Optimizing Multi-Factory Remanufacturing Scheduling Under Order Tardiness Constraints Using Reinforcement Learning

Wenjing Zeng, Nan Wang, and Moitrayee Chatterjee

***Abstract*—With the rapid development of manufacturing industries, particularly in remanufacturing, how to efficiently manage production scheduling across multiple factories has become an important research topic. The frequent occurrence of order tardiness, which refers to situations where orders are not completed within the required delivery time, poses a significant challenge, negatively impacting production efficiency and customer satisfaction. Therefore, optimizing multi-factory remanufacturing systems under order tardiness constraints has become a critical issue in both theory and practice. To address this problem, this study develops a mathematical model with the objective of minimizing total costs. By optimizing production scheduling through this model, the losses caused by order timeouts in disassembly operations can be effectively reduced. The model is first validated using the CPLEX optimization solver to ensure its feasibility and accuracy. Subsequently, the Deep Double Q-Network (DDQN) algorithm is applied to solve the problem, as it is well-suited for complex decision-making under dynamic conditions. The performance of DDQN is further compared with two widely used reinforcement learning approaches: Advantage Actor-Critic (A2C) and Proximal Policy Optimization (PPO). A simulation environment is constructed to align with the characteristics of the multi-factory remanufacturing system. Experimental results show that DDQN achieved the lowest total cost in all six benchmark cases, reducing the cumulative cost compared with PPO and A2C, respectively, while maintaining good scalability as the order quantity increased. The results demonstrate that the DDQN algorithm significantly outperforms A2C and PPO, achieving higher system efficiency and better stability, particularly in managing order timeout constraints. This research offers a novel approach to the optimization of multi-factory remanufacturing systems and provides valuable insights for industrial applications and future studies.**

***Index Terms*—Multi-factory remanufacturing, Order tardiness, Production scheduling optimization, Deep Double Q-Network, Reinforcement learning**

I. Introduction

WITH the rapid development of technology and the widespread use of electronic products [1], the amount of electronic waste has been continuously increasing, posing serious challenges to environmental protection and resource management. Governments and international organizations strengthen regulations on the recycling and treatment of electronic waste and promote the development of the circular economy by enhancing resource recovery, precious metal extraction, and the reuse of electronic components [2]. Meanwhile, the application of intelligent recycling systems and the Internet of Things improves recycling efficiency [3, 4]. In the product design stage, adopting easy-to-disassemble and recyclable design concepts helps achieve source reduction and lifecycle management. In China, electronic waste management policies continue to improve, and related technologies are being increasingly applied [5]. How to further enhance resource recovery efficiency and management performance remains an important issue that requires urgent attention.

Remanufacturing, as an effective approach to resource circulation, has attracted increasing attention from both academia and industry. By collecting, disassembling, repairing, and reusing end-of-life products, remanufacturing significantly reduces resource waste, improves material utilization, and lowers the emission of hazardous waste [6, 7, 8]. Compared with traditional manufacturing, remanufacturing significantly reduces energy consumption, raw material usage, and environmental impacts, making it a key strategy for achieving green manufacturing and sustainable development [9, 10]. However, as product life cycles shorten and market demands diversify, traditional single-factory remanufacturing models face increasing challenges in flexibility, capacity utilization, and operational efficiency. In response to these challenges, multi-factory collaborative remanufacturing has emerged as a promising approach to enhance production flexibility, improve resource utilization, and support sustainable operations.

Several studies have investigated various aspects of remanufacturing systems to enhance efficiency and sustainability. For example, Guo *et al*. [11] developed a disassembly process optimization model for multi-factory systems that considers worker posture to improve operational efficiency. Similarly, Wang *et al*. [12] proposed a data-driven collaborative production planning model for multi-factory systems within multi-channel supply chains. While these studies have advanced the understanding of multi-factory remanufacturing, they often fail to consider the impact of order fulfillment timeliness, particularly order tardiness, on overall system performance.

Manuscript received February 22, 2026; revised March 22 and March 29, 2026; accepted June 23, 2026. This article was recommended for publication by Associate Editor Shujin Qin upon evaluation of the reviewers' comments.

This work is supported in part by NSFC under Grants 61573089, 62073069, and 51405075 and in part by Natural Science Foundation of Shandong Province under Grant ZR2019BF004.

W. Zeng is with the College of Computer Science and Technology, Donghua University, Shanghai 201620, China (e-mail: zengwenjing1013@163.com).

N. Wang is with the Department of Computer Science, William Paterson University, 773 Yorkshire Drive, Breinigsville, PA 18031, USA (e-mail:nan.wang.88888@gmail.com).

M. Chatterjee is with the Department of Computer Science, New Jersey City University, Jersey City NJ, USA (e-mail: mchatterjee@njcu.edu).

Corresponding Author: Wenjing Zeng.

Order tardiness, defined as the delay in completing orders beyond the promised delivery time, is a crucial factor affecting operational efficiency, market competitiveness, and resource utilization. Addressing this issue is essential for improving production scheduling decisions, enhancing service levels, and promoting the sustainable development of multi-factory remanufacturing systems.

In addition to addressing order tardiness, this study also incorporates the disassembly line balancing problem (DLBP), which is recognized as a fundamental challenge in remanufacturing systems [13]. DLBP focuses on the balanced allocation of disassembly tasks across multiple workstations while ensuring precedence constraints and operational feasibility are met [14, 15, 16]. Effective disassembly line balancing not only enhances production efficiency and reduces resource waste but also facilitates downstream processes such as component transportation and subsequent remanufacturing operations. In the context of multi-factory systems, an optimized disassembly line improves inter-factory coordination, streamlines material flows, and strengthens overall operational performance. By integrating the consideration of order tardiness and DLBP into the decision-making process, this study aims to enhance both delivery performance and resource utilization within multi-factory remanufacturing systems.

Recent studies on remanufacturing systems have made significant progress in integrated optimization and intelligent decision-making[14, 15, 16]. In addition to traditional scheduling objectives, increasing attention has been paid to disassembly line balancing, resource coordination, and delivery performance [11]. Meanwhile, reinforcement learning [17, 18] has emerged as a promising tool for solving dynamic manufacturing problems. However, most existing studies focus on either disassembly line balancing or scheduling optimization separately, and relatively few consider the joint optimization of order tardiness and disassembly line balancing in multi-factory remanufacturing systems.

However, most existing studies to address the challenges of insufficient delivery timeliness and inefficient disassembly resource allocation in multi-factory remanufacturing systems, this study proposes the Order-Tardiness-oriented Multi-factory Remanufacturing (OTMR) model. The model jointly considers order fulfillment timeliness and disassembly line balancing, coordinating production scheduling and disassembly task allocation across multiple factories [19]. By optimizing order execution and resource allocation strategies in a dynamic manner, OTMR enhances system responsiveness and resource utilization.

To address the proposed OTMR problem, this study introduces a reinforcement learning (RL) [17, 18, 20]-based approach. By interacting with the environment and continuously learning through reward feedback, RL can generate adaptive and near-optimal decision policies without the need for explicit system modeling. In recent years, RL has been widely applied in fields such as manufacturing systems, production scheduling, and resource optimization. For example, Wang *et al*. [21] proposed a deep reinforcement learning approach for dynamic balancing of U-shaped robotic disassembly lines, and Wang *et al*. [22] introduced a multi-objective advantage actor-critic algorithm for hybrid disassembly lines with multi-skilled workers. A recent survey by Guo *et al*. [23] provides a comprehensive overview of RL applications in disassembly system optimization. These studies demonstrate the broad applicability of RL in complex industrial optimization and provide theoretical support for the method proposed in this study.

This study proposes an approach based on Double Deep Q-Networks (DDQN) [24, 25], aiming to minimize order tardiness while optimizing production scheduling and disassembly task allocation. DDQN mitigates the overestimation of action values common in traditional Deep Q-Networks [26], thereby enhancing learning stability and decision-making accuracy in dynamic production environments [27]. To support the effective training of the learning agent, a simulation environment [28] is designed to replicate the operational dynamics of multi-factory remanufacturing systems. Additionally, to reduce action space complexity and simplify decision-making in the transportation phase, a greedy heuristic [29] is employed to allocate transportation tasks based on immediate cost efficiency.

The contributions of this study are as follows:

1) This study investigates a multi-factory remanufacturing problem with order tardiness considerations and disassembly line balancing. A mathematical optimization model is developed to jointly optimize production scheduling and disassembly task allocation while maintaining balanced workloads across disassembly lines.
2) To solve the proposed problem, this study develops a solution framework with DDQN and a dedicated simulation environment that models the dynamic operations of multi-factory remanufacturing systems.
3) The correctness of the mathematical model is validated using IBM CPLEX, and the proposed DDQN is compared with other RL methods. Experimental results demonstrate the superior performance of the proposed approach in solving the remanufacturing scheduling problem.

The structure of the work is organized as follows. Section II provides a detailed description of the OTMR problem and establishes the mathematical model. Section III introduces DDQN and its environment setup; Section IV discusses the experimental design and presents a comparative analysis of the results. Finally, Section V concludes the paper and outlines potential directions for future research.

II. Problem Description

A. Problem Statement

In multi-factory remanufacturing systems, customer orders are fulfilled through the coordinated operation of recycling centers, disassembly centers, and manufacturing centers. End-of-life (EOL)[30] products are collected and disassembled to recover valuable components, which are subsequently used to meet production requirements. Existing studies primarily focus on optimizing production scheduling and disassembly planning, typically assuming that orders can be completed as long as sufficient resources are available, without explicitly considering strict delivery deadlines. However, in practical

remanufacturing environments, timely order fulfillment is crucial for maintaining service quality and ensuring customer satisfaction.

This study investigates the Multi-factory Remanufacturing Order Problem (MROP), where manufacturing centers submit one or more orders to recycling centers. Recycling centers then deliver EOL products to disassembly centers, which process these products based on specific component requirements and transport the recovered components to manufacturing centers for order completion. The objective is to minimize both total operational costs and order tardiness penalties throughout the remanufacturing process.

In practice, each customer order is usually associated with a strict delivery deadline. Disassembly centers must complete the required tasks within the specified time to ensure on-time delivery; otherwise, late penalties may be incurred. To reduce the risk of order tardiness, disassembly task scheduling must be carefully optimized. Specifically, tasks at each workstation should be sequenced efficiently to improve production rhythm, reduce idle time, and ensure timely completion. Proper task sequencing not only improves the operational efficiency of disassembly centers but also reduces tardiness-related losses and enhances customer satisfaction.

Furthermore, this study designs the order configuration in a way that allows multiple orders to require the same or different components, with each order having distinct delivery deadlines. This setting reflects the complexity of real-world production scenarios, where component demands and timing constraints vary across orders. To address these challenges, a comprehensive mathematical model is developed to optimize disassembly scheduling, transportation planning, and order fulfillment under dynamic, multi-order conditions. Fig. 1 illustrates an example of the OTMR process.

Common modeling approaches include AND/OR graphs (AOG) [31, 32], precedence graphs [33, 34], and Petri nets [35]. Among these methods, the AND/OR graph is employed in this study to model the disassembly process and represent task dependencies. The AOG adopts a top-down approach that effectively captures the hierarchical structure of the product, the logical relationships among components, and the execution sequence of disassembly tasks. In this graph, each node represents a component, and each directed edge between parent and child nodes denotes a disassembly operation.

Fig. 2 illustrates the composition of the washing machine, while Fig. 3 presents its corresponding disassembly AOG. The AOG incorporates various constraints, including conflict relationships, task precedence, and task-to-subcomponent mappings. As shown in Fig. 3, the disassembly process involves 19 subcomponents, 8 parts, and 14 disassembly tasks. Throughout the process, both conflict relationships and precedence constraints must be satisfied to ensure feasible disassembly sequences. For example, subcomponent <1> can be disassembled by performing task 1, which separates subcomponents <2> and <7>. Subsequently, subcomponent <2> can be further disassembled to obtain subcomponent <5>. In addition, subcomponent <1> is associated with tasks 3 and 4, but these tasks cannot be executed simultaneously due to conflict constraints.

The following assumptions are made to develop the model based on the above descriptions :

- Disassembly operations are conducted on a linear disassembly line within each disassembly center.
- Each disassembly line contains a fixed number of workstations.
- The quantity of components obtained from disassembly exceeds the order demand, ensuring that each disassembly center can meet the assignment requirements.

The following information is provided: the precedence relation matrix, conflict relation matrix, and disassembly relation matrix derived from the AOG, which describe the relationships between tasks and components of the product. In addition, matrices related to order information are defined.

1) The disassembly relation matrix $D = [d_{pji}]$ represents the relationship between component p and its disassembly task i. The matrix element d_{pji} is defined as:

$$d_{pji} = \begin{cases} 1, & \text{executing task } i \text{ of product } p \text{ yields component } j. \\ -1, & \text{executing task } i \text{ of product } p \text{ results in the loss of component } j. \\ 0, & \text{otherwise.} \end{cases}$$

2) The conflict relation matrix $R = [i^C_{pii'}]$ represents whether two tasks i and i' of product p can be executed simultaneously. The matrix element $i^C_{pii'}$ is defined as:

$$i^C_{pii'} = \begin{cases} 1, & \text{tasks } i \text{ and } i' \text{ of product } p \text{ have a conflict relationship.} \\ 0, & \text{otherwise.} \end{cases}$$

3) The precedence relation matrix $S = [i^P_{pii'}]$ indicates whether task i of product p is a predecessor of task i'. The matrix element $i^P_{pii'}$ is defined as:

$$i^P_{pii'} = \begin{cases} 1, & \text{task } i \text{ of product } p \text{ is the immediate predecessor of task } i'. \\ 0, & \text{otherwise.} \end{cases}$$

4) The demand quantity matrix $Q = [q_{mo}]$ represents the quantity of components required by manufacturing center m for order o. Assuming the order set is $\mathbb{O} = o_1, o_2, o_3$ and there are two manufacturing centers m_1 and m_2, the demand quantity matrix can be expressed as:

$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \end{bmatrix}$$

5) The matching relation matrix $U = [\mu_{pjo}]$ indicates whether component j of product p matches the component required by order o. The matrix element μ_{pjo} is defined as:

$$\mu_{pjo} = \begin{cases} 1, & \text{part } j \text{ of product } p \text{ is required by order } o. \\ 0, & \text{otherwise.} \end{cases}$$

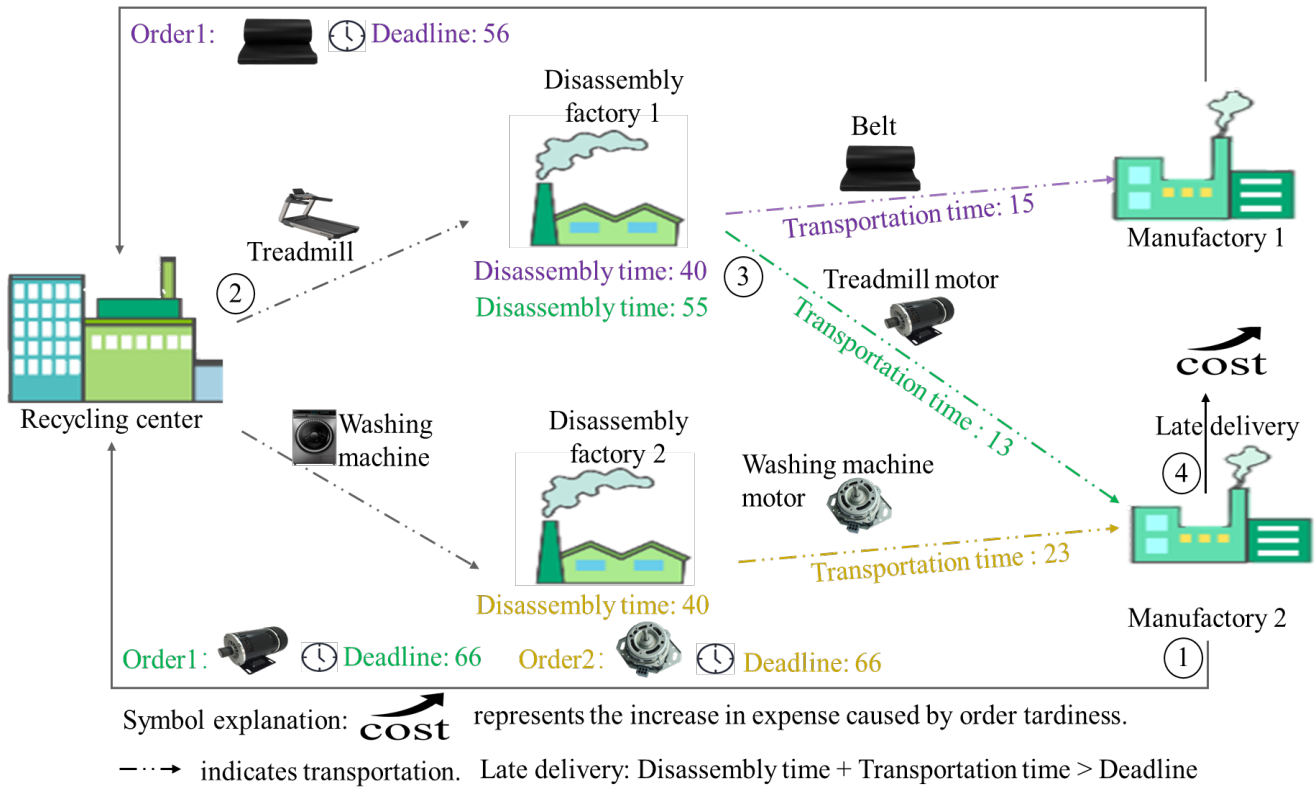


Fig. 1. Illustration of the OTMR process.

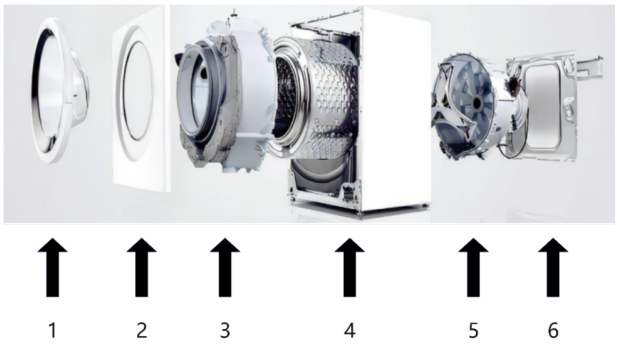


Fig. 2. The schematic diagram of a washing machine.

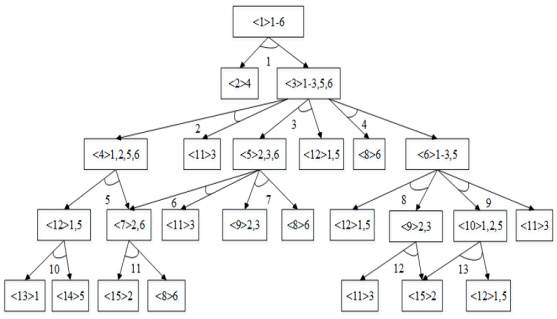


Fig. 3. The AND/OR graph of the disassembly process of a washing machine.

B. Notations and Decision Variables

1) Notations

$\mathbb{H}$	Set of indices of disassembly centers, $\mathbb{H} = \{1, 2, \ldots, H\}$.
$\mathbb{M}$	Set of indices of manufacturing centers, $\mathbb{M} = \{1, 2, \ldots, M\}$.
$\mathbb{P}$	Set of indices of products, $\mathbb{P} = \{1, 2, \ldots, P\}$.
$\mathbb{J}_p$	Set of indices of components of product p, $\mathbb{J}_p = \{1, 2, \ldots, J_p\}$, $p \in \mathbb{P}$.
$\mathbb{I}_p$	Set of indices of tasks of product p, $\mathbb{I}_p = \{1, 2, \ldots, I_p\}$, $p \in \mathbb{P}$.
$\mathbb{W}^h$	Set of indices of workstations in disassembly center h, $\mathbb{W}^h = \{1, 2, \ldots, W^h\}$, $h \in \mathbb{H}$.
$\mathbb{O}$	Set of indices of orders, $\mathbb{O} = \{1, 2, \ldots, O\}$.
$\mathbb{A}$	A virtual concept used to sort tasks assigned to the same workstation, $\mathbb{A} = \{1, 2, \ldots, A\}$.
c^T_{hmpj}	Transportation cost of component j from disassembly center h to manufacturing center m for product p.
t^D_{kpiw}	Disassembly time of task i for product p at workstation w in disassembly center k.
c^D_{kpi}	Disassembly cost (non-time-based) of task i for product p in disassembly center k.
c^O_h	Unit operating cost per time for disassembly center h.
c^S_{hw}	Fixed cost for opening workstation w in disassembly center h.
c^R_p	Outbound cost of product p from recycling center R to a disassembly center.
q_{mo}	Demand of order o at manufacturing center m.
c^P_{mo}	Penalty cost for tardiness of order o at manufacturing center m.
t^M_{mo}	Deadline time of order o requested by manufacturing center m.
t^T_{hm}	Transportation time from disassembly center h to manufacturing center m.

2) Decision Variables

$$z_{ph} = \begin{cases} 1, & \text{if product } p \text{ is assigned to disassembly center } h; \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{pihwa} = \begin{cases} 1, & \text{if task } j \text{ of product } p \text{ is assigned as the } a\text{-th} \\ & \text{task to workstation } w \text{ in disassembly center } k; \\ 0, & \text{otherwise.} \end{cases}$$

$$y_h = \begin{cases} 1, & \text{if disassembly center } h \text{ is opened;} \\ 0, & \text{otherwise.} \end{cases}$$

$$u_{hw} = \begin{cases} 1, & \text{if workstation } w \text{ in disassembly center } h \\ & \text{is opened;} \\ 0, & \text{otherwise.} \end{cases}$$

$$\alpha_{hmpjo} = \begin{cases} 1, & \text{if component } j \text{ of product } p \text{ is transported} \\ & \text{from disassembly center } h \text{ to manufacturing} \\ & \text{center } m \text{ to fulfill order } o; \\ 0, & \text{otherwise.} \end{cases}$$

$$\lambda_{mo} = \begin{cases} 1, & \text{if order } o \text{ at manufacturing center } m \\ & \text{is delayed;} \\ 0, & \text{otherwise.} \end{cases}$$

t^S_{hpiwa} Start time of the a-th disassembly task i for product p at workstation w in disassembly center h.

T_h Operating time of disassembly center h.

C. Mathematical Model

1) Objective of Optimization

$$\begin{aligned} \min \ & \sum_{h\in\mathbb{H}}\sum_{m\in\mathbb{M}}\sum_{p\in\mathbb{P}}\sum_{j\in\mathbb{J}_p}\sum_{o\in\mathbb{O}_p} c^T_{hmpj}\alpha_{hmpjo} \\ & + \sum_{h\in\mathbb{H}}\sum_{p\in\mathbb{P}}\sum_{i\in\mathbb{I}_p}\sum_{w\in\mathbb{W}^h}\sum_{a\in\mathbb{A}} c^D_{hpi}x_{pihwa} \\ & + \sum_{h\in\mathbb{H}} c^O_h T_h + \sum_{h\in\mathbb{H}}\sum_{w\in\mathbb{W}^h} c_{hw}u_{hw} \\ & + \sum_{h\in\mathbb{H}}\sum_{p\in\mathbb{P}} z_{pk}c^R_p + \sum_{m\in\mathbb{M}}\sum_{o\in\mathbb{O}} c^P_{mo}\lambda_{mo} \end{aligned} \tag{1}$$

The objective function 1 aims to minimize the total cost and order tardiness penalties after fulfilling all orders. Specifically, the first term represents the total transportation cost, the second term represents the total disassembly cost, the third term denotes the total cost of establishing disassembly centers, the fourth term denotes the total cost of setting up workstations, the fifth term accounts for the total transportation cost from recycling centers to disassembly centers, and the sixth term captures the penalty cost associated with order tardiness.

2) Fundamental Constraints

$$\sum_{h\in\mathbb{H}} z_{ph} \le 1, \quad \forall p \in \mathbb{P} \tag{2}$$

$$z_{ph} \le y_h, \quad \forall p \in \mathbb{P},\ \forall h \in \mathbb{H} \tag{3}$$

$$u_{hw} \le y_h, \quad \forall w \in \mathbb{W}^h,\ \forall h \in \mathbb{H} \tag{4}$$

$$y_h \le T_h, \quad \forall h \in \mathbb{H} \tag{5}$$

$$x_{pihwa} \le z_{ph}, \forall p \in \mathbb{P}, \forall i \in \mathbb{I}_p, \forall h \in \mathbb{H}, \forall w \in \mathbb{W}^h, \forall a \in \mathbb{A} \tag{6}$$

$$x_{pihwa} \le u_{hw}, \forall p \in \mathbb{P}, \forall i \in \mathbb{I}_p, \forall h \in \mathbb{H}, \forall w \in \mathbb{W}^h, \forall a \in \mathbb{A} \tag{7}$$

$$\sum_{h\in\mathbb{H}}\sum_{w\in\mathbb{W}^h}\sum_{a\in\mathbb{A}} x_{pihwa} \le 1, \quad \forall p \in \mathbb{P},\ \forall i \in \mathbb{I}_p \tag{8}$$

$$\forall p \in \mathbb{P} \sum_{i\in\mathbb{I}_p} t_{pi}x_{pihwa} \le 1, \quad \forall h \in \mathbb{H},\ \forall w \in \mathbb{W}^h, \forall a \in \mathbb{A} \tag{9}$$

Constraint (2) ensures that each product can only be assigned to one disassembly center. Constraint (3) specifies that a product can only be assigned to an opened disassembly center. Constraint (4) indicates that only workstations in opened disassembly centers can be utilized. Constraint (5) states that a factory is considered operational only when it has tasks assigned. Constraint (6) ensures that a disassembly center must be open if a product is assigned to it. Constraint (7) states that a task of a product can only be assigned as the k-th task to a workstation within the disassembly center to which the product is allocated. Constraint 8) ensures that each disassembly task of each product is performed at most once. Constraint (9) ensures that each workstation can perform at most one task in each time period.

3) Task Relationship Constraints

$$\begin{aligned} & \sum_{p\in\mathbb{P}}\sum_{i\in I_p} x_{pihwa} \ge \sum_{p\in\mathbb{P}}\sum_{i\in I_p} x_{pikw(a+1)}, \\ & \forall h \in \mathbb{H},\ \forall w \in \mathbb{W}^h,\ \forall a \in \{1,\ldots,A-1\} \end{aligned} \tag{10}$$

$$\begin{aligned} x_{pi_2hw_2a_2} \le & \sum_{i_1\in\mathbb{I}_p}\sum_{a_1=1}^{a_2-1} i^P_{pi_1i_2}\cdot x_{pi_1hw_2a_1} \\ & + \sum_{i_1\in\mathbb{I}_p}\sum_{h_1=1}^{h_2-1}\sum_{a_1\in\mathbb{A}} i^P_{pi_1i_2}\cdot x_{pi_1kw_1a_1}, \\ & \forall p \in \mathbb{P},\ \forall i_2 \in \mathbb{I}_p,\ \forall h \in \mathbb{H},\ \forall w_2 \in \mathbb{W}^h,\ \forall a_2 \in \mathbb{A},\ d_{pii_2} = 0 \end{aligned} \tag{11}$$

$$\begin{aligned} & \sum_{w\in\mathbb{W}^h}\sum_{a\in\mathcal{A}}(x_{pi_1hwa} - x_{pi_2hwa}) + W^h\left(\sum_{w\in\mathbb{W}^h}\sum_{a\in\mathbb{A}} x_{pi_2kwa} - 1\right) \\ & \le 0, \forall k \in \mathbb{K},\ \forall p \in \mathbb{P},\ \forall i_1, i_2 \in I_p, \text{ and } i^P_{pi_1i_2} = 1, \forall a \in \mathbb{A} \end{aligned} \tag{12}$$

$$\sum_{w\in\mathbb{W}^h}\sum_{a\in\mathcal{A}} x_{pqhwa} \le \sum_{i\in I_p}\sum_{w\in\mathbb{W}^h}\sum_{a\in\mathbb{A}} s_{piq}x_{pihwa}, \quad \forall p\in\mathbb{P},\ \forall i,q\in I_p,\ \forall h\in\mathbb{H},\ and\ d_{d1q}=0 \tag{13}$$

$$\sum_{w\in\mathbb{W}^h}\sum_{a\in\mathbb{A}}(x_{pi_1hwa}+x_{pi_2hwa})\le 1, \quad \forall h\in\mathbb{H},\ \forall p\in\mathbb{P},\ \forall i_1,i_2\in\mathbb{I}_p,\ i^C_{i_1i_2}=1 \tag{14}$$

$$t^S_{hpiwa}+t^D_{hpiw}\le T+T^{max}(1-y_h), \quad \forall h\in\mathbb{H},\ \forall w\in\mathbb{W}^h,\ \forall a\in\mathbb{A},\forall i\in\mathbb{I}_p \tag{15}$$

Constraint (10) ensures that tasks assigned to each workstation follow a compact sequence. Given that the number of workstations is sufficient, this constraint guarantees that tasks are executed in consecutive positions (e.g., 1st, 2nd, 3rd, etc.) without gaps. Constraint (11) defines the positional relationship between immediate predecessor tasks. Constraints (12)–(13) enforce the precedence requirements for task assignments of each product. Constraint (14) ensures that task assignments respect conflict constraints. Constraint (15) enforces the disassembly dependency relationships among tasks.

4) Order Allocation Constraints

$$\sum_{i\in I_p}\sum_{w\in\mathbb{W}^h}\sum_{a\in\mathbb{A}} d_{pji}x_{pihwa}\ge\sum_{m\in\mathbb{M}}\sum_{o\in\mathbb{O}}\alpha_{hmpjo}, \quad \forall h\in\mathbb{H},\ \forall p\in\mathbb{P},\forall j\in\mathbb{J}_p, \tag{16}$$

$$\sum_{h\in\mathbb{H}}\sum_{p\in\mathbb{P}}\sum_{j\in J_p}\mu_{pjo}\cdot\alpha_{hmpjo}=q_{mo},\forall m\in\mathbb{M},\ \forall o\in\mathbb{O} \tag{17}$$

$$\sum_{h\in\mathbb{H}}\alpha_{hmpio}\le\mu_{pjo},\quad \forall p\in\mathbb{P},\ \forall j\in\mathbb{J}_p,\ \forall m\in\mathbb{M},\ \forall o\in\mathbb{O} \tag{18}$$

$$t^S_{hpi_2w_2a_2}\ge T^{max}(x_{pi_2hw_2a_2}+x_{pi_1kw_1a_1}-2)+ t^S_{hpi_1w_1a_1}+t^P_{hpi_1w_1},\forall i_1,i_2\in\mathbb{I}_p,\ \forall w_1,w_2\in\mathbb{W}^h, \quad w_1\ne w_2,\ \forall h\in\mathbb{H},\ i^P_{pi_1i_2}=1 \tag{19}$$

$$t^S_{hp_2i_2wa}\ge T^{max}(x_{p_2i_2hwa}+x_{p_1i_1hw(a-1)}-2)+ t^S_{hp_1i_1w(a-1)}+t^D_{hp_1i_1w},\forall p_1,p_2\in\mathbb{P}, \quad \forall i_1,i_2\in I_p,\ i_1\ne i_2,\ \forall w\in\mathbb{W}^h,\ \forall h\in\mathbb{H} \tag{20}$$

$$t^S_{hpiwa}+t^D_{hpiw}+t^T_{hm}\le T^{max}\left(1-d_{pji}\cdot\alpha_{hmpjo}+\lambda_{mo}\right) +t^M_{mo},\forall p\in\mathbb{P},\ \forall i\in I_p,\ \forall h\in\mathbb{H}, \quad \forall w\in\mathbb{W}^h,\ \forall m\in\mathbb{M},\ \forall o\in\mathbb{O} \tag{21}$$

Constraint (16) ensures that the opening time of each disassembly center is not earlier than the latest completion time of all assigned disassembly tasks. Constraints (17)–(18) ensure that the decision variables satisfy the order allocation conditions. Constraint (19) defines the relationship between the start times of tasks belonging to the same product when executed on different workstations. Constraint (20) specifies the relationship among the start times of tasks from different products assigned to the same workstation. Constraint (21) defines the time relationship under conditions of order tardiness or on-time completion.

$$z_{ph}\in\{0,1\},\quad \forall p\in\mathbb{P},\ \forall h\in\mathbb{H} \tag{22}$$

$$x_{pihwa}\in\{0,1\},\forall p\in\mathbb{P},\ \forall i\in\mathbb{I}_p,\ \forall w\in\mathbb{W}^h,\forall h\in\mathbb{H},\forall a\in\mathbb{A} \tag{23}$$

$$y_h\in\{0,1\},\quad \forall h\in\mathbb{H} \tag{24}$$

$$u_{hw}\in\{0,1\},\quad \forall h\in\mathbb{H},\ \forall w\in\mathbb{W}^h \tag{25}$$

$$t^S_{hpiwa}\in\mathbb{R}_+,\forall p\in\mathbb{P},\ \forall i\in\mathbb{I}_p,\ \forall w\in\mathbb{W}^h,\forall h\in\mathbb{H},\forall a\in\mathbb{A} \tag{26}$$

$$T_h\in\mathbb{R}_+,\quad \forall h\in\mathbb{H} \tag{27}$$

Constraints (22)-(27) define the value domains of each decision variable.

III. Proposed Algorithm

This section applies the DDQN to solve the proposed OTMR problem. RL environment is constructed by defining the action space, state space, and reward function to suit the characteristics of the problem. In particular, the impact of order delays is explicitly incorporated into the environment design to enhance the policy's responsiveness to time-sensitive constraints.

A. Markov Decision Process

In the RL framework, the scheduling problem is typically modeled as a Markov Decision Process (MDP). An MDP consists of a state space, an action space, transition probabilities, and a reward function, which together describe the interaction between the agent and the environment. In the context of resource scheduling, the system state may include features such as the current time, the remaining processing requirements of orders, and the allocation status of factory resources. The actions correspond to task assignment decisions made at each time step. The environment transitions to the next state based on the current state and selected action, and returns a reward that reflects the quality of the scheduling decision. Due to the large and complex state-action space and the uncertainty in environment dynamics, traditional optimization methods struggle to find effective strategies. Therefore, this study employs a deep RL approach based on DDQN to optimize the scheduling policy [36].

B. Algorithm Description

This study adopts the DDQN to improve the learning performance in the proposed scheduling environment. DDQN is an extension of the standard Deep Q-Network (DQN) [37, 38], designed to mitigate the problem of overestimation in Q-value updates. It decouples action selection and evaluation by using separate networks, which leads to more stable value estimation during training. Specifically, DDQN computes the target Q-value as:

$$y^{\text{DDQN}} = r + \gamma Q_{\text{target}}\left(s', \arg\max_{a'} Q_{\text{online}}(s', a')\right) \tag{28}$$

Here, r denotes the immediate reward received after taking an action, and $\gamma \in [0, 1]$ is the discount factor that controls the weight of future rewards. s' represents the next state after the action is executed. $Q_{\text{online}}(s', a')$ is the value predicted by the online network for taking action a' in state s', and $\arg\max_{a'} Q_{\text{online}}(s', a')$ selects the action with the highest estimated value according to the online network. $Q_{\text{target}}(s', \cdot)$ is the target network used to evaluate the value of the selected action in the next state, thereby forming the target value y^{DDQN} for updating the Q-network. This formulation helps reduce the positive bias introduced by the max operator in standard DQN, resulting in more reliable policy learning in complex scheduling environments. The structure and specific workflow of DDQN are illustrated in Fig. 4.

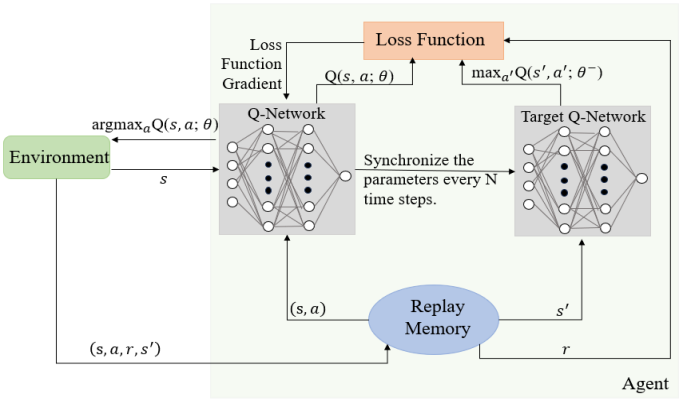


Fig. 4. Architecture of DDQN. The online Q-network selects actions, while the target Q-network evaluates the corresponding Q-values. A replay memory stores past transitions to improve training stability and sample efficiency.

In the context of complex scheduling with high-dimensional state spaces and delayed rewards, DDQN offers better convergence properties and improved decision robustness compared to traditional value-based methods. These advantages make it well-suited for solving the resource allocation and order-timing challenges in the proposed scenario.

C. State and Action Design

The state space defines the information observed by the agent at each time step and forms the basis for decision-making. To enable effective scheduling decisions, this study constructs a multi-dimensional dynamic state space based on key features, including the number of products p, the number of disassembly centers h, the number of workstations w, and the maximum number of tasks for each product i. These features reflect the current system configuration and task complexity, providing essential input for action selection. The structure of the state space is defined as:

$$\text{space}_{\text{state}} = (p_t, h_t, w_t, i_t) \tag{29}$$

The action space defines the set of possible decisions the agent can make at each time step, and it plays a critical role in both the convergence speed and effectiveness of the reinforcement learning algorithm.

In the proposed scheduling scenario, the action space mainly focuses on resource allocation within disassembly factories and order delay in the transportation process. Specifically, the agent must determine which product to disassemble, assign it to a disassembly center, allocate tasks to workstations, and decide the task sequence for each product. Based on these decisions, the initial action space is defined as:

$$\text{space}_{\text{action}} = P \times H \times W \times I \tag{30}$$

This formulation captures the core variables in disassembly task allocation. However, if the task distribution across different work positions and manufacturing centers is also considered, the action space further expands to:

$$\text{space}_{\text{action\_e}} = P \times H \times W \times I \times A \times M \tag{31}$$

where A denotes the number of work positions and M refers to the number of manufacturing centers. Although this extension provides a more comprehensive representation, it significantly increases the agent's decision space and reduces training efficiency. To address this challenge, we simplify the action space. At each time step, the agent sequentially assigns tasks to corresponding workstations based on the current state and historical actions, thus removing dimension A. For the transportation phase, a greedy algorithm [39] is applied to determine the routing plan. The environment records the disassembly completion time of each component, and order delays are evaluated by comparing this time with the transportation time t^T_{hm}. The overall cost and penalty are then computed using the transportation cost c^T_{hmpj} and delay status, which allows the removal of dimension M. This strategy significantly reduces the search space and improves the efficiency and stability of the learning process.

To further illustrate the execution process of an action within the proposed action space, Fig. 6 presents a flowchart depicting how the agent interacts with the environment after selecting an action. The process includes action decoding, state updates, order delay evaluation, and reward computation. This diagram reflects the integration of task-level decision-making and cost-based evaluation, providing an intuitive understanding of how the simplified action space interacts with the environment dynamics.

D. Reward Design

In RL, the agent selects actions based on the current state and the received reward. The reward function serves as a

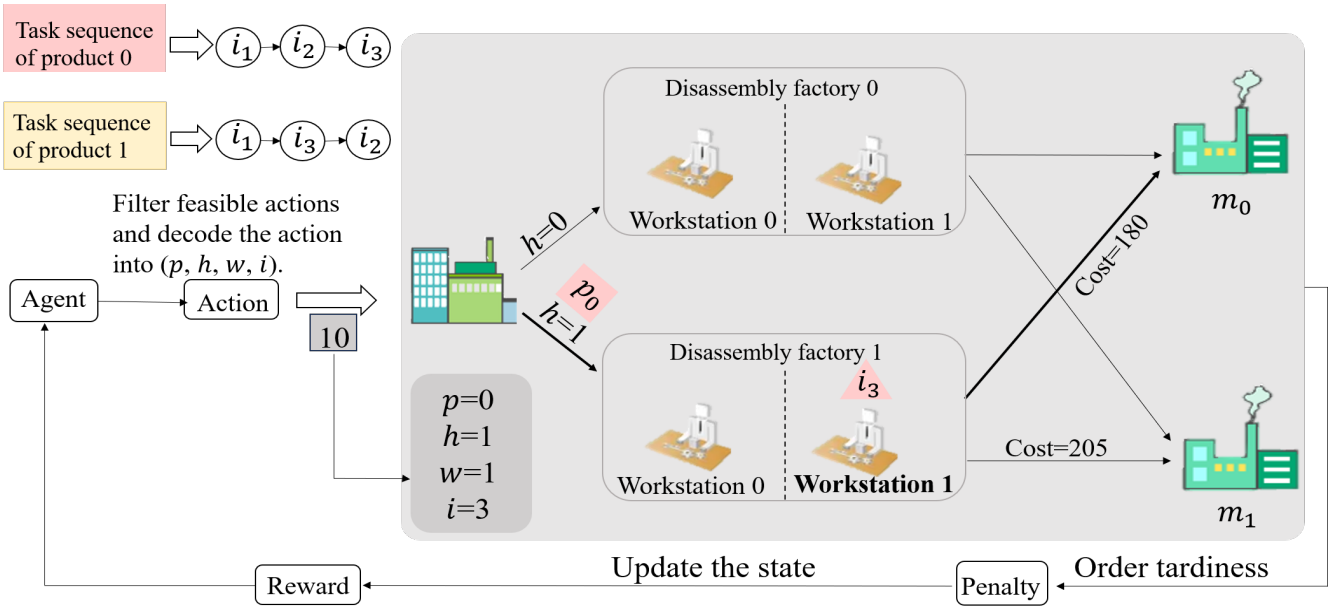


Fig. 5. The decision process of a DDQN agent in a disassembly scheduling environment.

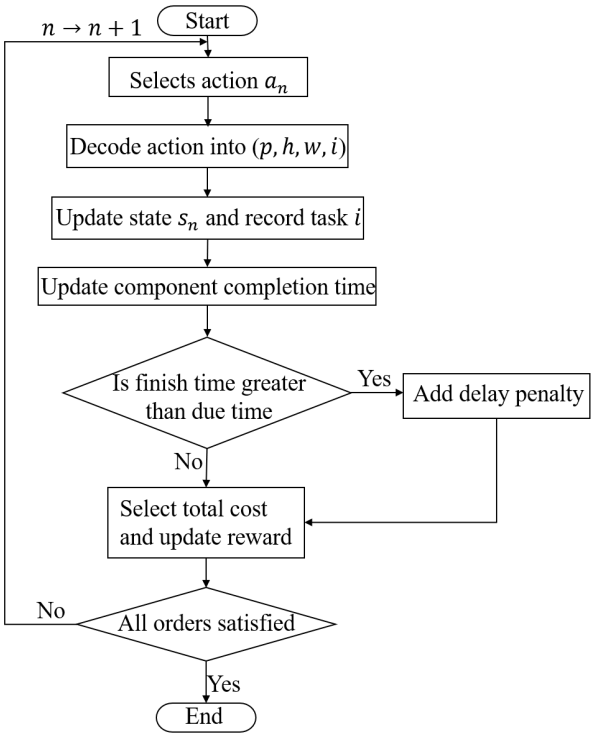


Fig. 6. The execution process of agent-selected actions within the proposed action space.

numerical signal that evaluates the quality of an action and guides the agent to improve its policy. In the multi-stage resource optimization problem, the total cost—including both operational expenses and order delay penalties—is calculated at each step. The negative of this total cost is used as the reward, which encourages the agent to learn strategies that minimize overall costs. As a result, lower costs lead to higher rewards, promoting more efficient decision-making in complex scheduling environments.

E. Environment Structure

The environment first loads the instance parameters and sets the initial state as $s_0 = (-1, 0, 0, 0)$. Feasible tasks are identified using the disassembly matrix D, the demand matrix Q, and the matching matrix U, which help filter out invalid options and simplify the action space. For example, in the washing machine case shown in Fig. 3, matrix D indicates that each component is associated with up to three tasks, leading to a reduced action space of $P \times H \times W \times 3$.

At each time step n, the agent selects an action a_n, which is first mapped to a set of valid actions according to the current task constraints. A modulo operation ensures that a_n corresponds to a feasible choice. The environment then decodes the action into its components (p, h, w, i), updates the state s_n, and adjusts the component completion time and workstation usage accordingly.

Based on these updates, the system calculates disassembly costs, transportation costs, and time-based resource costs using predefined parameters. The decision-making process of the DDQN agent in the disassembly scheduling environment is illustrated in Fig. 5. Since only feasible tasks are allowed, no penalty terms are introduced in the reward function, effectively reducing the agent's exploration space. The episode terminates when the number of completed orders satisfies the demand specified in matrix Q; otherwise, the process continues to the next time step. The complete procedure is summarized in Algorithm III-E.

Require: Action a_n, state s_n, environment parameters (P, H, W, I, D, Q, U)

Ensure: Next state s_{n+1}, reward r_{n+1}, termination signal *done*

1: Filter feasible tasks using matrices D, Q, and U
2: Map a_n to the valid action set
3: $a_n \leftarrow a_n$ mod size of feasible action set
4: Decode a_n into (p, h, w, i)
5: Update state s_n based on (p, h, w, i)

Record task i for product p, update component list and completion time
Update workstation time for workstation w at disassembly center h
/ Transportation cost and penalty */*
for each component j of product p to be transported **do**
  for each candidate manufacturing center m **do**
    Compute transport time $t_{trans} = t^T_{hm}$
    Compute finish time $t_{finish} = t_{comp} + t_{trans}$
    if t_{finish} > order due time t^d_{mo} **then**
      Apply delay penalty $penalty = c^d_{mo}$
    else
      $penalty = 0$
    end if
    Compute total transport cost $c_{trans} = c^T_{hmpj} + penalty$
  end for
  Decide manufacturing center m with minimum c_{trans}
end for
Compute disassembly cost C_d, transport cost C_t, and factory time cost C_f
Compute total cost $C = C_d + C_t + C_f$
Set reward $r_{n+1} = -C$
if completed orders match the demand matrix Q **then**
  $done \leftarrow$ True
else
  $done \leftarrow$ False
end if
return $s_{n+1}, r_{n+1}, done$

IV. Experimental Studies

This section applies RL algorithms to solve the OTMR problem. The proposed model is first validated using CPLEX, and the same instances are solved for comparative analysis. Specifically, DQN, A2C, and PPO from the Stable-Baselines3[40] library are adopted. All experiments are performed on a Windows 11 system with a 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz processor.

A. Instance Generation

To evaluate the performance of the proposed method, several case instances are constructed based on disassembly scheduling scenarios involving products such as washing machines[41], radios[42], and treadmills[43]. Each instance specifies the number of products, the number of tasks per product, and the configuration of disassembly centers and workstations.

a) Instance configuration: To test the effectiveness of the proposed approach under varying levels of complexity, six instances are constructed using different combinations of product types. Each product corresponds to a predefined disassembly task structure, represented by an AND/OR graph as described in the modeling section. These structures are converted into dependency matrices and used to build the RL environment. Table I summarizes the test instances, including the number of products of each type, the total disassembly tasks, and the corresponding order count for each case.

For small-scale instances, the solution quality of DDQN is evaluated against the CPLEX optimum using the optimality gap, defined as

$$\text{Gap}(\%) = \frac{Z_{\text{DDQN}} - Z_{\text{CPLEX}}}{Z_{\text{CPLEX}}} \times 100\%,$$

where Z_{CPLEX} and Z_{DDQN} denote the objective values obtained by CPLEX and DDQN, respectively.

b) Factory and workstation configuration: Each disassembly center is equipped with a predefined number of workstations, enabling task allocation across different stations within the same center. In this study, two disassembly centers are configured, each with three workstations. The setup cost per center ranges from $[3, 5]$, while the workstation operation costs vary between $[5, 10]$. All centers are capable of processing all types of products.

c) Order setting: Orders are defined based on required components, due dates, and associated delay penalties. Each manufacturing center processes specific orders, which require components disassembled from different products. The order structure is defined using three main matrices: the demand matrix q_{mo}, the matching matrix μ_{pjo}, and the deadline matrix t_{mo}. If the actual delivery time exceeds the deadline, a penalty c_{mo} is incurred. The delivery time is calculated based on the component completion time at the disassembly center and the transport time t_{hm} to the manufacturing center. The reward function incorporates these penalties to encourage on-time delivery. Table II illustrates the detailed order configurations across six experimental cases, including component requirements, source products, deadlines, and penalties. To simulate different urgency levels and customer priorities, orders with identical component types may be assigned different penalty values across cases.

d) Hyperparameter setting: The hyperparameters of DDQN were determined based on preliminary experiments to balance convergence speed and training stability. Specifically, the learning rate was set to 0.0003, the discount factor for future rewards was set to 0.9999, and the soft update coefficient was set to 0.005. The replay buffer size and batch size were set to 50,000 and 500, respectively. The exploration rate ϵ was initialized at 1.0 and gradually decreased with a decay rate of 5×10^{-6} until reaching 0. Two fully connected hidden layers with 500 neurons each were adopted in the Q-network. These settings were found to provide stable performance for the OTMR problem.

All instance parameters, including processing times, disassembly costs, transportation costs, and delivery time windows, are randomly generated based on practical estimates. These instances are designed to reflect the complexity of real-world

TABLE I Cases for Experiments

Case ID	Number of Products			Task Count	Order Count
	Washing Machine	Treadmill	Radio		
1	0	1	0	17	1
2	2	0	0	26	2
3	0	0	1	30	2
4	0	4	0	68	4
5	2	3	0	77	3
6	0	0	5	150	8

TABLE II Order Configuration in Case Studies

Case ID	Manufactory Center ID	Order Qty	Required Components	Source Products	Component Qty per Order	Deadline	Penalty Cost
Case 1	1	1	Belt	Treadmill	1	90	155
Case 2	1	2	Motor	Washing Machine	1	(130, 134)	(505, 529)
Case 3	1	1	Antenna	Radio	1	98	510
		1	Receiver	Radio	1	93	516
Case 4	1	4	Belt	Treadmill	1	(107, 196)	(520, 523)
Case 5	1	2	Belt	Treadmill	1	50	200
		1	Motor	Washing Machine	3	60	210
Case 6	1	2	Antenna	Radio	1	100	168
		2	Receiver	Radio	1	(101, 117)	(116, 155)
	2	2	Antenna	Radio	1	127	192
		2	Receiver	Radio	1	(124, 145)	(129, 134)

Note: For orders with a single instance, specific deadlines and penalties are listed; for repeated orders, only the range of deadlines and penalties is shown to avoid redundancy.

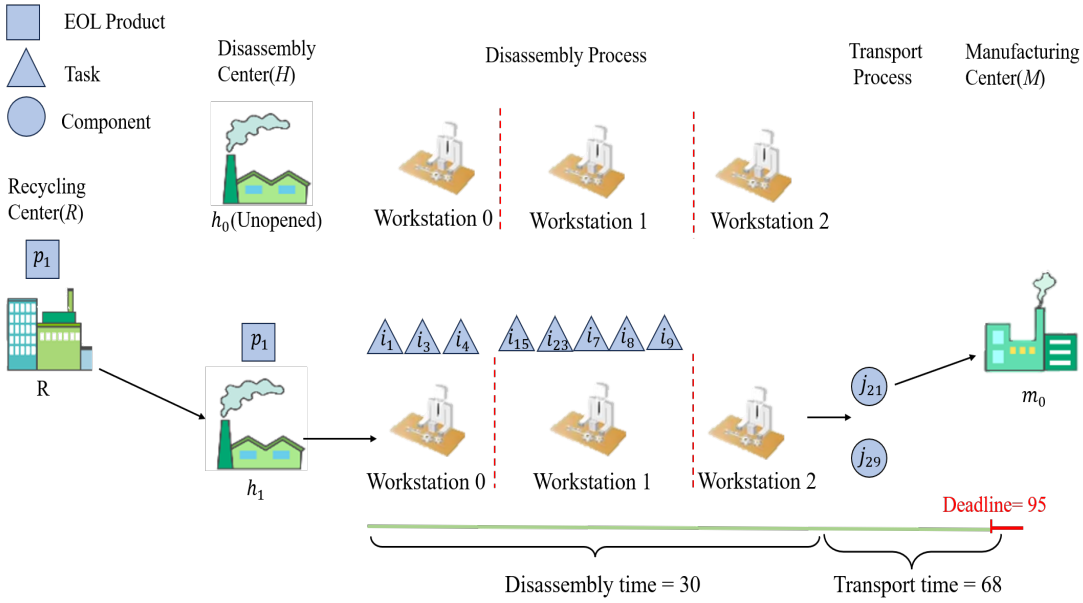


Fig. 7. Scheduling result of Case 3 under the OTMR model.

disassembly scheduling scenarios while maintaining controlled conditions for algorithm evaluation.

B. Case Study

Fig. 7 presents the experimental results of Case 3 under the proposed OTMR model. Different symbols represent the manufacturing, recycling, and disassembly centers, while triangles correspond to the extracted components. The red elements indicate the delivery deadlines, reflecting the time constraints imposed on order completion.

As shown in the figure, the proposed scheduling strategy effectively coordinates the flow of products and components among facilities, ensuring that most operations are completed before the deadlines. This demonstrates the model's capability to manage time-sensitive remanufacturing tasks while maintaining efficient resource utilization.

C. Sensitivity Analysis

To evaluate the robustness and adaptability of the proposed scheduling model, we conduct a sensitivity analysis by varying key input parameters while keeping the base configuration fixed. Case 5 is selected as it represents a moderate-scale scenario with diverse product types and sufficient scheduling complexity. The analysis examines the effects of three critical parameters: (i) the order deadline time coefficient t_{mo}, and (ii) the number of available disassembly lines. Their impact on total cost, delay penalties, and resource utilization is assessed.

a) Impact of Order Deadline Time (t_{mo}): To examine how order due times influence scheduling performance, we vary the delivery deadlines T_{mo} while keeping other parameters fixed. In particular, the total number of orders is 3, and the number of order types (O) is 2. As shown in Table III and Fig. 8, tighter delivery windows (e.g., $t_{mo} \in [20, 30]$) result in overdue orders and high penalty costs. In contrast, when t_{mo} is moderately relaxed (e.g., $[40, 50]$ or beyond), the algorithm completes all orders on time, eliminating delay penalties.

The total cost decreases as t_{mo} increases, due to the reduction in penalty cost and improved scheduling flexibility. However, after a certain threshold (e.g., $t_{mo} \geq [50, 60]$), further increases in due time yield limited benefits. This

TABLE III Order Deadline Time (t_{mo})

t_{mo}	Late Orders	Total Cost	Penalty Cost	Completion Times
[20, 30]	1	1063	410	[38, 20]
[30, 40]	1	853	200	[29, 31]
[35, 45]	1	861	200	[32, 42]
[40, 50]	0	641	0	[31, 33]
[50, 60]	0	649	0	[49, 53]
[70, 70]	0	664	0	[59, 46]
[80, 90]	0	637	0	[39, 41]
[90, 100]	0	628	0	[36, 35]

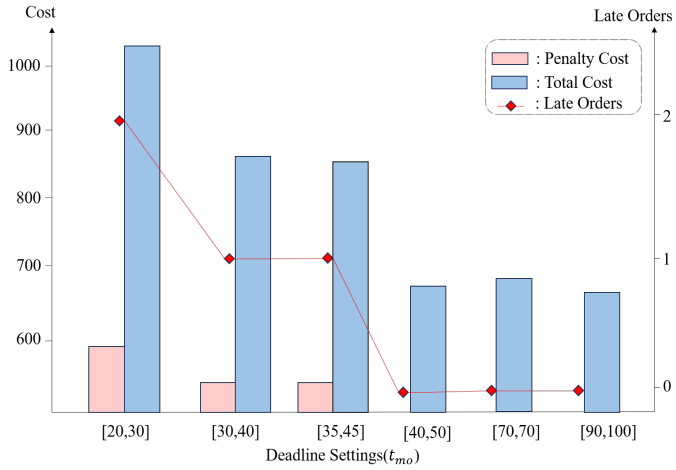


Fig. 8. Sensitivity analysis of total cost and lateness with respect to t_{mo}.

indicates that the model already achieves timely delivery once a feasible window is provided.

Overall, the proposed method shows robustness across different time constraints. It dynamically adjusts scheduling decisions to balance penalty avoidance and resource utilization, demonstrating strong adaptability to varying delivery requirements.

b) Impact of Available Disassembly Centers.: This experiment investigates how varying the number of available disassembly plants influences the overall performance of the scheduling model. The plant quantity is increased from 1 to 5, while keeping the number of orders, product types, and all other parameters unchanged. Table IV presents the results, including total cost, workstation utilization, and plant utilization.

The results show that the total cost first decreases and reaches its minimum when two disassembly centers are available. However, as the number of plants increases beyond two, the cost begins to rise again. This non-linear trend suggests that although additional plants may provide more scheduling flexibility, they also introduce higher coordination complexity and potential underutilization of resources.

Both workstation and factory utilization drop significantly as more plants are introduced. For instance, with only one plant, the workstation utilization is 100%, and the factory is fully utilized. In contrast, when five plants are available, only 6 out of 15 workstations and 2 out of 5 factories are used. This indicates a growing imbalance between available capacity and

TABLE IV Sensitivity to the Number of Manufacturing factories

No. of factories	Total Cost	Workstation Utilization	factory Utilization
1	649	1/1	1/1
2	604	3/6	1/2
3	682	5/9	2/3
4	715	5/12	2/4
5	659	6/15	2/5

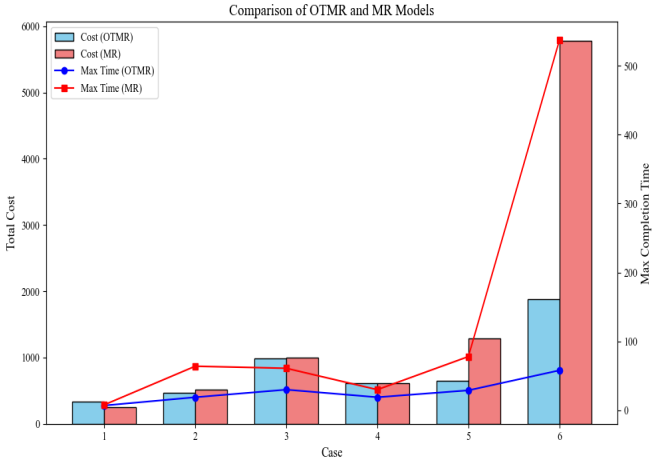


Fig. 9. Comparison of OTMR and MR models in cost and order completion time across different cases

actual task assignments.

In addition, opening more disassembly factories leads to increased fixed and operational costs, even if some factories are only partially utilized. Since the total disassembly workload remains constant, allocating tasks across more facilities fragments the resource usage, reducing overall efficiency. As a result, excessive plant expansion causes unnecessary cost increases without proportional performance gains.

Overall, these results demonstrate that excessive expansion of disassembly capacity may not lead to better performance. Instead, it can result in resource waste and inefficient scheduling. Therefore, optimizing the number of operational factories is critical for cost-effective and balanced system performance.

D. Order Delay Penalty Consideration

This section evaluates the effectiveness of the proposed OTMR model in handling delivery deadlines. Fig. 9 presents a comparative analysis between the OTMR model and the conventional Multi-Factory Remanufacturing (MR) model. The OTMR model integrates order tardiness into the scheduling objective, while the MR model focuses solely on minimizing cost without considering delivery timing.

As shown in the figure, the OTMR model consistently achieves lower total costs and shorter completion times across all cases. By integrating delivery constraints, the OTMR model dynamically balances disassembly and transportation scheduling to ensure timely order fulfillment. In contrast, the MR model often delays order completion, particularly in complex production scenarios, leading to higher overall costs and longer makespan.

TABLE V Experimental comparison between CPLEX and DDQN

Case ID	CPLEX		DDQN		Gap
	Profit	Time(s)	Profit	Time(s)	
1	333	35.07	333	83.92	0
2	466	3618.57	473	359.92	2.4%
3	964	18000.00	981	911.64	4.5%
4	-	-	641	956.85	-
5	-	-	649	946.60	-
6	-	-	2672	1871.61	-

These results demonstrate the importance of accounting for delivery deadlines in remanufacturing scheduling. By jointly optimizing cost and timeliness, the OTMR model produces more efficient, stable, and realistic scheduling outcomes compared with the conventional MR model.

E. Model Validation and Analysis

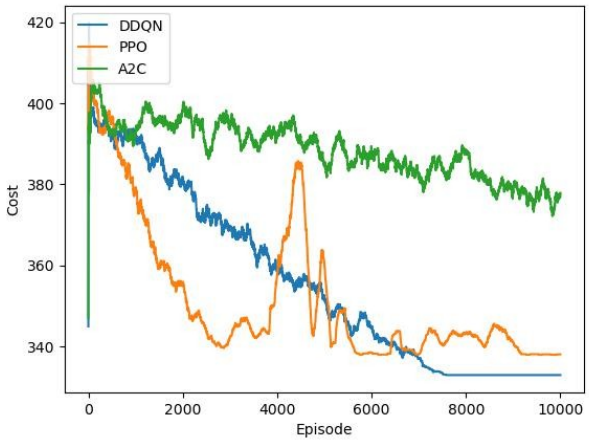
The results in Table V demonstrate the effectiveness of the proposed model and the advantages of the DDQN-based solution method. CPLEX can obtain optimal results for small-scale problems, confirming the model's correctness. However, as the problem scale increases, the CPLEX solver becomes inefficient and fails to produce feasible solutions within the time limit, revealing its computational bottleneck. In contrast, the DDQN algorithm efficiently generates high-quality solutions for all cases, showing strong scalability and practical applicability in large-scale multi-line remanufacturing scheduling problems. CPLEX is used in this study mainly to validate the correctness and feasibility of the proposed mathematical model on small-scale instances. As the problem scale increases, the exact solution approach becomes less suitable due to the rapidly growing computational burden, which motivates the use of RL-based methods for larger instances.

F. Algorithm Comparison and Performance Evaluation

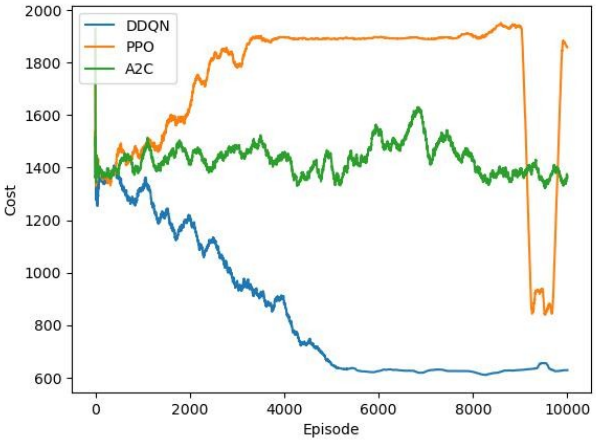
To evaluate the effectiveness of the proposed reinforcement learning methods, we compare three algorithms (DDQN, PPO, and A2C) with the exact solver CPLEX. The comparison considers two criteria: solution quality, measured by total cost, and computation time. All methods operate on six benchmark cases under consistent experimental settings.

a) Experimental Setup: All reinforcement learning algorithms are trained using consistent hyperparameter settings to ensure fair comparison. Each model runs for 10,000 episodes with a learning rate $\alpha = 0.0003$, a discount factor $\gamma = 0.9999$, a replay buffer size of 50,000, and a batch size of 500. An ϵ-greedy strategy is applied, where the exploration rate gradually decays to zero over time. This setup enables sufficient exploration in the early stage and encourages policy exploitation in later training, allowing the algorithms to learn effectively in complex scheduling environments.

b) Algorithm Comparison: Fig. 10 shows the training cost curves for Case 1 and Case 4. In the small-scale Case 1, all three algorithms reduce the cost over time. DDQN achieves the lowest and most stable performance. PPO converges moderately but exhibits noticeable fluctuations. A2C maintains a relatively flat curve with a higher overall cost.



(a) Case 1



(b) Case 4

Fig. 10. Training cost comparison across different problem scales.

TABLE VI Comparison of total costs and computation time

Case	PPO		A2C		DDQN	
	Cost	Time	Cost	Time	Cost	Time
1	338	500.62	399	134.17	**333**	83.92
2	620	2378.14	596	393.85	**473**	359.92
3	1007	2410.49	1489	192.40	**981**	911.64
4	1864	1064.82	830	248.45	**611**	956.85
5	2246	1953.78	4130	343.11	**649**	946.60
6	2790	3334.63	2694	737.89	**2672**	1871.61

In the more complex Case 4, DDQN continues to converge effectively, while PPO becomes unstable and fails to reduce costs. A2C remains stable but produces suboptimal results. These findings suggest that DDQN offers better robustness and scalability across different problem sizes.

Overall, DDQN achieves the best balance between cost and runtime. It converges quickly and handles large-scale instances that exact solvers cannot. PPO performs well in small-scale cases but lacks robustness as complexity increases. A2C is

TABLE VII Scalability test under increasing order quantities

Orders	Products	Cost	Penalty	Late Orders	Run Time
3	5	649	0	0	946.60
6	10	1747	200	1	3595.93
9	15	2686	410	2	6631.46
12	20	3311	410	2	10267.41

time-efficient but yields lower solution quality. These results confirm the effectiveness of deep RL methods for large-scale disassembly-manufacturing scheduling with order delays.

G. Scalability and Stability Analysis

To evaluate the scalability and stability of the proposed DDQN-based scheduling approach, four experimental scenarios are constructed with the number of customer orders increasing from 3 to 12, in increments of 3, while proportionally expanding the set of disassemblable products to ensure sufficient component availability.

Table VII presents the numerical results for each scenario, including the total cost, delay penalties, number of late orders, and computational time. The data show that the total cost rises from 649 to 3311 as the order quantity increases, accompanied by a significant growth in computational time due to the expanded decision space. Delay penalties are negligible in the first two scenarios but become more prominent when the order quantities exceed six, indicating greater scheduling difficulty under higher workloads.

Fig. 11 shows the changes in total cost, penalty cost, and runtime with increasing orders. The results reveal stable growth in total cost and a controlled rise in penalties, confirming that the DDQN-based method maintains robust performance and scalability for large-scale OTMR instances.

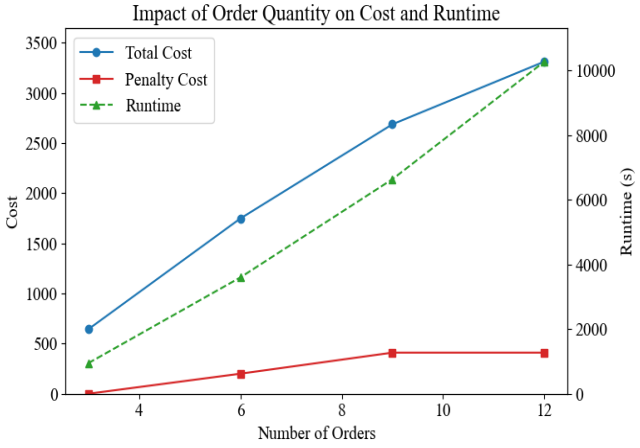


Fig. 11. Total cost and delay penalty under increasing order quantities

V. Conclusion

This study investigated the Order-Tardiness-oriented Multi-factory Remanufacturing (OTMR) problem by jointly considering order tardiness control and disassembly resource allocation in a multi-factory remanufacturing system. To address this problem, a total-cost-oriented mathematical model was developed to coordinate disassembly operations, production scheduling, and transportation decisions across multiple factories. The model was first validated by CPLEX on small-scale instances, confirming its correctness and feasibility. To handle larger-scale problems, a customized reinforcement learning environment was further constructed, and the DDQN algorithm was introduced to solve the OTMR problem under dynamic decision-making conditions.

The results demonstrate the effectiveness of the proposed approach from three aspects. First, the proposed RL environment can effectively capture the operational characteristics of the OTMR problem, including disassembly capacity limitations and order deadline constraints. Second, the problem-specific action design improves the efficiency of decision-making and enhances the applicability of RL to multi-factory remanufacturing scheduling. Third, compared with benchmark RL algorithms, DDQN achieved better solution quality and stronger stability, and it remained effective when exact optimization methods became difficult to apply to larger instances. These findings indicate that the proposed method provides an effective solution framework for multi-factory remanufacturing optimization under order tardiness constraints.

Future research will focus on the following directions: 1) Extending the current model to support multi-objective optimization[44, 45], such as balancing cost, delay, and energy consumption; 2) Incorporating dynamic and uncertain factors in production, including variable demand and machine breakdowns; 3) Further refining the RL environment to simulate real-world constraints more accurately and support broader scheduling scenarios[46, 47, 48].

References

[1] B. Tansel, "From electronic consumer products to e-wastes: Global outlook, waste quantities, recycling challenges," *Environment International*, vol. 98, pp. 35–45, 2017.

[2] J. Wang, M. Zhou, X. Guo, and L. Qi, "Multiperiod asset allocation considering dynamic loss aversion behavior of investors," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 1, pp. 73–81, 2018.

[3] P. K. Gupta, V. Shree, L. Hiremath, and S. Rajendran, "The use of modern technology in smart waste management and recycling: Artificial intelligence and machine learning," *Recent Advances in Computational Intelligence*, pp. 173–188, 2019.

[4] T. J. Sheng, M. S. Islam, N. Misran, M. H. Baharuddin, H. Arshad, M. R. Islam, M. E. H. Chowdhury, H. Rmili, and M. T. Islam, "An internet of things based smart waste management system using lora and tensorflow deep learning model," *IEEE Access*, vol. 8, pp. 148 793–148 811, 2020.

[5] C. Xu, H. Wei, X. Guo, S. Liu, L. Qi, and Z. Zhao, "Human-robot collaboration multi-objective disassembly line balancing subject to task failure via multi-objective artificial bee colony algorithm," *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 1–6, 2020.

[6] L. Qi, Q. Zeng, S. Liu, J. Wang, S. Qin, and X. Guo, "Twin delayed deep deterministic policy gradient algorithm for a heterogeneous multifactory remanufacturing optimization problem," *IEEE Transactions on Computational Social Systems*, 2025.

[7] A. M. King, S. C. Burgess, W. Ijomah, and C. A. McMahon, "Reducing waste: Repair, recondition, remanufacture or recycle?" *Sustainable Development*, vol. 14, no. 4, pp. 257–267, 2006.

[8] D. L. Diener and A.-M. Tillman, "Component end-of-life management: Exploring opportunities and related benefits of remanufacturing and functional recycling," *Resources, Conservation and Recycling*, vol. 102, pp. 80–93, 2015.

[9] B. Sarkar, M. Ullah, and M. Sarkar, "Environmental and economic sustainability through innovative green products by remanufacturing," *Journal of Cleaner Production*, vol. 332, p. 129813, 2022.

[10] M. Garetti and M. Taisch, "Sustainable manufacturing: Trends and research challenges," *Production Planning & Control*, vol. 23, no. 2-3, pp. 83–104, 2012.
[11] X. Guo, L. Chen, L. Qi, J. Wang, S. Qin, M. Chatterjee, and Q. Kang, "Multifactory disassembly process optimization considering worker posture," *IEEE Transactions on Computational Social Systems*, 2025.
[12] S. Wang, G. Yang, and S. Liu, "A data-driven multi-channel supply chain multi-factory collaborative production planning problem," *Soft Computing*, pp. 1–18, 2024.
[13] X. Wang, H. Zhang, and Q. Kang, "Human learning effect multi-period scheduling for circular disassembly line balancing based on impala reinforcement learning," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 3, pp. 144–154, September 2025.
[14] E. Özceylan, C. B. Kalayci, A. Ş. Güngör, and S. M. Gupta, "Disassembly line balancing problem: A review of the state of the art and future directions," *International Journal of Production Research*, vol. 57, no. 15-16, pp. 4805–4827, 2019.
[15] E. Tuncel, A. Zeid, and S. Kamarthi, "Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning," *Journal of Intelligent Manufacturing*, vol. 25, pp. 647–659, 2014.
[16] F. Guo, X. Guo, M. Zhou, W. Wang, S. Qin, and Q. Kang, "Hybrid disassembly line balancing for human-robot collaborative remanufacturing," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2024, pp. 2912–2917.
[17] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017. [Online]. Available: https://arxiv.org/abs/1701.07274
[18] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019. [Online]. Available: https://arxiv.org/abs/1904.12901
[19] S. Qin, F. Guo, J. Wang, L. Qi, J. Wang, X. Guo, and Z. Zhao, "Expanded discrete migratory bird optimizer for circular disassembly line balancing with tool deterioration and replacement," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 1, pp. 14–25, March 2025.
[20] F.-M. Luo, T. Xu, H. Lai, X.-H. Chen, W. Zhang, and Y. Yu, "A survey on model-based reinforcement learning," *Science China Information Sciences*, vol. 67, no. 2, p. 121101, 2024.
[21] K. Wang, Y. Li, J. Guo, L. Gao, and X. Li, "Dynamic balancing of u-shaped robotic disassembly lines using an effective deep reinforcement learning approach," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 4, pp. 6855–6865, 2024.
[22] J. Wang, G. Xi, X. Guo, S. Qin, and H. Han, "Multi-objective advantage actor-critic algorithm for hybrid disassembly line balancing with multi-skilled workers," *Information*, vol. 15, no. 3, p. 168, 2024.
[23] X. Guo, Z. Bi, J. Wang, S. Qin, S. Liu, and L. Qi, "Reinforcement learning for disassembly system optimization problems: A survey," *International Journal of Network Dynamics and Intelligence*, pp. 1–14, 2023.
[24] L. Xi, L. Yu, Y. Xu, S. Wang, and X. Chen, "A novel multi-agent ddqn-ad method-based distributed strategy for automatic generation control of integrated energy systems," *IEEE Transactions on Sustainable Energy*, vol. 11, no. 4, pp. 2417–2426, 2019.
[25] H. Zhang, M. Huang, H. Zhou, X. Wang, N. Wang, and K. Long, "Capacity maximization in ris-uav networks: A ddqn-based trajectory and phase shift optimization approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 4, pp. 2583–2591, 2022.
[26] Y. Huang, "Deep q-networks," *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pp. 135–160, 2020.
[27] Q. Zhang, Y. Xing, C. Zhang, X. Sun, B. Hu, and A. Das, "Column generation algorithms for two-dimensional cutting problem with surface defects," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 2, pp. 80–92, June 2025.
[28] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, "Mason: A multiagent simulation environment," *Simulation*, vol. 81, no. 7, pp. 517–527, 2005.
[29] V. Chvátal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.
[30] S. Yan, H. Guo, and S. Liu, "A partition stacking classification framework with oversampling for quality prediction of aluminum alloy ingots," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 1, 2025.
[31] J. C. Chen, Y.-Y. Chen, T.-L. Chen, and Y.-C. Yang, "An adaptive genetic algorithm-based and and/or graph approach for the disassembly line balancing problem," *Engineering Optimization*, vol. 54, no. 9, pp. 1583–1599, 2022.
[32] S. Qin, F. Guo, J. Wang, L. Qi, J. Wang, X. Guo, and Z. Zhao, "Expanded discrete migratory bird optimizer for circular disassembly line balancing with tool deterioration and replacement," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 1, 2025.
[33] K. Wang, X. Li, L. Gao, and P. Li, "Modeling and balancing for green disassembly line using associated parts precedence graph and multi-objective genetic simulated annealing," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 8, pp. 1597–1613, 2021.
[34] G. A. Kasapidis, D. C. Paraskevopoulos, P. P. Repoussis, and C. D. Tarantilis, "Flexible job shop scheduling problems with arbitrary precedence graphs," *Production and Operations Management*, vol. 30, no. 11, pp. 4044–4068, 2021.
[35] J. Wang, "Patient flow modeling and optimal staffing for emergency departments: A petri net approach," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 4, 2022.
[36] Y. J. Ji, Z. Y. Zhao, S. X. Liu, and X. Y. Yong, "Machine learning-based prediction of surplus material in intelligent production processes," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 1, pp. 25–34, Apr. 2025.
[37] I. Osband, C. Blundell, A. Pritzel, and B. V. Roy, "Deep exploration via bootstrapped dqn," *Advances in Neural Information Processing Systems*, vol. 29, 2016.
[38] F. Paschko, A. Krini, M. Kemke, and S. Knorn, "Reinforcement learning approach for decision making in disassembly processes," *Journal of Intelligent Manufacturing*, pp. 1–27, 2025.
[39] A. dos Santos Mignon and R. L. de A. da Rocha, "An adaptive implementation of ε-greedy in reinforcement learning," *Procedia Computer Science*, vol. 109, pp. 1146–1151, 2017.
[40] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: https://jmlr.org/papers/v22/20-1364.html
[41] M. A. Ilgin and S. M. Gupta, "Recovery of sensor embedded washing machines using a multi-kanban controlled disassembly line," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 2, pp. 318–334, 2011.
[42] E. Zussman and M. C. Zhou, "Design and implementation of an adaptive process planner for disassembly processes," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 2, pp. 171–179, 2000.
[43] R. L. Margolis and L. Wilson, "Microtubule treadmills—possible molecular machinery," *Nature*, vol. 293, no. 5835, pp. 705–711, 1981.
[44] B. Sanchez, C. Rausch, C. Haas, and R. Saari, "A selective disassembly multi-objective optimization approach for adaptive reuse of building components," *Resources, Conservation and Recycling*, vol. 154, p. 104605, 2020.
[45] T. Wei, X. Guo, M. Zhou, J. Wang, S. Liu, S. Qin, and Y. Tang, "A multiobjective discrete harmony search optimizer for disassembly line balancing problems considering human factors," *IEEE Transactions on Human-Machine Systems*, 2025.
[46] J. Beck, R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, and S. Whiteson, "A survey of meta-reinforcement learning," *arXiv preprint arXiv:2301.08028*, 2023. [Online]. Available: https://arxiv.org/abs/2301.08028
[47] R. Fakoor, P. Chaudhari, S. Soatto, and A. J. Smola, "Meta-q-learning," *arXiv preprint arXiv:1910.00125*, 2019. [Online]. Available: https://arxiv.org/abs/1910.00125
[48] S. Lee and S.-Y. Chung, "Improving generalization in meta-rl with imaginary tasks from latent dynamics mixture," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 222–27 235, 2021.

Wenjing Zeng received her Bachelor's degree in Computer Science and Technology from Changzhou University in China in 2022. She received her Master's degree from the School of Artificial Intelligence and Software, Liaoning Petrochemical University, and is currently a Ph.D. student at Donghua University. Her research interests include reinforcement learning.

Nan Wang is an Associate Professor of Computer Science at William Paterson University of New Jersey. She holds a Ph.D. in Computer Science from Mississippi State University (2009). Dr. Wang's research spans blockchain technology, generative AI, data science, and computational biology. She has published over 25 peer-reviewed journal articles and conference papers on topics including bioinformatics, gene regulatory networks, and machine learning, with recent work focusing on blockchain applications and self-evolving AI systems. She co-authored Blockchain for Beginners: A Project-Based Approach (Cogella, 2024–2025) and is preparing two additional textbooks on Python and calculus. She has participated on significant research funding as PI/Co-PI, including a $1M NSF grant (2020–2025) for the Math and Computer Science Scholars Program. Dr. Wang actively contributes to academic service, leading curriculum development for graduate programs, mentoring students in STEM initiatives, and advising the Computer Science student club. Prior to joining William Paterson in 2023, she held associate professor roles at New Jersey City University (2017–2023) and the University of Southern Mississippi (2008–2017).

Moitrayee Chatterjee received her Ph.D. and M.S. degree in Computer Science from Texas Tech University, Lubbock, TX, USA, in August 2020. Her current research work focuses on Analytical Data Sciences and Engineering. She has been interested in ensuring security in cyber space, and her research focuses on finding novel deep/machine learning based intelligent solutions through data driven learning, prediction, and decision-making processes in identifying, circumventing, and reacting to security threats.