Human Learning Effect Multi-Period Scheduling for Circular Disassembly Line Balancing Based on IMPALA Reinforcement Learning

Xu Wang, Haitao Zhang, and Qi Kang

***Abstract*—Driven by the growing demand for recycling, disassembly has emerged as a critical process, prompting the phased upgrade of traditional manual disassembly lines. To address the multi-period personnel scheduling problem within a cyclic disassembly-assembly line environment, this study innovatively introduces the Importance-Weighted Actor-Learner Architecture (IMPALA) reinforcement learning algorithm. The proposed method quantifies and incorporates worker learning effects to enable more rational dynamic task allocation, thereby maximizing production line profit and efficiency. Experimental results demonstrate that our approach exhibits a competitive advantage in both solution quality and efficiency when compared to exact solvers and other intelligent optimization algorithms (e.g., A2C, SAC, DQN, DDPG).**

***Key Words*—Disassembly line balancing problem, circular layout, learning effect, multi-skilled workers, IMPALA algorithm, multiple periods.**

I. Introduction

With the growing emphasis on resource conservation and environmental protection, producers are required to disassemble, reuse, and recycle waste products to support sustainable resource use and protect the environment[1, 2]. Product disassembly systematically separates parts, assemblies, and components from products. Only after the product is disassembled can the material be recycled and its usable parts be remade. The product disassembly line, referred to as the disassembly line, can realize automatic disassembly and assembly line operation. It is characterized by high operational efficiency and is suitable for both large products and high volumes of small products. Numerous factors affect the performance of the disassembly line, such as the type of disassembled product, the layout and speed of the disassembly line, the disassembly operation time, and the task allocation across workstations. The circular disassembly line considered in this paper offers high flexibility, making it easier to adapt to the disassembly needs from different products. By adjusting the position and configuration, it can quickly adapt to changes among the disassembly product tasks. To address these challenges, this study proposes a Circular Disassembly Line Balancing Problem considering Human Learning Effects and Worker Assignment (C-DLBP-HLWA), integrating multi-period personnel scheduling with worker learning effects to enhance disassembly efficiency.

In manufacturing enterprises, multi-period staff scheduling divides working hours into distinct time slots and assigns employees to shifts based on production demands and workforce availability, ensuring a better match between labor supply and operational needs. Existing studies have explored various task and personnel scheduling methods. Cao [3] *et al.* applied model predictive control to optimize scheduling in photovoltaic energy storage systems. Behnamian [4] proposed a colonial competition algorithm with variable neighborhood search, considering learning and deterioration in hybrid flow shop scheduling and worker assignment. Krishnamoorthy [5] *et al.* introduced the personnel task scheduling problem with its properties and benchmarks. Abualigah [6] *et al.* developed a hybrid multiverse optimization–genetic algorithm (MVO-GA) to improve scheduling efficiency, while Zhou Conghao [7] *et al.* addressed computing task scheduling in integrated air-ground networks under energy constraints. However, few studies have focused on disassembly lines, particularly the role of human learning. In practice, workers improve performance through repeated task execution, though efficiency gains tend to plateau as experience accumulates. Studies by Adler [8], Stansbury [9], and Mosheiov [10] have examined learning curves and their impact on scheduling and productivity. Learning not only shortens task times—particularly for skilled operations—but also reduces risks and operational losses.

In recent years, reinforcement learning algorithms have received extensive attention in various fields [11][12], and many studies have employed them to solve disassembly problems. Wang et al. [13] used efficient deep reinforcement learning technology to achieve dynamic balance optimization of U-shaped robot disassembly lines. Another study [14] introduced

Manuscript received September 8, 2025; revised September 16 and September 23, 2025; accepted October 7, 2025. This article was recommended for publication by Associate Editor Shujin Qin upon evaluation of the reviewers' comments.

This work was supported in part by NSFC under Grant Nos. 61573089 and 51405075, in part by Liaoning Province Education Department Scientific Research Foundation of China under Grant No. L2019027, in part by Liaoning Province Dr. Research Foundation of China under Grant No. 20170520135, in part by LiaoNing Revitalization Talents Program under Grant No. XLYC1907166, and in part by the Natural Science Foundation of Shandong Province under Grant ZR2019BF004.

X. Wang is with the school of Computer Science and Technology, Shenyang University of Chemical Technology, Shenyang 110100, China (e-mail: wangxu@hebuee.edu.cn).

H. Zhang is with the school of Computer and Communication Engineering College, Liaoning Petrochemical University, Fushun 113001, China (e-mail: ZhangHaitao4502@163.com).

Q. Kang is with the school of Electrical and Computer Engineering department, New Jersey Institute of Technology, New Jersey 07907-07399, USA (e-mail: qk22@njit.edu).

Corresponding author: Xu Wang

reinforcement learning methods to the problem of hybrid disassembly line balance, providing new ideas to optimize the disassembly process. C-DLBP-HLWA mainly involves the reasonable allocation of product disassembly tasks to workers with different skills, while considering constraints such as priority and conflict to ensure efficient execution of tasks on circular disassembly lines. In addition, Circular Disassembly Line Balancing Problem typically arises in a complex, dynamic [15], and large-scale environment, making the scalability and performance of the algorithm crucial. Reinforcement learning methods acquire optimal strategies autonomously through interaction with the environment. They are particularly well suited for dynamic and iterative decision-making scenarios, effectively handling high-dimensional state and action spaces while modeling complex nonlinear relationships between variables. The Importance-Weighted Actor–Learner Architecture (IMPALA) employs a distributed actor–learner framework that enables stable policy updates while maintaining high throughput. It introduces a V-trace policy correction mechanism to reduce policy deviation and improve gradient estimation accuracy, thereby enhancing training stability. Compared with other reinforcement learning approaches, IMPALA shows a stronger convergence. In particular, when addressing the circular disassembly line problem with human learning effects, it can more effectively maximize profits. Moreover, the algorithm can dynamically adjust product task allocation and worker scheduling, and sustain strong performance even as system scale expands.

This paper improves the circular disassembly line balance problem considering the human learning effect. First, more factors are considered affecting the circular layout disassembly line problem, and a mathematical model is developed to maximize the profit from disassembly. Second, the IMPALA algorithm is used and compared with four reinforcement learning algorithms, namely the dominant actor-critic algorithm (A2C[16]), the soft actor-critic algorithm (SAC[17]), the deep Q network (DQN[18]) and the deep deterministic policy gradient algorithm (DDPG[19]). Finally, the IMPALA algorithm outperforms other methods in terms of algorithm convergence, solution quality, and execution time.

This work aims to make the following three contributions:

1) This work incorporates the impact of workers' different learning abilities on multi-period personnel scheduling in DLBP. It further proposes a circular disassembly line layout model based on skill categories to assign workers and tasks to optimize the assignment of workers and tasks.
2) This work applies a new reinforcement learning algorithm, IMPALA, to the circular disassembly line balancing problem in shifts with human learning effects. The algorithm employs a V-trace policy correction strategy to effectively reduce strategy lag, improve gradient estimation accuracy, and significantly enhance data utilization and training stability in complex, large-scale scheduling environments.
3) Experimental results demonstrate that IMPALA outperforms other state of the art reinforcement learning algorithms(A2C, SAC, DQN, and DDPG) in solving the DLBP-HLWA problem, validating its effectiveness and practical value.

The rest of this paper is organized as follows: Section II explains the problem studied and its mathematical model. Section III introduces the IMPALA algorithm design. Section IV discusses the experimental comparison results. Section V is the conclusion of this paper.

II. Problem Description

A. Problem Statement

The DLBP solution includes the disassembly plan of the products to be disassembled and the allocation of disassembly tasks to workstations and workers who perform the tasks to obtain the maximum disassembly benefit[20–22]. Based on basic DLBP, this paper mainly considers the circular layout with worker learning ability. The expansion aspects mainly include the division of shifts according to the workers' experience values and the flexible allocation of disassembly tasks to workstations. At the same time, considering the effect of learning on workers, each task is assigned to the most suitable worker as much as possible. By achieving adaptation between tasks and workers, we aim to maximize the benefit of disassembly.

B. AND/OR Graph

AND/OR graphs represent all feasible disassembly sequences of EOL products, specifying precedence and conflict relations between operations [23],[24]. This paper uses AND/OR graphs to describe the disassembly relations between components, conflicts, and priority constraints between tasks. In an AND/OR graph, each node represents a subcomponent, indexed by an integer i in angle brackets, i.e., $<i>$, i=1, 2, $\ldots I$, where I represents the number of subcomponents. Straight arcs then connect the nodes. Disassembly operations are represented by multiple directed edges from the same starting node to its child nodes, forming an "AND" relationship. All operations are graphically labeled with an integer j, j=1, 2, $\ldots$, J, where J is the total number of disassembly operations for a given EOL product. A node can have multiple operations, forming an "OR" relationship.

Figure 1 shows a case of a compass consisting of 7 parts, and its DAOG is shown in Figure 2. The DAOG consists of boxes and directed arcs. The elements represent the parts and disassembly tasks of the product. As shown in Figure 2, there are two ways to disassemble the component. Executing disassembly task 5 can obtain sub-assemblies $<10>$ and sub-assembly $<5>$, and executing disassembly task 6 can obtain sub-assemblies $<4>$ and sub-assembly $<14>$. The component can only be disassembled using one of the two tasks. Therefore, task 5 and task 6 are two conflicting tasks. When one of the conflicting tasks is executed, the other tasks are prohibited. Since the component is disassembled by task 2, task 2 is the immediate predecessor of task 5 and task 6. Each task must be executed after its previous task in a disassembly sequence.

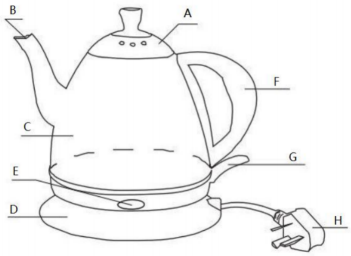


Fig. 1. Schematic diagram of a kettle.

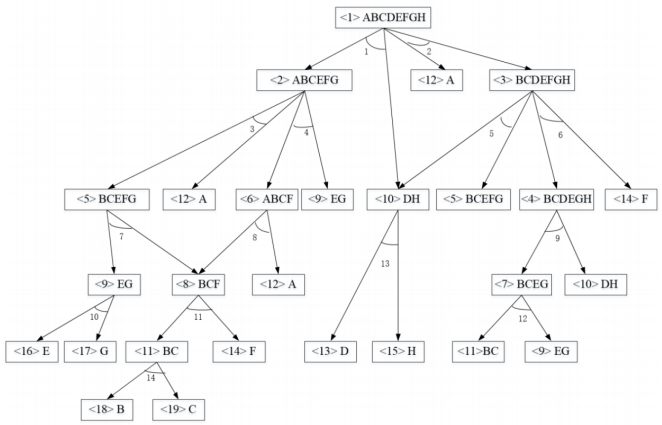


Fig. 2. The AND/OR graph of the disassembly process of a kettle.

To calculate the benefits of disassembly components, two matrices are used to describe the relationship between disassembly components and tasks and the relationship between disassembly tasks and disassembly skills.

1) Subassembly-task association matrix

A subassembly-task association matrix $D = [d_{pji}]$ is constructed to define the relationships between subassemblies and tasks:

$$d_{pji} = \begin{cases} 1, \text{if subassembly } j \text{ is obtained by task } i \text{ of product } p; \\ -1, \text{if subassembly } j \text{ is disassembled by task } i \text{ of} \\ \quad \text{product } p; \\ 0, \text{otherwise} \end{cases}$$

2) Task-skill association matrix

A task-skill association matrix $B = [b_{pis}]$ is used to describe the relationship between tasks and skills.

$$b_{pis} = \begin{cases} 1, \text{ if disassembly task } i \text{ of product } p \text{ involves skill } s \\ 0, \text{ if disassembly task } i \text{ of product } p \text{ does not} \\ \quad \text{involve skill } s; \end{cases}$$

In this work, we assume that:

1) Matrices D and B are known.

2) The conflicting and priority relationship among disassembly tasks is known.

3) A worker is assigned to one workstation, and each worker has an initial experience.

4) Each disassembly task requires a disassembly skill. The higher the level of disassembly skill, the less time it requires to accomplish the disassembly task.

5) At least one disassembly task is assigned to each active workstation.

6) There is no conflict and priority among the disassembly tasks of different products.

C. Differentiate between shifts

Taking into account the benefits of the enterprise and the personal life and work needs of employees, we reasonably allocate and schedule employees according to the production needs of different shifts. This paper plans to divide a day into three periods, namely 00:00-08:00, 08:00-16:00 and 16:00-24:00. The rotation rule is set to rotate every four days, with a cycle of 28 days. After a cycle of experience accumulation, when the different skill levels of different workers have changed, they will be regrouped according to their experience values and rotate shifts again, and so on. We provide flexible working time options to improve employee satisfaction and quality of life[25]. By arranging more workers during critical periods, we ensure enough people to cope with the peak production period, improving overall production efficiency. This flexible scheduling method can not only improve production efficiency[26], increase the diversity of employee work content within the allowable range of mobilization, and reduce the harm caused by long-term fixed work but also optimize employees' work experience and effectively reduce the company's operating costs. Reasonable scheduling is significant to improving work efficiency and reducing company operating costs.

D. Circular layout and learning effect

Traditional disassembly layouts include linear disassembly [27], U-shaped disassembly [28], parallel disassembly [29], and bilateral disassembly. However, traditional disassembly layouts are subject to time and space constraints at each workstation, which affects task allocation. This paper considers the impact of workers' learning ability on task allocation and adopts a circular layout disassembly system. This disassembly layout is more flexible and can transfer undisassembled components to suitable workers in a cycle on the disassembly line, effectively solving the time and space constraints on the workstation, thereby achieving more tasks on one workstation. Fig. 3 shows a circular layout disassembly line with three workstations, and the allocation of tasks on the workstations is illustrated using the kettle example from Fig. 1. The figure transfers components from the entrance to the disassembly line, and components can be transferred on the conveyor belt. When a subassembly is transferred to an appropriate worker position, a worker can remove it from the workstation for disassembly. In contrast, the remaining subassemblies to be disassembled can continue to be transferred to the disassembly line. Finally, the disassembled components are sent out from the outlet. It can be seen that tasks 1 and 13 are assigned to workstation one and executed by worker 1; tasks 3 and 15 are assigned to workstation three and executed by worker 3;

task 8 is assigned to workstation two and executed by worker 2. $1^{'}$ means that the component is disassembled during the first round of transportation, and $13^{''}$ means that component 13 is disassembled when it is transported to workstation 1 in the second round. Therefore, the task sequence and execution order are 1, 3, 8, 13, and 15.

The learning effect means that in an actual disassembly process, workers can continuously improve their work efficiency through learning, thereby gradually shortening the time required to complete a disassembly task. To quantify, we divide the skill level of workers according to the experience range interval. Note that the improvement of productivity in the learning curve is more significant in the initial stage and will be decreasing along with experiences accumulating.

As shown in Table I, experience is divided into different intervals to represent the corresponding skill level. From this table, the skill level of workers can be obtained based on their experience. By comprehensively considering the worker's skill level, learning ability, and task nature, the worker's work can be well adjusted, thereby improving the efficiency and quality of the entire disassembly line.

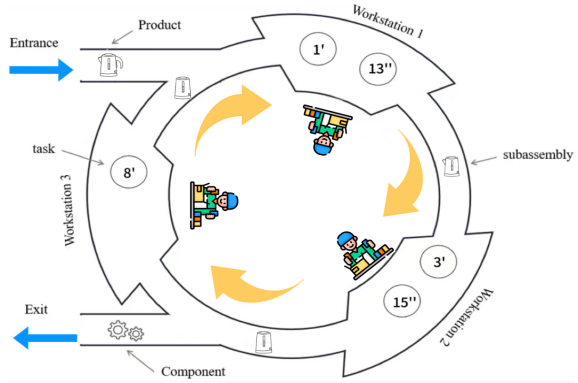


Fig. 3. Schematic diagram of compass disassembly task allocation.

TABLE I Division of skill levels based on experience

Skill level	Level 1	Level 2	Level 3	Level 4	Level 5
Exp. interval	[0,50)	[50,100)	[100,150)	[150,200)	[200,+∞)

E. Mathematical Model

Notations used in the model to be presented in Section II are summarized as follows:

Sets:

$\mathbb{W}$ Index set of workstations, $\mathbb{W} = \{1, 2, ..., W\}$.
$\mathbb{S}$ Index set of skills, $\mathbb{S} = \{1, 2, ..., S\}$.
$\mathbb{R}$ Index set of workers, $\mathbb{R} = \{1, 2, ..., R\}$.
$\mathbb{P}$ Index set of products, $\mathbb{P} = \{1, 2, ..., P\}$.
$\mathbb{D}$ Index set of shifts, $\mathbb{D} = \{1, 2, ..., D\}$.
$\mathbb{K}$ Index set of positions on each workstation, $\mathbb{K} = \{1, 2, ..., K\}$.
$\mathbb{L}$ Index set of skill level index, $\mathbb{L} = \{1, 2, ...L\}$.
$\mathbb{J}_p$ Subassembly set of product p.
$\mathbb{I}_p$ Task set of product p.
$\mathbb{I}_{pi}^{\text{con}}$ Task set that conflicts with task i of product p.
$\mathbb{I}_{pi}^{\text{pre}}$ Immediately preceding task set of task i of product p.

Parameters:

T_{pirl} The time needed for performing task i on product p by the r-th worker with skill level .
E_{rs}^{0} The initial experience of *r-th* worker's skill s.
E_{sl}^{lb} The lower bound of experience required for skill s at level l.
C_w Startup cost of workstation w.
V_{pj} Value of subassembly j of product p.
α_s Learning effect coefficient of skill s. The experiential benefit of workers for skill s is equal to the execution time multiplied by the coefficient.
C_{iw} The cost incurred per unit of time for disassembly activities (covering employee salaries, electricity) expenses, equipment depreciation, etc.
M A sufficiently large number.
T_t The total duration of shift t

Decision variables:

x_{piwkt} In shift t, if task i of product p is executed at the k-th position on workstation w, x_{piwkt} = 1; otherwise x_{piwkt} = 0.
z_{rwt} In shift t, if the r-th worker is assigned to the w-th workstation, z_{rwt}=1; otherwise z_{rwt} = 0.
u_{wt} In shift t, if the w-th workstation is started,u_{wt} = 1; otherwise $u_w t$= 0.
y_{piwkt}^{rsl} In shift t, if task i of product p is executed by r-th worker at the k-th position on workstation w and the level of skill $s$is l, then y_{piwkt}^{rsl} = 1; otherwise y_{piwkt}^{rsl} = 0.
e_{rst} In shift t, experience corresponds to skill s of worker r
s_{piwkt} In shift t, start time of task i of product p at position k on workstation w.

CDP-L is formulated as follows:

$$\max\left(\sum_{p\in\mathbb{P}}\sum_{j\in\mathbb{I}_p}\sum_{i\in\mathbb{I}_p}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}}\sum_{t\in\mathbb{D}}V_{pj}d_{pji}x_{piwkt}-\sum_{w\in\mathbb{W}}\sum_{t\in\mathbb{D}}C_w u_{wt}\right.$$
$$\sum_{p\in\mathbb{P}}\sum_{r\in\mathbb{R}}\sum_{k\in\mathbb{K}}\sum_{s\in\mathbb{S}}\sum_{w\in\mathbb{W}}\sum_{i\in\mathbb{I}_p}\sum_{l\in\mathbb{L}}\sum_{t\in\mathbb{D}}C_{iw}y^{rsl}_{piwkt}T_{pirl}-$$
$$\left.\sum_{r\in\mathbb{R}}\sum_{t\in\mathbb{D}}c_{rt}T_t z_{rwt}\right) \tag{1}$$

$$s.t.\quad \sum_{r\in\mathbb{R}} z_{rwt}=u_{wt},\forall w\in\mathbb{W},\forall t\in\mathbb{D} \tag{2}$$

$$\sum_{w\in\mathbb{W}} z_{rwt}\le 1,\forall r\in\mathbb{R},\forall t\in\mathbb{D} \tag{3}$$

$$z_{rwt}\ge y^{rsl}_{piwkt}T_{pirl},\forall r\in\mathbb{R},\forall s\in\mathbb{S},\forall l\in\mathbb{L},\forall p\in\mathbb{P},$$
$$\forall i\in\mathbb{I}_p,\forall w\in\mathbb{W},\forall k\in\mathbb{K},\forall t\in\mathbb{D} \tag{4}$$

$$\sum_{p\in\mathbb{P}}\sum_{i\in\mathbb{I}_p}x_{piwkt}\le Mu_{wt},\forall w\in\mathbb{W},\forall k\in\mathbb{K},\forall t\in\mathbb{D} \tag{5}$$

$$\sum_{t\in\mathbb{D}}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}}x_{piwkt}\le 1,\forall i\in\mathbb{I}_p,\forall p\in\mathbb{P} \tag{6}$$

$$\sum_{p\in\mathbb{P}}\sum_{i\in\mathbb{I}_p}x_{piwkt}\ge\sum_{p\in\mathbb{P}}\sum_{i\in\mathbb{I}_p}x_{piwk+1t},$$
$$\forall w\in\mathbb{W},\forall k\in\mathbb{K}\backslash\{K\},\forall t\in\mathbb{D} \tag{7}$$

$$\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}}\sum_{t\in\mathbb{D}}x_{piwkt}+\sum_{i'\in\mathbb{I}^{\mathrm{con}}_{pi}}\sum_{t\in\mathbb{D}}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}}x_{pi'wkt}\le 1,$$
$$\forall i\in\mathbb{I}_p,p\in\mathbb{P} \tag{8}$$

$$x_{piwkt}\le\sum_{i'\in\mathbb{I}^{\mathrm{pre}}_{pi}}\sum_{w'\in\mathbb{W}}\sum_{k'\in\mathbb{K}}\sum_{t'\in\mathbb{D}}x_{pi'w'k't'}$$
$$\forall \mathrm{i}\in\mathbb{I}_p,\mathrm{w}\in\mathbb{W},\forall k\in\mathbb{K},\forall p\in\mathbb{P},\forall t\in\mathbb{D} \tag{9}$$

$$e_{rs1}=E^0_{rs},\forall r\in\mathbb{R},\in\mathbb{S} \tag{10}$$

$$e_{rst}=e_{rst-1}+\sum_{p\in\mathbb{P}}\sum_{i\in\mathbb{I}_p}\sum_{k\in\mathbb{K}}\sum_{l\in\mathbb{L}}\sum_{w\in\mathbb{W}}\alpha_s T_{pirl}y^{rsl}_{piwkt}$$
$$\forall r\in\mathbb{R},s\in\mathbb{S},\forall t\in\mathbb{D}\backslash\{1\} \tag{11}$$

$$\sum_{k\in\mathbb{K}}\sum_{s\in\mathbb{S}}\sum_{i\in\mathbb{I}_p}\sum_{r\in\mathbb{R}}\sum_{l\in\mathbb{L}}\sum_{p\in\mathbb{P}}y^{rsl}_{piwkt}T_{pirl}\le T_t$$
$$\forall t\in\mathbb{D},\forall w\in\mathbb{W} \tag{12}$$

$$\sum_{l\in\mathbb{L}}\sum_{r\in\mathbb{R}}y^{rsl}_{piwkt}=x_{piwkt}b_{pis}$$
$$\forall p\in\mathbb{P},w\in\mathbb{W},s\in\mathbb{S},k\in\mathbb{K},i\in\mathbb{I}_p,t\in\mathbb{D} \tag{13}$$

$$E^{\mathrm{lb}}_{sl}-M\left(1-y^{rsl}_{piwkt}\right)\le e_{rst}$$
$$\le E^{\mathrm{lb}}_{sl+1}+M\left(1-y^{rsl}_{piwkt}\right)$$
$$\forall p\in\mathbb{P},w\in\mathbb{W},r\in\mathbb{R},s\in\mathbb{S},k\in\mathbb{K},l\in\mathbb{L},i\in\mathbb{I}_p,t\in\mathbb{D} \tag{14}$$

$$s_{p'i'wkt}+\sum_{s\in\mathbb{S}}\sum_{l\in\mathbb{L}}\sum_{r\in\mathbb{R}}y^{rsl}_{pi'wkt}T_{pi'rl}$$
$$\le s_{piwk+1t}+M\left(2-x_{piwk+1t}-x_{p'i'wkt}\right)$$
$$\forall w\in\mathbb{W},i\in\mathbb{I}_p,i'\in\mathbb{I}^{pre}_{pi},k\in\mathbb{K},k'\in\mathbb{K},p\in\mathbb{P},t\in\mathbb{D} \tag{15}$$

$$s_{pi'w'k't}+\sum_{s\in\mathbb{S}}\sum_{l\in\mathbb{L}}\sum_{r\in\mathbb{R}}y^{rsl}_{pi'w'k't}T_{pi'rl}$$
$$\le s_{piwkt}+M\left(2-x_{piwkt}-x_{p'i'w'k't}\right)$$
$$\forall w\in\mathbb{W},i\in\mathbb{I}_p,k\in\mathbb{K},t\in\mathbb{D},p\in\mathbb{P} \tag{16}$$

The objective function (1) is to maximize profit, equal to the total value of the disassembled parts minus the cost of starting the workstation and the level-determined disassembly and employee costs. Constraint (2) indicates that only one worker can assign each idle workstation. Constraint (3) indicates that workers can only be assigned one workstation in the same shift. Constraint (4) indicates that if worker r performs a specific task i at workstation w and time period t, then worker r must be assigned to workstation w and time period t to ensure the consistency of task execution and worker allocation, and avoid the contradictory situation that a worker is not assigned to a workstation in a certain time period but performs a task at the workstation. Constraint (5) indicates that tasks can only be assigned to enabled workstations. Constraint (6) indicates that each task can only be executed once in the same shift. Constraint (7) indicates that the positions of executing tasks are left-aligned; when a task is in the latter position, there must be a task in the previous position. Constraint (8) indicates that, at most, one conflicting task can be executed; task conflict is avoided. Constraint (9) indicates that the priority relationship of tasks is satisfied; at least one of the predecessor tasks is executed. Constraint (10) indicates the initial experience of the first shift. Constraint (11) states that during the same shift, cumulative skill experience is gained when performing disassembly tasks. The experience from this shift is calculated and added to the worker's experience before the start of the next shift. Constraint (12) indicates that the total duration of the disassembly task cannot exceed the duration of the shift. Constraint (13) indicates that the skill level of the workers in the same shift remains unchanged. Constraint (14) represents the experience range constraint of the skill level. Constraint (15) limits the execution order of tasks on workstations. Constraint (16) indicates that the start time of a task must be immediately after the start time of the previous task.

III. Proposed Algorithm

A. Algorithm Description

The traditional A3C method requires multiple working nodes to update gradients synchronously and use the actor-critic method to update the globally shared strategy and

value function. IMPALA [30] uses a decoupled Actor-Learner structure, which allows data collection and model learning to be optimized separately, thereby achieving efficient parallel computing. Its core innovation lies in the use of a decoupled Actor-Learner architecture, where multiple Actors (executing agents [31]) run the environment in parallel, generate trajectories and send them to a centralized Learner (learning agent), which performs efficient batch gradient updates on the GPU. The V-trace off-policy correction algorithm is used. Since the data generated by the Actor may lag behind the current Learner's strategy, IMPALA uses the V-trace algorithm to correct the gradient estimate to reduce the deviation caused by off-policy learning. Fig.4 shows the framework for solving C-DLBP-HLWA based on IMPALA. The IMPALA algorithm deploys multiple Actor processes, and each Actor interacts with the C-DLBP-HLWA environment independently. Each Actor receives the current state of the disassembly line, including features such as task assignment, workstation load, product information, and worker assignment. The Actor uses the policy network to generate action A, such as assigning task 1 to workstation 1, and sampling discrete actions according to the current policy. Each Actor generates an experience trajectory $(S_0, A_0, R_0, S_1, \ldots, S_t)$, and all Actors store the generated trajectory data (including state, action, reward, and policy probability) in a shared experience queue, and then perform importance sampling to record the probability distribution of the policy when generating the action, which is used to correct the off-policy deviation later. The Learner randomly extracts a batch of trajectory data from the experience queue. The V-trace algorithm is used to calculate the corrected value target, and the off-policy data distribution difference is adjusted by the importance weight. The value network parameters are then updated to minimize the mean square error between the predicted value and the V-target. At the same time, the learner calculates the policy gradient, combines the advantage function

$$(A_t = R_t + \gamma V(S_{t+1}) - V(S_t))$$

and the importance weight (ρ_t), and updates the policy network parameters. Network synchronization periodically synchronizes the updated policy network parameters to all actors.

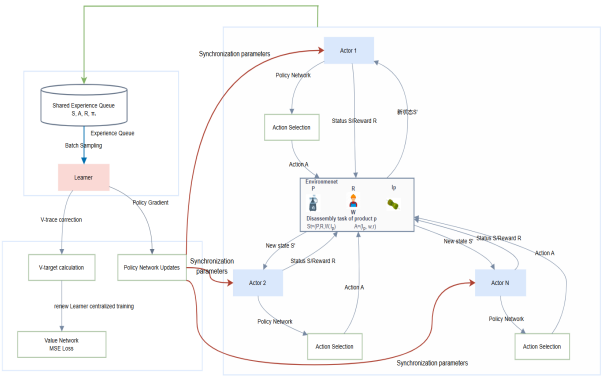


Fig. 4. Framework for DLBP-HLWA based on IMPALA.

B. The working principle of V-trace

The core idea of the V-trace algorithm is to process the data generated by multiple parallel Actor sampling through traditional Monte Carlo estimation and temporal difference learning, and eliminate the deviation between different strategies. The specific calculation steps are as follows: 1. Calculate the behavior strategy ratio:

$$\rho_t = \frac{\pi\left(a_t \mid s_t\right)}{\mu\left(a_t \mid s_t\right)}$$

Among them, π is the current strategy, μ is the old strategy (in IMPALA, it refers to the strategy of each Actor).

2. Calculate the corrected TD target: The core of V-trace is the corrected temporal difference (TD) target. For each time t, we calculate a corrected target value $V_t^{\text{V-trace}}$, which combines the behaviors sampled from the current strategy and the old strategy. The specific formula is as follows:

$$V_t^{\text{V-trace}} = r_t + \gamma\left(1-\rho_t\right) V_{t+1}^{\text{V-trace}}$$

Among them, r_t is the reward at the current moment, γ is the discount factor, and ρ_t is the ratio of the current strategy to the old strategy. The key to the correction step: By correcting the temporal difference target by ρ_t, the deviation introduced by the policy update is reduced. This allows even the experience generated under the old policy to be effectively used for learning under the current policy.

3. Update the advantage function: With the correction value obtained in the previous step, we can use it to update the advantage function of the strategy:

$$A_t = \hat{A}_t + \rho_t\left(V_t^{\text{V-trace}} - V(s_t)\right)$$

Among them, A_t is the traditional advantage function estimation, and $V(s_t)$ is the value function of the current state.

In this way, V-trace corrects the samples generated from different strategies so that all experiences can be used to update the current strategy, thereby reducing the deviation between strategy updates. Since the Actor makes decisions at an earlier time step, its strategy may be outdated during learning, so IMPALA performs off-strategy correction through the V-trace method. Unlike traditional importance sampling, V-trace uses truncated importance weights, which not only controls the estimation variance, but also effectively adjusts the error caused by strategy lag. In the scenario of experience replay, V-trace can still ensure a high data utilization rate, making IMPALA more stable and efficient in distributed reinforcement learning.

C. Aciton and State Space

In the IMPALA algorithm, the state space

$$S_t = [p, r, w, I_p]$$

includes disassembled product p, disassembly worker r, workstation w and disassembly task I_p, where $St_{[}0] = p, \quad St_{[}1] = r, \quad St_{[}2] = w, \quad St_{[}3] = I_p$. Whenever a valid disassembly step is taken, the state $S't$ is updated. The initial state of St is [0,0,0,0], indicating that disassembly has not yet started. When

disassembling a product, it starts from p = 1 and proceeds in sequence. When a product is disassembled, the state switches and is updated. Actions in the IMPALA algorithm

$$A = [I_P, w, r]$$

is used to determine the scheduling of products for workers, select product disassembly tasks, and assign them to workstation w. Where $A[0] = I_p, A[1] = w, A[2] = r$. When disassembling product p, use set Ic to store the disassembly tasks that need to be disassembled. Check state $St[3]$. If $St[3]$ is 0, it means that the product has no disassembly task. Then let $A[0]$ select a product with a disassembly task from set Ic and update $St[3]$. If $St[3]$ is not 0, it means that the product has task I that needs to be disassembled, so $St[3]$ is not updated. Similarly, in order to assign the disassembly product task to the appropriate workstation, based on the current state of the disassembly task $St[3]$, determine which workstation the disassembly task of product p is assigned to and store it in set Wc. Then, $A1$ selects which workstation the disassembly task is assigned to from Wc and updates $St[2]$. Finally, in order to assign the appropriate worker to the disassembly task, action $A2$ determines the disassembly worker based on state $St[1]$. The pseudo code is given in Algorithm 1. The action and state definitions are shown in Figure 5. The reward function Re guides the training process of the IMPALA algorithm. The disassembly state is determined by the state $S[t]$, the actor selects actions and executes them according to the current strategy, and the learner updates the strategy by training the model and ultimately decides on the change of strategy.

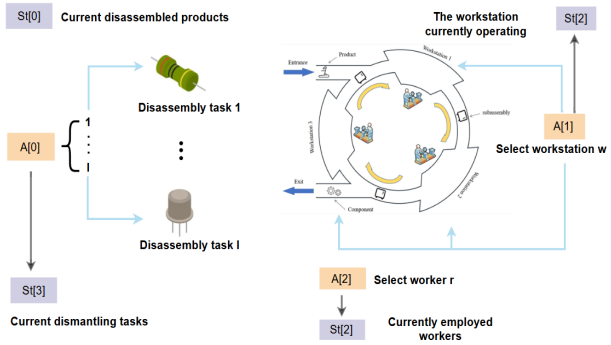


Fig. 5. Definition of actions and states.

Algorithm 1 The Selection on Actions

Input: Action set $A = [I_p, w, r]$, State $St = [p, r, w, I_p]$
Output: Updated state $St = [p, r, w, I_p]$
Initialize state $St = [0, 0, 0, 0]$
Initialize $I_c = \{\}$
Initialize $W_c = \{\}$
if $St[0] == 0$ **then**
  $p \leftarrow 1$
else
  $p \leftarrow St[0]$
end if
if $St[3] == 0$ **then**
  $I_p \leftarrow A[0] \mod |I_c| + 1$
else
  $I_p \leftarrow St[3]$
end if
$W_c = \{w \mid w\text{executable tasks}I_p\}$
if $W_c \neq \emptyset$ **then**
  $w \leftarrow A[1] \mod |W_c| + 1$
else
  $w \leftarrow St[2]$
end if
$r \leftarrow A[2] \mod$ (total number of available workers) $+ 1$
Update Status $St \leftarrow [p, r, w, I_p]$
if all tasks for the current product p are completed **then**
  $p \leftarrow p + 1$
  $St[3] \leftarrow 0$
end if
return St

D. Reward Definition

The reward Re represents the total profit generated by all shifts. After the disassembly operation is completed, the disassembly cost $dismantling\_cost$ is calculated. The $disassembly\_cost$ is the product of the $workstation_cost$ generated per unit time of the disassembly activity and the task execution time. From this, the profit $income$ obtained from the component can be determined. The $workstation\_cost$ $workstation\_cost$ is the product of the number of open workstations in all shifts and the cost of opening the workstation. The worker cost $worker\_cost$ is the product of the worker cost and working time of each shift. The formula for calculating the reward is as follows:

$$Re = income - dismantling\_cost - workstation\_cost - worker\_cost$$

where

$$income = \sum_{p\in\mathbb{P}}\sum_{j\in\mathbb{J}_p}\sum_{i\in\mathbb{I}_p}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}}\sum_{t\in\mathbb{D}} V_{pj} d_{pji} x_{piwkt}$$

$$dismantling\_cost = \sum_{p\in\mathbb{P}}\sum_{r\in\mathbb{R}}\sum_{k\in\mathbb{K}}\sum_{s\in\mathbb{S}}\sum_{w\in\mathbb{W}}\sum_{i\in\mathbb{I}_p}\sum_{l\in\mathbb{L}}\sum_{t\in\mathbb{D}} C_{iw} y^{rsl}_{piwkt} T_{pirl}.$$

$$workstation\_cost = \sum_{w\in\mathbb{W}}\sum_{t\in\mathbb{D}} C_w u_{wt}$$

$$worker\_cost = \sum_{r\in\mathbb{R}}\sum_{t\in\mathbb{D}} c_{rt} T_t z_{rwt}$$

E. Environment Description

In the IMPALA algorithm, each episode consists of multiple steps and generates an optimization plan for C-DLBP-HLWA. Multiple Actors observe the state of the environment and select a feasible task allocation plan based on the policy network, that is, select suitable workers and workstations for product tasks. After the Actor selects the action A_t, the environment will feedback the new state S_{t+1} and the corresponding reward

R_e, and then store the experience in the shared experience queue. Learner samples data from the experience queue, uses V-trace to correct policy deviations, calculates the value target (V-target), and optimizes the policy network based on policy gradients. At the same time, the updated policy parameters are synchronized to each Actor to ensure that the model can continue to improve. After training is completed, the maximum reward value of the average obtained during the training process is reported.

In the environment design, as shown in Fig. 6, a circular disassembly line with three workstations. We consider a washing machine product that is disassembled in the order of tasks 1, 4, 8, and 12. Initially, the Actor provides the environment with an action $A = [1, 1, 1]$, indicating that the product is disassembled by worker 1 in workstation 1 for task 1. After completing this disassembly step, the state is updated to $St = [1, 1, 1, 1]$, indicating that product 1 is being disassembled by worker 1 in workstation 1 for task 1. The Actor then receives the updated state and reward and proceeds to the next action. In the figure, steps represent multiple actions required to disassemble product 1.

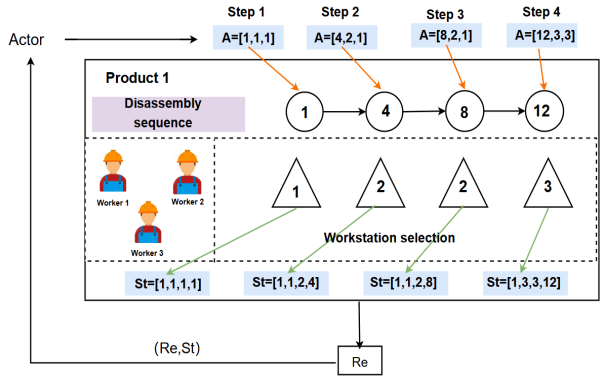


Fig. 6. Algorithm environment design for DLBP-HLWA.

IV. Experimental Studies

All code implementations were carried out using PyCharm Community Edition 2023.2.3 and executed with Python 3.10. All experiments were conducted on a workstation equipped with a 13th Gen Intel(R) Core(TM) i5-13500H processor (2.60 GHz) and 16 GB of RAM, ensuring a consistent and reliable computational environment. This section aims to verify both the correctness of the proposed model and the effectiveness of the IMPALA algorithm for training. First, the model's validity is confirmed by solving it using the CPLEX 12.8.0 solver. Subsequently, an environment tailored to the IMPALA algorithm is developed, and its training results are manually verified to ensure algorithmic accuracy. This manual verification process is essential for identifying potential coding errors or algorithmic issues that could impact result reliability. Finally, the performance of the IMPALA algorithm is compared against that of CPLEX, A2C, SAC, DQN, and DDPG to evaluate its effectiveness under consistent conditions.

A. Experimental Cases and Parameter Settings

This study focuses on multi-product DLBP. Five products of different scales, including washing machines, flashlights, electric kettles, computers, and compasses, were selected and assembled into various multi-product cases [32] for testing. Table II shows the detailed information of these products, and Table III shows the scale information of the combined cases.

TABLE II Product set

Product	Num. of tasks	Num. of subassemblies	Num. of skills
Washing machine	10	15	3
flashlight	13	15	3
Kettle	14	19	3
Computer	13	13	3
Compass	15	18	3

TABLE III Case information.

Case #	Product					Num. of tasks	Num. of skills
	Washing machine	Flash light	Kettle	Computer	Compass		
1	1	1	0	0	0	23	3
2	0	1	1	0	1	39	3
3	1	1	0	1	1	51	3
4	1	2	1	1	1	80	3
5	2	2	2	2	2	130	3

B. Analysis of Experimental Results

We use CPLEX to test the experimental cases, setting a maximum runtime of four hours to search for the optimal solution. The experimental results are shown in Table IV. The disassembly sequence represents the optimal solution that can be obtained. Taking Case 1 as an example, in the first shift, the disassembly sequence indicates that tasks 10 and 20 are assigned to workstation 1 and executed by worker 1, tasks 11, 19 and task 3 are assigned to workstation 2 and executed by worker 6, and task 1 is assigned to workstation 3 and executed by worker 12. The benefit column represents the target value corresponding to this disassembly sequence [33].

The value of the best upper bound does not necessarily correspond to a feasible solution. There are two solution states in one column. Best means that the current solution is the optimal solution of the model, and feasible means that the current solution is a feasible solution but not the optimal solution. For example, in Case 2, the optimal solution for the return is 1931, but the feasible solution is 1909. The gap indicates the gap between the current solution and the best upper bound, which can be used to judge the quality of the current solution. The gap value in Case 2 is 1.17 percent. The calculation time column indicates the time required to obtain the current sequence. For example, the calculation time in Case 2 is 6575.55 seconds. The results indicate that due to the complexity of the model and the high dimensionality of the decision variables, CPLEX performs inefficiently to solve this problem. In Case 1, CPLEX can find the optimal solution. However, as the scale of cases increases, the solution time

TABLE IV CPLEX solutions

Case #	Disassembly sequence	Profit (Best Bound)	Solution status	Gap	Computing time(s)
1	$1:(10,20)\rightarrow 1,(11,19,3)\rightarrow 6,(1)\rightarrow 12$	1971 **(1971)**	optimal	0.00%	88.98
2	$1:(1,11)\rightarrow 9,(25)\rightarrow 7,3:(9,10,39)\rightarrow 3$	1909 **(1931)**	feasible	1.17%	6575.55
3	$1:(11,37)\rightarrow 4,(1,9,24)\rightarrow 6,(13,20)\rightarrow 7,$ $3:(10,25,51)\rightarrow 3$	3131 **(3482)**	feasible	11.23%	14400.0
4	$1:(1,11,21,34,47,60,74)\rightarrow 10,$ $2:(9)\rightarrow 6,(88,23,30,49,36,43)\rightarrow 1,$ $3:(19,10)\rightarrow 6$	5146 **(5952)**	feasible	15.66%	14400.0
5	—[a]	—	—	—	—

[a] For Cases 4-6, CPLEX can't find a feasible solution within the set 4 hours.

TABLE V IMPALA Solutions

Case #	Iterations / Population	Disassembly sequence	Profit	Computing time(s)
1	100	$1:(1:4,8,12,10)\rightarrow 2,(15,20,17,19)\rightarrow 4$	**1971**	26.54
2	100	$1:(1,10,11,21,24)\rightarrow 10,(37,39)\rightarrow 12,2:(7,14,18,20)\rightarrow 5$ $3:(3,9,6,23)\rightarrow 7,(26,31,34,36)\rightarrow 9$	**1927**	26.46
3	100	$1:(2,41)\rightarrow 5,(7,5,8,9,10)\rightarrow 12,2:(11,24)\rightarrow 6,(14,19,23,20)\rightarrow 12$ $3:(27,33,35)\rightarrow 2,(47,46,51,44,49)\rightarrow 4$	**3461**	27.75
3	100	$1:(2,8,9,10,11,15,18,19,20)\rightarrow 7,(21,29,33,30)\rightarrow 2$ $2:(7,5,17,34,37,42,46,43)\rightarrow 8,(24,50,47,56,58)\rightarrow 3$ $3:(78,84,83,88,81,86)\rightarrow 9,(61,65,68,72,71,73,69)\rightarrow 1$	**5937**	23.65
5	100	$1:(1,11,73,87,107,122)\rightarrow 1,(21,23,34,36,47,49,60,62)\rightarrow 2$ $2:(7,10,17,20,110,112,115,125,127,130)\rightarrow 6,(27,32,40,45,59,72)\rightarrow 8$ $3:(3,9,6,13,19,16,83,100)\rightarrow 10,(102,113,117,128)\rightarrow 11,(30,43,53,66)\rightarrow 12$	**9806**	25.64

also increases. For cases 3 and 4, CPLEX fail to find the optimal solution within the time limited and only a feasible solution found. In cases 3 through 5, the increasing problem scale prevents CPLEX from generating even a feasible solution within the specified time.

Similarly, we use IMPALA to test the above cases. Because IMPALA maintains an approximately constant single-step update time in a distributed environment, the algorithm shows stable time efficiency across tasks of different sizes, resulting in similar times to reach the optimal solution. The experimental results are shown in Table V.

The complexity of the algorithm is the primary factor that influences its running time. Comparison of Table IV and Table V shows that, for the first four cases where CPLEX finds the optimal solution, the algorithm achieves the same solution with a much shorter running time. For case 2 where CPLEX gives a feasible solution, the proposed algorithm can find a higher quality solution in a shorter time. In case 5, where CPLEX cannot find a feasible solution within the time limit, the algorithm is able to obtain a high-quality solution in a very short time. During the solution process, IMPALA is more reasonable in task allocation between shifts, assigning tasks to workers who are most suitable for these disassembly tasks.

In order to compare the advantages of the IMPALA algorithm over other reinforcement learning algorithms, we selected four algorithms, A2C, SAC, DQN and DDPG, for experimental testing. These four reinforcement learning algorithms are representative methods covering value-based (DQN), policy-based (A2C), and actor-critic (DDPG, SAC) methods. They are widely used in both continuous and discrete action spaces, making them ideal benchmarks for comparing IMPALA in terms of stability, convergence speed, and scalability in complex scheduling problems [34]. In order to prove the advantages of the IMPALA algorithm, we conducted 20 independent experiments for each case of each algorithm and calculated the average value of each generation of 20 experiments. The iteration curve of each algorithm is plotted according to the average value. The iterator curve of case 4-5 is shown in Fig 7. Combining the iteration curves of different cases, it can be concluded that the convergence speed and solution quality of IMPALA are higher than those of other intelligent algorithms. In cases 1-5, the profit of IMPALA is always the highest value. For example, in case 5, the profit of IMPALA is 9806, while the profit of DDPG is 9586, and the calculation time is stable at about 25 seconds, which is much lower than DDPG's 150 seconds. In contrast, although A2C and SAC have shorter computation time (10-16 seconds), their profits are significantly lower than IMPALA. DQN performs close to IMPALA in some cases (such as cases 2 and 4), but its solution quality fluctuates greatly (profit 3205 in case 3 vs. 3461 of IMPALA). The results show that IMPALA achieves a better balance in the exploration-exploitation trade-off through distributed architecture and policy optimization, and is suitable for real-time optimization requirements of complex disassembly sequences.

V. Conclusion

This study established a mathematical model for C-DLBP-HLWA[35] that integrates a cyclic layout, multi-period scheduling, and worker learning effects, verifying its feasibility using CPLEX. The IMPALA algorithm was successfully

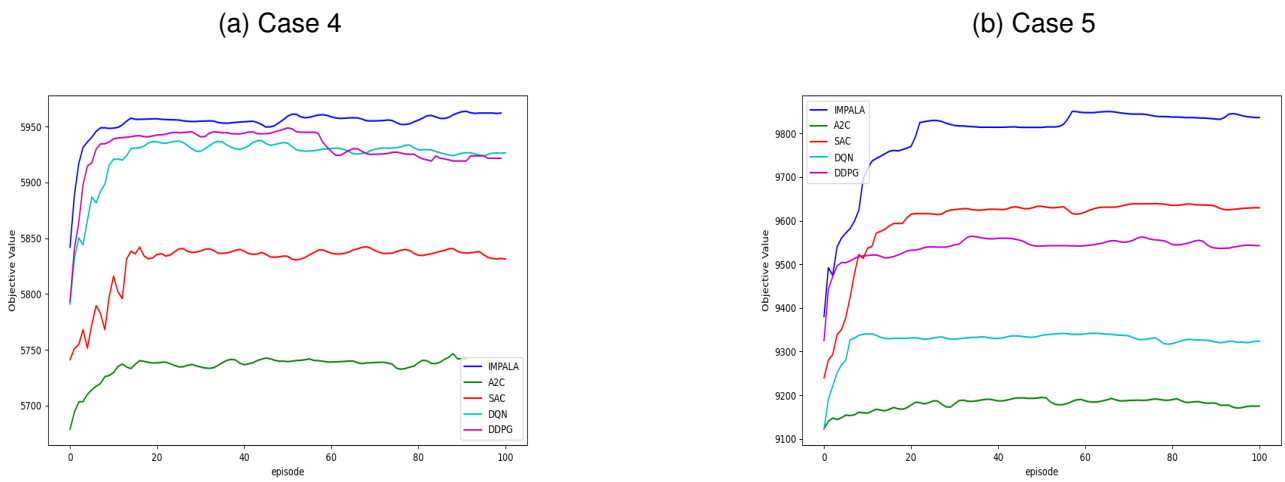


Fig. 7. Iterative comparison.

TABLE VI Algorithm performance comparison

Case #	Optimal profit					Computing time(s)				
	IMPALA	A2C	SAC	DQN	DDPG	IMPALA	A2C	SAC	DQN	DDPG
1	**1971**	1971	1576	1921	1934	**26.54**	9.48	8.31	6.12	132.95
2	**1921**	1919	1920	1921	1914	**26.46**	10.10	88.69	7.72	140.52
3	**3461**	3307	3320	3205	3396	**27.75**	10.63	8.57	8.69	139.25
4	**5937**	5754	5842	5906	5914	**23.65**	13.48	11.84	12.27	139.45
5	**9806**	9122	9678	9352	9586	**25.64**	16.61	13.57	14.54	150.27

Bold values indicate the bestperformance in each row.

applied for the first time to solve this problem, demonstrating its ability to obtain high-quality scheduling solutions stably and efficiently. Experiments confirm that the proposed method offers significant advantages in dynamic task allocation and resource optimization, providing new insights for real-time scheduling challenges in intelligent manufacturing. Future research will focus on exploring collaborative learning mechanisms within heterogeneous worker groups and extending the model to multi-objective[36–39] optimization scenarios, thereby further enhancing the algorithm's industrial applicability and generalizability. And adding future application scenarios (such as robot disassembly and large-scale remanufacturing) to expand the impact.

REFERENCES

[1] X. Guo, M. Zhou, S. Liu, and L. Qi, "Multiresource-constrained selective disassembly with maximal profit and minimal energy consumption," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 804–816, 2020.

[2] C. Z. X. S. B. H. Qi Zhang, Yang Xing and A. Das, "Column generation algorithms fortwo-dimensional cutting problem with surfacedefects," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 2, pp. 23–35, 2025.

[3] X. Cao, "Multi-time scale optimal scheduling of a photovoltaic energy storage building system based on model predictive control," *Energy Engineering*, p. 121.4, 2024.

[4] J. Behnamian, "Scheduling and worker assignment problems on hybrid flowshop with cost-related objective function," *The International Journal of Advanced Manufacturing Technology*, vol. 74, pp. 267–283, 2014.

[5] M. Krishnamoorthy and A. T. Ernst., "The personnel task scheduling problem," *Optimization methods and applications*, vol. 52, pp. 343–368, 2001.

[6] L. Abualigah and M. Alkhrabsheh, "Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing," *The Journal of Supercomputing*, vol. 78.1, pp. 740–765, 2022.

[7] C. Zhou, "Deep reinforcement learning for delay-oriented iot task scheduling in sagin," *IEEE Transactions on Wireless Communications*, vol. 20.2, pp. 911–925, 2020.

[8] K. B. C. Paul S. Adler, "Behind the learning curve: A sketch of the learning process," *Management Science*, vol. 37, pp. 267–281, 1991.

[9] L. H. S. Anna Stansbury, "The declining worker power hypothesis: An explanation for the recent evolution of the american economy," *Brookings Papers on Economic Activity*, vol. 2020, pp. 1–96, 2020.

[10] J. B. S. Gur Mosheiov, "Scheduling with general job-dependent learning curves," *European Journal of Operational Research*, vol. 147, pp. 665–670, 2003.

[11] M. Yang, G. Liu, Z. Zhou, and J. Wang, "Partially observable mean field multi-agent reinforcement learning based on graph attention network for uav swarms," *Drones*, vol. 7, no. 7, 2023.

[12] H. Shi, G. Liu, K. Zhang, Z. Zhou, and J. Wang, "Marl sim2real transfer: Merging physical reality with digital virtuality in metaverse," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 4, pp. 2107–2117, 2023.

[13] 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.

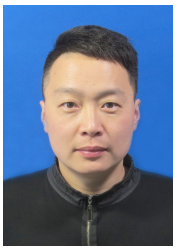
[14] J. Wang, G. Xi, X. Guo, S. Liu, S. Qin, and H. Han, "Reinforcement learning for hybrid disassembly line balancing problems," *Neurocomputing*, vol. 569, p. 127145, 2024.

[15] J. Xiao, Z. Zhang, S. Terzi, N. Anwer, and B. Eynard, "Dynamic task allocations with q-learning based particle swarm optimization for human-robot collaboration disassembly of electric vehicle battery recycling," *Computers Industrial Engineering*, vol. 204, p. 111133, 04 2025.

[16] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018, pp. 1861–1870.

[17] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PmLR, 2016, pp. 1928–1937.

[18] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep q-learning," in *Learning for dynamics and control*. PMLR, 2020, pp. 486–489.
[19] E. H. Sumiea, S. J. Abdulkadir, H. S. Alhussian, S. M. Al-Selwi, A. Alqushaibi, M. G. Ragab, and S. M. Fati, "Deep deterministic policy gradient algorithm: A systematic review," *Heliyon*, vol. 10, no. 9, p. e30697, 2024.
[20] 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*, pp. 1–12, 2025.
[21] X. Niu, X. Guo, P. Liu, J. Wang, S. Qin, L. Qi, B. Hu, and Y. Ji, "Hybrid disassembly line balancing of multi-factory remanufacturing process considering workers with government benefits," *Mathematics*, vol. 13, no. 5, p. 880, 2025.
[22] X. Guo, L. Zhou, Z. Zhang, L. Qi, J. Wang, S. Qin, and J. Cao, "Multi-objective optimization of multi-product parallel disassembly line balancing problem considering multi-skilled workers using a discrete chemical reaction optimization algorithm." *Computers, Materials & Continua*, vol. 80, no. 3, 2024.
[23] G. D. Tian, Y. P. Ren, Y. X. Feng, M. C. Zhou, H. H. Zhang, and J. R. Tan, "Modeling and planning for dual-objective selective disassembly using and/or graph and discrete artificial bee colony," *IEEE Trans. Ind. Inform.l Electronics Society*, vol. 15, no. 4, pp. 2456–2468, 2018.
[24] S. Qin, J. Li, J. Wang, X. Guo, S. Liu, and L. Qi, "A salp swarm algorithm for parallel disassembly line balancing considering workers with government benefits," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 1, pp. 282–291, 2024.
[25] A. J. Ruiz-Torres, "Scheduling to maximise worker satisfaction and on-time orders," *International Journal of Production Research*, vol. 53.9, pp. 2836–2852, 2015.
[26] H. Z. Liangbo Zhou and B. Akbari, "Multi-objective optimization of multi-factory remanufacturing process considering worker fatigue," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 2, pp. 36–50, 2025.
[27] S. Qin, J. Wang, J. Wang, X. Guo, L. Qi, and Y. Fu, "Linear disassembly line balancing problem with tool deterioration and solution by discrete migratory bird optimizer," *Mathematics*, vol. 12, no. 2, p. 342, 2024.
[28] Z. Li and M. N. Janardhanan, "Modelling and solving profit-oriented u-shaped partial disassembly line balancing problem," *Expert systems with applications*, vol. 183, p. 115431, 2021.
[29] K. Wang, L. Gao, X. Li, and P. Li, "Energy-efficient robotic parallel disassembly sequence planning for end-of-life products," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, pp. 1277–1285, 2021.
[30] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," pp. 1407–1416, 2018.
[31] J. Xiao, J. Gao, N. Anwer, and B. Eynard, "Multi-agent reinforcement learning method for disassembly sequential task optimization based on human–robot collaborative disassembly in electric vehicle battery recycling," *Journal of Manufacturing Science and Engineering*, vol. 145, 11 2022.
[32] J. Gao, G. Wang, J. Xiao, P. Zheng, and E. Pei, "Partially observable deep reinforcement learning for multi-agent strategy optimization of human-robot collaborative disassembly: A case of retired electric vehicle battery," *Robotics and Computer-Integrated Manufacturing*, vol. 89, p. 102775, 10 2024.
[33] J. Xiao, N. Anwer, W. Li, B. Eynard, and C. Zheng, "Dynamic bayesian network-based disassembly sequencing optimization for electric vehicle battery," *CIRP Journal of Manufacturing Science and Technology*, vol. 38, 07 2022.
[34] J. Xiao and S. Terzi, "Large language model-guided graph convolution network reasoning system for complex human-robot collaboration disassembly operations," *Procedia CIRP*, vol. 134, pp. 43–48, 01 2025.
[35] D. T. P. Haitao Zhang and Q. Kang, "Improved fruit fly algorithm for multi-objective disassembly line balancing problem considering learning effect," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 2, pp. 51–62, 2025.
[36] Z. Zhao, S. Liu, M. Zhou, and A. Abusorrah, "Dual-objective mixed integer linear program and memetic algorithm for an industrial group scheduling problem," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 6, pp. 1199–1209, 2022.
[37] X. Guo, T. Wei, J. Wang, S. Liu, S. Qin, and L. Qi, "Multiobjective u-shaped disassembly line balancing problem considering human fatigue index and an efficient solution," *IEEE Transactions on Computational Social Systems*, 2022.
[38] X. W. Guo, S. X. Liu, M. C. Zhou, and G. D. Tian, "Dual-objective program and scatter search for the optimization of disassembly sequences subject to multiresource constraints," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 3, pp. 1091–1103, 2018.
[39] X. Guo, S. Liu, M. Zhou, and G. Tian, "Dual-objective program and scatter search for the optimization of disassembly sequences subject to multiresource constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 1091–1103, 2017.



Xu Wang received his B.S. degree in Automation from Liaoning university of technology, Jinzhou, China, in 2006, M.S. degree in Control theory and control engineering from Liaoning university of technology, Jinzhou, China, in 2010, and Ph.D. degree in System Engineering from Northeastern University, Shenyang, China, in 2017. He is presently an Associate Professor of Computer Science and Technology at Shenyang University of Chemical Technology. His research focuses on logistics and supply chain management, project management and human resource optimization, intelligent optimization algorithm. He has published over 30+ journal and conference papers in the above research areas.

Haitao Zhang received her B.S. degree in Computer Science and Technology from Linyi University, China in 2022. Currently, she is a graduate student at the School of Artificial Intelligence and Software, Liaoning Shihua University, Fushun, China. Her research interests are disassembly line balancing problems and intelligent optimization algorithms.

Qi Kang received his Ph.D degree from Electrical and Computer Engineering department of New Jersey Institute of Technology. His research focuses the neural modulation with transcranial electrical stimulation and focused ultrasound stimulation. Prior to arriving at NJIT, he holds a Master of Science degree in Electrical and Computer Engineering from New York Institute of Technology in Old Westbury.