Compared to the existing DRL-based methods, this approach employs multi-head attention to perceive warehouse situations. Additionally, three improvements are proposed to further strengthen the solution quality and generalization, including (1) the extra location representation to align the batch length during training, (2) the dynamic decoding to integrate real-time information of the warehouse environment during inference, and (3) the proximal policy optimization with entropy bonus to facilitate action exploration during training. Finally, comparison experiments based on thousands of order-picking instances from the Swedish warehouse validated that the proposed ADRL could outperform the other twelve DRL-based methods at most by 40.6%, considering the optimization objective. Furthermore, the performance gap between ADRL and seven evolutionary algorithms is controlled within 3%, while ADRL can be hundreds or thousands of times faster than these EAs regarding the solving speed.

1. Introduction

Industry 5.0 complements the existing industry paradigm by emphasizing research and innovation as catalysts for a transition towards a human-centric, sustainable, and resilient European industry [1,2]. As a crucial component in the European industry, warehouse management directly impacts production efficiency and operator workload in manufacturers [3]. Within warehouse management, manual order picking can constitute up to 60% of the total human-labor activities and might represent as high as 65% of the overall operational costs [4,5]. Especially in an automotive-part warehouse, the order-picking activity can largely influence the production efficiency because the manufacturing of one automotive usually requires the assembly of thousands of automotive parts [6]. According to a survey in an automotive-part warehouse in Scania, more than 50% of operators thought their picking activity could be further optimized to reduce repeated fetching back and forth. Therefore, an efficient order-picking approach is of great significance in an automotive-part warehouse.

Although existing research has paid much attention to the order-picking problems under various manufacturing backgrounds, most overlooked the warehouse scenario's dynamics and its practical constraints, such as the cart capacity and the operator's workload. Specifically, most research regards warehouse order picking as a variant of the Steiner traveling salesman problem (STSP) with assumptions of plenty of cart capacity and unlimited workload [7,8]. Although they provided promising thoughts and highly abstract models for simplifying order-picking models, the accompanying assumptions may not suit practical order-picking problems.

Therefore, this paper considered the existing limitations and studied the dynamic order picking of automotive parts in a Swedish warehouse.

* Corresponding author at: School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China.
** Corresponding author.
E-mail addresses: zhanglin@buaa.edu.cn (L. Zhang), wangxi@kth.se (X.V. Wang).

https://doi.org/10.1016/j.rcim.2025.102959
Received 15 April 2024; Received in revised form 29 November 2024; Accepted 7 January 2025
Available online 16 January 2025
0736-5845/© 2025 Published by Elsevier Ltd.

Dynamic order picking describes a scenario in which an operator expeditiously picks up dozens of automotive parts distributed in a warehouse corridor, considering uncertain changes of order requirements and part placements. In the practical warehouse, the cart may be unable to transport all the necessary parts with just one tour. Additionally, considering the operator's workload during each tour, it is imperative to maintain a balanced work intensity. However, existing order-picking models rarely involve these practical factors simultaneously. Furthermore, the warehouse environment exhibits dynamics and diversity due to the unpredictable order requirements and the distinguishable part placements in warehouses. Therefore, the dynamics of the multi-tour order-picking tasks must be taken into account. The dynamic environment elevates the demand for generalization and solving speed of the approach, and increasing efforts have been invested in order-picking approaches.

The commonly utilized methods for order-picking problems can be roughly divided into four categories, namely mathematical methods, heuristic rules, evolutionary algorithms (EAs), and deep reinforcement learning-based methods (DRLs) [9,10]. Mathematical methods solve the order-picking problems with constructed mixed-integer programming models, which can find globally optimal solutions [7,11,12]. However, their time complexity tends to be unacceptable because of the NP-hard nature of the dynamic order-picking problems. Heuristic rules refer to hand-craft programs defined by experienced engineers [13]. Although they spend little solving time, their solution quality might be unsatisfying when facing diverse warehouse scenarios. In contrast to mathematical methods and heuristic rules, EAs demonstrate the capability to attain satisfactory solutions within a reasonable time. This characteristic renders them widely applicable to order-picking problems [8,14]. However, a satisfactory solution usually consumes much search time, limiting its application in dynamic scenarios with rapid decision-making requirements. Besides, the generalization of EAs is relatively limited, necessitating a new search when scenarios change. In contrast to EAs, DRLs indicate high generalization to dynamic environments. A DRL agent trained in one instance can easily suit other unseen instances because of the reasoning ability of the neural networks [15]. DRLs typically have fast solving speed, only needing feed-forward calculations after the agents' training. The strong generalization and rapid solving speed make DRLs well-suited for dynamic order-picking problems: They can quickly generate a re-planning solution when the warehouse scenario changes. Nonetheless, their generalization and solution quality heavily depend on the feature extraction efficiency of the problem, i.e., the state representation of the MDP environment. Consequently, enhancing the extraction efficiency of DRLs emerges as a promising direction to further improve the order-picking performance.

With the widespread of large language models (LLM), attention-based neural architectures have been proven effective in capturing the intercorrelations among features [16]. Some research considered combining the attention-based neural networks and DRL-based training algorithms to solve traveling salesman problems (TSPs) and vehicle routing problems (VRPs), achieving competitive results compared to conventional methods [17,18]. However, little research considers expanding the attention-based approach to practical order-picking problems. Furthermore, exploratory experiments illustrate that three algorithmic issues may largely influence the performance of attention-based DRLs for the dynamic multi-tour order-picking problem:

(1) The solution lengths for different order-picking instances tend to be inconsistent because of diverse order requirements for the number of parts. Consequently, the length of training samples cannot be aligned for training as batches, which may essentially decrease the training efficiency.

(2) The information for the remaining parts in each location will change in real-time as the implementation of order picking. However, most existing algorithms only encode the location information once at the initiation of the order picking, overlooking the real-time changes of these remaining parts.

(3) The data distribution of training datasets derived from the Swedish warehouse tends to be unbalanced and biased, posing challenges to training a policy model of high quality. In contrast to standard datasets for TSPs or VRPs, the data distribution within practical order-picking datasets indicates chaos and complexity, which puts forward higher requirements for the exploration ability of the agent and policy generalization.

With motivations to efficiently address these challenges and in line with the principles of Industry 5.0, this paper proposes a novel attention-aware DRL-based approach (ADRL) for dynamic multi-tour order-picking problems within the automotive-part warehouse. The expression 'attention-aware' highlights the capture of the state information through the attention mechanism. The studied problem originates from the Swedish automotive-part warehouse, where one order-picking task tends to be split into multiple tours because of the constraints of the cart weight and the operator's workload. Regarding the dynamic and diverse warehouse scenarios, we utilize the strong generalization of ADRL to rapidly re-generate new solutions when the scenario changes. The proposed ADRL is based on the actor-critic DRL architecture with four neural network models, including the policy encoder, the policy decoder, the value encoder, and the value decoder. Each of the policy encoder and value encoder possesses an identical network employing multi-head attention (MHA) to capture intercorrelations among location features. The policy decoder calculates the probability distribution of actions according to the encoder output, the context information, and the real-time information of the remaining parts. In contrast, the value decoder employs two linear layers to calculate the value for the current state. These four neural network models are trained using proximal policy optimization with entropy bonus (Entropy-PPO). The entropy bonus in the loss function facilitates action exploration within fixed training datasets and alleviates the influence of unbalanced data distribution. Experimental results on 6600 practical instances validated the effectiveness and superiority of the proposed ADRL compared to seven popular DRLs, seven commonly utilized EAs, and four other attention-based methods. To conclude, the contribution of the paper can be summarized as follows:

(1) This paper introduces a nonconventional order-picking model, i.e., a multi-tour order-picking model, to resolve the practical warehouse order picking in an automotive-part warehouse. The formulated model aims to minimize the total order picking time (OPT) considering the cart weight and the operator's workload constraints. The mixed integer programming formulation is provided first, followed by the MDP reformulation with states, actions, and rewards specifically defined for this model.

(2) A novel attention-aware DRL approach, namely ADRL, is proposed to solve the studied problem effectively. ADRL employs MHA to capture state features of the shelf locations and trains the neural networks with an actor-critic DRL algorithm. Three improvements are proposed to handle the algorithmic issues and further enhance ADRL's performance. First, we formulate an extra location embedding representing the order-picking's complement to align the training batch. Second, we utilize the dynamic decoding calculation to integrate the real-time information for the remaining parts. Third, we adopt the entropy-PPO to train the agent policy within unbalanced data distributions.

(3) The application of the proposed ADRL contributes to improving the generalization and solution quality of existing order-picking methods. In contrast to twelve learning-based methods, ADRL exhibits a performance advantage, surpassing them by up to 40.6% in terms of the optimization objective. The performance gap between ADRL and seven popular EAs is no more than 3%. However, ADRL demonstrates a substantially higher computational efficiency, hundreds or thousands of times faster than EAs in solving an order-picking instance.

The rest of the paper is organized as follows: Section 2 reviews the related work on warehouse order picking problems and DRLs for production optimization. Section 3 introduces the system model of the multi-tour order-picking problem and the corresponding MDP formulation. The detailed framework of the proposed ADRL is described in Section 4. The comparison experiments based on the practical case study are conducted in Section 5 before concluding the paper in Section 6.

2. Related work

This section reviews related literature from the perspective of the problem and method. The existing research on warehouse order-picking problems is introduced first, followed by the recent work on DRL-based approaches for production optimization. In this study, the warehouse can also be regarded as an automotive-part supermarket or a logistics platform.

2.1. Warehouse order-picking problems

As the core of a warehouse management system, manual order picking has been extensively studied for decades [19]. Among the related work, some studies consider the joint optimization of order batching and order picking [9,11,20,21]. Joint optimization can simultaneously improve batching and picking efficiency compared to only optimizing the order-picking procedure. The required parts of one order in their scenarios tend to be less than the maximum capacity of the cart. Therefore, several similar orders can be packed into one batch to improve the picking efficiency. However, this strategy is unsuitable for automotive-part warehouses because the required parts of an order usually exceed the cart capacity. In contrast to conventional order-picking problems modeled as an STSP [8], some related studies introduced multiple optimization objectives and various picking constraints [22]. Considering the optimization objectives among these studies, the commonly discussed ones include total travel time [11,23], makespan of pickers [11], picking delays [23], workload balancing [24], picking costs [25], and operator's energy consumption [14]. The commonly involved order-picking constraints include the traffic congestion [23], the picking capacity [26], the diverse order requirements [27], and the operator's workload [14]. Subsequently, considering the dynamic order picking, related research usually concerns the uncertain events such as random order arrival [13,28] and the unexpected disruption [20] during the picking procedure. For example, Dauod et al. considered the urgent orders' arrival and the possible disruption events during picking, and the conflicts between operators in the aisles were involved [20]. Yang et al. studied a flow-picking problem where the existing picking list was updated in real-time, thus providing efficient solutions for e-commerce warehouses [29]. Besides, Mahmoudinazlou et al. addressed the fluctuating order arrivals in a single-block warehouse with an autonomous picking machine [30].

Although the existing research has considered numerous constraints, objectives, and dynamics for multiple warehouse order-picking problems, few focus on operator well-being. Besides, most studies assumed that the picking list could be fulfilled through one-tour picking, ignoring the situation in which the required parts of a picking list may exceed the maximum capacity of one tour. In summary, dynamic multi-tour order picking has rarely been considered by previous research, stressing the need for further research.

2.2. DRL-based methods for production optimization

In recent years, DRLs have been widely applied across various dynamic production scenarios [31], such as cyber–physical systems [32], process control [33], resource management [34], and production scheduling [35]. Among these scenarios, production scheduling based on DRLs has attracted significant attention [15]. These related studies tend to employ deep Q-network (DQN) and its variants to select dispatching heuristics [36], capture dynamic characteristics [37], and handle disturbance events [38]. Compared to research on production scheduling, the resource management or task allocation in cloud environments preferred to adopt actor-critic-based DRLs to solve the resource allocation [39], service composition [40], and workflow scheduling [41]. Nonetheless, most research only considered directly applying existing DRL algorithms to their optimization problems without further algorithmic improvements. To this end, Kool et al. proposed the attention-based policy trained by DRL to effectively capture the intercorrelations among features for combinatorial optimization problems [17]. Furthermore, some researchers found that utilizing the symmetry of combinatorial optimization problems and attention-based methods can further improve policy generalization [18,42]. Nonetheless, these papers are only concerned with highly abstract models such as TSPs rather than practical production problems. Despite the increasing interest in applying DRLs across various production fields, to the best of our knowledge, little research resolves order-picking problems using DRLs. For example, Cals et al. solved the online order batching problem with a DRL-based method and argued that the DRLs could develop strategies that perform better than heuristics [9]. Mahmoudinazlou et al. addressed the dynamic order-picking problem with a neural policy trained by DQN, and the experimental results validated the outstanding performance over benchmark methods [30]. Furthermore, Dunn et al. solved the order-picking problem with an attention-based policy trained by DRL [43]. Nonetheless, they rarely considered further improving the algorithm performance based on the characteristics of practical order-picking problems, and their studied problems were not consistent with the multi-tour order-picking in an automotive-part warehouse. Therefore, developing an effective DRL-based approach with strong generalization and rapid solving speed for the studied problem is necessary.

3. System model of multi-tour order picking

This section first introduces the multi-tour order-picking model under the background of an automotive-part warehouse, followed by the formulated mathematical model with decision variables, objectives, and constraints. Subsequently, the optimization problem is reformulated as an MDP with specialized definitions of states, actions, and rewards. As part of this research, Scania AB, a Swedish automotive manufacturer, was viewed as the test bed, contributing to the problem description and the research data.

3.1. System description

As indicated by Fig. 1, multi-tour order picking describes an abstract model where an operator gathers automotive parts from shelves according to the order requirement in a warehouse's corridor. Suppose n shelf locations store multiple types of automotive parts in a corridor. The ith location ($i \in \{1, 2, \dots, n\}$) is described by four properties of $\{Lc_i, Tp_i, Wt_i, Nm_i\}$, specifying the location coordinates, the part type, the part weight, and the part quantity at this location, respectively. The transfer point represents a location where the operator starts the picking task and unloads the parts from the cart. The transfer point can also be regarded as the start and the end of a tour. One tour includes leaving from the transfer point, picking up parts from shelves, and returning to the transfer point. The cart has to be emptied when its current weight exceeds the maximum weight threshold, or the operator's workload of this tour reaches the maximum limitation. Therefore, one order-picking task may contain multiple tours among the transfer point and shelf locations. An order-picking task is completed when all parts required by the order have been picked and delivered to the transfer point, considering the constraints of cart weight and operator's workload.

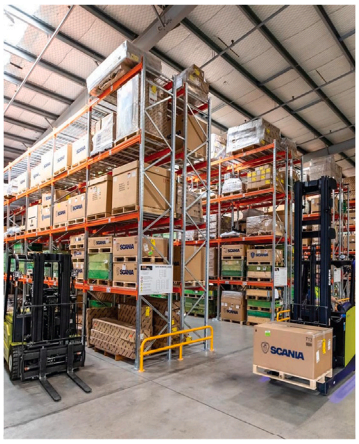
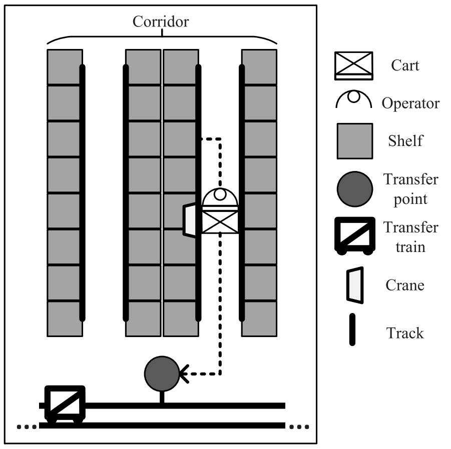

Fig. 1. Order picking in an automotive-part warehouse. The transfer train will carry picked parts to other production departments at each transfer point.

3.1.1. Constraints of cart weight and operator's workload

In each tour, the cart must be emptied at the transfer point when (1) the current cart weight reaches the maximum threshold or (2) the operator's workload exceeds the maximum limitation. Suppose the sequential set of picked parts in the cart is noted as $\mathcal{T}$. For each picked part $\tau \in \mathcal{T}$, $\{u_\tau, \mathcal{L}_\tau\}$ specify the part weight and its corresponding location, respectively. Let Λ and Υ be the thresholds for the maximum weight and the maximum workload, respectively. Then, the constraints for each tour are described by (1) and (2), respectively.

$$\sum_{\tau\in\mathcal{T}} u_\tau \le \Lambda \tag{1}$$

$$\sum_{\tau,\tau'\in\mathcal{T}} \mu_f \cdot \|\mathcal{L}_\tau - \mathcal{L}_{\tau'}\|_2 \cdot \sum_{\tau\in\mathcal{T}} u_\tau \cdot g \le \Upsilon \tag{2}$$

where $\mathcal{L}_{\tau'}$ represents the location of the next part τ', μ_f represents the rolling workload coefficient, and g denotes the gravity acceleration coefficient. Eq. (2) also means that the energy consumption of one tour should not exceed a certain threshold.

3.1.2. Time consumption in one tour

The optimization objective of the order-picking problem is to minimize the order-picking time (OPT), which mainly contains the time consumption of several tours. The ath tour includes four categories of time consumption, namely the visiting time $T_v^{(a)}$, the picking time $T_p^{(a)}$, the loading time $T_l^{(a)}$, and the unloading time $T_u^{(a)}$. Suppose the ath tour needs to pick $|\mathcal{T}|$ parts, and $\tau \in \mathcal{T}$ represents one part. The visiting time $T_v^{(a)}$ is defined by (3).

$$T_v^{(a)} = \sum_{\tau,\tau'\in\mathcal{T}} \frac{\|\mathcal{L}_\tau - \mathcal{L}_{\tau'}\|_2}{\mathcal{V}} \tag{3}$$

where $\mathcal{V}$ denotes the cart's average moving speed. The picking time $T_p^{(a)}$ represents the time consumption of taking the part off the shelf, as defined by (4).

$$T_p^{(a)} = \sum_{\tau\in\mathcal{T}} \{C(\tau)\cdot(t_w + Q_c\cdot u_\tau) + \{1 - C(\tau)\}\cdot u_\tau \cdot t_p\} \tag{4}$$

where $C(\tau) \in \{0,1\}$ represents the crane function to indicate whether to utilize the crane to assist the picking ($C(\tau) = 1$ represents utilizing the crane; otherwise $C(\tau) = 0$), Q_c denotes the picking time of the crane per unit weight, t_w means the startup time of the crane, t_p denotes the picking time of the operator per unit weight. The loading time $T_l^{(a)}$ is the time consumption of loading the picked parts into the cart, as defined by $|\mathcal{T}|\cdot t_l$, where t_l denotes the average loading time for an automotive part. The unloading time $T_u^{(a)}$ refers to the time consumption of unloading the parts in the cart at the transferring point, as defined by $\sum_{\tau\in\mathcal{T}} u_\tau \cdot t_u / \Lambda$, where t_u represents the unit unloading time for each part, and Λ denotes the maximum weight threshold.

3.1.3. Dynamics analysis and assumptions

In the studied problem, the dynamics mainly reflects in three aspects:

(1) The order requirement. The required part types and part quantities may change dynamically. For example, according to a survey of Swedish automotive-part warehouses, an order may contain 45 ∼ 75 automotive parts and 10 ∼ 15 part types.

(2) The part distribution (also known as the part placement). The part distributions in different corridors indicate differences. For example, the part types, quantities, and locations can be distinguishable in different corridors.

(3) The threshold values. The maximum cart weight $\mathcal{A}$ and the maximum workload Υ may be diverse in different corridors. Practical factors such as operator strengths & genders, cart differences, and warehouse requirements influence their values.

Due to the complicated production environment, the algorithm is expected to handle these dynamics effectively and provide solutions within a fast response time. For simplicity but without losing generality, we make the following assumptions:

(1) For one order-picking task, the types and quantities of parts stored on shelves are adequate compared to the required parts of an order.

(2) The replenishment of parts on the shelves only occurs after an order-picking task is completed.

(3) The maximum capacity of the cart is proportional to the maximum weight of the cart. The cart cannot carry more automotive parts if the current weight of the cart has reached the maximum threshold.

(4) The maximum weight of a part cannot exceed the maximum cart weight threshold, and the workload of delivering one part from a shelf to the transfer point cannot exceed the maximum workload constraint.

(5) The number of parts at one shelf location is usually more than one.

3.2. Mathematical formulation

This subsection formulates the multi-tour order-picking problem as a mixed integer programming model based on the above-mentioned elements and concepts. Suppose $x_{i,j,a}$ represents whether the route $i \to j$ in the ath tour is visited, $y_{k,i,a}$ means whether the part of the kth type is picked up at the ith location in the ath tour, $\mathcal{O}_a$ represents the set of required parts in the ath tour, and z represents the number of tours in one order-picking task. Subsequently, the multi-tour order-picking model is defined as follows:

$$\min F(\mathbf{X},\mathbf{Y},\mathbf{Z},\mathcal{O}_a) = \sum_a \{T_v^{(a)} + T_p^{(a)} + T_l^{(a)} + T_u^{(a)}\} \tag{5}$$

$$\propto \min \sum_a \sum_i \sum_j \|\mathcal{L}_i - \mathcal{L}_j\|_2 \cdot x_{i,j,a} \tag{6}$$

$$s.t.\quad z \in \{1,2,3,\dots\} \tag{7}$$

$$x_{i,j,a} \in \{0,1\}, \forall i,j \in \{0,1,\dots,n\}, \forall a \in \{1,2,\dots,z\} \tag{8}$$

$$y_{k,i,a} \in \{0,1\}, \forall k \in \{1,2,\dots,m\}, \forall i \in \{0,1,\dots,n\}, \forall a \in \{1,2,\dots,z\} \tag{9}$$

$$\sum_a |\mathcal{O}_a| = |\mathcal{O}|, \mathcal{O}_a \neq \emptyset \tag{10}$$

$$\sum_i \sum_k u_k \cdot y_{k,i,a} \leq \Lambda, \forall a \in \{1,2,3,\dots,z\} \tag{11}$$

$$\sum_i \sum_j \sum_k \mu_f \cdot \|\mathcal{L}_i - \mathcal{L}_j\|_2 \cdot x_{i,j,a} \cdot u_k \cdot g \cdot y_{k,i,a} \leq \Upsilon, \forall a \in \{1,2,\dots,z\} \tag{12}$$

$$\sum_i x_{i,j,a} = 1, \forall j \in \{0,1,\dots,n\}, \forall a \in \{1,2,\dots,z\} \tag{13}$$

$$\sum_j x_{i,j,a} = 1, \forall i \in \{0,1,\dots,n\}, \forall a \in \{1,2,\dots,z\} \tag{14}$$

$$x_{i,j,a} + x_{j,i,a} \leq 1, \forall i,j \in \{0,1,\dots,n\}, \forall a \in \{1,2,\dots,z\} \tag{15}$$

$$\sum_k \sum_i y_{k,i,a} = |\mathcal{O}_a|, \forall a \in \{1,2,\dots,z\} \tag{16}$$

$$\sum_i \sum_j x_{i,j,a} \leq \sum_k \sum_i y_{k,i,a}, \forall a \in \{1,2,\dots,z\} \tag{17}$$

where (5) represents the objective function of one order-picking task's order-picking time (OPT). Eq. (6) denotes the simplified objective that is proportional to (5) when the problem instance is given. Eqs. (7)∼(10) denote the range of the decision variables. Eqs. (11)∼(12) are constraints for the cart weight and the operator's workload. Eqs. (13)∼(14) means that each location is only visited once per tour. Eq. (15) avoids circular subsets in a tour, (16) maintains that the quantity of picked parts equals the required quantity in each tour, and (17) means that one or more than one part should be picked up in each visited location. The detailed meanings of symbols and variables used in the formulation are summarized by Table 1.

3.3. MDP reformulation

To solve the multi-tour order picking model with DRLs, we reformulate this optimization model as an MDP consisting of a tuple of $\{A, S, R, \pi, \gamma\}$, as introduced by Fig. 2. $a_t \in A$ represents the action in the tth step, and we define the action a_t as the index of one location. $a_t = 0$ represents returning to the transfer point, which also splits the tours. Consequently, the action space $|A|$ equals $n+1$. $s_t \in S$ represents the state containing the environmental statuses of the cart and the shelf locations in the tth step. The state s_t can be described as $\{a_{t-1}, Wl_t, Cw_t, Ol_t, \Omega_1, \Omega_2, \dots, \Omega_n\}$, where a_{t-1}, Wl_t, Cw_t, and Ol_t specify the last selected action, the current percentage of workload, the current percentage of cart weight, and a list of left parts in the current order, respectively. For $i \in \{1,2,\dots,n\}$, $\Omega_i = \{Lc_i, Tp_i, Wt_i, Nm_i\}$ represent the location coordinate, the part type, the part weight, and the part quantity in the ith location. $r_t \in R$ represents the reward, and we define the reward function by $-1 \cdot F(\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathcal{O}_a)$ if t is the terminal step; otherwise, $r_t \leftarrow 0$. π denotes the policy, and $a_t = \pi(s_t)$. DRL usually fits the policy with neural network models. $\gamma \in (0,1)$ represents the discounted factor to compute the expected return in DRL. The state transition is triggered when the operator has just finished one picking activity or returned to the transfer point.

4. Proposed attention-aware DRL-based planning approach

4.1. Approach framework

As illustrated by Fig. 3, the framework mainly contains an encoder–decoder-structured policy model and the training algorithm of entropy-PPO. The policy input is the state information from the formulated MDP

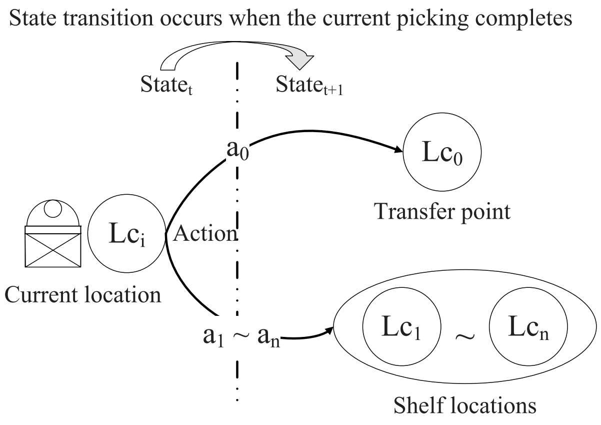

Fig. 2. The MDP reformulation for the studied problem.

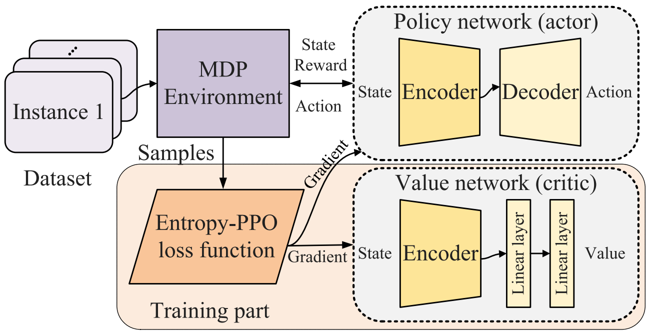

Fig. 3. Mainframe of the proposed ADRL. The proposed approach utilizes an encoder–decoder neural network as the policy network, and the policy is trained under the actor-critic framework. The pink area denotes the training part for updating the policy network.

environment, and the policy output represents a solution L for this instance. The solution $L = \{l_1, l_2, l_3 \dots\}$ contains a sequential combination of candidate locations. The Section 4.2 introduces the policy model, and Section 4.3 illustrates the training algorithm.

4.2. Encoder–decoder-structured policy model

As shown in Fig. 4, the proposed end-to-end policy model for the multi-tour order-picking problem contains an encoder and a decoder. The encoder converts the original information of shelf locations into node embeddings with three attention layers. Then, the decoder generates the picking solution L step by step with the node embeddings from the encoder and the real-time context information. We introduce the structure of the policy model following the sequence from the input to the output.

4.2.1. Node representations for picking locations

Each location in the corridor is regarded as a node, and the node representation of the ith location is represented by $x_i, i \in \{0,1,2,3,\dots,n,n+1\}$. x_i is defined by (18).

$$x_i = \begin{cases} W_1 \cdot [Lc_0] + b_1, & i = 0, \\ W_2 \cdot LN([Lc_i, Tp_i, Wt_i, Nm_i]) + b_2, & \forall i \in \{1,2,3,\dots,n\}, \\ W_3 \cdot [-1 \cdot Lc_0] + b_3, & i = n+1. \end{cases} \tag{18}$$

where Lc_i, Tp_i, Wt_i, Nm_i specify location coordinates, the part type, the part weight, and the part quantity in the ith location. $LN(\cdot)$ represents the layer normalization that scales the data within the scope of $0 \sim 1$. The transfer point is denoted by $i = 0$, and we add an extra location

Table 1
Primary symbols and variables.

Notations	Explanations
m	The number of part types.
n	The number of shelf locations.
τ	The automotive part to be picked.
$\mathcal{T}$	The list of parts to be picked in a tour.
u_τ	The part weight.
$\mathcal{L}_\tau$	The part coordinate.
Λ	The threshold for the maximum cart weight.
Υ	The threshold for the maximum workload.
μ_f	The rolling workload coefficient.
g	The gravity acceleration coefficient.
$\mathcal{V}$	The average cart speed.
$C(\tau)$	The crane function for the part τ.
Q_c	The picking efficiency of the crane.
t_w	the startup time of the carne.
t_p	The unit picking time.
t_l	The average loading time for a part.
t_u	The unit unloading time for each part.
$T_v^{(a)}$	Visiting time of the ath tour.
$T_p^{(a)}$	Picking time of the ath tour.
$T_l^{(a)}$	Loading time of the ath tour.
$T_u^{(a)}$	Unloading time of the ath tour.
$x_{i,j,a}$	A decision variable determining whether the route $i \rightarrow j$ is visited.
$y_{k,i,a}$	A decision variable determining whether the part of the kth type is picked up at the ith location in the ath tour.
$\mathcal{O}_a$	The set of required parts in the ath tour.
z	The number of tours in one order-picking task.
a	The index for the tour.
i, j	The indexes for the shelf locations.
k	The index for the part type.
t	The index of the step in MDP.
s_t	The state in the tth step.
a_t	The action in the tth step.
Wl_t	The percentage of the operator's workload in the tth step.
Cw_t	The percentage of the cart weight in the tth step.
Ol_t	The list of the remaining parts in the tth step.
Tp_i	The type of the stored parts in the ith location.
Nm_i	The number of parts stored in the ith location.
Lc_i	The coordinate of the ith location.
Wt_i	The weight of a part stored in the ith location.

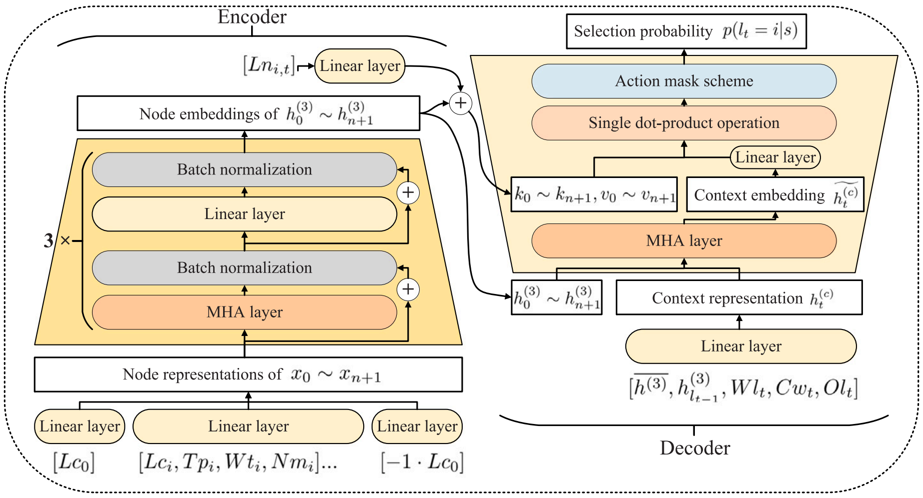
Encoder
$[Ln_{i,t}]$
Linear layer
Node embeddings of $h_0^{(3)} \sim h_{n+1}^{(3)}$
Batch normalization
Linear layer
Batch normalization
MHA layer
3×
Node representations of $x_0 \sim x_{n+1}$
Linear layer
Linear layer
Linear layer
$[Lc_0]$
$[Lc_i, Tp_i, Wt_i, Nm_i]$...
$[-1 \cdot Lc_0]$
Selection probability $p(l_t = i|s)$
Action mask scheme
Single dot-product operation
Linear layer
$k_0 \sim k_{n+1}, v_0 \sim v_{n+1}$
Context embedding $\widetilde{h_t^{(c)}}$
MHA layer
$h_0^{(3)} \sim h_{n+1}^{(3)}$
Context representation $h_t^{(c)}$
Linear layer
$[\overline{h^{(3)}}, h_{l_{t-1}}^{(3)}, Wl_t, Cw_t, Ol_t]$
Decoder

Fig. 4. Encoder–decoder-structured policy model. The left part shows the framework of the encoder, and the right part indicates the structure of the decoder.

($i = n + 1$) to align the solution length during training. Specifically, when the order-picking task is completed, the policy will continuously select x_{n+1} until the length of the solution L is aligned with the batch length. We define this extra location rather than using x_0 to allow the policy model to differentiate between splitting the order and padding the batch length. Ablation experiments validate the effectiveness of this operation. $w_1 \sim w_3, b_1 \sim b_3$ represent three groups of neural networks that project the location information into node representations with consistent tensor dimensions.

4.2.2. Encoder structure

Based on the calculations reported by (18), the node representations are delivered into the encoder module. The encoder contains three attention layers, each layer containing one MHA layer and one linear layer. For $\forall i \in \{0, 1, 2, \ldots, n+1\}, \forall l \in \{1, 2, 3\}$, the hidden embedding $\hat{h}_i^{(l)}$ of the ith node in the lth layer is defined by (21).

$$\hat{h}_i^{(l)} = BN(h_i^{(l-1)} + MHA_i^{(l)}(h_0^{(l-1)}, \ldots, h_{n+1}^{(l-1)})), \tag{19}$$

$$\widetilde{h_i^{(l)}} = ReLU(W_1^{(l)} \hat{h}_i^{(l)} + b_1^{(l)}), \tag{20}$$

$$h_i^{(l)} = BN(\hat{h_i^{(l)}} + \widetilde{h_i^{(l)}}), \tag{21}$$

where $BN(\cdot)$ denotes batch normalization, $ReLU(\cdot)$ represents the ReLU activation function. $\hat{h_i^{(l)}}, \widetilde{h_i^{(l)}}$ are the hidden embeddings calculated by the MHA layer and the linear layer, respectively. The plus operations in (19) and (21) represent the skip connections. $MHA_i^{(l)}(\cdot)$ represents the MHA layer extracting intercorrelations among nodes with eight attention heads, which is explained by (22).

$$MHA_i^{(l)}(h_0^{(l-1)}, \dots, h_{n+1}^{(l-1)}) = \sum_{o=1}^{8} W_o^{(head)} \cdot Head_{i,o}^{(l)}(h_0^{(l-1)}, \dots, h_{n+1}^{(l-1)}) \tag{22}$$

where $Head_{i,o}^{(l)}, o \in \{1, 2, \dots, 8\}$ represents the calculation of one attention head, $W_o^{(head)}$ denotes the neural weight parameter that connects all attention heads. For $\forall o \in \{1, 2, \dots, 8\}$, the $Head_{i,o}^{(l)}$ is defined by the dot-product calculation of query, key, and value [44]. In the lth attention layer, the query, key, and value for the ith node are represented by (23).

$$q_i = W_q h_i^{(l-1)}, k_i = W_k h_i^{(l-1)}, v_i = W_v h_i^{(l-1)} \tag{23}$$

where W_q, W_k, W_v are neural weight parameters. For $\forall o \in \{1, 2, \dots, 8\}$, $\forall i, j \in \{0, 1, 2, \dots, n+1\}$, the $Head_{i,o}^{(l)}$ is defined by (24).

$$Head_{i,o}^{(l)}(h_0^{(l-1)}, \dots, h_{n+1}^{(l-1)}) = \sum_{j=0}^{n+1} SoftMax(\frac{q_i k_j^T}{\sqrt{d_k}}) v_j. \tag{24}$$

where $SoftMax(\cdot)$ denotes the SoftMax function, d_k is the dimension of the key. After three attention layers, the original node representations $\{x_0, x_1, \dots, x_{n+1}\}$ are converted into node embeddings $\{h_0^{(3)}, h_1^{(3)}, h_2^{(3)}, \dots, h_{n+1}^{(3)}\}$.

4.2.3. Context representation

After the encoding procedure, the $n+2$ embeddings are sent to the decoder with the context information. The context information reflects the real-time representation of the cart and the operator. The context representation $h_t^{(c)}$ in the tth step is defined by (26).

$$h_t^{(c)} = W_4 \cdot [\overline{h^{(3)}}, h_{l_{t-1}}^{(3)}, Wl_t, Cw_t, Ol_t] + b_4 \tag{25}$$

where $\overline{h^{(3)}}$ represents the average value of node embeddings, $h_{l_{t-1}}^{(3)}$ denotes the node embedding of the selected location in the $(t-1)$-th step. Wl_t, Cw_t specify the current percentages of the operator's workload and the cart weight compared to their maximum thresholds. Ol_t notes a list of the remaining parts in the current step. A linear layer with parameters W_4, b_4 projects the original information into the context representation $h_t^{(c)}$, thus aligning the tensor dimensions with node embeddings.

4.2.4. Decoder structure

With both the embeddings from the encoder and the context representation, the decoder employs an MHA layer and a single dot-product operation to generate the choosing probabilities of locations. In the tth step, MHA first integrates the context representation into node embeddings. The context embedding $\widetilde{h_t^{(c)}}$ is calculated by (26).

$$\begin{aligned}\widetilde{h_t^{(c)}} &= \sum_{o=1}^{8} W_o^{(head)} \cdot Head_o(h_t^{(c)}, h_0^{(3)}, \dots, h_{n+1}^{(3)}) \\ &= \sum_{o=1}^{8} W_o^{(head)} \cdot [\sum_{i=0}^{n+2} softmax(\frac{q_c k_i^T}{\sqrt{d_k}}) v_i] | o.\end{aligned} \tag{26}$$

where the symbol $[\cdot]|o$ represents the calculation for the oth attention head. The q_c, k_i, v_i in (26) are defined by (27).

$$q_c = W_c h_t^{(c)} \tag{27}$$

$$k_i = W_k h_i^{(3)} + W_{kd} \cdot [Ln_{i,t}] \tag{28}$$

$$v_i = W_v h_i^{(3)} + W_{vd} \cdot [Ln_{i,t}] \tag{29}$$

where $Ln_{i,t}$ denotes the number of the remaining parts in the ith location in the tth step, W_{kd}, W_{vd} are neural weight parameters. We formulate this key and value in (28) and (29) to integrate the real-time status of location information. This dynamic decoding calculation enables the policy to perceive the real-time changes in the number of remaining parts without the necessity to re-execute the encoding procedure. Suppose that s represents the initial state of an instance, then we construct a query q_l by $W_c \widetilde{h_t^{(c)}}$ for a single dot-product to produce the probability $p(l_t = i|s)$ by (30).

$$p(l_t = i|s) = softmax(tanh(\frac{q_l k_i^T}{\sqrt{d_k}}) \cdot M_{t,i}) \tag{30}$$

where $M_{t,i} \in \{1, -\infty\}$ denotes the output mask converting the selection probability of the next infeasible action to negative infinity, and k_i has been defined by (28). Then, the selection probabilities of all locations in the tth step can be calculated, namely $p(\cdot, t|s) = \{p(l_t = 0|s), p(l_t = 1|s), \dots, p(l_t = n+1|s)\}$. During training, the next action is selected based on these probabilities to leave some exploration space (Selected by sampling). During evaluation, we directly choose a location with the highest probability (Greedy selection).

4.3. Training algorithm: Entropy-PPO

The distribution of training instances derived from the practical warehouse is usually unbalanced and biased, posing challenges to training a high-quality policy model with DRL. We introduce the Entropy-PPO algorithm under the actor–critic architecture to encourage more exploration and improve training efficiency. The policy model serves as the actor. To construct a reasonable critic whose representation ability can be equivalent to the actor's, we define the critic network by (33).

$$\hat{h_{critic,i}} = MHAencoder(x_0, x_1, \dots, x_{n+1}), \forall i \in \{0, 1, \dots, n+1\}, \tag{31}$$

$$h_{critic,i} = W_{critic} \cdot \hat{h_{critic,i}} + b_{critic}, \forall i \in \{0, 1, \dots, n+1\}, \tag{32}$$

$$V(s) = \sum_{i=0}^{n+1} h_{critic,i}/(n+2), s = \{x_0, x_1, \dots, x_{n+1}\} \tag{33}$$

where $MHAencoder(\cdot)$ represents the same encoding procedure as described in Section 4.2.2, W_{critic}, b_{critic} represent neural parameters, and $V(s)$ denotes the value estimation for the initial state s. We train the critic network and the policy model together based on the loss function defined by (34).

$$\begin{aligned}Loss = &- E_{p(\cdot|s)}\{\min\{ratio(p(\cdot|s)), ratio^{(clip)}(p(\cdot|s))\} \cdot (R(s) - V(s))\} \\ &+ \lambda_v \cdot E_s\{(R(s) - V(s))^2\} \\ &+ \lambda_e \cdot \sum_{p(\cdot|s)} p(\cdot|s) \cdot \log(p(\cdot|s))\end{aligned} \tag{34}$$

where $ratio(p(\cdot|s)) = \frac{p(\cdot|s)}{p'(\cdot|s)}$ represents the probability ratio of the current policy $p(\cdot|s)$ and an old policy $p'(\cdot|s)$, $ratio^{(clip)}(p(\cdot|s)) = clip(ratio(p(\cdot|s)))$, $1-\epsilon, 1+\epsilon$ represents constraining $ratio(p(\cdot|s))$ within the range of $(1-\epsilon, 1+\epsilon)$, and $\epsilon \in (0, 1)$. $E_{p(\cdot|s)}\{\cdot\}$ and $E_s\{\cdot\}$ represent expectations for $p(\cdot|s)$ and s, respectively. $\lambda_v, \lambda_e \in (0, 1)$ denote two factors. Eq. (34) contains three items: The first one is the original PPO loss, the second one represents the mean square error for training $V(s)$, and the third one represents the entropy regularization. The training algorithm is reported by Algorithm 1, where lines $1 \sim 7$ represent initialization, lines $10 \sim 13$ denote batch-based sampling from the MDP environment with the policy model, line 14 means updating the neural parameters based on the Adam optimizer.

5. Case study

This section presents the comparison results on optimization objectives, solving time, and other detailed problem variables. First, the configurations for experimental datasets and comparison algorithms are introduced. Subsequently, the comparison results on the training and evaluation datasets are analyzed. Finally, detailed comparisons of solution quality are implemented to exhibit the differences in the algorithms.

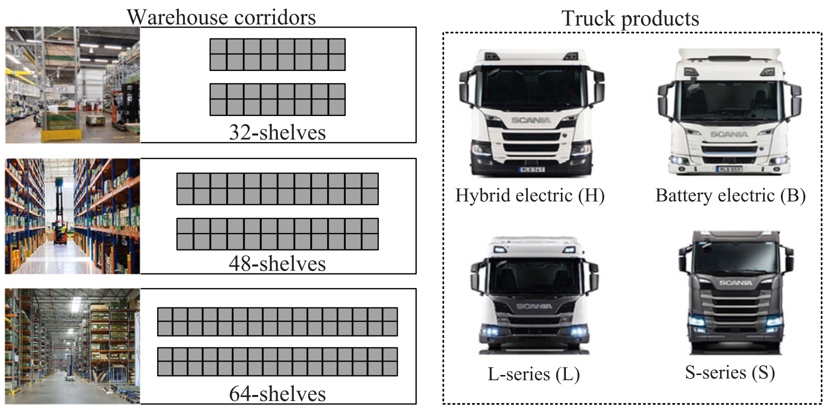

Fig. 5. Demonstration of the automotive-part warehouse. The warehouse corridors are divided into three categories according to the shelf scales of 32, 48, and 64. The picked parts are mainly the components for four types of truck productions, namely hybrid electric, battery electric, L-series, and S-series.

Algorithm 1: Training Algorithm

```
1  Initialize:
2  $\theta$: The policy model with neural parameter $\theta$;
3  $\phi$: The critic model with neural parameter $\phi$;
4  $N$: The number of training epochs;
5  $M$: The number of steps;
6  $B$: The number of training batches;
7  $D$: The training dataset;
8  foreach $epoch = 1 : N$ do
9      foreach training batch sampled from $D$ do
10         Sample an instance $s_i$ randomly from the training batch
             for $\forall i \in [1, B]$;
11         Generate the solution $L_i$ and calculate the output
             probability $p(\cdot|s_i)$ for $\forall i \in [1, B]$;
12         Calculate the value $V(s_i)$ for $\forall i \in [1, B]$;
13         Obtain the episode reward $R(s_i)$ for $\forall i \in [1, B]$;
14         $\theta, \phi \leftarrow Adam(\theta, \phi, \Delta Loss)$;
15     end
16 end
```

5.1. Experimental configuration

Dataset. The training and evaluation datasets are mainly derived from the practical warehouses of Scania AB, a Swedish manufacturing company that provides automotives for heavy transport applications (https://www.scania.com/). Thousands of order-picking instances are generated based on the practical data distribution in warehouse corridors. Some of the parameters in the datasets are masked due to confidentiality issues. According to the differences in corridor scales, these instances are divided into three categories: 32-shelves, 48-shelves, and 64-shelves, as shown in Fig. 5. For example, the 32-shelves represents an operator collecting automotive parts in a corridor with 32 shelves distributed at different locations. The demonstration for the part types is reported by Table 2. In general, an order contains 45 ~ 75 parts for one order-picking task. The cart weight constraint is within the range of 800 kg ~ 900 kg, and the workload constraint tends to be within the range of 650kJ ~ 750kJ. The average picking time consumptions are usually 5s ~ 15s and 10s ~ 20 s for picking by hand and picking with a crane, respectively. In these instances, the environmental parameters may fluctuate according to specific requirements, and we uploaded these parameters online in the format of Python Numpy Zipfile.[1] The training and evaluation instances are divided according to the ratio of $10 : 1$. In each of the three datasets of 32-shelves, 48-shelves, and 64-shelves, the training dataset contains 2000 instances, and the evaluation dataset contains 200 instances.

Compared algorithms. To make comprehensive comparisons, we implement three categories of compared algorithms, i.e., EAs, DRLs, and attention-based methods. EAs and DRLs are popular planning methods for order picking problems, which have attracted considerable interest recently. Furthermore, we also compared the existing attention-based methods to strengthen the significance of the proposed improvements.

EAs contain seven popular algorithms, namely artificial bee colony algorithm (ABCA), bee algorithm (BA), ant colony optimization (ACO), differential evolution (DE), particle swarm optimization (PSO), genetic algorithm (GA), and random differential evolution (RNDE). These EAs share the same evolutionary encoding scheme in which all selected locations for one episode are encoded as Real numbers in the chromosome. In contrast, the mutation operators and the selection mechanism separate these diverse EAs. For all these EAs, the evolutionary generation is 100, and the population size is 30. For ABCA, the employed bee exploration probability is 0.5, and the onlooker bee exploration probability is 0.5. For BA, the number of selected sites is 5, and the initial patch size is 0.1. For ACO, the pheromone evaporation rate is 0.85, and the heuristic factor is 3. For DE and RNDE, the scaling factor is 0.6, and the crossover rate is 0.5 (RNDE selects the differential equations randomly). For PSO, the acceleration coefficient is 0.8, and the velocity limit is 0.1. For GA, the mutation rate is 0.3, and the crossover rate is 0.4.

DRLs include eight common algorithms, namely proximal policy optimization (PPO), soft actor-critic (SAC), double deep Q-network (DDQN), deep Q-network (DQN), DQN with fixed Q targets (FDQN), dueling double deep Q-network (D3QN), advantage actor-critic (A2C), and asynchronous advantage actor-critic (A3C). The short descriptions of these DRL algorithms are reported in the Appendix section.

To validate the effectiveness of the proposed three improvements on the studied problem, we conduct three ablation versions, namely attention-based method-1 (AM1), attention-based method-2 (AM2), and attention-based method-3 (AM3), which remove the extra location, the dynamic decoding, and the entropy-PPO, respectively. For AM1, we utilize x_0 to pad the remaining length of solutions in the training batch as a straightforward operation to serve as a comparison. For AM3, the REINFORCE with a baseline replaces the entropy-PPO to serve as a comparison. Furthermore, we also implement a recently proposed attention-based method that utilizes the symmetry of instances to improve the training efficiency of attention-based methods. This method is named SYMNCO, and the detailed descriptions of SYMNCO can be found in [18]. For all these attention-based methods, the policy model shares the same structure as ADRL described in this paper. The training parameters of these DRLs and attention-based methods are reported

[1] https://github.com/MISTCARRYYOU/ADRL4ScaniaOrderPicking.git

Table 2
Some demonstrations of automotive parts. L, S, B, and H represent four truck products: L-series, S-series, Battery electrical, and Hybrid electrical.

Types	Weights	Products	Average picking time	
			By hands	By a crane
Electrical motor transmission	10 kg	B, H	5 s	10 s
Diesel engine 7 l transmission	10 kg	L, H	5 s	10 s
Diesel engine 9 l transmission	15 kg	L	10 s	15 s
Diesel engine 13 l transmission	20 kg	S	10 s	15 s
Diesel engine 16 l transmission	25 kg	S	15 s	20 s
6 shift gearbox	150 kg	L, S	/	20 s
8 shift gearbox	180 kg	B, H	/	20 s
Right break L	10 kg	L, S, B, H	5 s	10 s
Right break S	10 kg	L, S, B, H	5 s	10 s
Wheel pairs	50 kg	L, S, B, H	/	20 s
Battery pack 90kwh	90 kg	B, H	/	20 s
Screw pack D	1 kg	L, S, H	5 s	10 s
Screw pack El	1 kg	B, H	5 s	10 s

Table 3
The training hyper-parameters of DRLs.

Hyper-parameters	PPO	SAC	DDQN	DQN	FDQN	D3QN	A2C	A3C	AM1	AM2	AM3	SYMNCO	ADRL
Training episodes	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
Training instances	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000
Learning rate (actor)	0.02	0.0001	/	/	/	/	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
Learning rate (critic)	0.02	0.0001	0.001	0.001	0.001	0.001	0.0001	0.0001	/	/	/	/	/
Buffer size	/	40 000	40 000	40 000	40 000	40 000	/	/	/	/	/	/	/
Batch size	256	256	256	256	256	256	256	256	256	256	256	256	256
Discounted factor	0.95	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1	1	1	1	1
Hidden dimensions	512, 64	512, 64	512, 64	512, 64	512, 64	512, 64	512, 64	512, 64	128	128	128	128	128
Soft updating rate	1	0.005	0.01	0.01	0.01	0.01	0.005	0.005	1	1	/	/	1

in Table 3. During training, we save the agent policy with the best ever-seen performances in each dataset. For PPO, SAC, A2C, and A3C, we apply the soft updating strategy to keep the training smoothly. Specifically, only 1% of the parameters of the critic model will be updated during training in each iteration.

Running environment. The programming language is Python 3.8.0. The deep learning platform is based on PyTorch 1.6.0, and the DRL environment extends the base Class of the OpenAI gym.Env. All experiments are executed on a Ubuntu server with a CPU of Intel Core i7-4790 CPU@3.60 GHz and a GPU of NVIDIA GTX 1080Ti. The solving time is calculated based on the result on a single-core CPU.

5.2. Comparison results with DRLs and EAs

The comparison results of DRLs on the training datasets are reported by Table 4, where table cells represent the mean and the standard deviation of obtained objectives of 2000 instances in each dataset. The mean can reflect the average performance, and the standard deviation denotes the performance fluctuation. The training results indicate that PPO performs nearly the best among the eight DRLs. Noticing that these eight DRLs have the same neural structure. The results illustrate the superiority of the PPO training algorithm when facing dynamic multi-tour order-picking problems. Considering these attention-based methods, ADRL achieves the lowest mean and standard deviation in datasets of 32-shelves and 48-shelves, while AM3 performs best in 64-shelves. The standard deviation can reflect the policy robustness as each algorithm utilizes the sampling strategy to select actions during training. The proposed ADRL has the lowest standard deviations compared to other attention-based methods, which means that ADRL's policy has higher confidence than other attention-based methods. In the 64-shelves dataset, AM3 outperforms other attention-based methods. Combining the evaluation results, we guess AM3 may overfit this training dataset. This phenomenon illustrates the benefits of Entropy-PPO when facing this order-picking dataset. The results in Table 4 are collected using the sampling strategy during training, which cannot comprehensively reflect the algorithms' performance. The instances to be solved in practical scenarios are usually unseen to the training. Consequently, the evaluation results are more representative of the algorithms' performance.

Subsequently, the evaluation results of EAs, DRLs, and attention-based methods are compared on evaluation datasets. Each algorithm of DRLs and attention-based methods utilizes the greedy strategy to select actions in the evaluation datasets. As reported by Table 5, considering the seven EAs, RNDE and PSO can find lower objectives than other EAs regarding the mean values. Considering the eight DRLs, SAC indicates the apparent advantages regarding the mean values. Considering the thirteen learning-based methods, the proposed ADRL achieves the best performance regarding the mean and the standard deviation in datasets of 48-shelves and 64-shelves. In 32-shelf, ADRL achieves the second-best performance, but the gap between ADRL and the best-performing one (AM3) is only 0.7%. Compared to these twelve learning-based methods, ADRL can outperform them at most by 27.5%, 33.8%, and 40.6% in datasets of 32-shelves, 48-shelves, and 64-shelves, respectively. Additionally, the results also confirm the indispensability of the proposed three improvements regarding the comparisons between ADRL and AM1 $\sim$ AM3. Although the performances of all these learning-based methods may be slightly inferior to that of EAs regarding the evaluation results in Table 5, the table of solving time reported by Table 6 illustrates the vast differences between EAs and learning-based methods. On average, EAs consume 1639 $\sim$

Table 4
Training results of learning-based methods.

Algorithms	32-shelves		48-shelves		64-shelves	
	Mean	Std.	Mean	Std.	Mean	Std.
PPO	**917.6**	**110.8**	**1306.2**	**129.8**	**1585.4**	**130.3**
SAC	971.8	125.7	1336.9	137.1	1709.5	159.2
DDQN	1148.9	125.0	1663.4	159.8	2421.3	176.5
DQN	1132.9	125.1	1695.7	146.1	2265.7	152.8
FDQN	1143.5	136.7	1719.9	149.3	2268.5	153.5
D3QN	1150.9	135.0	1627.1	198.3	2193.0	625.0
A2C	1079.6	159.1	1496.5	206.1	2159.4	328.8
A3C	1104.3	140.3	1493.4	189.1	1924.0	268.6
AM1	1160.6	129.3	1676.3	217.4	2124.3	289.9
AM2	1058.5	146.0	1438.6	170.8	1971.1	295.1
AM3	970.4	171.1	1537.6	307.8	**1787.9**	435.5
SYMNCO	1146.7	109.6	1620.6	183.7	2163.6	345.0
ADRL	**879.7**	**91.6**	**1329.4**	**137.2**	2106.3	**217.6**

Table 5
Evaluation results of EAs, DRLs, and attention-based methods.

Algorithms	32-shelves		48-shelves		64-shelves	
	Mean	Std.	Mean	Std.	Mean	Std.
GA	829	92.7	1133.3	105.7	1451.7	**99.9**
PSO	831.6	**92.5**	1131.8	**105.5**	**1441.5**	101.2
ACO	830.1	94.1	1133.8	107.1	1448.5	102.3
ABCA	834.2	93.1	1141.8	106.2	1463.6	102.2
BA	838.2	97.1	1151.5	113.9	1485.9	112.8
DE	831.4	94.1	1143.8	108.5	1471.4	104.0
RNDE	**826.3**	93.3	**1131.2**	106.7	1451.5	102.7
PPO	943.7	**110.4**	1444.3	138.2	1713.4	**112.6**
SAC	**903.4**	113.0	**1323.6**	**121.2**	**1625.3**	134.0
DDQN	1145.5	125.9	1668.3	144.7	2462.6	150.1
DQN	1130.0	121.3	1694.0	153.6	2289.9	144.2
FDQN	1155.9	125.8	1726.3	148.0	2296.8	152.9
D3QN	1173.1	128.6	1738.8	150.2	2273.6	149.7
A2C	1167.0	123.2	1747.5	150.8	2478.7	155.5
A3C	1163.2	125.0	1711.5	144.3	2422.6	158.0
AM1	964.6	88.7	1366.6	108.5	1769.5	133.4
AM2	904.1	89.6	1326.0	108.0	1607.8	107.1
AM3	**845.2**	**81.8**	1192.3	97.3	1598.3	105.6
SYMNCO	1138.6	108.2	1617.2	182.9	2172.3	329.4
ADRL	851.0	83.0	**1157.6**	**95.8**	**1473.8**	**97.9**

Table 6
Evaluation results on solving time (seconds).

Algorithms	32-shelves		48-shelves		64-shelves	
	Mean	Std.	Mean	Std.	Mean	Std.
GA	**128.9932**	**11.3605**	**169.8652**	11.2124	**218.3882**	10.2742
PSO	134.5859	11.5286	175.4519	**11.1533**	223.4324	10.4549
ACO	135.4753	11.8080	178.1445	11.5354	226.9609	10.1755
ABCA	365.3785	33.1730	483.2409	31.9308	619.6440	28.4736
BA	185.6733	16.4050	278.4046	18.3906	426.8118	18.8418
DE	136.2036	11.7926	181.0270	11.9173	230.7188	10.4805
RNDE	132.5961	11.6864	176.9356	11.5353	226.2204	**9.9510**
PPO	0.0919	**0.0101**	0.1376	0.0121	0.1488	**0.0098**
SAC	**0.0837**	0.0118	**0.1224**	**0.0098**	**0.1289**	0.0099
DDQN	0.1657	0.0136	0.2295	0.0149	0.3223	0.0332
DQN	0.1655	0.0137	0.2273	0.0169	0.3181	0.0299
FDQN	0.169	0.0177	0.2317	0.0152	0.3259	0.0337
D3QN	0.127	0.0129	0.1684	0.0127	0.2178	0.0204
A2C	0.1337	0.0127	0.1793	0.0129	0.2163	0.0115
A3C	0.1293	0.0128	0.1758	0.0136	0.2152	0.0128
AM1	0.0757	0.0001	0.1033	0.0001	0.1434	0.0001
AM2	0.0589	0.0001	0.0995	0.0001	0.0997	0.0001
AM3	**0.0546**	0.0001	0.0785	0.0001	**0.0912**	0.0001
SYMNCO	0.0565	0.0001	0.0747	0.0001	0.0936	0.0001
ADRL	0.0577	0.0001	**0.0746**	0.0001	0.0943	0.0001

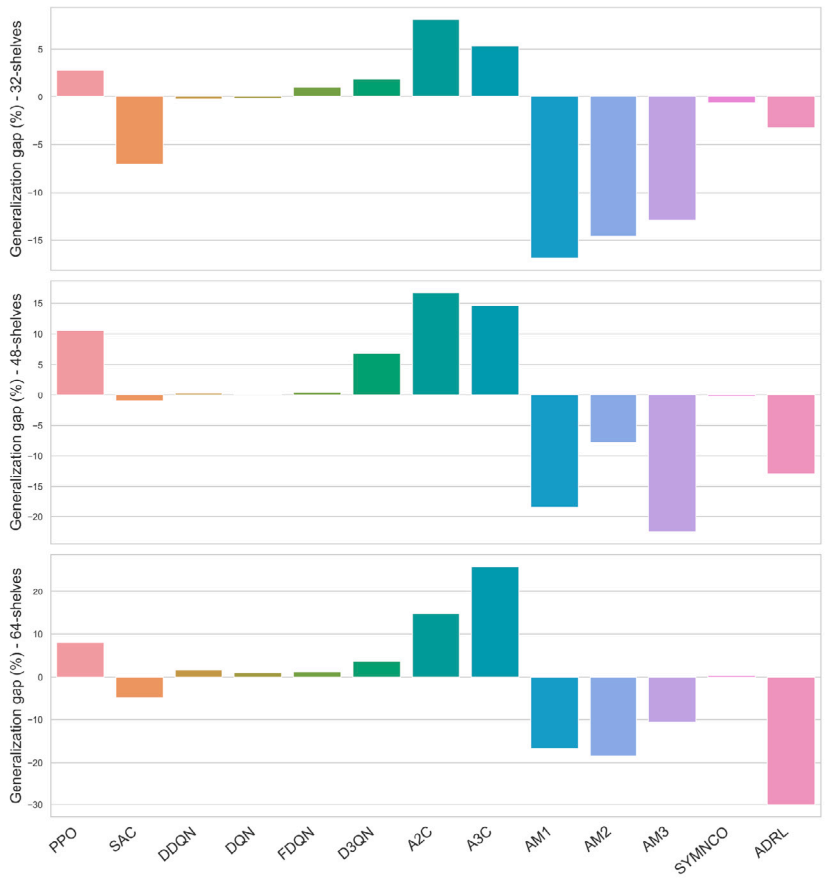

Fig. 6. The generalization gap of learning-based methods on the training and evaluation datasets.

1705 times the solving time than these learning-based methods. The fastest GA still requires at least 128 s to obtain a satisfying result. This high time consumption can hardly meet the requirements of dynamic changing environments. The performance gap between ADRL and the best-performed EAs in datasets of 32-shelves, 48-shelves, and 64-shelves are only 3.0%, 2.4%, and 2.2%, respectively. In contrast, EAs usually consume thousands of times the solving time of ADRL.

Fig. 6 depicts the generalization gaps for learning-based methods across three datasets. These gaps are quantified as the percentage difference between the objectives obtained in the training and evaluation datasets. With the exception of SAC, most DRLs tend to overfit the training data distribution. We hypothesize that this is because SAC's entropy training may confer some benefits in the studied problem. Furthermore, the results also showed that attention-based methods consistently demonstrate positive generalization capabilities, highlighting their superior generalization performance. SYMNCO performs nearly the same in the training and evaluation datasets, and we infer that this is because the symmetry in the studied problem is not apparent. Therefore, the entropy training and the attention mechanism benefit from avoiding overfitting issues in the studied problem. Fig. 7 shows the searching curves of EAs with the ADRL horizons in three datasets, where the X-axis denotes the searching iterations, and the Y-axis represents the objectives. The shadow areas represent the fluctuation range of an algorithm, and the solid line represents the average objective of 200 evaluation instances. These curves yield two main phenomenons: (1) RNDE and PSO indicate obvious superiority over other EAs, validating their effectiveness in solving the studied problem. (2) The ADRL horizons become more competitive when the dataset becomes more complicated. For example, most EAs find better solutions than ADRL only after dozens of iterations in 64-shelves. This result reveals that the searching performance of EAs may decrease when facing complicated order-picking datasets, and the gap between ADRL and EAs will decrease.

5.3. Comparison of solution quality with other learning-based methods

In contrast to searching-based methods, the essence of learning-based methods is reasoning out the next action based on the current state. This subsection explores and analyzes the learned policy of each learning-based method by comparing several problem variables. The lengths of the visited tours of these learning-based methods in one picking task are compared first, as the visiting distance can largely influence the final objective. Fig. 8 indicates the cumulative tour lengths in one order-picking task, where each violin plot represents the distribution of an algorithm. The width of a violinplot at any given point denotes the density of the data, with wider portions indicating higher density. PPO, SAC, AM1, AM2, AM3, and ADRL usually obtain lower values than others in the three datasets, and ADRL achieves nearly the lowest values. These results illustrate that one of the learned strategies is shortening the visited distance as much as possible. Additionally, the widths of ADRL's plots are wider than others, representing that ADRL's trained policies may be more robust. Subsequently, the average tour lengths are compared to analyze the learned policies further. Fig. 9 indicates average tour lengths in one picking task, where the Y-axis represents the quotient of the cumulative tour length and the number of tours. Although lower average lengths may represent lower cumulative lengths, excessive repetitive round tours may cause a waste of visiting time. For example, SYMNCO usually indicates the lowest average lengths, but it still performs worse than ADRL regarding the obtained objective. Therefore, ADRL learns to control the average length within reasonable ranges.

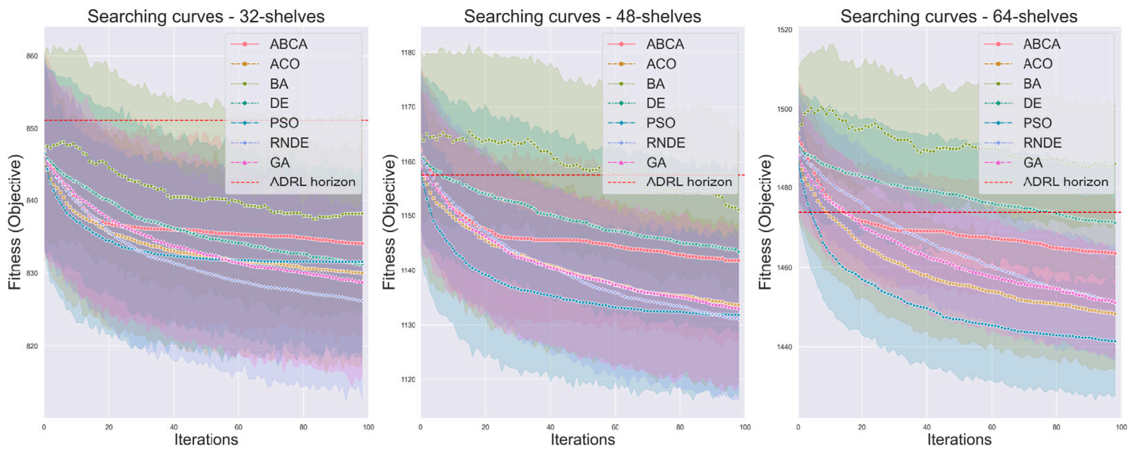

Fig. 7. Searching curves of EAs with ADRL horizon lines. The ADRL horizon lines represent the objective achieved by ADRL in each scenario.

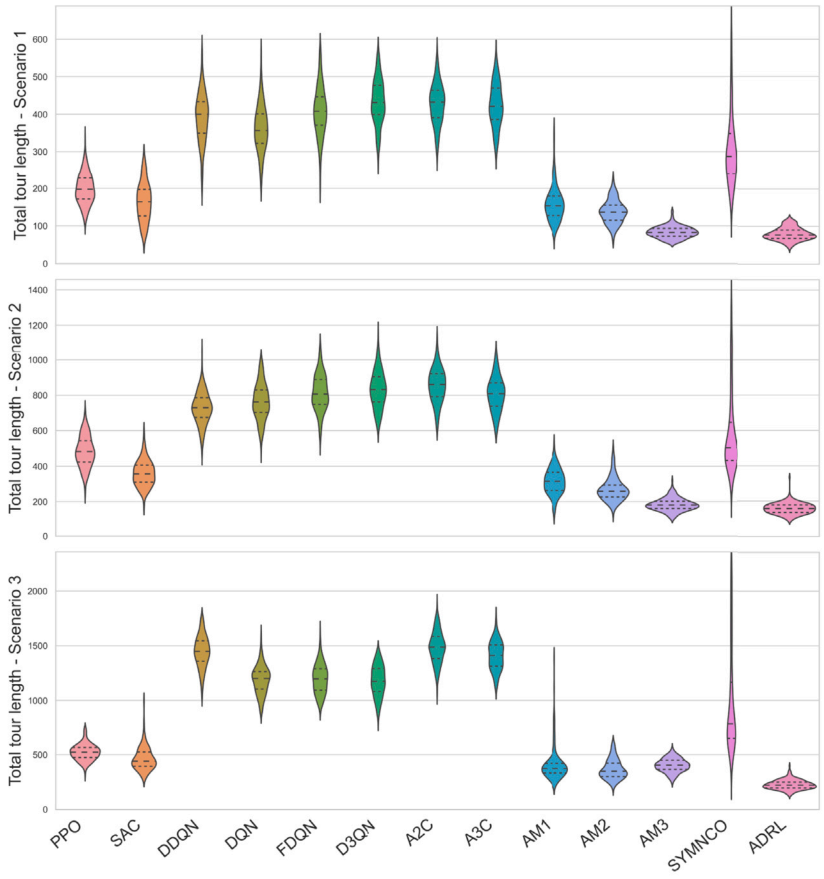

Fig. 8. Cumulative tour lengths of learning-based methods in three scenarios.

Subsequently, we compare the percentages of the cart weight and the workload in each tour. The quotient of maximum values and constraint thresholds calculates the percentages. For example, the cart weight percentage equals the quotient of the maximum weight during one tour and the weight constraint threshold. Fig. 10 indicates the cart weight percentages of these learning-based methods in three datasets, where each box line represents the distribution of percentages of a learning-based method. ADRL usually has higher cart weight percentages than others, especially in the 64-shelves dataset. Higher percentages usually represent a more rational tour division of the order, which can effectively reduce the travel time between shelves and the transfer point. Fig. 11 indicates the workload percentages of these learning-based methods in three datasets. Like the cart weight percentages analysis, higher workload percentages usually mean more sufficient utilization of the workload constraint. For example, the percentages of ADRL can reach near 100% in Fig. 11, illustrating the effectiveness of the learned policies. Furthermore, we find that AM3 indicates higher percentages than ADRL in the dataset of 64-shelves, but the average objective obtained by AM3 is lower according to the results in Table 5. Therefore, simply increasing the workload percentages does not necessarily lead to optimizing objectives.

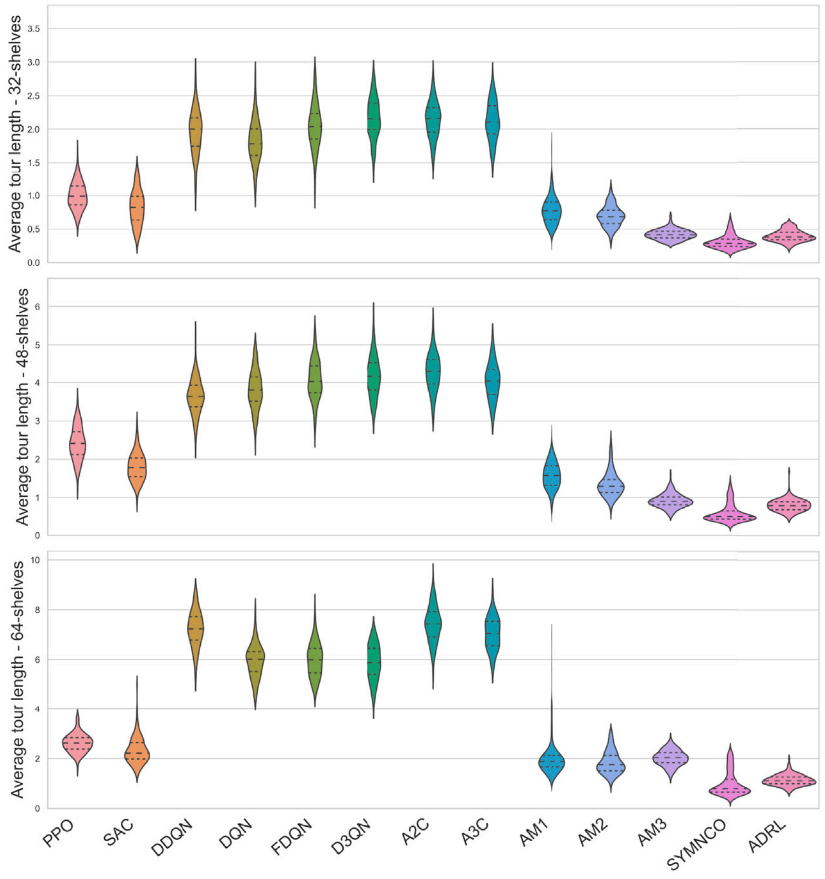

Fig. 9. Average tour lengths of learning-based methods in three scenarios.

5.4. Result discussion

The comparison results with EAs, DRLs, and other attention-based methods proved the solving efficiency of ADRL. The variance of performance across scenarios mainly originates from the inconsistency of data distributions in different order picking configurations. Most neural network-based approaches generate fluctuant solutions facing diverse data distributions, which can be regarded as random errors. Nonetheless, numerous experimental results validated that ADRL could outperform most learning-based methods across diverse test scenarios of the three datasets. Furthermore, the objective gaps between ADRL and EAs are no more than 3%. The solving time of ADRL tends to be around 0.1 s, while EAs may consume thousands of times solving time than ADRL. In summary, the strong generalization enables ADRL to solve dynamic problems efficiently.

The comparison of solution quality with other learning-based methods reveals some potential order-picking strategies. Combining the figures shown by Figs. 7 ∼ 11, we find that a satisfying picking policy needs to trade off the different metrics according to environmental diversity. Simply optimizing one certain metric may be harmful to policy generalization. Therefore, we can infer that enabling the strategy to learn to adjust the importance of each metric in different environments qualitatively is the key to effectively solving order-picking problems using learning-based methods. Effective learning requires both an efficient state feature extraction and stable training algorithms. The comparison results illustrate that the MHA-based encoder–decoder networks and the entropy-PPO training algorithm indicate some merits for the studied problem.

6. Conclusion

This paper solved the dynamic multi-tour order-picking problem in Scania's warehouse with ADRL. In contrast to existing research that usually models the order-picking problem as a variant of the STSP, the studied problem describes a practical production process that involves both the order split and the order picking. We formulate this dynamic multi-tour order-picking problem as a mathematical model to minimize the OPT with the considerations of constraints of the cart weight and the operator's workload. With the motivation to address this problem efficiently, we construct an encoder–decoder-structured agent policy and train it with Entropy-PPO under the actor-critic framework. Three improvements to the extra location, dynamic decoding, and entropy-PPO are proposed to further enhance the performance of attention-based methods. Experimental results on 6600 instances from Scania's warehouses validate the effectiveness and superiority of ADRL compared to the other nineteen approaches. The comparison results between ADRL and DRLs validate the effectiveness of MHA in capturing the intercorrelations of order-picking problems. Furthermore, the comparisons between ADRL and AM1 ∼ AM3 serve as ablation experiments, proving the indispensability of the proposed improvements. Besides, experimental results indicate that the average solving time of ADRL tends to be less than 0.1 s. The fast solving speed and strong generalization of ADRL make it suitable for dynamic scenarios because a rescheduling solution can be rapidly generated when some scenario parameters change. The formulated mathematical model may also fit order-picking problems in other warehouse management systems, and the proposed ADRL can also be expanded to other similar optimization

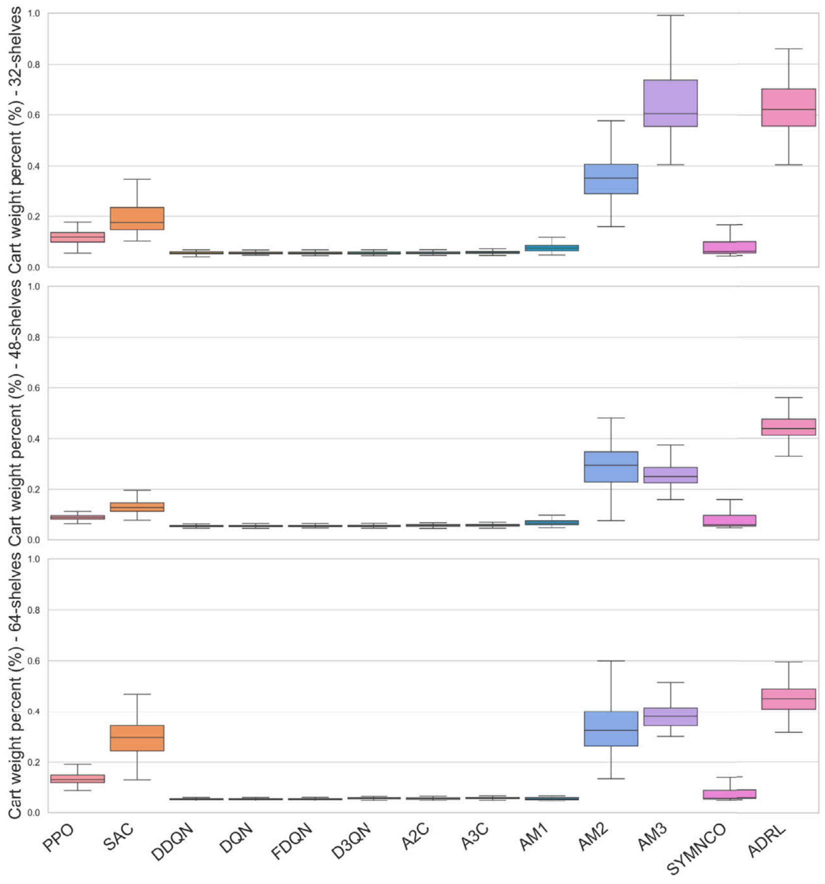

Fig. 10. Percentages of cart weights of learning-based methods in three scenarios.

problems. Furthermore, this study considers both the human-centric problem and the AI-based solving methods, promoting the further development of Industry 5.0 in European manufacturers.

Future work will concentrate on two aspects: First, the number of model parameters and the amount of training data for ADRL will be further expanded. In recent years, the rapid development of LLM has proven the effectiveness of large-parameter models in solving practical problems. Therefore, future work will focus on effectively training a large-parameter policy model for dynamic order-picking problems. Related technologies such as offline pre-training and online finetuning will be considered. Second, we only consider the end-to-end approaches that generate the solution directly based on the input of order-picking instances, overlooking the consideration of the hierarchical model that can split the multi-tour order picking into one splitting problem and one picking problem. Some recent literature reveals that a hierarchical model may be more effective in solving complex combinatorial optimization problems [45]. Therefore, a hierarchical model will be developed to divide the multi-tour order-picking problem into two subproblems: order-splitting and order-picking. Then, each subproblem will be solved with an independent neural policy.

CRediT authorship contribution statement

Xiaohan Wang: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Lin Zhang:** Supervision, Funding acquisition, Formal analysis, Conceptualization. **Lihui Wang:** Supervision, Resources, Project administration, Funding acquisition, Data curation, Conceptualization. **Enrique Ruiz Zuñiga:** Writing – review & editing, Resources, Data curation. **Xi Vincent Wang:** Writing – review & editing, Supervision, Resources, Project administration, Investigation, Funding acquisition, Formal analysis. **Erik Flores-García:** Writing – review & editing, Project administration, Funding acquisition, Data curation.

Declaration of competing interest

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

Acknowledgments

The authors would like to acknowledge the support of Swedish Innovation Agency (VINNOVA). This study is part of the Dynamic Scheduling of Assembly and Logistics Systems using AI (Dynamic SALSA) project. This research is also supported by the National Key R&D Program of China (No. 2023YFB3308201).

Appendix A. Brief descriptions of compared approaches

The formulated MDP for the dynamic multi-tour order-picking problem requires discrete action-based DRL algorithms. These algorithms

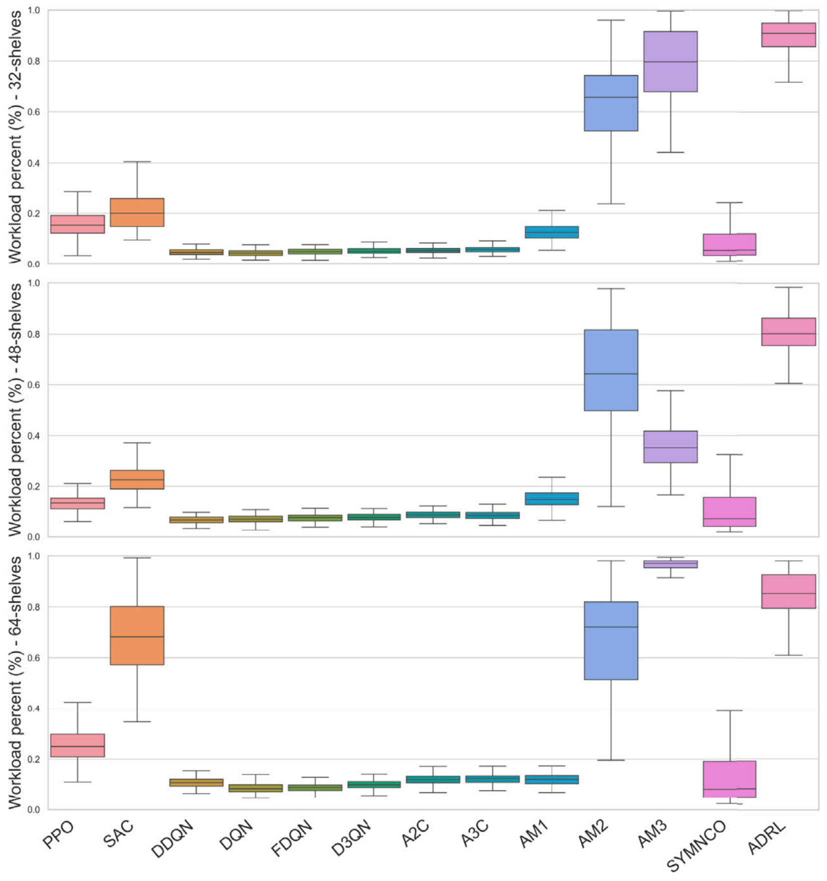

Fig. 11. Workload percentages of learning-based methods in three scenarios.

can be divided into two categories: value-based and policy-based algorithms. Value-based algorithms tend to learn a value function represented by a neural network and then select the action based on the prediction results of estimated values. Policy-based algorithms usually directly learn the policy function represented by a neural network and then select the action by sampling. The short descriptions of DRLs used as compared approaches are briefly introduced as follows:

PPO (Proximal Policy Optimization) is a policy-based method simplifying the trust region optimization process. PPO introduces a clipped objective function to avoid large policy updates that could lead to poor performance.

SAC (Soft Actor-Critic) is a policy-based method that combines the benefits of maximum entropy with the stability of actor-critic methods. It learns a policy that maximizes both the expected return and the entropy of the policy, which encourages exploration and can lead to more robust solutions.

A3C (Asynchronous Advantage Actor-Critic) is a distributed policy-based method that allows multiple agents to learn simultaneously in parallel environments. The key innovation in A3C is its asynchronous training, where multiple agents can interact with the environment and update the global model without waiting for each other.

A2C (Advantage Actor-Critic) is a distributed policy-based method operating synchronously, meaning that it updates the model using experiences from a single thread or agent at a time. A2C tends to have more consistent exploration patterns because all updates come from a single policy. However, it may be slower during training, especially in environments with a large state or action space.

DQN (Deep Q-Network) is a value-based method that combines Q-learning with deep neural networks to learn the value function directly from high-dimensional sensory input. DQN was one of the first deep-learning models to achieve human-level performance on Atari games.

DDQN (Double Deep Q-Network) is an improvement over DQN. DDQN uses two separate Q-networks: One for selecting actions (the online network) and the other for evaluating actions (the target network). The double-network architecture helps to reduce overestimation bias in DQN.

FDQN (DQN with Fixed Q-Targets) is a variant of DQN where the target-Q network is fixed and will not be updated during training. This may help the training become more stable in some situations.

Dueling Double Deep Q-Network (D3QN) is an extension of the DDQN that incorporates the dueling architecture. It has two separate estimators for the state value function and the advantage function. The state value function estimates the expected return of being in a particular state, while the advantage function estimates the relative advantage of taking different actions in that state.

Appendix B. Supplementary experimental results

This section presents the supplementary results for Tables 4 and 5. Table B.7 reports the best, worst, and median objectives obtained during training for these learning-based methods. Table B.8 reports the best, worst, and median objectives in the evaluation datasets for EAs, DRLs, and attention-based methods.

Table B.7
Best, worst, and median values of training results of learning-based methods.

Algorithms	32-shelves			48-shelves			64-shelves		
	Best	Worst	Median	Best	Worst	Median	Best	Worst	Median
PPO	630.6	1256.6	915.4	990.9	1737.3	1304.2	1236.6	2078.0	1582.2
SAC	629.6	1472.5	969.2	939.7	1977.9	1333.7	1308.4	2434.2	1701.6
DDQN	766.2	1471.7	1145.8	979.6	2093.8	1673.9	1480.2	2902.8	2430.4
DQN	767.6	1462.5	1128.6	1056.7	2155.9	1701.2	1503.8	2715.4	2264.8
FDQN	671.7	1503.4	1144.2	1025.7	2141.7	1725.6	1551.1	2706.3	2267.0
D3QN	650.3	1505.6	1152.8	958.8	2211.4	1628.3	1394.9	22 143.3	2176.0
A2C	674.8	1506.6	1077.5	972.4	2153.3	1493.6	1396.7	2859.7	2240.5
A3C	715.0	1496.0	1099.3	1009.5	2132.5	1486.0	1348.9	2775.1	1874.4
AM1	743.4	1558.8	1161.8	1053.9	2370.6	1691.3	1364.5	3284.3	2113.7
AM2	663.7	1580.2	1045.8	1030.6	2370.6	1414.2	1332.2	3284.3	1908.0
AM3	643.6	1553.6	936.4	977.2	2422.6	1484.4	1339.8	3307.6	1647.3
SYMNCO	854.8	1556.3	1143.9	1204.4	2370.6	1586.8	1568.6	3294.1	2052.0
ADRL	645.9	1315.6	873.6	936.8	2045.6	1313.9	1247.8	2655.9	2137.6

Table B.8
Best, worst, and median values of evaluation results of EAs, DRLs, and attention-based methods.

Algorithms	32-shelves			48-shelves			64-shelves		
	Best	Worst	Median	Best	Worst	Median	Best	Worst	Median
GA	633.8	1039.3	829.6	909.6	1369.6	1144.3	1225.5	1665.1	1482.9
PSO	637.9	1040.5	831.7	912	1362	1140.6	1211.1	1669.4	1474.6
ACO	633	1041.1	830.4	920.6	1368.6	1139.1	1205.7	1655	1480.7
ABCA	639.3	1047.7	834.5	909.4	1377.9	1150.1	1225.5	1667.4	1495.2
BA	633.9	1065.9	833	924.8	1420	1154.3	1232.7	1776.4	1512.3
DE	624.6	1041.8	830.6	910.4	1380	1153.4	1215.6	1686.7	1503.2
RNDE	620.1	1031.8	827.5	915.5	1369.2	1143.2	1224.9	1649.2	1481.1
PPO	710.7	1216.3	941.4	1147.3	1823.1	1456.9	1392.5	2004.3	1716.4
SAC	661.6	1169.9	896.9	1032.2	1632.4	1331.7	1295.9	2081.4	1618.7
DDQN	853.0	1451.5	1129.7	1328.0	2116.5	1681.6	2025.2	2794.4	2464.4
DQN	880.1	1446.6	1120.9	1332.9	2069.2	1706.9	1884.9	2631.5	2280.6
FDQN	914.2	1479.3	1139.4	1379.6	2106.8	1733.6	1897.1	2723.0	2305.7
D3QN	886.8	1471.1	1186.6	1389.2	2120.6	1742.4	1874.3	2586.3	2278.9
A2C	918.3	1453.8	1171.7	1402.3	2143.9	1755.4	1975.2	2882.3	2499.0
A3C	897.7	1453.1	1155.5	1409.4	2091.6	1726.1	2020.8	2804.5	2427.2
AM1	746.7	1237.0	958.2	1099.6	1631.5	1367.1	1479.4	2297.4	1760.6
AM2	691.7	1125.4	900.2	1083.7	1634.3	1326.5	1361.0	1883.5	1603.4
AM3	646.7	1039.3	845.4	981.7	1429.5	1189.6	1329.8	1811.1	1608.7
SYMNCO	886.2	1542.0	1129.5	1249.3	2285.4	1591.9	1659.6	3215.8	2073.9
ADRL	650.7	1047.0	850.9	940.6	1369.1	1157.9	1236.9	1672.9	1485.8

Data availability

Data will be made available on request.

References

[1] B. Wang, H. Zhou, X. Li, G. Yang, P. Zheng, C. Song, Y. Yuan, T. Wuest, H. Yang, L. Wang, Human digital twin in the context of industry 5.0, Robot. Comput.-Integr. Manuf. 85 (2024) 102626.

[2] A. Martínez-Gutiérrez, J. Díez-González, H. Perez, M. Araújo, Towards industry 5.0 through metaverse, Robot. Comput.-Integr. Manuf. 89 (2024) 102764.

[3] R. De Koster, T. Le-Duc, K.J. Roodbergen, Design and control of warehouse order picking: A literature review, European J. Oper. Res. 182 (2) (2007) 481–501.

[4] Y. Bukchin, E. Khmelnitsky, P. Yakuel, Optimizing a dynamic order-picking process, European J. Oper. Res. 219 (2) (2012) 335–346.

[5] L. Pansart, N. Catusse, H. Cambazard, Exact algorithms for the order picking problem, Comput. Oper. Res. 100 (2018) 117–127.

[6] S. Molder, Improving the Order Picking Performance at Scania Production Zwolle (Master's thesis), University of Twente, 2022.

[7] Y. Su, M. Li, X. Zhu, C. Li, Steiner TSP based on aisle as a unit for order picking, Comput. Ind. Eng. 168 (2022) 108026.

[8] R.L. Daniels, J.L. Rummel, R. Schantz, A model for warehouse order picking, European J. Oper. Res. 105 (1) (1998) 1–17.

[9] B. Cals, Y. Zhang, R. Dijkman, C. van Dorst, Solving the online batching problem using deep reinforcement learning, Comput. Ind. Eng. 156 (2021) 107221.

[10] M. Masae, C.H. Glock, E.H. Grosse, Order picker routing in warehouses: A systematic literature review, Int. J. Prod. Econ. 224 (2020) 107564.

[11] İ. Muter, T. Öncan, Order batching and picker scheduling in warehouse order picking, IISE Trans. 54 (5) (2022) 435–447.

[12] L. Pansart, N. Catusse, H. Cambazard, Exact algorithms for the order picking problem, Comput. Oper. Res. 100 (2018) 117–127.

[13] Y. Bukchin, E. Khmelnitsky, P. Yakuel, Optimizing a dynamic order-picking process, European J. Oper. Res. 219 (2) (2012) 335–346.

[14] B. Gajsek, G. Dukic, M. Kovacic, M. Brezocnik, A multi-objective genetic algorithms approach for modelling of order picking, Int. J. Simul. Model. 20 (4) (2021) 719–729.

[15] J. Park, J. Chun, S.H. Kim, Y. Kim, J. Park, Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning, Int. J. Prod. Res. 59 (11) (2021) 3360–3377.

[16] X. Wang, L. Zhang, Y. Liu, C. Zhao, Logistics-involved task scheduling in cloud manufacturing with offline deep reinforcement learning, J. Ind. Inf. Integr. 34 (2023) 100471.

[17] W. Kool, H. Van Hoof, M. Welling, Attention, learn to solve routing problems!, 2018, arXiv preprint arXiv:1803.08475.

[18] M. Kim, J. Park, J. Park, Sym-nco: Leveraging symmetricity for neural combinatorial optimization, Adv. Neural Inf. Process. Syst. 35 (2022) 1936–1949.

[19] T. De Lombaert, K. Braekers, R. De Koster, K. Ramaekers, In pursuit of humanised order picking planning: methodological review, literature classification and input from practice, Int. J. Prod. Res. 61 (10) (2023) 3300–3330.

[20] H. Dauod, D. Won, Real-time order picking planning framework for warehouses and distribution centres, Int. J. Prod. Res. 60 (18) (2022) 5468–5487.

[21] O. Briant, H. Cambazard, D. Cattaruzza, N. Catusse, A.-L. Ladier, M. Ogier, An efficient and general approach for the joint order batching and picker routing problem, European J. Oper. Res. 285 (2) (2020) 497–512.

[22] G. Casella, A. Volpi, R. Montanari, L. Tebaldi, E. Bottani, Trends in order picking: a 2007–2022 review of the literature, Prod. Manuf. Res. 11 (1) (2023) 2191115.

[23] I.G. Lee, S.H. Chung, S.W. Yoon, Two-stage storage assignment to minimize travel time and congestion for warehouse order picking operations, Comput. Ind. Eng. 139 (2020) 106129.

[24] S. Vanheusden, T. Van Gils, A. Caris, K. Ramaekers, K. Braekers, Operational workload balancing in manual order picking, Comput. Ind. Eng. 141 (2020) 106269.

[25] M. Vazquez-Noguerol, J. Comesaña-Benavides, R. Poler, J.C. Prado-Prado, An optimisation approach for the e-grocery order picking and delivery problem, CEJOR Cent. Eur. J. Oper. Res. 30 (3) (2022) 961–990.
[26] E. Ardjmand, I. Ghalehkhondabi, W.A. Young, II, A. Sadeghi, G.R. Weckman, H. Shakeri, A hybrid artificial neural network, genetic algorithm and column generation heuristic for minimizing makespan in manual order picking operations, Expert Syst. Appl. 159 (2020) 113566.
[27] J.P. van der Gaast, F. Weidinger, A deep learning approach for the selection of an order picking system, European J. Oper. Res. 302 (2) (2022) 530–543.
[28] R. D'Haen, K. Braekers, K. Ramaekers, Integrated scheduling of order picking operations under dynamic order arrivals, Int. J. Prod. Res. 61 (10) (2023) 3205–3226.
[29] P. Yang, Z. Zhao, Z.-J.M. Shen, A flow picking system for order fulfillment in e-commerce warehouses, IISE Trans. 53 (5) (2021) 541–551.
[30] S. Mahmoudinazlou, A. Sobhanan, H. Charkhgard, A. Eshragh, G. Dunn, Deep reinforcement learning for dynamic order picking in warehouse operations, 2024, arXiv preprint arXiv:2408.01656.
[31] C. Li, P. Zheng, Y. Yin, B. Wang, L. Wang, Deep reinforcement learning in smart manufacturing: A review and prospects, CIRP J. Manuf. Sci. Technol. 40 (2023) 75–101.
[32] X. Liu, H. Xu, W. Liao, W. Yu, Reinforcement learning for cyber-physical systems, in: 2019 IEEE International Conference on Industrial Internet, ICII, IEEE, 2019, pp. 318–327.
[33] V. Samsonov, K.B. Hicham, T. Meisen, Reinforcement learning in manufacturing control: Baselines, challenges and ways forward, Eng. Appl. Artif. Intell. 112 (2022) 104868.
[34] X. Wang, L. Zhang, Y. Liu, F. Li, Z. Chen, C. Zhao, T. Bai, Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning, J. Manuf. Syst. 65 (2022) 130–145.
[35] X. Wang, L. Zhang, T. Lin, C. Zhao, K. Wang, Z. Chen, Solving job scheduling problems in a resource preemption environment with multi-agent reinforcement learning, Robot. Comput.-Integr. Manuf. 77 (2022) 102324.
[36] S. Luo, Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning, Appl. Soft Comput. 91 (2020) 106208.
[37] R. Liu, R. Piplani, C. Toro, Deep reinforcement learning for dynamic scheduling of a flexible job shop, Int. J. Prod. Res. 60 (13) (2022) 4049–4069.
[38] Y. Li, W. Gu, M. Yuan, Y. Tang, Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network, Robot. Comput.-Integr. Manuf. 74 (2022) 102283.
[39] J. Huang, J. Wan, B. Lv, Q. Ye, Y. Chen, Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning, IEEE Syst. J. (2023).
[40] Y. Liu, H. Liang, Y. Xiao, H. Zhang, J. Zhang, L. Zhang, L. Wang, Logistics-involved service composition in a dynamic cloud manufacturing environment: A DDPG-based approach, Robot. Comput.-Integr. Manuf. 76 (2022) 102323.
[41] T. Dong, F. Xue, C. Xiao, J. Zhang, Workflow scheduling based on deep reinforcement learning in the cloud environment, J. Ambient Intell. Humaniz. Comput. (2021) 1–13.
[42] Y.-D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, S. Min, Pomo: Policy optimization with multiple optima for reinforcement learning, Adv. Neural Inf. Process. Syst. 33 (2020) 21188–21198.
[43] G. Dunn, H. Charkhgard, A. Eshragh, S. Mahmoudinazlou, E. Stojanovski, Deep reinforcement learning for picker routing problem in warehousing, 2024, arXiv preprint arXiv:2402.03525.
[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[45] Z. Yang, Z. Pan, M. Li, K. Wu, X. Gao, Learning based 2D irregular shape packing, ACM Trans. Graph. 42 (6) (2023) 1–16.