Neurocomputing 569 (2024) 127145

Contents lists available at ScienceDirect

Neurocomputing

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

Reinforcement learning for Hybrid Disassembly Line Balancing Problems

Jiacun Wang [a,*], GuiPeng Xi [b], XiWang Guo [b], Shixin Liu [c], ShuJin Qin [d], Henry Han [e]

[a] Monmouth University, NJ, 07764, USA
[b] LiaoNing Petrochemical University, Fu shun, 113005, China
[c] Northeastern University, Sheng yang, 110167, China
[d] Shangqiu Normal University, Shang qiu, 476000, China
[e] Baylor University, Waco, 76798, USA

ARTICLE INFO

Communicated by N. Zeng

Keywords:
Hybrid disassembly line balancing problem
Soft actor–critic algorithm
Deep reinforcement learning
Optimization

ABSTRACT

With the rapid development of the economy and technology, the rate of product replacement has accelerated, resulting in a large number of products being discarded. Disassembly is an important way to recycle waste products, which is also helpful to reduce manufacturing costs and environmental pollution. The combination of a single-row linear disassembly line and a U-shaped disassembly line presents distinctive advantages within various application scenarios. The Hybrid Disassembly Line Balancing Problem (HDLBP) that considers the requirement of multi-skilled workers is addressed in this paper. A mathematical model is established to maximize the recovery profit according to the characteristics of the proposed problem. To facilitate the search for optimal solution, a new strategy for agents in reinforcement learning to interact with complex and changeable environments in real-time is developed, and deep reinforcement learning is used to complete the distribution of multi-products and disassembly tasks. On this basis, we propose a Soft Actor–Critic (SAC) algorithm to effectively address this problem. Compared with the Deep Deterministic Policy Gradient (DDPG) algorithm, Advantage Actor–Critic (A2C) algorithm, and Proximal Policy Optimization (PPO) algorithm, the results show that the SAC can get the approximate optimal result on small-scale cases. The performance of SAC is also better than DDPG, PPO, and A2C in solving large-scale disassembly cases.

1. Introduction

With the continuous development of science and technology, the pace of product replacement is getting faster. As a consequence, a large number of waste products are generated. To alleviate the environmental burden and enhance resource efficiency, the recycling of used products is imperative. Disassembly is an important way for resource recycling. Through disassembly, valuable materials and parts can be reused, and materials that cause environment pollution are separated, ultimately achieving the goal of green remanufacturing [1].

In a disassembly line, workload balancing among workstations and workers directly impacts the cost-effectiveness of disassembly operations. Imbalanced workloads can lead to inefficient utilization of resources, increased cycle times, and labor inefficiencies. Balancing the workload across multi-skilled workers ensures that each worker's skills and capabilities are leveraged optimally, enhancing productivity and reducing costs associated with idle time, overtime, or underutilization of skilled labor. Ultimately, a well-balanced hybrid disassembly line can contribute to cost savings and improved profitability for organizations. Workload imbalances can negatively affect worker satisfaction and safety. Excessive workloads on certain workers can lead to fatigue, stress, and decreased job satisfaction. Moreover, if workers are assigned tasks that exceed their skill levels, it can increase the risk of errors, accidents, and injuries. Effective workload balancing in disassembly lines with multi-skilled workers is essential for promoting sustainability, cost-effectiveness, worker satisfaction and safety, as well as ensuring product quality and reliability. By addressing this problem through research and developing appropriate strategies, we can optimize disassembly operations and contribute to a more sustainable and efficient circular economy. The Disassembly Line Balancing Problem (DLBP) is an NP problem [2–4] that has been studied by many scholars.

The common layouts of disassembly lines are linear, U-shaped [5,6], two-sided [7], and parallel [8,9]. Linear disassembly lines are mainly used to disassemble small-scale products. For example, Wang et al. [10] study the partial disassembly line balancing problem in the linear disassembly line. Li et al. [11] compare linear disassembly lines with the U-shaped disassembly lines. Experiments show that U-shaped disassembly lines are more flexible and efficient and require fewer operators.

* Corresponding author.
E-mail addresses: jwang@monmouth.edu (J. Wang), xiguipeng@stu.lnpu.edu.cn (G. Xi), guoxiwang@lnpu.edu.cn (X. Guo), sxliu@mail.neu.edu.cn (S. Liu), qinshujin@sqnu.edu.cn (S. Qin), Henry_Han@baylor.edu (H. Han).

https://doi.org/10.1016/j.neucom.2023.127145
Received 14 August 2023; Received in revised form 28 October 2023; Accepted 6 December 2023
Available online 11 December 2023

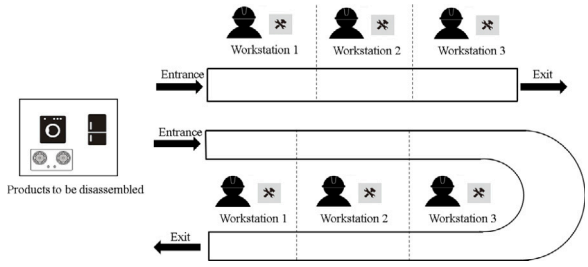


Fig. 1. An example of HDLBP.

Zhang et al. [12] propose an improved forbidden search algorithm to solve the multi-robot hybrid disassembly line balancing problem. Zhu et al. [13] use the union variable neighborhood descent algorithm to solve the multi-product hybrid disassembly line balancing problem considering workstation resource allocation. Linear disassembly lines and U-shaped disassembly lines have different characteristics and applicable scenarios, but there is a lack of research in academia to combine them to form a hybrid line that employs multi-skilled workers. This paper will explore the HDLBP considering the requirement of multi-skilled workers. Fig. 1 depicts an exemplification of the aforementioned HDLBP.

In the disassembly process of waste products, the number of workstations, disassembly cost, and idle time are mainly used as equilibrium indicators, and the hazard degree and smoothing rate are among objective functions. Many studies focus on the complexity and diversity of disassembly lines. For example, Paksoy et al. [14] establish a fuzzy objective mixed model and apply the binary fuzzy objective planning and fuzzy multi-objective planning methods to solve it. Guo et al. [15] propose multi-resource-constrained selective disassembly with maximal profit and minimal energy consumption. The above studies about DLBP mainly focus on profit and efficiency, ignoring the impact of employee factors on DLBP. Zhu et al. [16] consider the hazard parts in a disassembly line, indicating that the hazard products threaten workers' physical and mental health. Xu et al. [17] study that the disassembly skills acquired by employees have an impact on the disassembly balance. Therefore, it is of practical importance to consider personal factors in the disassembly line.

DLBP belongs to the class of NP problems, and its solution space expands exponentially with the size of the problem. To discover optimal solutions, numerous heuristic search techniques have been used. These include swarm intelligence algorithms, neighborhood search algorithms, and genetic algorithms [18–20]. The DLBP is also a combinatorial optimization problem. Traditional algorithms use heuristics to construct solutions sequentially, which can be suboptimal as the complexity of the problem increases. In contrast, reinforcement learning proposes an alternative approach by training an agent to automatically search for these heuristics in a supervised or self-supervised manner. Nina Mazyavkina [21] presents the current status and trends of applications of reinforcement learning in the field of combinatorial optimization. The combination of reinforcement learning and combinatorial optimization has broad application prospects and can be applied to many practical problems, such as route planning, scheduling optimization, resource allocation, etc. Researchers have proposed many reinforcement learning methods to solve different types of combinatorial optimization problems, such as Traveling Salesman Problem [22], Maximum Cut Problem [23], Bin Packing Problem [24], etc. In the experimental section, the authors also compare the performance of different reinforcement learning methods on various combinatorial optimization problems and provide some enlightening case studies that demonstrate the feasibility and effectiveness of reinforcement learning on combinatorial optimization problems.

Machine learning has played important roles in many areas in which learning is a core to the solution of a problem [25–27]. Reinforcement learning, as a subfield of machine learning, has showcased its efficacy in addressing complex combinatorial optimization and scheduling problems on a large scale. Reinforcement learning algorithms exhibit the ability to adapt to the challenges presented by multiple intricate products when addressing DLBP, ultimately facilitating the acquisition of high-quality solutions within a constrained timeframe. For example, Zhao et al. [28] study the optimal disassembly sequence with an uncertainty of EOL product structure, which uses the reinforcement learning method to deal with DLBP and could operate in a stochastic environment with determinism. In this paper, SAC is used to solve hybrid disassembly line balancing problem (HDLBP).

In this study, we use OpenAI Gym to define a hybrid disassembly line environment. In Gym, discrete problems can be defined using the Discrete class and MultiDiscrete class, while the action space for continuous problems can be defined using the Box class.

The main contributions of this paper are as follows:

(1) A hybrid disassembly system that is composed of both linear and U-shaped disassembly lines is presented for the first time. The mathematical model of the HDLBP is constructed with the objective of maximizing the disassembly profit, in which multi-skilled workers and their assignment to disassembly tasks are considered.

(2) A simulation environment specifically designed to realize the application of reinforcement learning algorithms for the hybrid disassembly line is developed.

(3) In addition, simulation experiments are carried out for real-world disassembly cases. The experimental findings indicate that the SAC algorithm exhibits superior performance compared to DDPG, A2C, and PPO algorithms in effectively addressing the HDLBP.

This paper is an extension of our previous work reported in [29], a conference paper, with significant improvements. First, according to the disassembly line layout characteristics, we establish a mathematical model of the HDLBP to maximize disassembly profit and verify the model with CPLEX to ensure the accuracy of the model. Secondly, we establish a hybrid disassembly line environment and choose four popular reinforcement learning algorithms to solve the model. The experiment shows that the SAC algorithm has advantages in solving HDLBP. Finally, we add more cases to explore the relationship between multi-skilled workers and disassembly task assignments. The experimental results prove the necessity of considering multi-skilled workers in HDLBP.

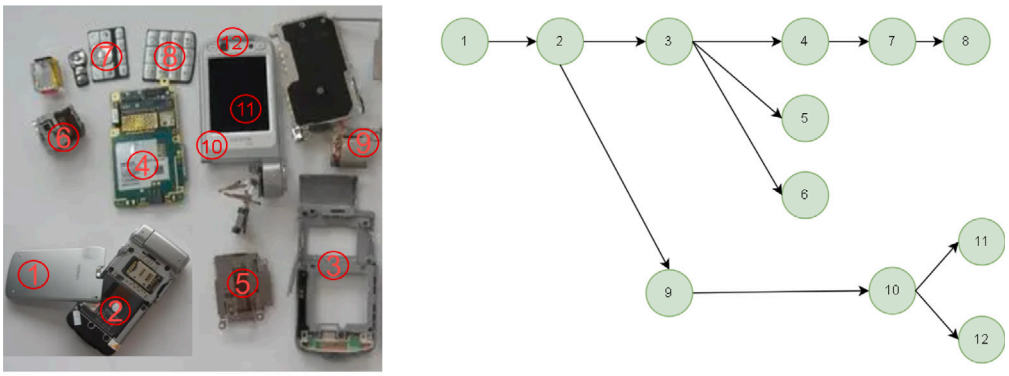
The rest of this paper is organized as follows. Section 2 describes the problem of HDLBP. Section 3 proposes the SAC algorithm. Section 4 presents the experimental outcomes and the performance of the SAC algorithm. Section 5 concludes this paper and discusses the future research direction.

2. Multi-product hybrid disassembly line balancing

2.1. Problem description

The hybrid disassembly line studied in this paper is a combination of a single-row linear disassembly line and a U-shaped disassembly line. Multiple products can be disassembled simultaneously on both lines. Different parts have different degrees of difficulty in the disassembly process, which requires workers to have different disassembly skills. For example, disassembling high-precision parts and dismantling dangerous parts require workers to have the corresponding but different skills. If a worker does not have any of the skills for a disassembly task, additional skill training is required before the worker can perform the task. Alternatively, tasks can be assigned to workers with relevant disassembly skills.

The HDLBP mainly consists of task assignment and disassembly line assignment. Task assignment involves how to assign the disassembly of different products to workers with different skills, and disassembly line assignment concerns how multiple products are assigned to different disassembly lines. With reinforcement learning, by learning the hybrid



(a) Components of a cellphone (b) Precedence graph of the cellphone disassembly tasks

Fig. 2. Cellphone components and precedence graph of disassembly tasks.

disassembly line environment, the agent transmits the decision scheme to the product distribution station and conveyor belt. The product distribution station distributes the product to different disassembly lines, and the conveyor belt transfers the part to the workstation suitable for the disassembly task. The optimization objective of the HDLBP problem in this work is to maximize profit.

The establishment of the disassembly product information model is the first step to study the HDLBP. Common disassembly information models mainly include AND/OR graph [30], precedence graph [31], Petri nets [32–35], and so on. The AND/OR graph employs a hierarchical representation technique, employing a top-down approach, to comprehensively and unambiguously depict the interrelationships among diverse components, tasks, and all disassembly sequences. However, with the increase in the complexity of a product, the combination explosion problem is easy to occur. The precedence diagram can clearly express the precedence relationship between tasks, which is easy to understand. Therefore, In this study, a precedence graph is used to represent the interdependencies and order of disassembly tasks.

Table 1 lists the disassembly tasks of the cell phone. Fig. 2(b) illustrates the precedence graph that depicts the interdependencies among the specified tasks. According to the graph, the execution of task 1 precedes that of task 2. In order to proceed with task 3 or task 9, completion of task 2 is a prerequisite. However, there exists no constraint relationship between task 3 and task 9, allowing flexibility in their execution sequence once task 2 has been concluded.

We define a matrix to describe the relationship between tasks, and another matrix to specify the relationship between tasks and skills.

(1) Precedence matrix

The precedence matrix $D = [d^p_{jk}]$ describes the relationship between any two tasks, where j and k are task indexes, p is the product ID.

$$d^p_{jk} = \begin{cases} 1, & \text{if task } j \text{ can be performed before task } k \\ & \text{in product } p \\ 0, & \text{otherwise.} \end{cases}$$

For example, assume the production ID of the cell phone is 1. Then based on Fig. 3 we have

$d^1_{11} = 0$;

$d^1_{1k} = 1$, for $k = 2, \ldots, 12$.

(2) Task-skill association matrix

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

$$b^p_{jn} = \begin{cases} 1, & \text{if disassembly task } j \text{ requires skill } n \text{ in product } p \\ 0, & \text{if disassembly task } j \text{ does not require skill } n \\ & \text{in product } p \end{cases}$$

Table 2 shows an example relationship between tasks and skills in disassembling he cellphone, in which Skill 1 is removing precision parts, and Skill 2 is removing dangerous parts. In the execution of Task 2, due to the flammable and explosive characteristics of the battery, we need workers to have experience handling dangerous products to disassemble them. In the execution of Task 4, as many small electronic components are on the motherboard, we require workers to have experience disassembling precision components.

Table 1
Cell phone disassembly tasks.

Task index	Task name	Task index	Task name
1	remove battery cover	7	remove the main button
2	remove battery	8	remove number button
3	remove back case	9	remove junction
4	remove board	10	remove front case
5	remove microphone	11	remove LCD
6	remove camera	12	remove speaker

Table 2
Relationship between tasks and skills.

Task index	Skill		Task index	Skill	
	1	2		1	2
1	0	0	7	0	0
2	0	1	8	0	0
3	0	0	9	1	0
4	1	0	10	0	0
5	1	0	11	1	0
6	1	0	12	1	0

Table 3
Relationship between workers and skills.

Disassembly line type	Worker	Skill	
		1	2
Linear	1	0	1
	2	0	0
	3	1	0
U-shaped	1	1	1
	2	0	0
	3	1	1

(3) Worker-skill association matrix

A worker-skill association matrix $A = [a^l_{wn}]$ is used to describe the relationship between worker and skills.

$$a^l_{wn} = \begin{cases} 1, & \text{if Worker } w \text{ has Skill } n \text{ on line } l \\ 0, & \text{if Worker } w \text{ does not have Skill } n \text{ on line } l \end{cases}$$

For example, the relationship between worker and skills is shown in Table 3.

To build up a hybrid disassembly line, we assume that:

(1) Matrices D, A, and B are known.

(2) All parts of the product need to be disassembled.
(3) Each disassembly task can only be performed once.
(4) Multi-skill workers are fixed to corresponding workstations and cannot switch workstations.

2.2. Mathematical model

This section presents the mathematical model of the HDLBP problem. The model primarily focuses on the allocation of multiple products on a hybrid disassembly production line. The constraints include those ensuring the precedence relationship of disassembly tasks, tasks are assigned to workstations with required skills, a task can only be assigned to one workstation, etc. The goal is to maximize the profit of the disassembly line. The notations and decision variables for modeling HDLBP are defined as follows.

(1) Notations:

$\mathbb{P}$ Set of all End-of-life products, $\mathbb{P} = \{1, 2, \dots, p\}$.
$\mathbb{J}_p$ Number of tasks in product p.
$\mathbb{L}$ Set of all disassembly lines.
$\mathbb{W}^1$ Set of all linear disassembly line workstations, $\mathbb{W}^1 = \{1, 2, \dots, W^1\}$.
$\mathbb{W}^2$ Set of all U-shaped disassembly line workstations, $\mathbb{W}^2 = \{1, 2, \dots, W^2\}$.
$\mathbb{S}$ Set of the two sides of U-shaped disassembly line workstation, $\mathbb{S} = \{1, 2\}$.
$\mathbb{N}$ Set of all additional skills, $\mathbb{N} = \{1, 2, \dots, N\}$.
v_{pj} The value of performing the jth task of the pth product.
t_{pj} The time of executing the jth task of the pth product.
c_{pj} The time unit cost of executing the jth task of the pth product.
c_l The time unit cost of executing the lth disassembly line.
c_w The time unit cost of executing the wth workstation.
C Individual skill training cost.
α_{lwn} The worker on workstation wth on the lth disassembly line has the nth skill.
β_{pjn} The jth task of the pth product requires the use of nth skill.

(2) Decision variables

$$z_{pl} = \begin{cases} 1, & \text{product } p \text{ assigned to disassembly line } l \\ 0, & \text{otherwise} \end{cases}$$

$$x^1_{pjw} = \begin{cases} 1, & \text{task } j \text{ of product } p \text{ is assigned to linear disassembly line workstation } w \ (w \in \mathbb{W}^1) \\ 0, & \text{otherwise} \end{cases}$$

$$x^2_{pjws} = \begin{cases} 1, & \text{task } j \text{ of product } p \text{ is assigned to the } s \text{ side of U-shaped disassembly line workstation } w \ (w \in \mathbb{W}^2) \\ 0, & \text{otherwise} \end{cases}$$

$$y_l = \begin{cases} 1, & \text{open disassembly line } l \\ 0, & \text{otherwise} \end{cases}$$

$$u_{lw} = \begin{cases} 1, & \text{open disassembly line } l \text{ of the workstation } w \\ 0, & \text{otherwise} \end{cases}$$

$$\xi_{lwn} = \begin{cases} 1, & \text{if a worker on workstation } w \text{ of the disassembly line } l \text{ uses skill } n (n \in \mathbb{N}) \\ 0, & \text{otherwise} \end{cases}$$

T^l, cycle time of disassembly line l.

(3) Mathematical model of maximizing disassembly profit:

$$\begin{aligned} \max \sum_{p\in\mathbb{P}} \sum_{j\in\mathbb{I}_p} \Bigg(& \sum_{w\in\mathbb{W}^1} \left(v_{pj} - c_{pj}t_{pj}\right) x^1_{pjw} \\ & + \sum_{w\in\mathbb{W}^2} \sum_{s\in\mathbb{S}} \left(v_{pj} - c_{pj}t_{pj}\right) x^2_{pjws} \Bigg) \\ & - \sum_{l\in\mathbb{L}} \left(c_l T^l + \sum_{w\in\mathbb{W}^l} c_{lw} u_{lw} \right) \\ & - C \sum_{l\in\mathbb{E}} \sum_{w\in\mathbb{W}^l} \sum_{n\in\mathbb{N}} \left(\xi_{lwn} - \alpha_{lwn}\right) \end{aligned} \tag{1}$$

The objective function (1) represents the maximization of profit of the hybrid disassembly line, which consists of three parts. In Part 1: We calculate the profit from disassembling the product. For each task j in each product p, we compute the sum of the following two terms: For the linear disassembly line workstation $\mathbb{W}^1$, we compute $(v_{pj} - c_{pj}t_{pj})$ multiplied by the decision variable x^1_{pjw}, which is the portion that is the profit of disassembly for the linear disassembly line. For each side s in the U-shaped disassembly line workstation $\mathbb{W}^2$, we compute $(v_{pj} - c_{pj}t_{pj})$ multiplied by the decision variable x^2_{pjws}, which is the portion that is the profit of disassembly for the U-shaped disassembly line. In Part 2: We subtract the cost of all disassembly lines l. This includes: The time unit cost of the disassembly line c_l multiplied by the cycle time of the disassembly line T^l. The time unit cost of the disassembly line workstation c_{lw} multiplied by the decision variable u_{lw} indicating whether the workstation is open or not. In Part 3: We subtract the cost of skills training. This includes: For each disassembly line $l \in \mathbb{L}$, workstation $w \in \mathbb{W}^l$, and skill $n \in \mathbb{N}$, we compute $(\xi_{lwn} - \alpha_{lwn})$ multiplied by the constant C.

$$\sum_{l\in\mathbb{L}} z_{pl} = 1, \forall p \in \mathbb{P} \tag{2}$$

The formula (2) indicates that each product p must be and can only be assigned to one disassembly line l. In other words, each product must be assigned to a unique disassembly line.

$$z_{pl} \le y_l, \forall p \in \mathbb{P}, \forall l \in \mathbb{L} \tag{3}$$

The formula (3) ensures that if product p is assigned to disassembly line l (i.e., $z_{pl} = 1$), then that disassembly line must be open (i.e., $y_l = 1$). If the disassembly line is not open, then the product cannot be assigned to that line.

$$u_{lw} \le y_l, \forall w \in \mathbb{W}^l, \forall l \in \mathbb{L} \tag{4}$$

The formula (4) indicates that if the workstation w is open on the disassembly line l (i.e., $u_{lw} = 1$), then that disassembly line must be open (i.e., $y_l = 1$).

$$\xi_{lwn} \ge \alpha_{lwn}, \forall l \in \mathbb{L}, \forall w \in \mathbb{W}^l, \forall n \in \mathbb{N} \tag{5}$$

The formula (5) indicates that if a worker uses a skill on a particular workstation, then he must have at least that skill. ξ_{lwn} cannot take a value less than α_{lwn} or the skill requirement constraint is violated.

$$x^1_{pjw} \le z_{pl}, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_p, l = 1, \forall w \in \mathbb{W}^1 \tag{6}$$

The formula (6) ensures that if the product p of task j is assigned to the linear disassembly line workstation w (i.e., $x^1_{pjw} = 1$), then that product must be assigned to the linear disassembly line l (i.e., $z_{pl} = 1$). If the product is not assigned to that line, then the task cannot be assigned to that workstation either.

$$x^1_{pjw} \le u_{lw}, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_p, l = 1, \forall w \in \mathbb{W}^1 \tag{7}$$

The formula (7) ensures that if the product p of task j is assigned to the linear disassembly line workstation w (i.e., $x^1_{pjw} = 1$), then that workstation must be open (i.e., $u_{lw} = 1$). If the workstation is not open, then the task cannot be assigned to that workstation.

$$x^1_{pjw} \le \sum_{n\in\mathbb{N}} \beta_{pjn}\xi_{lwn}, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_p, l = 1, \forall w \in \mathbb{W}^1 \tag{8}$$

The formula (8) ensures that if the product p of task j is assigned to a linear disassembly line workstation w (i.e., $x^1_{pjw} = 1$), then the workers at that workstation must have the skills required to perform the task.

$$x^2_{pjws} \le z_{pl}, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_p, l = 2, \forall w \in \mathbb{W}^2, \forall s \in \mathbb{S} \tag{9}$$

The formula (9) ensures that if the product p of task j is assigned to the side s of the U-shaped disassembly line workstation w (i.e., $x^2_{pjws} = 1$), then that product must be assigned to the U-shaped disassembly line

l (i.e., $z_{pl} = 1$). If the product is not assigned to that line, then the task cannot be assigned to that workstation either.

$$x^2_{pjws} \le u_{lw}, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_p, l = 2, \forall w \in \mathbb{W}^2, \forall s \in \mathbb{S} \tag{10}$$

The formula (10) ensures that if the product p of task j is assigned to the side s of the U-shaped disassembly line workstation w (i.e., $x^2_{pjws} = 1$), then that workstation must be open (i.e., $u_{lw} = 1$). If the workstation is not open, then the task cannot be assigned to that workstation.

$$x^2_{pjws} \le \sum_{n\in\mathbb{N}} \beta_{pjn}\xi_{lwn}, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_p, l = 2, \forall w \in \mathbb{W}^2, \forall s \in \mathbb{S} \tag{11}$$

The formula (11) ensures that if the product p of task j is assigned to the side s of the U-shaped disassembly line workstation w (i.e., $x^2_{pjws} = 1$), then the workers at that workstation must have the skills required to perform the task.

$$\sum_{p\in P}\sum_{j\in\mathbb{J}_p} t_{pj}x^1_{pjw} \le T^1, \forall w \in \mathbb{W}^1 \tag{12}$$

$$\sum_{p\in P}\sum_{j\in\mathbb{J}_p}\sum_{s\in\mathbb{S}} t_{pj}x^2_{pjws} \le T^2, \forall w \in \mathbb{W}^2 \tag{13}$$

The formula (12)–(13) ensures that the total time for all tasks performed on a disassembly line workstation w does not exceed the cycle time for that disassembly line T^l

$$\sum_{w\in\mathbb{W}^1} w\left(x^1_{pjw} - x^1_{pkw}\right) + W^1\left(\sum_{w\in\mathbb{W}^1} x^1_{pkw} - 1\right) \le 0, \quad \forall p \in \mathbb{P}, \forall j, k \in \mathbb{J}_p \text{ and } d_{pjk} = 1 \tag{14}$$

$\sum_{w\in\mathbb{W}^1} w\left(x^1_{pjw} - x^1_{pkw}\right)$ denotes the sum of the difference between the workstation number of task j multiplied by the number of tasks j assigned to it on that workstation, and the workstation number of task k multiplied by the number of tasks k assigned to it on that workstation., which must be less than or equal to 0. This condition ensures that there are not multiple tasks running simultaneously on the same workstation under the same product. Second, $W^1\left(\sum_{w\in\mathbb{W}^1} x^1_{pkw} - 1\right)$ denotes that the sum of the differences between the sum of the number of assignments of a task k on all workstations and 1, multiplied by the total number of workstations W^1, must also be less than or equal to 0. This condition ensures that the number of assignments of a task on all workstations This condition ensures that the sum of the number of assignments of a task on all workstations is 1, i.e., each task can be assigned to only one workstation.

$$\sum_{w\in\mathbb{W}^2}\left(w\left(x^2_{pjw1} - x^2_{pkw1}\right) + \left(2W^2 - w\right)\left(x^2_{pjw2} - x^2_{pkw2}\right)\right) + 2W^2\left(\sum_{w\in\mathbb{W}^2}\sum_{s\in\mathbb{S}} x^2_{pkws} - 1\right) \le 0, \quad \forall p \in \mathbb{P}, \forall j, k \in \mathbb{J}_p \text{ and } d_{pjk} = 1 \tag{15}$$

This Eq. (15) is a constraint on the U-shaped disassembly line and acts similarly to Eq. (14).

$$d_{pjk}\left(\sum_{w\in\mathbb{W}^1} x^1_{pkw} - \sum_{w\in\mathbb{W}^1} x^1_{pjw}\right) \le 0 \tag{16}$$

$$d_{pjk}\left(\sum_{w\in\mathbb{W}^2}\sum_{s\in\mathbb{S}} x^2_{pkws} - \sum_{w\in\mathbb{W}^2}\sum_{s\in\mathbb{S}} x^2_{pjws}\right) \le 0 \tag{17}$$

Eqs. (16)–(17) indicate that for any two tasks j and k of product p, if there is a sequential relationship (i.e. $d_{pjk} = 1$), the following condition needs to be satisfied on a linear disassembly line: $\sum_{w\in\mathbb{W}^1} x^1_{pkw} - \sum_{w\in\mathbb{W}^1} x^1_{pjw}$ must be less than or equal to 0. On the U-shaped demolition line, the following condition needs to be satisfied: $\sum_{w\in\mathbb{W}^2}\sum_{s\in\mathbb{S}} x^2_{pkws} - \sum_{w\in\mathbb{W}^2}\sum_{s\in\mathbb{S}} x^2_{pjws}$ must be less than or equal to 0. This condition ensures that the order of the tasks j and tasks k with sequential relationships in terms of the number of assignments under the same product conforms to their sequential relationships.

$$\sum_{w\in\mathbb{W}^1} x^1_{pjw} + \sum_{w\in\mathbb{W}^2}\sum_{s\in\mathbb{S}} x^2_{pjws} \le 1, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_p \tag{18}$$

Eq. (18) indicates that for any task j of product p, the sum of the number of assignments on time period $\mathbb{W}^1$ and the sum of the number of assignments on time period $\mathbb{W}^2$ plus the sum of the number of assignments of that task on all time periods and workstations must be less than or equal to 1. This condition ensures that the number of assignments of each task across the entire time period does not exceed 1, i.e., each task can only be assigned to one time period and one workstation.

The value range of decision variables:

$$z_{pl} \in \{0, 1\}, \forall p \in \mathbb{P}, \forall l \in \mathbb{L} \tag{19}$$

$$x^1_{pjw} \in \{0, 1\}, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_p, w \in \mathbb{W}^1 \tag{20}$$

$$x^2_{pjws} \in \{0, 1\}, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_p, w \in \mathbb{W}^2, s \in \mathbb{S} \tag{21}$$

$$y_l \in \{0, 1\}, \forall l \in \mathbb{L} \tag{22}$$

$$u_{lw} \in \{0, 1\}, \forall l \in \mathbb{L}, \forall w \in \mathbb{W}^l \tag{23}$$

$$\xi_{lwn} \in \{0, 1\}, \forall l \in \mathbb{L}, \forall w \in \mathbb{W}^l, \forall n \in \mathbb{N} \tag{24}$$

$$T^l \in \mathbb{R}_+, \forall l \in \mathbb{L} \tag{25}$$

Constraints (19)–(25) restrict the value ranges of decision variables.

The present study proposes a hybrid disassembly line model, which is demonstrated with the example of disassembling multiple cell phones. The first step is to allocate the cell phones to different disassembly lines, followed by the assignment of disassembly tasks. These tasks can be assigned to the same or different workstations, but subject to satisfying the priority relation for each cell phone. It should be noted that each disassembly task requires a specific skill, and not all workers on the workstation possess this skill. Therefore, the unskilled workers need to undergo appropriate training before undertaking the disassembly task. The disassembly process continues until all parts are disassembled and the product leaves the conveyor.

3. The Soft Actor–Critic algorithm for HDLBP

In reinforcement learning, the learner is called an *agent*, which is also the *actor* of the learning system. The agent interacts with the *environment*, the surrounding of the agent, by taking *actions*. For each action taken in a *state* of the environment, the agent receives a *reward*, and the system transitions to a new state. The ultimate goal of reinforcement learning is to find a strategy (a sequence of actions) that can achieve the highest reward cumulatively, that is, the best solution to a problem. In the context of DLBP, a reinforcement learning algorithm learns the optimal allocation scheme through real-time interactive data and obtains the optimal solution. In this study, we use the Soft Actor–Critic algorithm for the HDLBP optimization.

The SAC algorithm [36] is the first deep reinforcement learning algorithm that combines the off-policy algorithm, actor–critic methods, and maximum entropy framework. SAC has advantages in dealing with complex tasks. SAC has a high sample utilization rate and can avoid the local optimum through the maximum entropy, has better exploration ability, and is easier to make adjustments in the face of interference to achieve rapid convergence. The theoretic framework of SAC, like any RL algorithm, is Markov decision process (MDP). We introduce MDP in Section 3.1, and present the mathematical formulations of SAC in Section 3.2. In Section 3.3, we discuss the application of SAC to HDLBP.

3.1. Markov Decision Process (MDP)

MDP is a theoretical framework for describing the decision-making process of an RL agent over a finite set of states and a set of actions. The agent chooses an action by observing the current state and receives a reward, then updates its state and continues to make decisions based on the new state, and so on. The goal of the agent is to maximize the sum of long-term rewards. It can be described as a quintuple (S, A, P, R, γ), where S represents the set of all possible states in the environment, and A is the set of all actions that the agent can take. The transition probability function $P(s, a, s')$ indicates the probability of state transition to s' after taking action a in state s. The reward function $R(s, a, s')$ represents the reward obtained when the state s takes action a and moves to the state s'. The discount factor γ indicates the degree of consideration for future awards. The greater the γ, the greater the influence of future rewards on current decisions.

3.2. The Soft Actor–Critic (SAC) algorithm

The core feature of the SAC algorithm is entropy regularization, and the training of strategy makes a trade-off between maximizing expected return and maximizing entropy. In the realm of reinforcement learning, the primary objective is to optimize the accumulation of rewards, that is, to maximize the value of state–action $Q_\pi(s, a)$. A deterministic strategy can directly select the action with the largest Q_π to make a state transition, but that could lead to repeated selection of the same action and the thus a local optimal strategy. The objective function of the SAC algorithm requires that strategy π to maximize not only the ultimate return but also the entropy. The optimal policy π^* is mathematically defined in Eq. (26):

$$\pi^* = \arg\max_{\pi} \mathbb{E}[\sum_t \underbrace{R\left(s_t, a_t\right)}_{\text{reward}} + \underbrace{\alpha H\left(\pi\left(\bullet \mid s_t\right)\right)}_{\text{entropy}}] \tag{26}$$

where s_t and a_t are the state and action at t, respectively, R the reward, H the entropy function, and α the temperature parameter. By exploring the maximum entropy strategy, the actions with similar Q values are assigned approximately equal probabilities, to prevent recurrent selection of the same action and potential convergence to local optimal. Simultaneously, to enhance training efficiency and stability, the SAC algorithm incorporates two Q-networks that facilitate the selection of the smaller target Q-value at each step. This method can effectively reduce the variance and make the training process smoother and faster. Moreover, the adoption of the maximum entropy strategy promotes more uniform exploration, thereby accelerating the training process.

Similar to standard reinforcement learning, policy evaluation and improvement is also achieved by training neural networks using stochastic gradient descent. The Q function $Q_\theta\left(s_t, a_t\right)$ is trained through the neural network with parameter θ parameterization. The soft Q function is trained by minimizing the error defined in Eq. (27):

$$J_Q(\theta) = E_{(s_t, a_t)\sim D}\left[Q_\theta\left(s_t, a_t\right) - \hat{Q}\left(s_t, a_t\right)\right]^2 \tag{27}$$

The update of the policy network is to minimize the Kullback–Leibler divergence as shown in Eq. (28).

$$J_\pi(\phi) = D_{KL}\left(\pi_\phi\left(\cdot \mid s_t\right) \| \exp\left(\frac{1}{\alpha} Q_\theta\left(s_t, \cdot\right)\right) - \log Z\left(s_t\right)\right)\right) \tag{28}$$

Because the reward changes constantly, it is not reasonable to use a fixed temperature. When the agent is exploring a new area, the optimal action is not clear. The temperature needs to be adjusted to allow the agent to explore more space. When the optimal action is determined, it can be reduced appropriately. Construct a constrained optimization problem where the average entropy weight is limited, but the entropy weight is variable at different states, as in Eqs. (29) and (30)

$$\begin{aligned} &\max_{\pi_{0:T}} \mathbb{E}_{\rho_\pi}\left[\sum_{t=0}^{T} r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right] \text{ s.t.} \\ &\mathbb{E}_{\left(\mathbf{s}_t, \mathbf{a}_t\right)\sim\rho_\pi}\left[-\log\left(\pi_t\left(\mathbf{a}_t \mid \mathbf{s}_t\right)\right)\right] \geq \mathcal{H} \forall t \end{aligned} \tag{29}$$

Algorithm 1: SAC for HDLBP

Input: $\theta_1, \theta_2, \phi, \bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2, \mathcal{D} \leftarrow \emptyset$
Output: θ_1, θ_2, ϕ
for *each iteration* **do**
  for *each environment step* **do**
    Sample action from the policy: $a_t \sim \pi_\phi\left(a_t \mid s_t\right)$
    Action mapping:
    new $a_t = mapping(a_t)$
    Sample transition from the environment: $s_{t+1} \sim p\left(s_{t+1} \mid s_t, a_t\right)$
    Store the transition in the replay pool: $D \leftarrow D \cup \left\{\left(s_t, a_t, r\left(s_t, a_t\right), s_{t+1}\right)\right\}$
  end
  for *each gradient step* **do**
    Update Q-function parameters: $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q\left(\theta_i\right) \quad i \in \{1, 2\}$
    Update policy weights: $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    Adjust temperature parameter: $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$
    Update target network weights: $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i \quad i \in \{1, 2\}$
  end
end

$$J(\alpha) = \mathbb{E}_{\mathbf{a}_t\sim\pi_t}\left[-\alpha \log \pi_t\left(\mathbf{a}_t \mid \mathbf{s}_t\right) - \alpha\overline{\mathcal{H}}\right] \tag{30}$$

The implementation details of Eqs. (27)–(30) can be found in [37] and will not be discussed here due to space limit.

The pseudo-code of SAC for HDLBP is shown in Algorithm 1. The main flow of the code is as follows: The target network weights are initialized, and an empty replay pool is created in order to facilitate the training of a neural network model. Generate actions based on the current policy and convert the values generated in the space of continuous actions. Perform a state transfer based on the current action and compute the current reward. Store the information in the replay pool. Update the corresponding parameters. Output parameters after training is complete.

The distribution of training data has an impact on the performance of reinforcement learning (RL) agents. Training data heavily biased towards certain states or actions may cause RL agents to be too specialized in these areas, resulting in low generalization or exploration capabilities. In the SAC algorithm, the following strategies can indirectly alleviate the impact of data distribution skew:

1. Experience playback memory: Using experience playback memory is a key component of SAC algorithm. It randomly samples past experience and uses it for training, which helps to balance the distribution between different states and actions and reduces the impact of skewed data distribution.

2. Random strategy exploration: SAC algorithm uses deterministic strategy and random strategy for training. Random strategy exploration can make the agent explore the state and action space more diversely, and alleviate the data distribution skew to a certain extent.

3. Entropy regularization: The SAC algorithm promotes exploration by maximizing the entropy of the strategy, that is, encouraging the strategy to select actions with greater uncertainty in a given state. This can help the agent to explore different states and actions more evenly during the training process, thus alleviating the problem of data distribution skew.

3.3. Apply SAC to HDLBP

In this section, we provide a detailed account of the steps involved in integrating the SAC algorithm with the HDLBP. To begin with, we

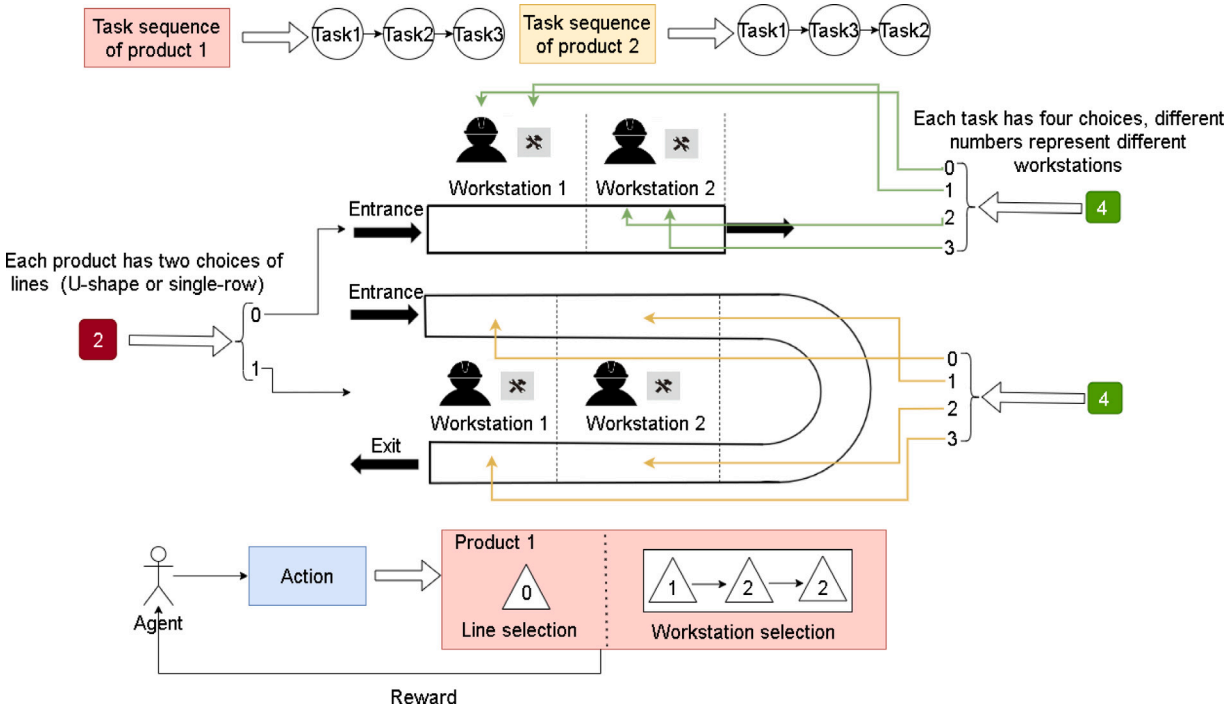


Fig. 3. Action design for an HDLBP instance with a linear disassembly line and a U-shape disassembly line. There are two products to be disassembled. Each disassembly line has two workstations and each product has three disassembly tasks. The action is to select a disassembly line first and then a workstation. For a workstation in the U-shape line, an entry side or exit side must also be selected. The example action code (0, 1, 2, 2 for product 1 shown in the pink box means product 1 is to be disassembled on the single-row linear disassembly line, its task 1 is assigned to workstation 1, and its tasks 2 and 3 are assigned to workstation 2.

Table 4
Action selection code.

Line selection		Workstation selection	
0	Linear line	0 or 1	W1
		2 or 3	W2
1	U-shaped line	0	entry side of W1
		1	entry side of W2
		2	exit side of W1
		3	exit side of W2

first define the disassembly state space, the final disassembly requirement is complete disassembly, so we only consider the initial state and the complete disassembly state as the disassembly process state. The next step involves defining the action space encoding, which employs distinct numerical values to represent the different disassembly lines and workstations. This allows for a clear representation of the various task assignments to be carried out during the disassembly process. Finally, the reward function is defined using the profit obtained from complete disassembly. This serves as an appropriate measure of performance able to quantify the effectiveness of the system in achieving the desired outcome.

During the dynamic interplay between an agent and its environment, the agent's actions exert influence on the state, consequently generating associated rewards. The efficacy of action selection lies crucial in facilitating the agent's attainment of optimal rewards within an expedited timeframe. While initially conceived for interactive environments featuring continuous action spaces, the SAC algorithm has seen adaptations that render it amenable to discrete environments as well.

The Discrete action space in OpenAI consists of a discrete non-negative integers. For example, Discrete(4) can represent a maze problem with four possible actions: 0 for moving left, 1 for moving right, 2 for moving up, and 3 for moving down. MultiDiscrete allows us to control multiple discrete actions simultaneously. On the other hand, the Box action space defines an n-dimensional space with real numbers, where each dimension represents an action that can take any real value. Box action space is suitable for environments that require continuous action characteristics from agents. The Box action space is usually defined by two parameters: low and high. Low is an n-dimensional real vector representing the lower bound of the action space; high is also an n-dimensional real vector representing the upper bound of the action space. When we want to generate a Box action, we need to sample an n-dimensional real vector such that this vector is within the range defined by low and high. Specifically, for the ith dimension, generate a random real number a_i that satisfies $low[i] \leq a_i \leq high[i]$, a_i falls within the range. Eventually, the generated vector a is a legal Box action. For example, to control a robot's knee and elbow joints, we can define a two-dimensional Box action space, where the first dimension represents the knee angle and the second dimension represents the elbow angle. Assuming low=[−1.0, −1.0] and high=[1.0, 1.0], we can sample a real-valued vector of length 2, where each element ranges from −1.0 to 1.0 to control the robot's motion. For example, [0.2, −0.5] represents rotating the knee by 20% and raising the elbow by 50%. As the complexity of the problem increases, more actions need to be designed. Using Discrete to encode the actions often leads to unsatisfactory returns within reasonable training times, Whether we use a small number of actions through different group sums to represent a large number of possible actions, or directly define a large action space. Therefore, we try to use the Box class that defines a continuous action space in solving the discrete problem and use the SAC and DDPG. A2C and PPO use the MultiDiscrete class.

(1) State Space

The state space is primarily used to store the disassembly state, which is divided into two states: the undisassembled state ($S = 0$) and the fully disassembled state ($S = 1$). We use $S = 0$ to represent the initial undisassembled state, and $S = 1$ to indicate that the product is in the fully disassembled state after all tasks have been completed. The transition of the disassembly process from state 0 to state 1 represents the change in the overall disassembly state, without considering the intermediate states and progress during disassembly.

(2) Action Space

We explain how to select a disassembly line in the continuous action space. Suppose we have two products that need to be assigned to one of two different disassembly lines, the ith dimension represents the selection of the disassembly line for the ith product. We define an action space with low=[0.0, 0.0] and high=[2.0, 2.0]. Then, we can sample a vector of real numbers with a length of 2 and each element between 0.0 and 2.0 to represent the assignment of the product to a specific disassembly line. We map this interval to the specific value. If the sampled value falls within the range of [0.0, 1.0), we map it to 0 to mean the product is assigned to the linear line. Otherwise, if the value falls within the range of [1.0, 2.0], we map it to 1 to represent the product is assigned to the U-shaped line. After the mapping, the specific encoding and decoding can be done with the MultiDiscrete class in OpenAI Gym.

For the presented HDLBP problem, as illustrated in Fig. 3, involving the disassembly of two products and two workstations per line, the encoded meaning of the action selection is elaborated upon in Table 4. The execution sequence for Product 1 is 1-2-3, while that for Product 2 follows 1-3-2. A total of five actions are involved in the disassembly selection process. The first action entails the selection of the line for Product 1, which can be either the linear line or the U-shaped line. The second action pertains to the line selection for Product 2. The third action involves the selection of the task on the workstations. Considering the U-shaped disassembly line, there are four options available for the two workstations in order to distinguish between the entry side and exit side. Should the product be allocated to the linear disassembly line, a task assignment code of 0 or 1 signifies assignment to workstation 1, while a code of 2 or 3 indicates assignment to workstation 2. In the case of the U-shaped disassembly line, when the task selection code is 0, it implies assignment to the entrance side of workstation 1. If the code is 1, the task is assigned to the entrance side of workstation 2; for a code of 2, the task is designated to the exit side of workstation 2; and if the code is 3, the task is allocated to the exit side of workstation 1. Different actions within the disassembly selection are denoted by distinct values, with the corresponding reward value being computed based on the selected actions.

For illustrative purposes, as depicted in Fig. 3, if the agent randomly selects an action code of [0, 1, 2, 2] for Product 1, it signifies that Product 1 is to undergo disassembly on the linear disassembly line. Furthermore, its first task is assigned to workstation 1, while its second and third tasks are allocated to workstation 2.

(3) Reward Function

We use the profit of a disassembly task as the reward of the agent on the disassemble action, as shown in Eq. (1). The corresponding actions are converted to the disassembly allocation of the product to calculate the final disassembly profit, and the meaning of the specific actions is illustrated in Fig. 3. A different random sample of actions will produce different rewards, to further stimulate the agent's choice of actions through rewards. Regarding the selection of disassembly line types, given that the linear line and the U-shaped line possess distinct characteristics, agents adopt a random assignment strategy for products during the initial attempt. Subsequently, as a superior payoff is attained in the current state, the probability of selecting that particular action is augmented. In the selection of workstations, the task requires some skill to disassemble because the workers on different workstations have different skills. The agent has two choices. If the worker whose task is assigned to this workstation does not have the skill, we can choose to spend extra money to learn the skill so that we can complete the disassembly task. The reward in the current state will be reduced because of the additional training cost incurred. If many tasks require this skill, then it is worth the training expense, because it will allow more workstations to perform tasks with this skill. Thus, in the long term, the efficiency of disassembly tasks will be improved, the cost will be reduced, and the total profit will be increased. If the agent chooses not to do additional training but to assign tasks to workstations with workers with that skill, the current reward is higher than after training. In the long run, if there are too many disassembly tasks that require this skill, assigning it to the same workstation many times instead of skill training will lead to an increase in disassembly line cycle time and thus the increase of cost, and the final reward may not be as good as that after training. The agent endeavors to optimize the final return value by undertaking various actions in accordance with the observed return value within different states.

Table 5
Products for disassembly test.

Product	Number of tasks	Number of required skills
TV	12	5
Cellphone	12	5
Tesla battery	37	5

4. Experiments

DDPG [38], A2C [39], and PPO [40] are widely recognized as effective deep reinforcement learning algorithms. They have been successfully applied to various domains, including robotics, gaming, and control tasks. Reinforcement learning algorithms need to balance exploration and exploitation to discover new solutions while exploiting existing knowledge. Reinforcement learning algorithms need to balance exploration and exploitation to discover new solutions while exploiting existing knowledge. DDPG, A2C, and PPO have different strategies for handling this trade-off, such as using exploration noise, policy gradients, or trust regions. In summary, DDPG, A2C, and PPO were chosen for comparison in SAC based on their effectiveness in reinforcement learning, their compatibility with continuous action spaces, their approach to the exploration–exploitation trade-off, and their status as state-of-the-art algorithms. By evaluating and comparing these algorithms, we aim to provide a meaningful assessment of our proposed methods and their performance in solving specific problems.

To assess the efficacy of the model and obtain the optimal solution, we employ the IBM ILOG CPLEX solver. Subsequently, we utilize the reinforcement learning algorithm to address identical disassembly instances and conduct a comparative analysis of the outcomes. We use the stable-baselines3 framework [41] to implement and train the reinforcement learning algorithm. In the experiment, we adopt the default parameter settings provided by the framework. The operating environment is Windows 10 with Inter(R) Core(TM) i7-7300HQ CPU (2.50GHz/16.00 GB RAM).

4.1. Disassembly instances

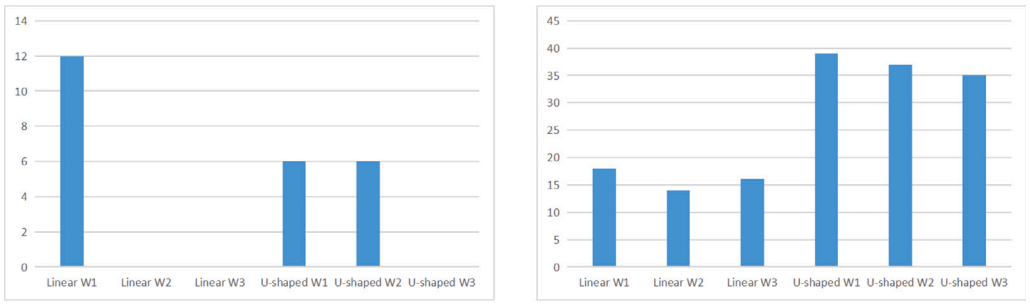
We select TV, cell phones [42], and Tesla batteries [43] as the products for disassembly. Among them, TV and cell phones are used as small-scale disassembly products, while Tesla batteries are large-scale products. Assume that the number of skills needed to complete the disassembly tasks for each product is 5, as listed in Table 5. The related parameters such as disassembly skills possessed by workers and disassembly task profit are generated by interval uniform distribution. Each disassembly instance is a combination of different products. Instance information is listed in Table 6.

4.2. Evaluation indicators

We used the following performance indicators in this study: *Steady timesteps*: A measure of the number of training timesteps required for an algorithm to reach a steady state.

Disassembly profit: The profit of the product under the current disassembly scheme.

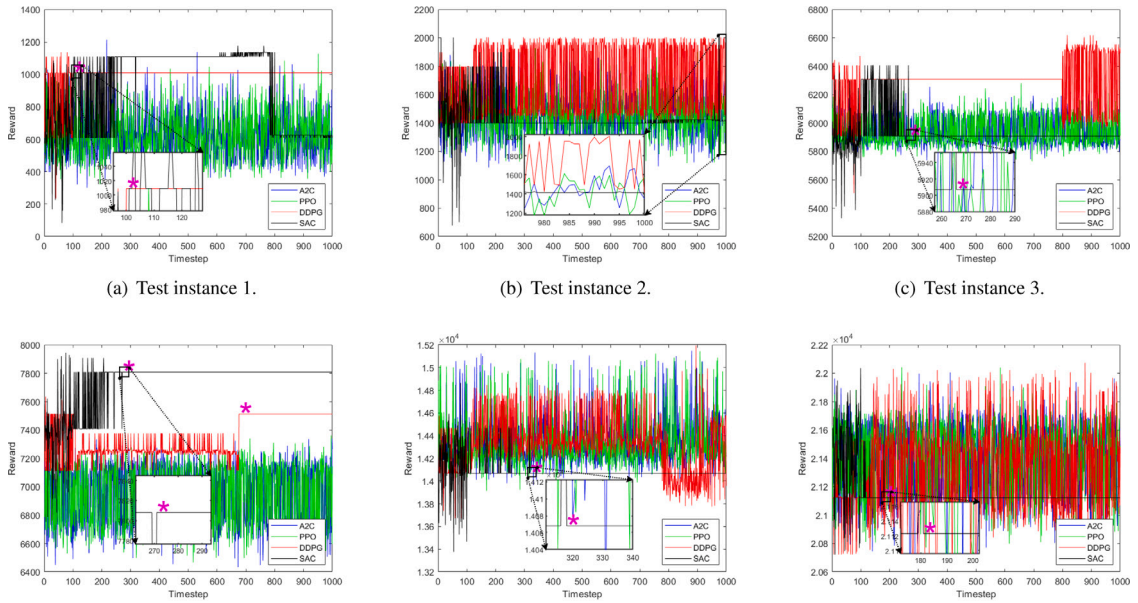
Runtime: Running time of the algorithm after completing the specified number of training timesteps.



(a) Small-scale tasks distribution in configuration 1

(b) Large-scale tasks distribution in configuration 1

Fig. 4. Different disassembly scales distribution in configuration 1.



(a) Test instance 1.

(b) Test instance 2.

(c) Test instance 3.

(d) Test instance 4.

(e) Test instance 5.

(f) Test instance 6.

Fig. 5. Trend of reward throughout training.

Table 6
Test instances.

Instance ID	Number of products			Number of tasks
	TV	Cellphone	Tesla battery	
1	1	1	0	24
2	1	2	0	36
3	0	1	1	49
4	1	1	1	61
5	1	2	2	110
6	2	2	3	159

Table 7
Workstation skill configuration.

Config. ID	Number of skills					
	Linear			U-shaped		
	W1	W2	W3	W1	W2	W3
1	1	0	0	1	0	0
2	2	0	0	2	0	0
3	3	0	0	3	0	0
4	4	0	0	4	0	0
5	5	0	0	5	0	0

4.3. Analysis of experimental results

Different employees may have different skills. Table 7 shows the number of skills that workers in each workstation have in five different skill configurations, in which Wi means workstation i, i = 1, 2, 3. In a different configuration, workers on one of the workstations are designed to have a different number of skills.

Table 8 shows the result of different configurations. Figs. 4(a) – 4(b) show the distribution of tasks on workstations at different disassembly scales of Configuration 1. In Configuration 1, when the number of disassembly tasks is small, the tasks on the linear disassembly line tend to be assigned to workers with more skills, and no additional training is given to workers on other workstations. When the disassembly task gradually increases, workers will be trained to learn skills, to shorten the total operation time of the disassembly line and improve the disassembly profit. U-shaped disassembly line workers can operate on both sides of the workstation, and additional skill training for workers is conducive to shortening the total operation time of the disassembly line and improving the disassembly profit. The assignments on the U-shaped disassembly line are approximately evenly distributed. In Configuration 5, when the number of disassembly tasks is small, it is not entirely assigned to one workstation. The staff will be trained in corresponding skills, and part of the task will be assigned to the newly trained workers to complete, to shorten the operation time of the disassembly line and improve the profit. When there are more disassembly tasks, the

Table 8
Results of different configurations.

Config. ID	Instance ID	The number of tasks on each workstation					
		Linear			U-shaped		
		W1	W2	W3	W1	W2	W3
1	1	12	0	0	6	6	0
	2	14	10	0	5	3	4
	3	37	0	0	5	3	4
	4	37	0	0	9	8	7
	5	17	18	0	26	25	23
	6	18	14	16	39	37	35
2	1	12	0	0	6	6	0
	2	5	2	5	13	11	0
	3	37	0	0	5	3	4
	4	37	0	0	9	7	8
	5	9	15	12	26	23	25
	6	17	16	15	39	35	37
3	1	6	0	6	6	0	6
	2	9	3	0	13	0	11
	3	37	0	0	8	4	0
	4	10	4	10	24	13	0
	5	26	22	26	12	12	11
	6	30	27	29	27	23	23
4	1	6	0	6	6	0	6
	2	5	7	0	13	0	11
	3	12	0	0	24	0	13
	4	12	0	12	24	0	13
	5	26	23	25	12	13	11
	6	34	32	32	23	18	20
5	1	6	0	6	6	0	6
	2	5	7	0	13	0	11
	3	12	0	0	24	0	13
	4	12	0	12	24	0	13
	5	26	23	25	12	13	11
	6	34	32	32	23	18	20

number of workstations used increases, and the tasks assigned to each workstation are approximately evenly distributed.

In the process of product disassembly, we blindly assign tasks to workers with more skills, which may lead to a long-running time of individual workstations and the rest of the workstations in an idle state. The training of multi-skilled workers can effectively reduce the operation time of the disassembly line, thus increasing the profit. In the actual disassembly process, reasonable training for employees is conducive to improving the efficiency of disassembly.

The model of each case is trained 1000 timesteps. Fig. 5 shows the results of the different case model training runs. In Fig. 5(a), the DDPG algorithm reaches a steady state in 100 rounds, while the SAC algorithm has small fluctuations. The A2C algorithm and PPO algorithm are still in a volatile state and cannot stabilize in 1000 training sessions. In Fig. 5(b), SAC has small fluctuations, while other algorithms have large fluctuations. In Fig. 5(c), the SAC algorithm has reached a steady state in 266 rounds, while other algorithms fail to stabilize and have a large degree of fluctuation. In Fig. 5(d), both SAC and DDPG are in a steady state, while other algorithms have fluctuations, and SAC reaches a steady state in 271 rounds, DDPG reaches a steady state in 676 rounds. The reward of SAC in the steady state is higher than DDPG. In the case of large-scale disassembly, as shown in Figs. 5(e) – 5(f). In Fig. 5(e) the SAC algorithm reaches a steady state in 318 rounds, and in Fig. 5(f) SAC algorithm reaches a steady state in 181 rounds, while the other algorithms fluctuate greatly and fail to reach a steady state.

As shown in Eq. (26), in the SAC algorithm α is the temperature parameter that determines the correlation between entropy and reward. Entropy randomizes the strategy so that each action output has as uniform a probability as possible, rather than being concentrated in one place to encourage exploration. In some states, it may be optimal to have multiple actions, so the selection probability of these actions is the same, which can improve the learning speed. Therefore, the convergence speed of the SAC algorithm is always the fastest in different cases. After the training of the DDPG algorithm, a deterministic policy is obtained, which only considers one optimal action for one state. In small-scale cases, the DDPG algorithm is as good as SAC. In large-scale cases, the DDPG algorithm's reward value fluctuates greatly due to its deterministic strategy, and it is difficult to converge within a limited number of training times. The SAC algorithm is superior to PPO and A2C in HDLBP.

Table 9 shows that the model is re-solved after 1000 timesteps for each algorithm, and the reward is shown in the table. Among them, the value obtained by SAC algorithm is the closest to the value in CPLEX. The slight disparity observed between the SAC algorithm and CPLEX can predominantly be attributed to the disassembly sequences employed for product decomposition. With the existence of numerous potential disassembly sequences for each product, it becomes impractical to exhaustively evaluate the optimal allocation for every conceivable sequence within the given time constraints. Consequently, a feasible disassembly sequence is selected instead, leading to a discrepancy in maximal rewords. In forthcoming research endeavors, we intend to adopt a multi-intelligence approach to systematically explore each viable disassembly sequence, thereby narrowing the gap towards attaining an optimal solution. Table 10 shows the running times of each algorithm. The A2C algorithm exhibits faster execution times due to its employment of a single actor network and a single critic network. Conversely, the SAC algorithm consists of one actor network and four Q critic networks, resulting in comparatively slower runtime performance. As a whole, SAC has great advantages in HDLBP application.

5. Conclusions

This study represents the first investigation into the workload balancing problem in hybrid disassembly lines that incorporate multi-skilled disassembly workers. The optimization goal of the problem

Table 9
The profit after training.

Instance ID	A2C	DDPG	PPO	SAC	CPLEX
1	638	1009	396	1403	1518
2	1534	1983	1498	2098	2429
3	5871	6555	5836	6600	6864
4	6560	7513	7169	7808	8135
5	14 221	14 316	14 197	14 754	15 507
6	21 688	21 939	21 510	22 063	23 040

Table 10
Computation time of algorithms.

Instance ID	Computation time (seconds)			
	A2C	DDPG	PPO	SAC
1	6.167	12.633	12.357	19.614
2	6.292	12.756	12.922	19.482
3	14.045	12.479	27.510	20.547
4	14.686	13.742	28.696	20.780
5	15.696	13.976	30.454	21.254
6	16.898	14.238	32.931	22.641

is to achieve maximal disassembly profit, which takes into account the cost of disassembly skill training. To solve this problem effectively, discrete action space and continuous action space are designed to establish the interaction environment between the reinforcement learning agent and the hybrid disassembly line for different algorithm characteristics. The use of interval mapping in continuous action spaces converts continuous actions into discrete actions, which allows the use of continuous action space algorithms to solve discrete problems. Simulation experiments are conducted on disassembly cases of varying scales, including small, medium, and large-scale instances, to validate the accuracy of the model and assess the performance of the employed SAC algorithm. In examining the training trajectories of each algorithm, it is observed that the SAC algorithm exhibits a higher degree of stability, characterized by smoother and more consistent convergence patterns. Furthermore, the optimal solution attained through the SAC algorithm displays closer proximity to the solution obtained via CPLEX. The experimental results confirmed that SAC performs better than other well-known reinforcement learning algorithms in solving HDLBP.

Future work includes: (1) We plan to incorporate external environmental factors (e.g., equipment breakdowns, supply chain disruptions) and develop appropriate strategies to deal with external uncertainties in future research. (2) Further research on multi-agent SAC algorithm or multiobjective SAC algorithm for solving HDLBP. (3) We will consider balancing multiple conflicting goals in a dynamic environment [44–46] (4) We will further validate in more real-world scenarios that the proposed methodology will strengthen the validity and applicability of our research.

CRediT authorship contribution statement

Jiacun Wang: Conceptualization, Writing – review & editing. **GuiPeng Xi:** Software, Writing – original draft. **XiWang Guo:** Conceptualization, Investigation. **Shixin Liu:** Conceptualization, Investigation. **ShuJin Qin:** Investigation. **Henry Han:** Validation.

Declaration of competing interest

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

Data availability

No data was used for the research described in the article.

References

[1] X. Guo, Z. Zhang, L. Qi, S. Liu, Y. Tang, Z. Zhao, Stochastic hybrid discrete Grey Wolf Optimizer for multi-objective disassembly sequencing and line balancing planning in disassembling multiple products, IEEE Trans. Autom. Sci. Eng. (2021).

[2] Z. Zhang, X. Guo, M. Zhou, S. Liu, L. Qi, Multi-objective discrete Grey Wolf Optimizer for solving stochastic multi-objective disassembly sequencing and line balancing problem, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics, SMC, IEEE, 2020, pp. 682–687.

[3] X. Guo, M. Zhou, S. Liu, L. Qi, Lexicographic multi-objective scatter search for the optimization of sequence-dependent selective disassembly subject to multiresource constraints, IEEE Trans. Cybern. 50 (7) (2019) 3307–3317.

[4] X. Guo, S. Liu, M. Zhou, G. Tian, Dual-objective program and scatter search for the optimization of disassembly sequences subject to multiresource constraints, IEEE Trans. Autom. Sci. Eng. 15 (3) (2017) 1091–1103.

[5] Z. Li, I. Kucukkoc, Z. Zhang, Iterated local search method and mathematical model for sequence-dependent U-shaped disassembly line balancing problem, Comput. Ind. Eng. 137 (2019) 106056.

[6] X. Guo, T. Wei, J. Wang, S. Liu, S. Ain, L. Qi, Multiobjective U-shaped disassembly line balancing problem considering human fatigue index and an efficient solution, IEEE Trans. Comput. Soc. Syst. (2022).

[7] J. Liang, S. Guo, B. Du, Y. Li, J. Guo, Z. Yang, S. Pang, Minimizing energy consumption in multi-objective two-sided disassembly line balancing problem with complex execution constraints using dual-individual simulated annealing algorithm, J. Clean. Prod. 284 (2021) 125418.

[8] S. Hezer, Y. Kara, A network-based shortest route model for parallel disassembly line balancing problem, Int. J. Prod. Res. 53 (6) (2015) 1849–1865.

[9] L. Zhu, Z. Zhang, C. Guan, Multi-objective partial parallel disassembly line balancing problem using hybrid group neighborhood search algorithm, J. Manuf. Syst. 56 (2020) 252–269.

[10] K. Wang, X. Li, L. Gao, Modeling and optimization of multi-objective partial disassembly line balancing problem considering hazard and profit, J. Clean. Prod. 211 (2019) 115–133.

[11] Z. Li, M.N. Janardhanan, Modelling and solving profit-oriented U-shaped partial disassembly line balancing problem, Expert Syst. Appl. 183 (2021) 115431.

[12] S. Zhang, P. Liu, X. Guo, J. Wang, S. Qin, Y. Tang, An improved tabu search algorithm for multi-robot hybrid disassembly line balancing problems, in: 2022 International Conference on Cyber-Physical Social Intelligence, ICCSI, 2022, pp. 315–320, http://dx.doi.org/10.1109/ICCSI55536.2022.9970618.

[13] J. Zhu, Y. Han, X. Guo, J. Wang, S. Qin, L. Qi, J. Zhao, Union variable neighborhood descent algorithm for multi-product hybrid disassembly line balancing problem considering workstation resource configuration, in: 2022 IEEE International Conference on Systems, Man, and Cybernetics, SMC, 2022, pp. 860–865, http://dx.doi.org/10.1109/SMC53654.2022.9945108.

[14] T. Paksoy, A. Güngör, E. Özceylan, A. Hancilar, Mixed model disassembly line balancing problem with fuzzy goals, Int. J. Prod. Res. 51 (20) (2013) 6082–6096.

[15] X. Guo, M. Zhou, S. Liu, L. Qi, Multiresource-constrained selective disassembly with maximal profit and minimal energy consumption, IEEE Trans. Autom. Sci. Eng. 18 (2) (2020) 804–816.

[16] L. Zhu, Z. Zhang, Y. Wang, A Pareto firefly algorithm for multi-objective disassembly line balancing problems with hazard evaluation, Int. J. Prod. Res. 56 (24) (2018) 7354–7374.

[17] S. Xu, X. Guo, S. Liu, L. Qi, S. Qin, Z. Zhao, Y. Tang, Multi-objective optimizer with collaborative resource allocation strategy for U-shaped stochastic disassembly line balancing problem, in: 2021 IEEE International Conference on Systems, Man, and Cybernetics, SMC, IEEE, 2021, pp. 2316–2321.

[18] X. Guo, Z. Zhang, L. Qi, S. Liu, Y. Tang, Z. Zhao, Stochastic hybrid discrete Grey Wolf Optimizer for multi-objective disassembly sequencing and line balancing planning in disassembling multiple products, IEEE Trans. Autom. Sci. Eng. (2021).

[19] S.M. McGovern, S.M. Gupta, A balancing method and genetic algorithm for disassembly line balancing, European J. Oper. Res. 179 (3) (2007) 692–708.

[20] Y. Qiu, L. Wang, X. Xu, X. Fang, P.M. Pardalos, A variable neighborhood search heuristic algorithm for production routing problems, Appl. Soft Comput. 66 (2018) 311–318.

[21] N. Mazyavkina, S. Sviridov, S. Ivanov, E. Burnaev, Reinforcement learning for combinatorial optimization: A survey, Comput. Oper. Res. 134 (2021) 105400.

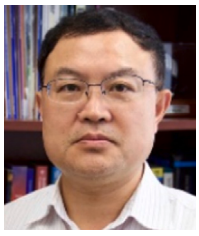
[22] Q. Cappart, T. Moisan, L.-M. Rousseau, I. Prémont-Schwarz, A.A. Cire, Combining reinforcement learning and constraint programming for combinatorial optimization, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, no. 5, 2021, pp. 3677–3687.

[23] S. Gu, Y. Yang, A deep learning algorithm for the max-cut problem based on pointer network structure with supervised learning and reinforcement learning strategies, Mathematics 8 (2) (2020) 298.

[24] J. Song, R. Lanka, Y. Yue, M. Ono, Co-training for policy learning, in: Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 1191–1201.

[25] W. Zhang, J. Wang, F. Lan, Dynamic hand gesture recognition based on short-term sampling neural networks, IEEE/CAA J. Autom. Sin. 8 (1) (2021) 110–120.

[26] B. Hu, J. Wang, Deep learning based hand gesture recognition and UAV flight controls, Int. J. Autom. Comput. 17 (1) (2020) 17–29.
[27] X. Heng, Z. Wang, J. Wang, Human activity recognition based on transformed accelerometer data from a mobile phone, Int. J. Commun. Syst. 17 (1) (2016) 17–29.
[28] X. Zhao, C. Li, Y. Tang, J. Cui, Reinforcement learning-based selective disassembly sequence planning for the end-of-life products with structure uncertainty, IEEE Robot. Autom. Lett. 6 (4) (2021) 7807–7814.
[29] G. Xi, J. Wang, X. Guo, S. Liu, S. Qin, L. Qi, Hybrid disassembly line optimization with reinforcement learning, in: 2023 32nd Wireless and Optical Communications Conference, WOCC, IEEE, 2023, pp. 1–5.
[30] X. Guo, M. Zhou, A. Abusorrah, F. Alsokhiry, K. Sedraoui, Disassembly sequence planning: A survey, IEEE/CAA J. Autom. Sin. 8 (7) (2020) 1308–1324.
[31] R.J. Riggs, O. Battaïa, S.J. Hu, Disassembly line balancing under high variety of end of life states using a joint precedence graph approach, J. Manuf. Syst. 37 (2015) 638–648.
[32] J. Wang, Patient flow modeling and optimal staffing for emergency departments: A Petri net approach, IEEE Trans. Comput. Soc. Syst. (2022).
[33] Z. Zhao, S. Liu, M. Zhou, D. You, X. Guo, Heuristic scheduling of batch production processes based on petri nets and iterated greedy algorithms, IEEE Trans. Autom. Sci. Eng. (2020).
[34] J. Wang, W. Tepfenhart, Formal Methods in Computer Science, Chapman and Hall/CRC, 2019.
[35] J. Wang, M. Zhou, Y. Deng, Throughput analysis of discrete event systems based on stochastic Petri nets, Int. J. Intell. Control Syst. 3 (3) (1999) 343–358.
[36] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.
[37] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al., Soft actor-critic algorithms and applications, 2018, arXiv preprint arXiv:1812.05905.
[38] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, S. Russell, Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, no. 01, 2019, pp. 4213–4220.
[39] Y. Zheng, X. Li, L. Xu, Balance control for the first-order inverted pendulum based on the advantage actor-critic algorithm, Int. J. Control Autom. Syst. 18 (12) (2020) 3093–3100.
[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.
[41] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, Stable-Baselines3: Reliable reinforcement learning implementations, J. Mach. Learn. Res. (ISSN: 1532-4435) 22 (1) (2021).
[42] K. Igarashi, T. Yamada, M. Inoue, 2-stage optimal design and analysis for disassembly system with environmental and economic parts selection using the recyclability evaluation method, Ind. Eng. Manag. Syst. 13 (1) (2014) 52–66.
[43] T. Wu, Z. Zhang, T. Yin, Y. Zhang, Multi-objective optimisation for cell-level disassembly of waste power battery modules in human-machine hybrid mode, Waste Manag. 144 (2022) 513–526.
[44] H. Li, Z. Wang, C. Lan, P. Wu, N. Zeng, A novel dynamic multiobjective optimization algorithm with non-inductive transfer learning based on multi-strategy adaptive selection, IEEE Trans. Neural Netw. Learn. Syst. (2023).
[45] N. Zeng, H. Li, Z. Wang, W. Liu, S. Liu, F.E. Alsaadi, X. Liu, Deep-reinforcement-learning-based images segmentation for quantitative analysis of gold immunochromatographic strip, Neurocomputing 425 (2021) 173–180.
[46] H. Li, Z. Wang, C. Lan, P. Wu, N. Zeng, A novel dynamic multiobjective optimization algorithm with hierarchical response system, IEEE Trans. Comput. Soc. Syst. (2023).



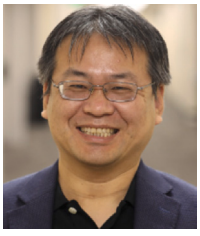
Jiacun Wang (SM'00) received the Ph.D. degree in computer engineering from Nanjing University of Science and Technology (NUST), China, in 1991. He is currently a Professor of software engineering at Monmouth University, West Long Branch, New Jersey, USA. His research interests include software engineering, discrete event systems, formal methods, machine learning, and real-time distributed systems. He authored 4 books and published over 200 research papers in journals and conferences. Dr. Wang is an Associate Editor of IEEE Transactions on Systems, Man and Cybernetics: Systems and IEEE/CAA Journal of Automatica Sinica. He has served as general chair, program chair, and special sessions chair or program committee member for many international conferences.

GuiPeng Xi received his degree in statistics from Yancheng Teachers University in Yancheng, China, in 2021. He is currently a graduate student at the School of Computer and Communication Engineering, Liaoning Petrochemical University. His research interest covers reinforcement learning algorithms.

XiWang Guo received his B.S. degree in Computer Science and Technology from Shenyang Institute of Engineering, Shenyang, China, in 2006, M.S. degree in Aeronautics and Astronautics Manufacturing Engineering. from Shenyang Aerospace University, Shenyang, China, in 2009, Ph. D. degree in System Engineering from Northeastern University, Shenyang, China, in 2015. He is currently an associate professor of the College of Computer and Communication Engineering at Liaoning Petrochemical University. From 2016 to 2018, he was a visiting scholar of Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. He has authored 60+ technical papers in journals and conference proceedings, including IEEE Transactions on Cybernetics, IEEE Transactions on System, Man and Cybernetics: Systems, IEEE Transactions on Intelligent Transportation Systems, and IEEE/CAA Journal of Automatica Sinica. His current research interests include Petri nets, remanufacturing, recycling and reuse of automotive, intelligent optimization algorithm.

Shixin Liu received his B.S. degree in Mechanical Engineering from Southwest Jiaotong University, Sichuan, China in 1990, M.S. degree and Ph. D. degree in Systems Engineering from Northeastern University, Shenyang, China in 1993, and 2000. He is currently a Professor of the College of Information Science and Engineering, Northeastern University, Shenyang, China. His research interests are in intelligent optimization algorithm, project management, and the theory and method of planning and scheduling. He has over 100 publications including 1 book.

ShuJin Qin received his B.S. degree in Information and Computing Science from Tianjin Polytechnic University, Tianjin, China in 2008, M.S. degree in Operational Research and Cybernetics from University of Science and Technology Liaoning, Anshan, China in 2011, and Ph. D. degree in System Engineering from Northeastern University, Shenyang, China in 2019. He joined Shangqiu Normal University, Shangqiu, China in 2019, and is now a lecturer of logistics management. His current research interests include large-scaled integer programming, cutting stock problem, vehicle routing problem, traveling repairman problem and intelligent optimization algorithm.



Henry Han received the Ph.D. degree from the University of Iowa, in 2004. He is currently a Professor with the Department of Engineering and Computer Science, Baylor University, Waco, Texas, USA. He is also the Director of the Laboratory of Big Data and Analytics. He was the Founding Director of Fordham's master program in cybersecurity besides the Department Associate Chair at Lincoln center campus. He has published more than 80 articles in leading journals and conferences in data science, bioinformatics, digital health, big data, fintech, machine learning, and data analytics fields. He has been supervising about a total of 60 Ph.D., master's, and bachelor's students, since 2005. His current research interests include data science, big data, digital health, AI, fintech, and cybersecurity. His research has been supported by NSF, NIH, and research contracts from industry.