Engineering Applications of Artificial Intelligence 150 (2025) 110588

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Dynamic scheduling in flexible and hybrid disassembly systems with manual and automated workstations using reward-shaping enhanced reinforcement learning

Jinlong Wang, Qihuiyang Liang, Min Li, Zelin Qu, Yuanyuan Zhang *

School of Information and Control Engineering of Qingdao University of Technology, 777 Jialingjiang East Road, Huangdao District, Qingdao, China

ARTICLE INFO

Keywords:
Sustainable production
Hybrid disassembly system
Reinforcement learning
Reward-shaping
Dynamic scheduling

ABSTRACT

As e-waste grows at an alarming rate, efficient disassembly systems have become crucial for sustainable production practices. Existing disassembly systems rely heavily on fixed automation and limited manual intervention, making it challenging to adapt to dynamic issues such as workstation failures and system bottlenecks, leading to inefficiencies and suboptimal resource allocation. To address these issues, a hybrid disassembly system is developed that integrates manual and automated workstations, allowing for the flexible variation of manual resources as needed to optimize the disassembly process, with a focus on reducing time and maximizing profit. Through the proposal of a Proximal Policy Optimization (PPO) algorithm enhanced with Reward-Shaping, the research effectively tackles key challenges of uncertainty and dynamic conditions in disassembly systems, including workstation failures and system bottlenecks. These issues are explored through a refrigerator disassembly simulation model. The results demonstrate that the PPO algorithm significantly outperforms traditional rule-based methods and two other reinforcement learning techniques in managing complex dynamic scheduling and resource allocation tasks, offering greater efficiency and flexibility. These findings contribute to the advancement of automated disassembly processes and their integration into modern industrial systems.

1. Instruction

As global development progresses, e-waste, especially waste from used electronic equipment that is difficult to destroy or dispose of, has made remanufacturing a critical global issue. The Global E-Waste Monitor 2023 report highlights the significant and accelerating growth in global electronic waste generation, with 62 million tons of e-waste generated worldwide in 2022—a sharp increase of 82 % compared to 2010. However, only about 22.3 % of e-waste in 2022 was properly collected and recycled. To address these recycling industry challenges and regulatory pressures, remanufacturing has emerged as a crucial closed-loop cycle that involves the recovery, repair, and reuse of high-value components (Glöser-Chahoud et al., 2021), (Ullah et al., 2021). Over the past decade, the potential of remanufacturing has shown substantial appeal; however, the complexity and uncertainty of managing returned products make remanufacturing an extremely challenging endeavor (Rizova et al., 2020).

The optimization of the disassembly line, as the initial step in remanufacturing, improves product utilization from a manufacturing standpoint, reduces remanufacturing costs, and effectively lowers the carbon emissions associated with product disassembly (Tian et al., 2018). Therefore, many governments are encouraging Original Equipment Manufacturers (OENs) to design eco-friendly products and engage in sustainable practices to mitigate their products' environmental impact (Yuan et al., 2024).

A substantial body of research in disassembly has focused on optimizing disassembly sequence planning (DSP) and disassembly line balancing (DLBP), addressing cost, time, and efficiency goals. DSP primarily seeks optimal disassembly sequences by minimizing operations through methods such as Genetic Algorithm (GA) (Sampson, 1976), Artificial Bee Colony (ABC) (Karaboga, 2005), Ant Colony Optimization (ACO) (Colorni et al., 1991), and Particle Swarm Optimization (PSO) (Eberhart and Kennedy, 1995). The primary goal is to find an optimal disassembly sequence, deemed feasible/optimal/near-optimal if it

This article is part of a special issue entitled: Meta for Sust Mfg - EGE: Anand Kulkarni published in Engineering Applications of Artificial Intelligence.

* Corresponding author.
E-mail addresses: wangjinlong@qut.edu.cn (J. Wang), l75369852@163.com (Q. Liang), liminmin2023@163.com (M. Li), qzlin162@163.com (Z. Qu), yyzhang1217@163.com (Y. Zhang).

https://doi.org/10.1016/j.engappai.2025.110588
Received 31 October 2024; Received in revised form 3 March 2025; Accepted 13 March 2025
Available online 21 March 2025

meets specific objectives (Chand and Ravi, 2023). Similarly, DLBP aims to allocate tasks effectively along disassembly lines to meet rising disassembly demands and improve cost-effectiveness (Hu et al., 2023). Meta-heuristic approaches like ABC (Wang et al., 2022), Simulated Annealing (SA) (Hu et al., 2023; Wang et al., 2021), and Ant-Lion Optimization (ALO) (Liang et al., 2024) have proven effective for multi-product line environment.

However, as disassembly environments become increasingly complex, dynamic scheduling has emerged as a crucial challenge. Traditional meta-heuristic algorithms, while capable of high-quality solutions, are typically time-intensive and lack the responsiveness needed to handle unpredictable events or varying product conditions in real time (Luo, 2020), (Gao et al., 2020). It is well recognized that in any reverse logistics system handling the collection and dismantling of discarded products, the variability in product conditions introduces considerable uncertainty into the dismantling process, increasing the likelihood of errors (Altekin and Akkan, 2012). Reinforcement Learning (RL) has been introduced as a promising solution to this limitation, with its ability to autonomously learn optimal strategies through continuous environmental interaction. In DSP, for example, RL can adaptively explore efficient disassembly paths based on product structures and historical data (Allagui, 2023), (Zhao et al., 2021).

Reinforcement learning also offers significant potential in disassembly scheduling, as it can accommodate real-time adjustments in response to resource conflicts, time delays, and variable product conditions (Colledani and Battaïa, 2016; Liu, 2022), (Zhang, 2023). Its adaptive learning approach enables more flexible and efficient task allocation, particularly in multi-task and multi-objective disassembly systems (Wang et al., 2024a). By leveraging accumulated data over time, RL further maximizes long-term scheduling gains, making it highly suitable for the complex requirements of modern disassembly operations (Schulman et al., 2017). Therefore, we have decided to explore the application of reinforcement learning in scheduling, aiming to address these dynamic and complex challenges effectively.

In traditional automated production, equipment follows established procedures to complete tasks. However, current automated systems are not yet capable of handling the inherent uncertainty in waste components, hindering the development of fully automated disassembly processes (Junior and Filho, 2012; Poschmann et al., 2020). Research suggests that hybrid production systems, combining manual and automated workstations, offer the most practical solution for effective disassembly (Kim et al., 2007). Manual operators remain essential due to their flexibility in handling tasks that involve real-time problem-solving, such as reallocating faulty parts to adaptable manual workstations when needed (Han et al., 2023), (Chen and Vongbunyong, 2015). Marco Wurster et al. (2022) emphasize that loosely connected workstations suit disassembly's decentralized material flow, predicting an increase in hybrid systems integrating both manual and automated stations. Poschmann et al., (2020) foresee automated disassembly advancing to industry-standard technology within the next decade. Reinforcement learning and hybrid systems with loosely connected workstations are being increasingly applied for disassembly line optimization, as demonstrated by Muyue Han et al.'s real-time scheduling method (Han et al., 2023) and Jiacun Wang et al.'s hybrid balancing approach that combines linear and U-shaped disassembly lines (Wang et al., 2024b).

In terms of resource utilization, relying solely on automated stations is impractical due to the risk of station failure, necessitating backup from manual stations (Wurster et al., 2022). Additionally, existing research on reinforcement learning for disassembly scheduling is limited. To address this gap, our study explores a hybrid disassembly model that integrates manual and automated workstations. We represent manual and automated resources as two workstation types, leveraging the flexibility of manual stations and the cost-effectiveness of automated stations. This configuration enables a simulated production environment with loosely connected workstations, fostering adaptability. A reinforcement learning model enhanced with Reward-Shaping is further developed to optimize environmental interaction and dynamic learning, achieving notable improvements in efficiency and resilience within disassembly processes.

1.1. Main contributions

1. To validate the impact of dynamically adjustable manual workstations on disassembly system performance, we established a highly adaptable hybrid disassembly system for the dynamic scheduling problem of the disassembly line. This system integrates manual and automated workstations in a discrete layout, allowing for flexible adjustment of manual stations to maximize profit and minimize time.
2. An enhanced proximal policy optimization (PPO) algorithm with Reward-Shaping is introduced to address the challenge of sparse rewards in reinforcement learning, making it highly suited for the dynamic and flexible environment of hybrid disassembly lines.
3. Simulation experiments with a refrigerator disassembly case demonstrate that the PPO algorithm significantly outperforms random and rule-based methods in dynamic scheduling, boosting both efficiency and adaptability in hybrid disassembly systems.

The rest of this paper is organized as follows. Section 2 describes the problem of HDLBP. Section 3 proposes the PPO algorithm enhanced by Reward-Shaping. Section 4 presents the experiments and performance of the PPO algorithm. Section 5 concludes this paper and discusses the future research direction.

2. System model and problem description

In our approach, the disassembly sequence of end-of-life appliances is predetermined, inspired by Marco Wurster et al. (2022). Our focus is on the real-time scheduling of the system (hereafter referred to as Dynamic scheduling), aiming to achieve system balance and enhance profitability. This approach accounts for common uncertainties in disassembly, such as variable disassembly times, the likelihood of automated disassembly failures, and potential system bottlenecks. To balance the cost-efficiency of automated stations with the greater adaptability of manual stations, products are redirected to a manual station with equivalent functionality when an automated disassembly failure occurs. This approach, which has also been referenced in (Wurster et al., 2022), ensures operational continuity and enhances system resilience. In cases of system blockage, the flexibility of manual stations is leveraged by increasing the number of these stations to alleviate bottlenecks, as illustrated in Fig. 1. Given that automated workstations typically involve more complex deployment processes, their numbers generally cannot be adjusted in real time. We grouped manual and automated workstations capable of processing the same type of product into a single workstation group. Global scheduling, managed by the selected reinforcement learning method, prioritizes long-term benefits, achieving better cost-effectiveness compared to conventional single-rule scheduling.

The discrete event simulation model is deployed as a digital twin, simulating disassembly and logistics processes triggered and controlled by a single decision-making agent. Based on this foundation, we further developed constraints to refine the system model, tailoring it to the functional characteristics of automated and manual stations and aligning with the specific requirements of disassembly tasks.

2.1. System model

The system based on the designs that distinguish between manual and automated stations (Wurster et al., 2022). Additionally, it takes into account the variability of manual resources and is able to handle scheduling problems with greater problem depth and complexity to prevent blocking situations. The entire disassembly system consists of loosely coupled disassembly workstations, and the following hypotheses

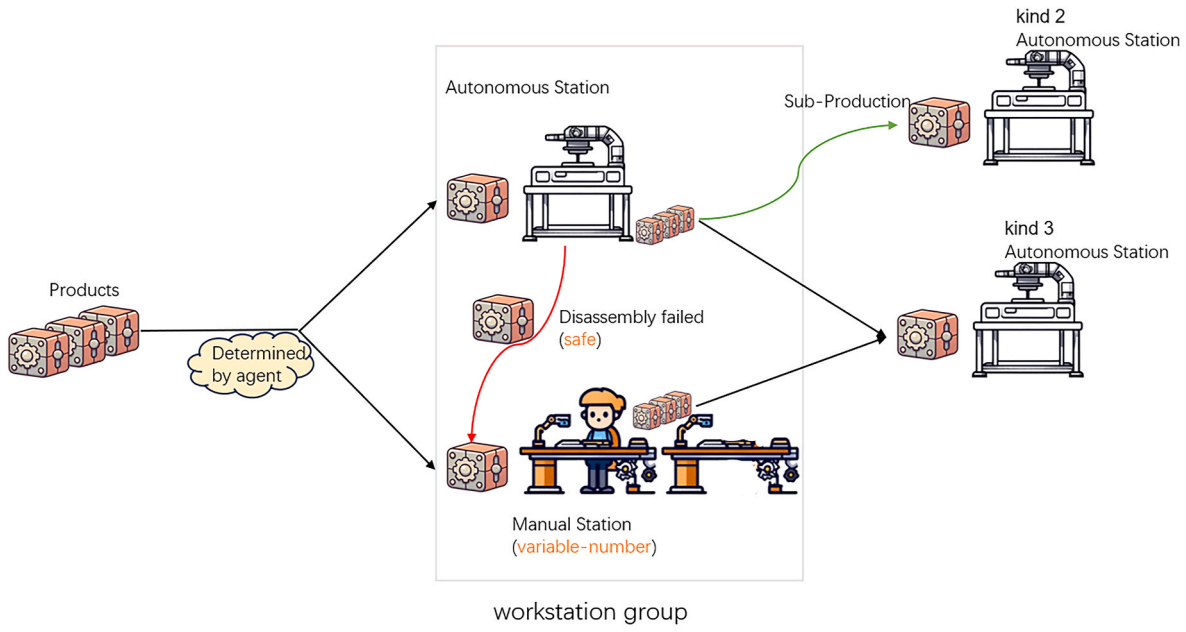

Fig. 1. Illustration of material flow and state change decisions, including operation failure handling, sub-order generation, and agent-based decisions for workstation selection and adjustment of the number of workstations (adapted from Marco Wurster et al. (Wurster et al., 2022), and added the concept of workstation groups and variable manual workstations).

are proposed.

1. Assuming that the disassembly order of product components is predetermined, the system considers only the selection of workstations and the assignment of components to the target workstations.
2. A hazardous component contains a hazardous gas or substance, the dismantling of which can only be handled by a machine to prevent hazards to workers.
3. Non-hazardous components can be handled by both automated and manual workstations, but automated workstations have a probability of failure.
4. Manual workstations can replace the work of all automated workstations handling non-hazardous components at a higher cost than automated workstations, but the number can be dynamically increased or decreased.
5. The default workstation can handle a portion of a product's neighboring components, rather than just a specific one, to simplify the disassembly process.
6. The system prioritizes disassembling appliances to completion, rather than disassembling only a portion of the important components. Leaving the remaining product components in the system tends to cause blockages.
7. Thinking about the ideal case, ignore the time required for manual station increment and decrement due to the low increment and decrement time of the manual station relative to disassembly.

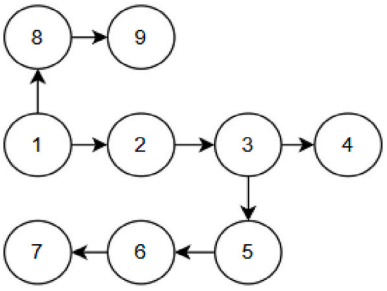

Fig. 2. Component priority graph.

	C1	C2	C3	C4	C5	C6	C7	C8	C9
C1	0	1	0	0	0	0	0	1	0
C2	0	0	1	0	0	0	0	0	0
C3	0	0	0	1	1	0	0	0	0
C4	0	0	0	0	0	0	0	0	0
C5	0	0	0	0	0	1	0	0	0
C6	0	0	0	0	0	0	0	0	0
C7	0	0	0	0	0	0	0	0	0
C8	0	0	0	0	0	0	0	0	1
C9	0	0	0	0	0	0	0	0	0

Fig. 3. Component priority matrix.

Components are streamlined and categorized through a disassembly priority map and an analysis of component relevance relationships. Given the presence of relatively fragmented parts, such as screws, the scheduling volume, and associated costs would be significantly higher if each workstation were limited to a single function. To address this, we group physically adjacent components to create a simplified disassembly priority hierarchy. This revised hierarchy allows for an optimized order of disassembly based on the new priority relationships.

Below is a sample priority graph Fig. 2 along with its corresponding priority matrix Fig. 3. A priority graph is a directed acyclic graph (DAG) used to represent the precedence relationships between tasks or components. The nodes in the graph represent tasks or components, and the directed edges indicate that one task must be completed before another. The priority matrix is the matrix representation of the priority graph, where rows and columns correspond to the nodes in the graph. The matrix element $M[i][j]$ has a value of 1 if there is a precedence relationship from node i to node j, and a value of 0 if there is no direct dependency between the two. The priority graph and its matrix allow for intuitive analysis and optimization of task sequences, enhancing system efficiency and reliability.

As shown in Table 1, we assume that the system has a starting point W_{start} and an ending point W_{end}. The number of workstations performing the disassembly task N consists of j automated stations W_j^A, k manual stations W_k^M, and the number of manual stations of each type can be

Table 1
Table of notations.

Notations	Description
N	Total number of workstations in the system.
W_j^A	The j th automated workstation in the system.
W_k^M	The k th automated workstation in the system.
D_N	A 0–1 array indicates whether an automated workstation can handle hazardous components: 1 means it can, and 0 means it cannot.
B_N	Line-side buffer indicating the number of tasks the workstation stored.
P	Collection of products to be disassembled.
S_i	The sequence of disassembly steps for the first i product to be disassembled
l	Total number of disassembly steps for the product.
$Sub_{i,l}$	A 0–1 array indicates whether a sub-product will be generated at a specific step in the disassembly process: 1 indicates generation, and 0 indicates no generation.
P_{Sub}	A collection of sub-products, if the product generates sub-products, the sub-products will be processed according to the established disassembly steps.
$P_{\text{actual\_step}}$	Practical steps for product disassembly
$W_{\max}$	Maximum number of manual workstations of a single type
$W_{\min}$	Maximum number of manual workstations of a single type

increased or decreased, denoted as $W_{k,i}^M$, where i must not exceed the maximum number of manual stations of a single type $W_{\max}$. Use $D_N = \{d_1, d_2, d_3, \ldots, d_N\}$ to indicate whether an automated station handles hazardous parts (yes is 1, no is 0). Each workstation has a line-side buffer B_N with a capacity of C , divided into input $B_{N,in}$ and output $B_{N,out}$ buffers with a common capacity of C. The flow of products to be disassembled in the input buffer follows the first-in-first-out (FIFO) processing rule. Appliances are to be disassembled in the system from the output buffer of W_{start} waiting to enter the end of the input buffer, only these two buffers do not set the capacity, the content of the number of products to decide.

The basic goal of the system is to process i incoming products to be disassembled $P = \{P_1, P_2, P_3, \ldots, P_i\}$, each product to be disassembled has l disassembly steps $S_i = \{S_{i,1}, S_{i,2}, S_{i,3}, \ldots, S_{i,l}\}$, and each step can be done independently by one of the corresponding workstations. $S_{i,l} = W_j^A$ or $S_{i,l} = W_k^M$. An array Sub_l indicates whether the product being disassembled generates a child product P_{Sub}, drawing upon the concept of sub-products, which are referred to as sub-orders in (Wurster et al., 2022) at step (generating is 1, not generating is 0), which also has a disassembly step according to its index. However, we assign sub-products a status similar to that of the main product, granting them their own distinct disassembly paths. Each disassembly step corresponds to a workstation type and whether it is a hazardous part or not, and if it is a non-hazardous part, it can be handled by a manual station or an automated station.

The system is driven by the selection of the target location of the products in each step. The appliance disassembly step S_i corresponds to each workstation, with $P_{\text{actual\_step}}$ as the index of the appliance disassembly step, and every time we go through a disassembly $P_{\text{actual\_step}}$ plus one, if the target workstation $S_{i,P_{\text{actual\_step}}+1}$ is an automated workstation W_N^A and the workstation buffer occupancy $B_N < 10$ and the workstation is a hazardous component disassembly station, such as $d_N = 1$, it means that the next step is feasible. If $S_{i,P_{\text{actual\_step}}+1}$ is an automated workstation W_N^A but $d_N = 0$, then the next step can be either an automated workstation W_N^A or a manual station of the same type W_N^A.

2.2. Problem description

In summary, the first optimization objective is to minimize the overhead of N workstations:

$$\min F_1 = \sum_{n=1}^{N} \left(c_{n,working} \times t_{n,working} + c_{n,idling} \times t_{n,idling} \right) \tag{1}$$

where $c_{n,working}$ represents the overhead per unit of time for the disassembly operation of the n th workstation, $t_{n,working}$ represents the processing time of the n th workstation in the time interval between the last dispatch and this one, $c_{n,idling}$ represents the idling overhead of the n th workstation, and $t_{n,idling}$ represents the idling time of this workstation at present.

The second optimization objective is to consider i product completion time minimization:

$$\min F_2 = \sum_{\mathrm{i}=1}^{\mathrm{i}} \left(t_{i,process} + t_{i,waiting} \right) \tag{2}$$

where $t_{i,process}$ represents the processing time of the disassembled completed appliances and $t_{i,waiting}$ represents the waiting time of the disassembled completed appliances.

The overall optimization objective is

$$\min F = w_{\cos\, t} \times F_1 + w_{time} \times F_2 \tag{3}$$

where $w_{\cos\, t}$ and w_{time} are the weights of workstation overhead and time cost, respectively.

3. Method description

3.1. Markov Decision Process

The scheduling aspect of the disassembly process for appliances is modeled as a Markov Decision Process (MDP). The system can be defined as a quintuple (S, O, A, T, R). Here, S represents the state of the entire system, including tasks, workstations, and other relevant information. O denotes the set of observations available to the intelligent agent, which it uses to make decisions. A is the set of actions the intelligent agent can select from, with decisions made based on the agent's observations O and its strategy $\pi : O \rightarrow A$.Changes in the system's state follow a Markov state transition function $T : S \times A \rightarrow S'$, which defines how the current state and selected actions lead to the next state S'. At the end of each time step, the intelligent agent receives a reward based on its actions, represented by $R : S \times A \rightarrow R$, where the reward function reflects the quality of the agent's performance. The primary objective of the agent is to maximize cumulative rewards by optimizing its policy through continuous interaction with the environment.

3.1.1. State space

The state space should accurately reflect the real-time status of the current system. Beyond merely representing the system's characteristics, it is also crucial to minimize the computational resource demands of the algorithm. To achieve this, this paper adopts a three-part structure for the state space representation.

The first component includes the action space size, represented by the vector u_1. Each index of this vector corresponds to an action at the current time step t indicating whether the execution of a component's action is pending. A value of 1 represents a feasible action, while a value of 0 indicates no pending action. The total size of u_1 equals the sum of the action space size $n + m$.

The second component is the current time t and the buffer occupancy of each workstation, represented by u_2. Given that there are N workstations, each workstation has both an input and an output buffer, resulting in $2N$ buffers. Additionally, there is an output buffer at the start point, making the total buffer size $2N + 1$. The buffer occupancy is expressed as an integer value representing the number of components currently in the buffer.

We account for the dynamic adjustment of manual workstations, enabling the system to increase or decrease the number of manual workstations as needed. To manage this, we sum the buffers of all manual stations of the same type, using the state transfer of that type of station. This approach helps avoid unnecessary changes to the overall

size of the state space.

The third component is a 0–1 vector u_3, representing workstation failures. The size of this vector is N, corresponding to the total number of workstations.

The total state space S is defined as the concatenation of the sets $\{u_1, u_2, u_3\}$, with the combined length expressed as $(n+m)+(2N+1)+N$. To ensure diversity and complexity in disassembly scenarios for training the intelligent agents, the occupancy of individual workstations changes dynamically after the initiation of the disassembly process. This ongoing variation ensures a sufficient diversity of data for training the agents effectively. Ultimately, the system's ability to make profit-driven decisions—by selecting the most optimal actions based on real-time conditions—maximizes efficiency in terms of time and energy consumption throughout the process.

3.1.2. Action space

Based on the designed loosely structured disassembly system, actions are defined as feasible material flows between two specific workstations. If the components handled by the two workstations have a reciprocal relationship in the established disassembly sequence, the transportation of components between them is considered part of the action space. While the total number of possible actions in the action space is fixed, the number of effective actions dynamically changes based on factors such as the buffer occupancy of the workstations and potential workstation failures. This approach ensures system flexibility, allowing actions to be executed only when buffer capacity or other action constraints are met, thus adapting to real-time conditions without altering the overall structure of the action space.

Therefore, the action space is discrete and comprises two parts: the action space for component transportation, represented by feasible material flow between workstations q_1, and the action space for increasing or decreasing the number of manual workstations q_2. The sizes of these two action spaces are n and m, respectively.

Each index in a products transportation action maps to a pair of workstations. For example, the action a_n corresponds to $\left[W_2^A, W_3^A\right]$ when the index n in $q_{1,n}$ is 1. This indicates a transportation relationship where the product moves from the output buffer of automated workstation W_2^A to the input buffer of automated workstation W_3^A.

The increase and decrease actions for manual stations are represented by corresponding action indexes in two-dimensional lists. For example, the action a_n is indexed at $n+m$ in $q_{2,m}$, corresponding to $\left[W_2^M, W_2^M\right]$ when $m=1$, indicating the increased action for manual workstation W_2^M. There is a corresponding constraint that the number of manual workstations of type k must not exceed the limit for that type of workstation, expressed as:

$$W_{\min} < k < W_{\max} \tag{4}$$

The goal is to address potential congestion that the intelligent system may face during the training process. Specifically, when feasible target workstations are fully occupied, increasing the number of target workstations compensates for the time required to complete product disassembly, albeit with an increased overhead for the additional workstations.

The total action space A is the concatenation $\{q_1, q_2\}$ of q_1 and q_2 with a fixed size $n+m$.

The following constraints exist concerning the feasibility of the system's actions.

- If the buffer of the target workstation is full, the workstation temporarily cannot accept the agent's product transfer action; however, disassembly operations can still proceed.
- If the target automated workstation fails, that workstation may not perform any actions either.
- When the target workstation is a manual workstation, prioritize the countermeasures of moving to a manual workstation.
- If all target manual workstations are at full capacity or in a failed state, the operation is not executable. However, if a manual workstation of the same type exists and that workstation can be expanded, the system combines the capacity judgment of all manual workstations of the same type.

3.1.3. Reward functions and reward-shaping

The component processing time is used as design reference data for the reward function. Since the disassembly steps are generally similar for all home appliances, this time can be utilized as an evaluation criterion for system optimization. Specifically, the total processing time of the most recently processed component T_{process} and the waiting time T_{waiting} are divided by the number of processing steps l of the product P_i, giving a time-based measure. This calculation is expressed as the current processing time, which serves as the basis for the reward function r_{time}:

$$T_1 = T_{i,process}/l \tag{5}$$

$$T_2 = T_{i,waiting}/l \tag{6}$$

$$r_{time} = 1/(1+T_1) + 1/(1+T_2) \tag{7}$$

where T_1 is the average processing time and T_2 is the average waiting time.

The overhead of the environmental workstation between two actions is also a crucial factor in the reward function. It not only helps control the number of active workstations but also balances the occupancy rates between manual and automated stations. The reward for each workstation is calculated by multiplying the workstation's processing time $T_{working}$ by the cost of running the workstation $C_{working}$, and adding the workstation's idle time T_{idling} multiplied by the cost of idling C_{idling}. Time is calculated as the difference between the current time and the time of the last incentive calculation. This generates the reward function $r_{\cos t}$ for workstation overhead between each move:

$$r_{\cos t} = \sum_{n=1}^{N} 1/\left(1 + T_{n,working} \times C_{n,working} + T_{n,idling} \times C_{n,idling}\right) \tag{8}$$

where $T_{n,working}$ is the processing time of workstation n, $C_{n,working}$ is the cost of running workstation n during processing, $T_{n,idling}$ is the idle time of workstation n, $C_{n,idling}$ is the cost of idling workstation n.

Since the system accounts for the uncertainty of time, the workstation processing time is randomly simulated using a normal distribution, resulting in significant volatility in $r_{\cos t}$, which can hinder the training of intelligent agents. Additionally, the reward r_{time}, based on product component processing time, is calculated only after each appliance has been fully processed. To facilitate the learning of intelligent agents and prevent difficulties arising from sparse feasible actions within the action space, we apply the principle of Reward-Shaping (Laud, 2004). Rewards are designed for each action step, with the short-term reward increasing as the current feasible action approaches the completion of product disassembly. This mechanism trains the intelligent agent to prioritize completing disassembly while simultaneously encouraging more effective exploration:

$$r_{action} = \underset{\text{step}}{P_{\text{i,actual\_}}}/l \tag{9}$$

where $P_{\text{actual\_step}}$ is the current processing step for the product P_i and l is the number of all steps for the product.

The weighted combination of several reward functions as an overall objective function is also equivalent to the reward of the action r_{step} :

$$r_{step} = w_{time} \times r_{time} + w_{\cos t} \times r_{\cos t} + w_{action} \times r_{action} \tag{10}$$

where w_{time} , $w_{\cos t}$ and w_{action} represent the weights of the time bonus and workstation overhead bonus, respectively. These hyperparameters are empirically set to balance the numerical differences among the

various rewards, with the primary objective of demonstrating the feasibility of the algorithm.

To further enhance the training effect, considering that while the action space is fixed, certain actions in specific states are ineffective (they do not lead to changes in the number of workstations or inventory levels), these actions are assigned a constant reward value r_p. This design ensures that the agent still receives feedback without altering the dynamics of the system.

Total Rewards R for:

$$R = r_{step} + \omega r_p \begin{cases} \omega = 0, \textit{if legal action} \\ \omega = 1, \textit{if illegal action} \end{cases} \tag{11}$$

where r_p is the reward (penalty value) for illegal action, and ω is the discriminating value of whether the action is illegal or not.

3.1.4. State transition

The state transitions T within a disassembly system can be effectively modeled using a Markov process, in which the system is divided into discrete states.

In the initial state and normal transmission state, two primary actions are considered feasible: transferring the product from the output buffer at the start point to either the target automated workstation or the manual workstation in the same group, as mentioned in Fig. 4. Additionally, adjusting the number of manual workstations—either increasing or decreasing—within the allowable limits for that specific workstation type is also classified as a feasible action. All other actions are deemed infeasible. If an illegal action is taken, the agent receives a penalty reward as a consequence. Importantly, no workstation exceptions are allowed, ensuring strict adherence to the defined rules and constraints of the system.

The automated workstation handles the failure state and does not process the products according to the actions of the intelligences, but directly puts them into the input buffer of the manual station in the same group, as in Fig. 5.

In the event of an automated workstation fault or a full automated input buffer, product scheduling actions targeting other types of workstations remain feasible. Actions directed towards the faulty or full automated workstation become infeasible, but actions targeting the corresponding manual workstation of the same type remain feasible, provided there is available space in the buffer of the manual workstation. If no space is available, only product scheduling actions for other workstation types are feasible. Additionally, changes in the number of manual workstations of the current type, within the defined constraints, are also considered feasible, as illustrated in Fig. 6At the same time, if the buffer of the same type (processing the same component) of manual

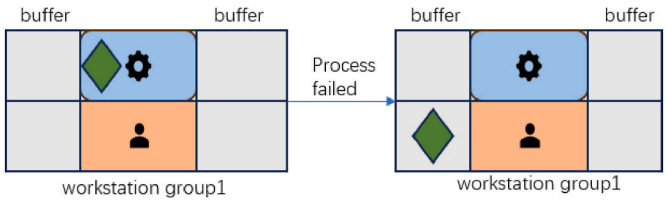

Fig. 5. Disassembly failure handling in the system.

workstations is fully occupied, the action space allows an increase in the number of manual stations for that workstation group with a value of 1 (feasible); otherwise, it is 0 (not feasible). Conversely, if the buffer is empty, the action of reducing the number of manual stations is feasible.

3.1.5. Specific Example

There is an example of a simple disassembly system in Fig. 7, detailing the logic for calculating the state space, action space, and reward function.

In the initial stage of the system, there is one automated workstation and one manual workstation of the same type. All workstations start with empty buffers. The source, where products enter the system, has an ID of 0, and the sink, where disassembled products are collected, has an ID of 1. Workstations W_1^A and W_2^A are different types of automated workstations, with IDs 2 and 3, respectively. W_1^M is the manual workstation of the same type as W_1^A, with an ID of 4. The initial state space of the system is represented as: $[1.0,1.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0]$.The first 8 indices correspond to the action space $[[0,2],[0,3],[0,4],[2,1],[3,1],[4,1],[3,3],[3,0]]$.The first 6 indices represent product transfers between workstations, while the last 2 indices represent increasing or decreasing the number of manual workstations of the same type as W_1^A and W_2^A, respectively. The middle 7 indices in the state space represent the occupancy of the input and output buffers of the three workstations, as well as the occupancy of the source's output buffer. Initially, only the source's buffer is fully occupied, while the buffers of other workstations are empty. The last index in the state space indicates the failure status of the workstations, with all values initially set to 0, indicating no failures.

When an action corresponding to indices 0 to 2 in the state space is taken, such as $[0,2]$ (index 0), the occupancy of the target workstation 2, W_1^M, changes to 1/10 of its buffer capacity, i.e., 0.1. If, under the given constraints, the action $[3,3]$ is taken, which increases the number of manual workstations of the same type as W_1^M, the change is reflected in the buffer capacity and the number of products that can be processed simultaneously, given the fixed size of the state space.

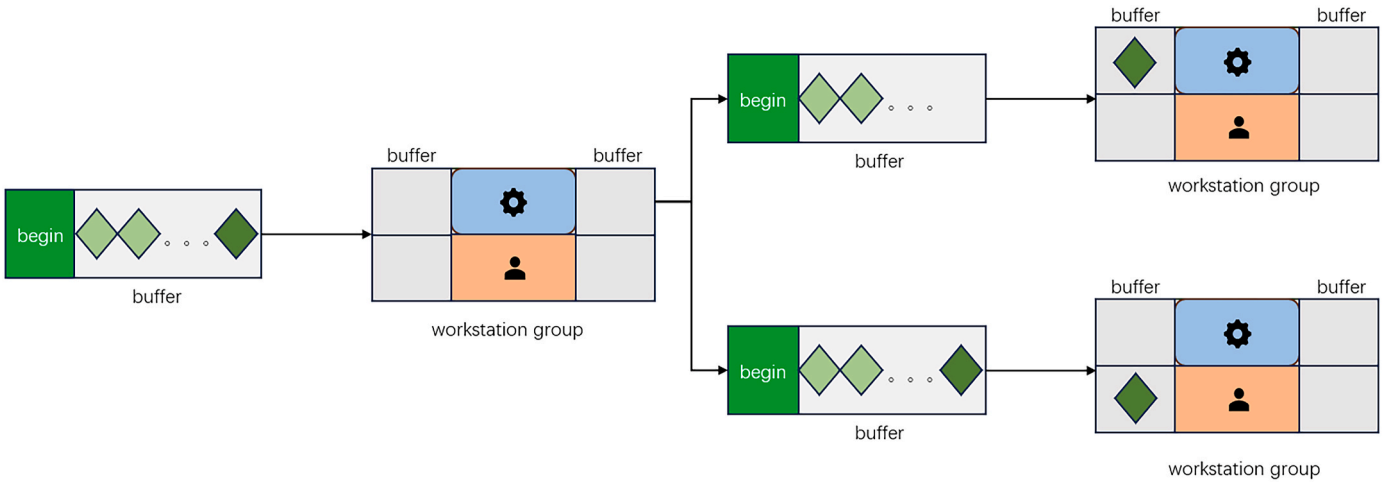

Fig. 4. Normal state transitions in the system.

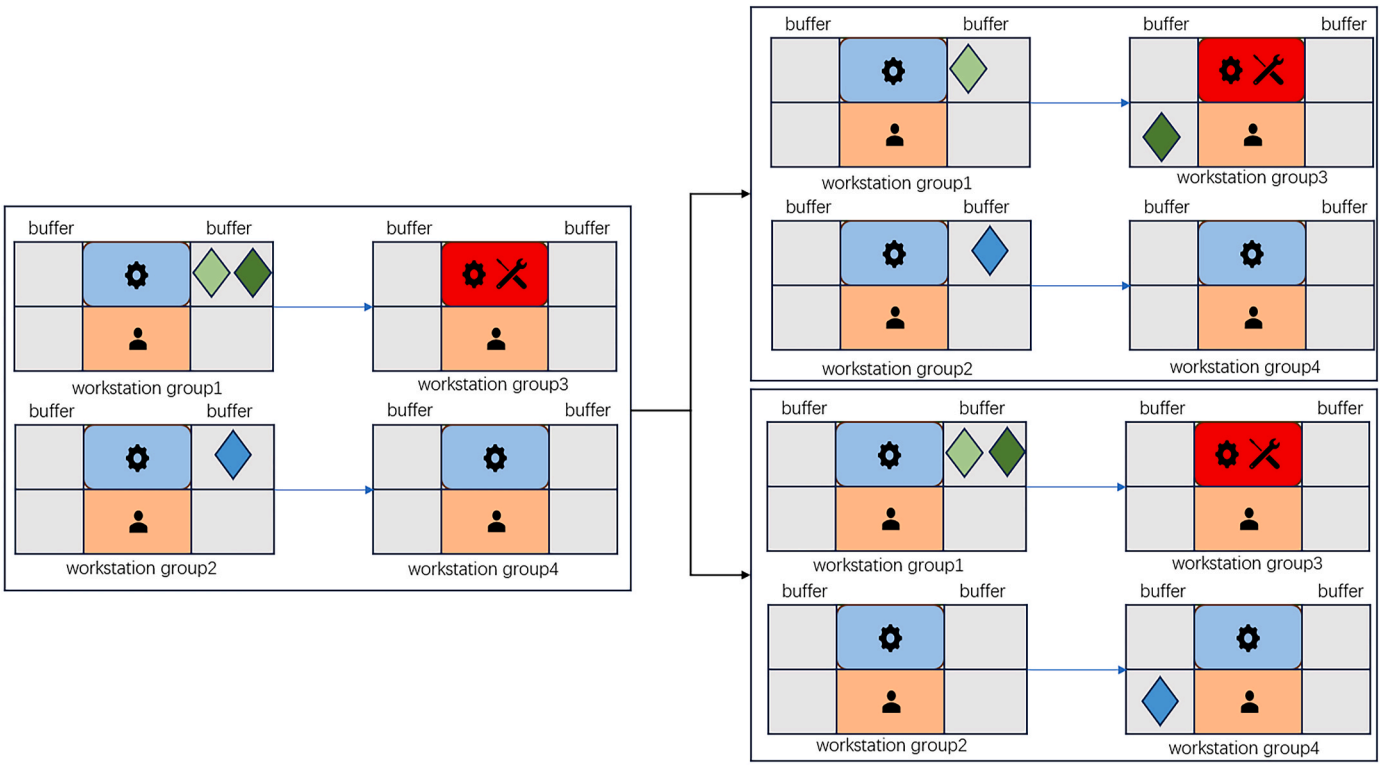

Fig. 6. State transitions in case of system failure and blocking.

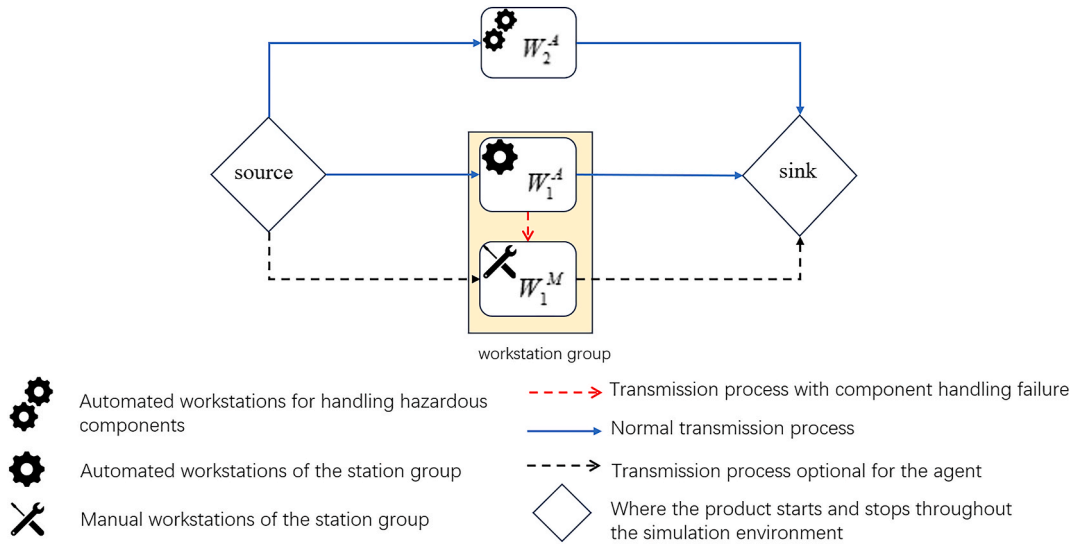

Fig. 7. The system structure of the Specific Example.

3.2. Proximal policy optimization algorithm

PPO is a widely used algorithm in the field of reinforcement learning, which centers on improving the decision-making ability of intelligence in complex environments through the collaborative updating of policy networks and value networks. The policy network is used to generate the probability distribution of actions in the current state, while the value network estimates the expected long-term payoff of each state. Through the combination of these two, the PPO algorithm is able to effectively handle complex problems in dynamic environments.

In PPO, the policy update is based on the policy ratio $r(\theta)$, which is used to indicate the relative change between the old and new policies for a given action, with the following formula:

$$r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{12}$$

where θ represents the network parameters, The policy network parameters θ and value network parameters ϕ were initialized using He (Kaiming) initialization method (He et al., 2015), which is commonly used for ReLU-based networks to maintain variance stability. Initialize an old policy network with parameters θ_{old}, setting it to be identical to the policy network at initialization. θ_{old} represents the old network parameters, π is the probability distribution of the action, s_t is the state at time t, and a_t is the action made in this state.

In order to prevent the instability caused by too large policy updates, the PPO uses a pruning operation to limit the magnitude of policy updates, and the objective functions of the policy network and the value network are as follows:

$$L^{CLIP}(\theta) = E[\min(r(\theta)A_t, clip(r(\theta), 1-\varepsilon, 1+\varepsilon)A_t)] \tag{13}$$

$$L^{VALUE}(\phi) = E\left[\left(R_t - V_\phi(s_t)\right)^2\right] \tag{14}$$

where θ and ϕ represent the network parameters, ε is a hyperparameter for the clip cropping operation, which limits the update magnitude to avoid excessive strategy changes, and A_t is a generalized advantage estimation (GAE), which is used to smooth the estimation of the advantage function and reduce the variance in the estimation. R_t is the actual reward computed through the reward function, and $V_\phi(s_t)$ is the value estimate of the current state. By minimizing the prediction error, the intelligence is able to more accurately assess the potential payoffs of each state and thus make better decisions.

Before each optimization step, the old policy parameters θ_{old} are hard updated by directly copying the current policy parameters θ, ensuring stable importance sampling and preventing excessive policy divergence: $\theta_{old} \leftarrow \theta$ (15)

PPO optimizes the parameters of the policy network and the value network by gradient descent. The update rule for the strategy network is:

$$\theta \leftarrow \theta - \alpha \nabla_\theta L^{CLIP}(\theta) \tag{16}$$

The rules for updating the value network are:

$$\phi \leftarrow \phi - \alpha \nabla_\phi L^{VALUE}(\phi) \tag{17}$$

α is learning rates. Taking a fixed value. Both the policy network and the value network are updated synchronously in each optimization step using the same learning rate α.

This updating mechanism ensures that the intelligences are able to progressively optimize their decisions and find the optimal strategy in a dynamic environment. The procedure of PPO is given in Algorithm 1.

Algorithm 1: Proximal Policy Optimization

Input: Initial policy parameters θ and old policy parameters θ_{old}, value function parameters ϕ, discount factor γ, hyperparameters of GAE λ, clip parameters ε, learning rate α

1: Initialize policy network π_θ and value network V_ϕ
2: **for** k = 1,2, …, K **do**
/*k means an episode. */
3: **for** t = 1, 2, …, T **do**
/*t means a timestep. */
4: Collect trajectories (s_t, a_t, r_t, s_{t+1}) using policy network π_θ
5: Compute TD error δ_t using value network V_ϕ: $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$
6: Compute generalized advantage estimate A_t: $A_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$
7: **if** the number of time steps collected reaches the update frequency do:
8 Update old policy parameters θ_{old}: $\theta_{old} \leftarrow \theta$
9 Update policy network π_θ using the objective:
$L^{CLIP}(\theta) = E[\min(r(\theta)A_t, clip(r(\theta), 1-\varepsilon, 1+\varepsilon)A_t)]$
10 Update value network V_ϕ by minimizing the loss: $L^{VALUE}(\phi) = E\left[\left(R_t - V_\phi(s_t)\right)^2\right]$
11 Apply gradient descent to update θ and ϕ: $\theta \leftarrow \theta - \alpha \nabla_\theta L^{CLIP}(\theta)$, $\phi \leftarrow \phi - \alpha \nabla_\phi L^{VALUE}(\phi)$
12 **End for**
13: **End for**
Output: Optimized policy π_θ and value function V_ϕ

3.3. PPO with reward-shaping in disassembly system

In a flexible hybrid disassembly system, task scheduling and resource allocation are critical components for improving overall system efficiency. PPO, as a reinforcement learning method, intelligently optimizes workstation selection, task allocation, and material flow by integrating both the policy network and the value network. The policy network generates potential actions based on the system's state (workstation load, component priority, buffer space), helping the agent make optimal decisions, while the value network evaluates the long-term return of the current state, continuously refining the scheduling strategy through comparison with actual returns, as illustrated in Fig. 8.

To avoid the problem of sparse rewards (a reward is only given after all components of a product have been disassembled), we design a short-term reward mechanism for every effective action. This Reward-Shaping technique provides feedback for intermediate steps, improving learning efficiency. The mechanism mitigates the inefficiency caused by sparse rewards, enabling the agent to find an optimized strategy more quickly.

PPO employs a clipping mechanism that limits the magnitude of policy updates, ensuring stable learning in complex systems by preventing large adjustments that could destabilize the system. This mechanism ensures that policy adjustments do not deviate too far, promoting stability in the learning process. PPO effectively handles uncertainties within the system, such as variations in product status, complex disassembly sequences, and random workstation failures, ensuring efficient adaptation and optimization in dynamic environments.

Traditional rule-based scheduling methods, such as FIFO, struggle to achieve optimal results in complex environments. In contrast, reinforcement learning methods interact dynamically with the environment to continuously optimize workstation and resource allocation, minimizing resource waste and waiting times through the use of an appropriate Reward-Shaping mechanism. Additionally, the clipping mechanism in PPO plays a critical role in enhancing decision stability, ensuring that each policy update does not negatively impact system performance. Compared to other reinforcement learning methods, such as Deep Q-Network (DQN) and Advantage Actor-Critic (A2C), PPO improves training stability by constraining the magnitude of policy updates, making it particularly suitable for real-time scheduling scenarios. This stability allows PPO to adapt more effectively to dynamic conditions, resulting in robust and efficient scheduling in flexible disassembly systems.

4. Experiments

In the previous sections, we established a loosely coupled disassembly system that integrates both manual and automated workstations. We also accounted for the flexibility in the number of manual workstations to address potential congestion within disassembly tasks. To manage the material flow and dynamically adjust the number of active workstations, a policy-based reinforcement learning approach was selected, enabling the system to optimize operations based on real-time conditions.

4.1. Evaluation indicators

In this section, we first establish an experimental scenario under relatively simple conditions to support workstation number adjustments and target workstation selection. This scenario demonstrates the superiority of the Proximal Policy Optimization (PPO) method in material flow scheduling and verifies the agent's ability to make reasonable adjustments to the number of workstations. We then proceed to a more complex and variable scenario, where the agent explores and trains under comprehensive constraints. This approach proves that our Reward-Shaping method and reinforcement learning technique can effectively perform exploration and training in dynamic, complex disassembly scenarios, achieving better long-term results.

To quantify the effectiveness of the scheduling approach, this study defines three core metrics: disassembly cost, processing time, and reward convergence. Disassembly cost measures the total resource consumption throughout the disassembly process, including the operational and idle costs of workstations, as well as the cost differences among various types of workstations. Processing time evaluates overall efficiency by recording the completion time of product disassembly and

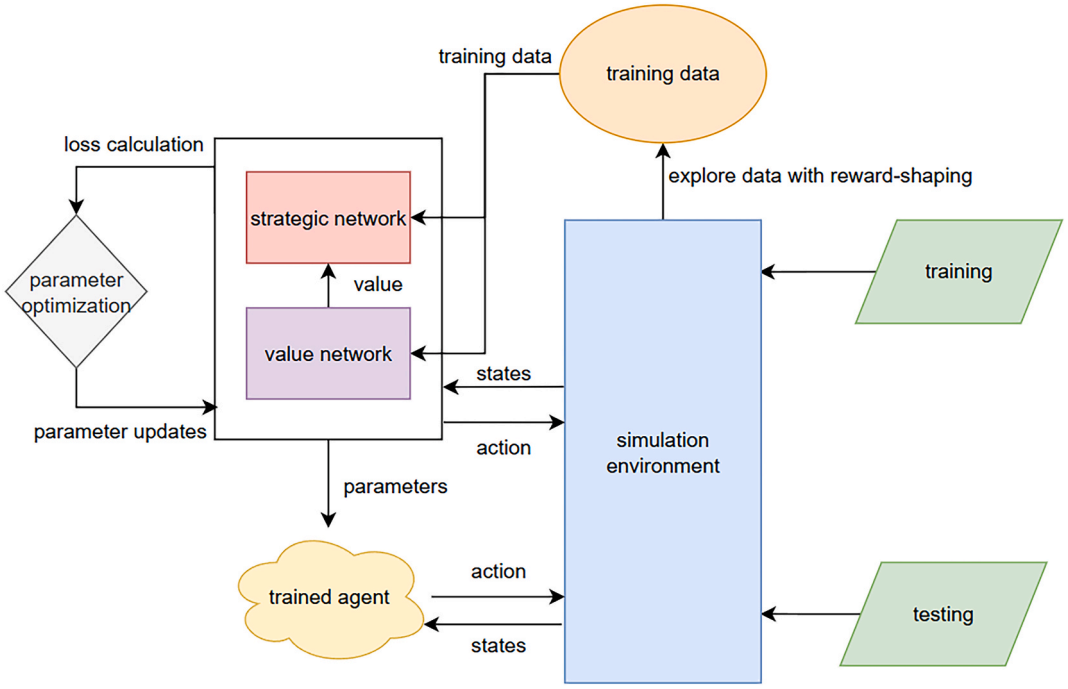

Fig. 8. The framework of the proposed Reward-Shaping-based scheduling method.

calculating the average disassembly time per step based on the total disassembly time and the number of steps. All operation processing times follow a normal distribution to simulate realistic variations in production. Reward convergence assesses the training performance of the reinforcement learning model by computing the average reward every 100 steps, providing insights into the algorithm's exploration efficiency and policy stability. Additionally, the variance and mean values after stabilization are analyzed to evaluate both the convergence speed and stability of the PPO algorithm in dynamic scheduling tasks.

To facilitate observation and comparison, we applied an Exponentially Weighted Moving Average (EWMA) smoothing technique to the data. In this study, we used the following performance metrics: disassembly cost, completion time, and stability.

Furthermore, a more comprehensive and detailed comparison of the algorithm's performance is conducted by examining the number of products processed within the same step, the adjustments made by the agent to optimize disassembly paths, and the specific quantification of product processing time and waiting time. These additional comparisons enhance the evaluation of the algorithm's effectiveness and adaptability.

4.2. Performance evaluation

This case study references (Wurster et al., 2022). We assume, excluding the start and end points, that the system consists of three automated workstations W_1^A , W_2^A , W_3^A and one manual workstation W_1^M. Among them, W_1^A is responsible for handling hazardous components, while W_2^A and W_3^A are designated for processing general components.

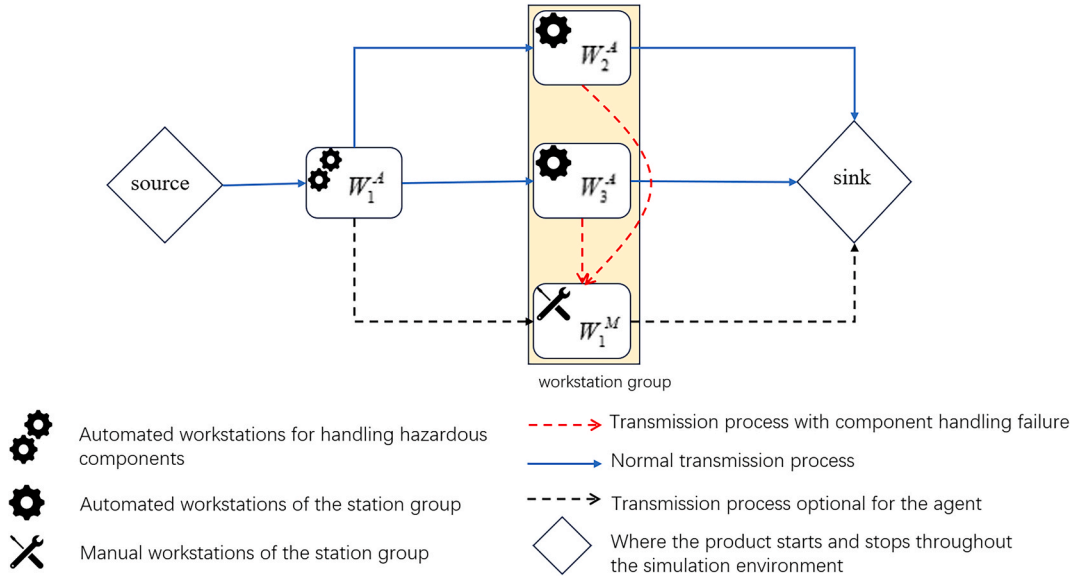

Fig. 9. The system structure of the experimental scenario for congestion handling.

The manual workstation W_1^M is capable of handling the same type of tasks as W_2^A and W_3^A, and the number of manual stations of the same type as W_1^M is variable, as shown in Fig. 9.

Due to the relative simplicity of the structure, we assume that the refrigerator requires three scheduling steps to complete the disassembly. The disassembly steps S_i for the product P_i are $\{source, W_1^A, W_2^A, \sin k\}$ and $\{source, W_1^A, W_3^A, \sin k\}$, where W_2^A and W_3^A are automated workstations. Automated workstations have lower processing costs and shorter processing times compared to manual workstation. However, these two workstations have a processing failure probability of 0.2 %. When processing fails, the product must be transferred to the manual workstation W_1^M for reprocessing. Additionally, when either of the automated workstations fails, they must be taken offline for repair, and they cannot receive product during this period. The components can only be handed over to the manual workstation W_1^M or an increased number of similar manual workstations $W_{1,i}^M$ for processing. The reinforcement learning agent is responsible for deciding whether to switch processing to the manual workstation W_1^M.

With an unlimited number of products during training, we compared the training speed and convergence of several reinforcement learning methods, with test cases characterized by changes in the number of workstations, the number of disassembly tasks completed, and the convergence of rewards within a fixed number of steps. The weight ratios of the three rewards are $w_{\cos t} : w_{action} : w_{time} = 0.2 : 5 : 1000$. The purpose is to normalize multiple objectives to balance their impact.

In experimental tests, PPO demonstrated faster convergence and better overall performance compared to other policy-based algorithms such as A2C and DQN. As illustrated in Figs. 10 and 11, the horizontal axis represents the number of training sets, where each set consists of 100 scheduling operations, while the vertical axis represents the average reward per 100 operations. Due to its focus on long-term returns, PPO exhibits superior overall performance throughout training. Furthermore, the reinforcement learning agent effectively identifies system bottlenecks, such as mitigating congestion by increasing the number of manual workstations. As shown in Table 2, results from five independent experiments indicate that PPO consistently stabilizes the number of manual workstations at 8, whereas A2C and DQN show fluctuations of ±1, demonstrating PPO's robustness in dynamic resource adjustments.

Fig. 12 presents the total number of completed products, the number of path changes generated by the agent's scheduling, the total disassembly operation time, and the total product waiting time for the three

Fig. 11. The total reward and training process of agents.

Table 2
The convergent number of Manual Workstation.

Method	1st	2nd	3rd	4th	5th
PPO	8	8	8	8	8
A2C	7	8	7	7	7
DQN	4	4	4	4	5

reinforcement learning methods under the same training steps. In terms of completed product quantity, PPO significantly outperforms A2C and DQN within the same training steps, achieving 30,382 completed products, compared to 26,745 for A2C and 10,406 for DQN. The core advantage of PPO lies in its clipping mechanism, which effectively balances exploration and exploitation, enabling the agent to quickly identify high-reward actions (e.g., prioritizing task allocation to low-load workstations) while avoiding excessive greed that could lead to suboptimal policies (e.g., prematurely fixing the number of workstations). This advantage is further evidenced by the fact that PPO agents adjust product routing more frequently, making full use of the additional manual workstations.

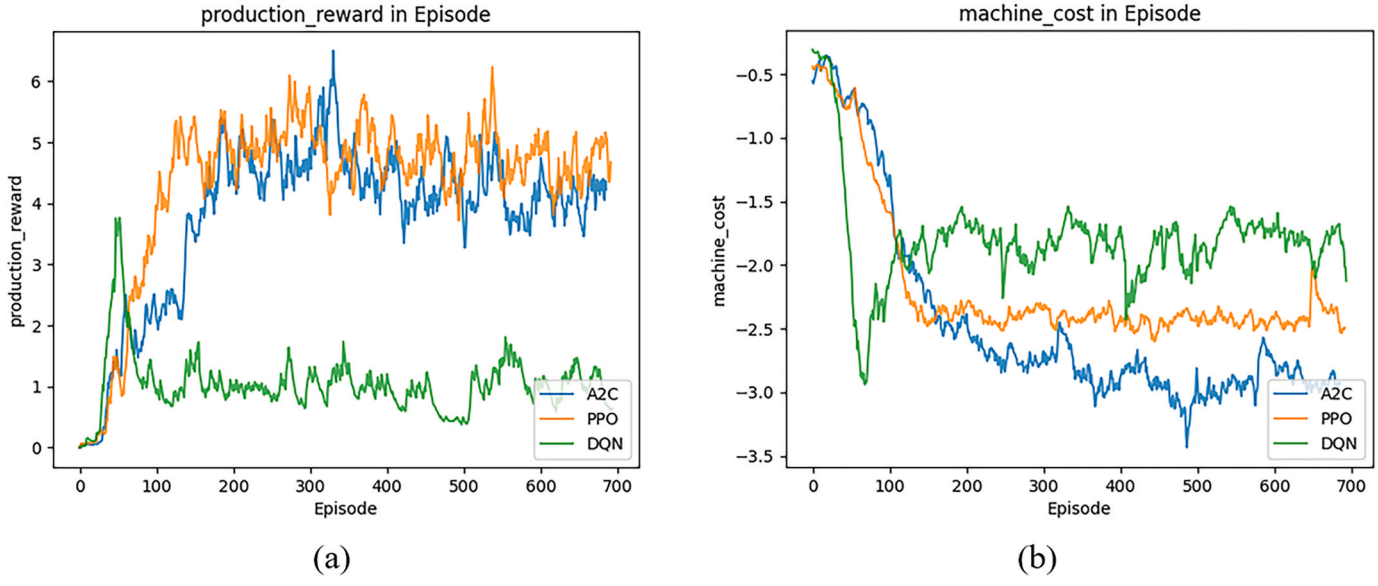

Fig. 10. The reward function trained for the agents.

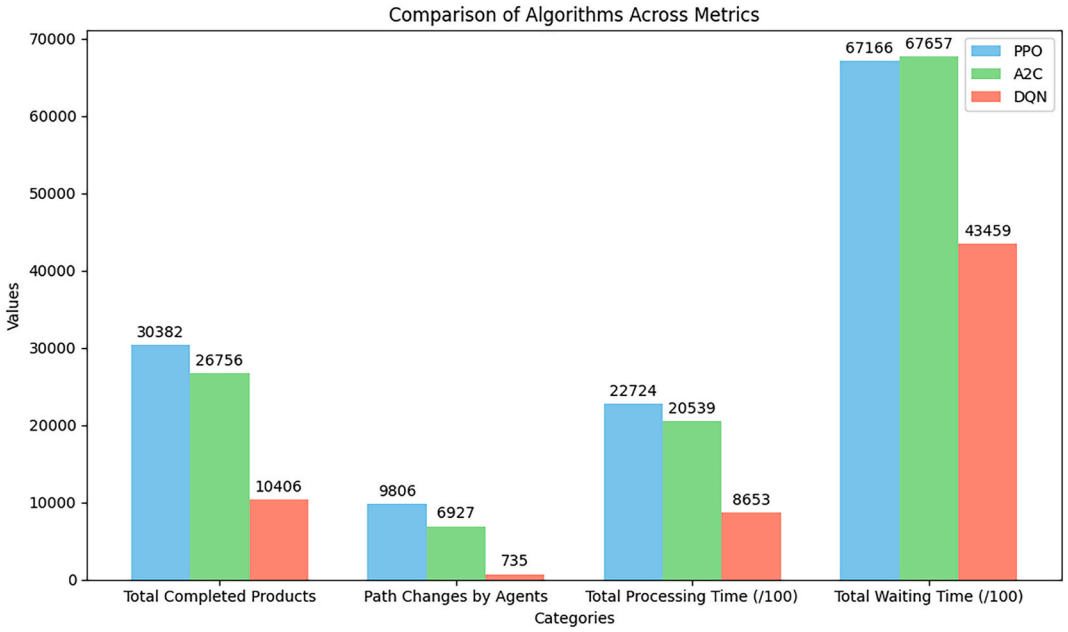

Fig. 12. The comparison of algorithms across metrics.

In terms of processing efficiency, PPO achieves a total processing time to waiting time ratio of approximately 0.34, higher than A2C (approximately 0.30) and significantly better than DQN (approximately 0.20), indicating superior work efficiency. This improvement stems from GAE, which smooths multi-step returns, ensuring that policy updates focus on long-term efficiency rather than single-step reward fluctuations.

Based on Fig. 10(a)(b), it is evident that although the DQN method demonstrates relatively fast convergence and maintains lower workstation overhead compared to the other two methods, it falls into a local optimal solution early. This limits its ability to learn better scheduling decisions, leading to minimal increases in the number of workstations. While this results in lower workstation overhead, consistently stabilizing at around −1.82, the method also yields the lowest disassembly completeness reward, with a value of only 0.98, as shown in Table 3.

A2C method performs better in balancing the trade-off between rewards. It can more effectively navigate the relationship between workstation overhead and disassembly rewards. However, its convergence speed and stability lag behind those of PPO. Although the final rewards obtained by A2C are close to those of PPO, there remains a noticeable gap, demonstrating that while A2C performs well, it does not reach the same level of optimality as PPO.

By stabilizing policy updates through its clipping mechanism, optimizing long-term return integration via GAE, and adapting seamlessly to dynamic action spaces, PPO achieves comprehensive multi-metric superiority in disassembly scheduling tasks. Its notable advantages in total product completion, processing time efficiency, and scheduling stability validate the unique value of policy optimization-based reinforcement learning in dynamic and uncertain environments, providing theoretical support and practical benchmarks for intelligent upgrades in industrial-scale disassembly systems.

Table 3
Mean and Standard Deviation of Each Method (300–700 episodes).

Method	Reward	Machine_cost	Production_reward
PPO	10.21, 0.65	−2.42, 0.15	4.76, 0.41
A2C	9.36, 0.65	−2.89, 0.15	4.34, 0.53
DQN	7.04, 0.43	−1.82, 0.15	0.98, 0.29

4.3. Results

We developed a complex, multi-level flexible disassembly system that includes multiple automated workstations and their corresponding manual workstations. Each manual workstation can handle product components of automated stations in the same group. Based on the correlation between refrigerator components and disassembly elements (Wang et al., 2022), we simplified the prioritization hierarchy for disassembly and designed the corresponding steps, as illustrated in Fig. 13. In this figure, nodes represent individual components, while directed line segments indicate the priority relationships between them. Path branches are used to generate sub-products within the disassembly process. This study incorporates two specific sub-product disassembly paths: (2, 3, 13) and (1, 12). For each product component, the system specifies the type of workstation capable of disassembling it, with a one-to-many relationship between workstation types and component types. The component information corresponding to the nodes in the figure is detailed in Table 4.

This design allows for flexible task distribution across workstations, facilitating optimized disassembly processes that can adapt to the dynamic nature of multi-component systems.

Each episode consists of 200 time-steps, with a total of 1000 episodes conducted during testing. Fig. 14 presents the performance of various scheduling methods under identical scenarios, revealing the strengths and weaknesses of different algorithms when dealing with complex and dynamic environments. The result shows that despite the increasing complexity of the environment, the reinforcement learning agent gradually adapts to the scenario's intricacies and uncertainties through the application of an incremental Reward-Shaping mechanism. In particular, the PPO algorithm performs exceptionally well, demonstrating a high degree of adaptability in complex task environments. As training progresses, PPO ensures that the agent incrementally optimizes its decision-making process, eventually achieving a higher average reward. Compared to rule-based methods, PPO also excels in pursuing long-term objectives.

Additionally, we evaluated three heuristic-based workstation selection strategies, namely Empty, Random, and FIFO, to analyze the performance of the PPO method in scheduling. The Random strategy assigns tasks to workstations randomly. While this method does not actively optimize resource allocation, it serves as a useful baseline for

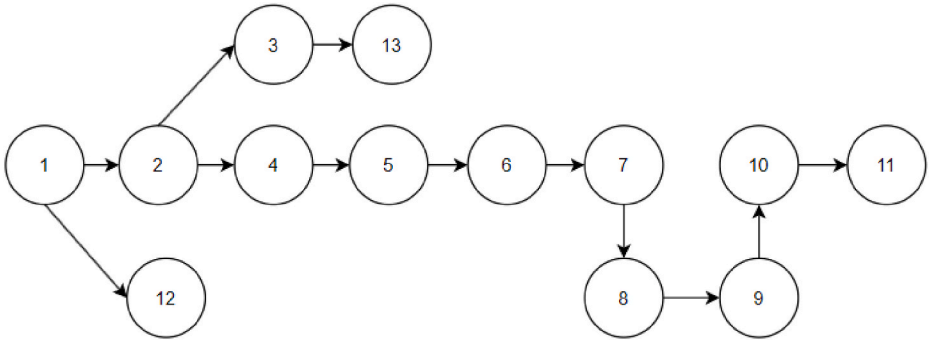

Fig. 13. Simplified component index and corresponding disassembly process.

Table 4
Disassembly tasks of the refrigerator.

ID	Component	Workstation ID	Processing Time	Processing Method	Hazardous
1	Upper Door	5	35	Workstation Group	–
2	Top Table Part 1	5	11	Workstation Group	–
3	Top Table Part 2	2	18	Automated Station	TRUE
4	Lower Door	6	25	Workstation Group	–
5	Pipe Cover	6	11	Workstation Group	–
6	Evaporator Part 1	7	32	Workstation Group	–
7	Evaporator Part 2	3	38	Automated Station	TRUE
8	Compressor Part 1	7	44	Workstation Group	–
9	Compressor Part 2	4	37	Automated Station	TRUE
10	Power Supply Part 1	6	23	Workstation Group	–
11	Power Supply Part 2	8	22	Workstation Group	–
12	Leg Component	9	28	Workstation Group	–
13	Light and Switch	9	39	Workstation Group	–

evaluating structured approaches. The Empty strategy prioritizes selecting the workstation with the smallest load, thereby balancing resource utilization and minimizing bottlenecks. The FIFO strategy schedules tasks based on their arrival order, aiming to account for time-based costs as much as possible but lacking optimization for long-term returns.

The data in Fig. 14 and Table 5 clearly illustrates that the average reward achieved by PPO significantly surpasses that of rule-based methods. In terms of adaptability and performance, PPO achieves the highest average reward (44.16), significantly outperforming all rule-based and reinforcement learning methods, such as FIFO (35.83), EMPTY (36.34), and RANDOM (14.85). This result highlights reinforcement learning method, such as PPO's ability to optimize long-term objectives by incorporating future returns into its decision-making process. Unlike FIFO, which focuses on immediate resource allocation efficiency, PPO can dynamically adjust to changes such as workstation failures and system bottlenecks, demonstrating a stronger capacity for handling uncertainty and complex scheduling challenges. The higher performance of PPO also emphasizes its effective Reward-Shaping strategy, which enables it to balance trade-offs between short-term efficiency and long-term profitability.

The fixed decision logic of rule-based methods prevents them from adapting to real-time changes, such as resource imbalances or bottlenecks, which reinforces their limitations in dynamic scheduling scenarios. RANDOM, with the lowest reward (14.85), serves as a baseline, illustrating the inefficiency of non-strategic decision-making in complex systems.

And if stability is taken as the primary condition, despite its superior average performance, PPO shows a higher standard deviation (5.14) compared to A2C (4.51) and the deterministic FIFO method (4.26). This variability is largely attributed to PPO's aggressive approach to maximizing workstation utilization, which, while increasing average rewards, introduces fluctuations in performance under scenarios with high uncertainty in disassembly times. By contrast, A2C achieves a slightly lower reward (41.22) but with better stability, suggesting that its policy-gradient approach effectively balances exploration and exploitation, making it more suited for scenarios prioritizing operational stability over absolute performance.

The comparison of methods underscores an important trade-off: while PPO delivers the highest rewards by leveraging future-oriented optimization and dynamic adaptability, this comes at the cost of increased variability. A2C offers a compromise, achieving competitive rewards with better stability, making it preferable for applications that require a balance between operational consistency and performance. In contrast, rule-based methods may be more suitable for simpler or more predictable systems where adaptability is less critical.

At the beginning of the testing phase, reinforcement learning methods experienced a brief period of lower rewards, due to the initial state in which all workstation buffers were empty. The agent required several steps to adjust buffer occupancy, bringing the system to a state capable of sustaining higher long-term rewards. However, the methods quickly reached the average reward levels seen at the end of training, demonstrating the strong adaptability of reinforcement learning approaches. Overall, the PPO algorithm demonstrates clear advantages in complex scenarios, particularly in terms of higher training efficiency and stability. In contrast, both DQN and A2C offer weaker final rewards compared to PPO, and DQN exhibits greater volatility during the training process. In practical testing, the value-based DQN method shows a notably weaker ability to handle uncertainty compared to policy-based methods. Although A2C, which is also a policy-based reinforcement learning method, performs better than DQN, it falls short of PPO in both final outcomes and convergence speed.

5. Conclusion

This study presents a novel reinforcement learning-based scheduling approach that directly addresses the pressing challenges outlined in the introduction. As global e-waste generation accelerates, traditional disassembly systems struggle with the complexities of unpredictable product conditions, workstation failures, and real-time resource allocation. By leveraging an enhanced Proximal Policy Optimization (PPO) algorithm with Reward-Shaping, this research introduces a dynamic scheduling framework capable of significantly improving the efficiency

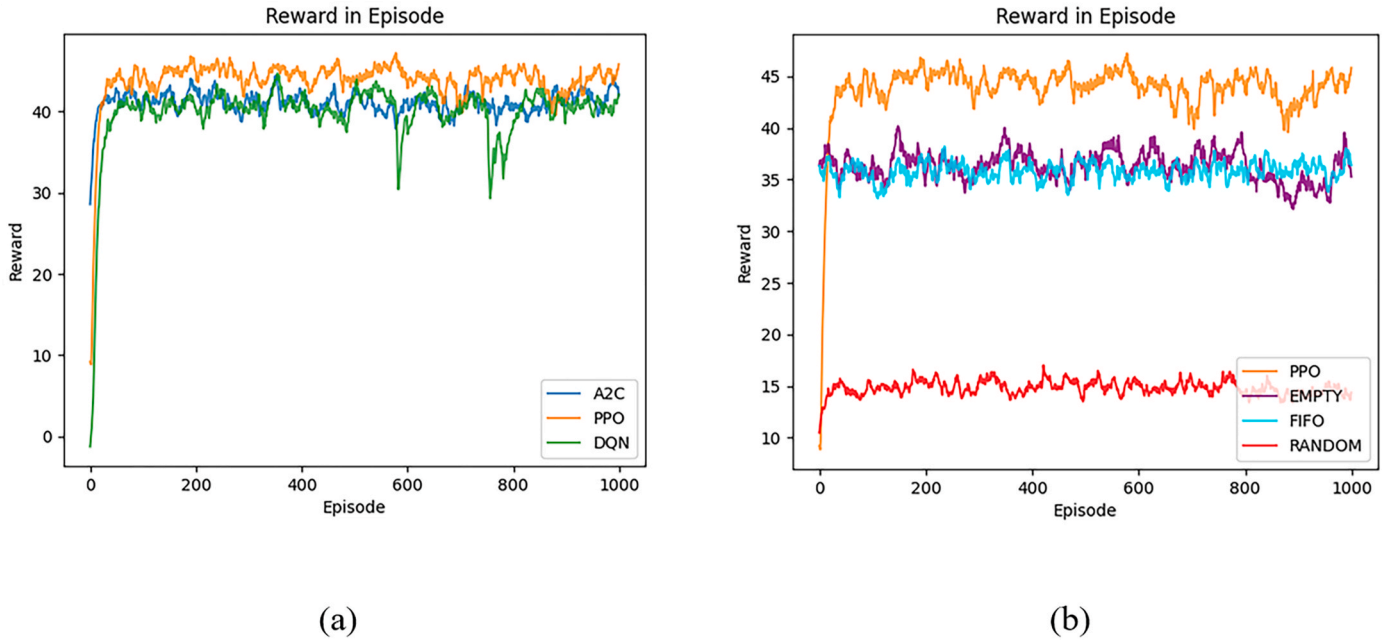

Fig. 14. Comparison of results from a PPO algorithm enhanced with Reward-Shaping in complex scenarios against other reinforcement learning methods and rule-based approaches.

Table 5
Mean and standard deviation (Std)of each method.

Method	Mean	Std
PPO	44.16	5.14
A2C	41.22	4.51
DQN	40.22	5.97
EMPTY	36.34	4.97
FIFO	35.83	4.26
RANDOM	14.85	2.71

and adaptability of disassembly operations. The proposed method directly responds to the limitations of traditional meta-heuristic and rule-based approaches, which lack the flexibility to handle the uncertainties inherent in hybrid disassembly environments.

By integrating reinforcement learning into a hybrid disassembly system that combines manual and automated workstations, this study demonstrates how intelligent scheduling can optimize system throughput, minimize delays, and enhance resource utilization. Unlike conventional automated systems, which struggle to adapt to unexpected failures or variations in product conditions, the proposed method dynamically reallocates resources, ensuring operational continuity even under uncertainty. This adaptability is particularly crucial for industries aiming to enhance sustainability and profitability in remanufacturing.

Furthermore, the findings underscore the potential for reinforcement learning to become a core technology in future industrial disassembly applications. The ability of PPO to outperform both rule-based scheduling and alternative reinforcement learning methods highlights its suitability for real-world implementation. By reducing operational time and maximizing the balance between cost and efficiency, this research provides a foundation for developing next-generation disassembly systems that align with circular economy principles. The advancements presented here address the immediate challenges of e-waste processing.

During our experiments, we found that when faced with more complex and multi-node real-world scenarios, individual intelligences are gradually unable to perfectly cope with the scheduling demands of the system as a whole at the temporal level. Future work will build on multi-agent research foundations to explore integrating more complex product types and further refine the Reward-Shaping mechanism to enhance real-time decision-making in industrial environments.

CRediT authorship contribution statement

Jinlong Wang: Supervision, Resources, Project administration, Conceptualization. **Qihuiyang Liang:** Writing – original draft, Methodology, Investigation, Conceptualization. **Min Li:** Investigation, Formal analysis. **Zelin Qu:** Investigation, Formal analysis. **Yuanyuan Zhang:** Writing – review & editing, Validation, Supervision, Conceptualization.

Declaration of competing interest

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

Acknowledgments

The authors thank the anonymous referees for their many valuable and helpful suggestions. This work was supported by Research and Application Demonstration of the Full-Process Digital Control Technology for the Recycling of Home Appliance Products in Industrial Internet (No. 24-1-2-qljh-25-gx).

Data availability

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

References

Allagui, A., 2023. Reinforcement learning for disassembly sequence planning optimization. Comput. Ind. 151, 103992. https://doi.org/10.1016/j.compind.2023.103992.
Altekin, F. T., Akkan, C., 2012. Task-failure-driven rebalancing of disassembly lines. Int. J. Prod. Res. 50 (18), 4955–4976. https://doi.org/10.1080/00207543.2011.616915.
Chand, M., Ravi, C., 2023. A state-of-the-art literature survey on artificial intelligence techniques for disassembly sequence planning. CIRP Journal of Manufacturing Science and Technology 41, 292–310. https://doi.org/10.1016/j.cirpj.2022.11.017.
Chen, W.H., Vongbunyong, S., 2015. Disassembly automation: automated systems with cognitive abilities. Sustainable production.

Colledani, M., Battaïa, O., Jan. 2016. A decision support system to manage the quality of end-of-life products in disassembly systems. CIRP Ann. 65 (1), 41–44. https://doi.org/10.1016/j.cirp.2016.04.121.

Colorni, A., Dorigo, M., Maniezzo, V., 1991. Distributed optimization by ant colonies. In: Proceedings of the First European Conference on Artificial Life, pp. 134–142. Paris, FranceAccessed: Oct. 08, 2024. [Online]. Available: https://faculty.washington.edu/paymana/swarm/colorni92-ecal.pdf.

Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43. https://doi.org/10.1109/MHS.1995.494215.

Gao, K. Z., He, Z. M., Huang, Y., Duan, P. Y., Suganthan, P. N., 2020. A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing. Swarm Evol. Comput. 57, 100719. https://doi.org/10.1016/j.swevo.2020.100719.

Glöser-Chahoud, S., et al., 2021. Industrial disassembling as a key enabler of circular economy solutions for obsolete electric vehicle battery systems. Resour. Conserv. Recycl. 174, 105735. https://doi.org/10.1016/j.resconrec.2021.105735.

Han, M., Yun, L., Li, L., 2023. Deep reinforcement learning-based approach for dynamic disassembly scheduling of end-of-life products with stimuli-activated self-disassembly. J. Clean. Prod. 423, 138758. https://doi.org/10.1016/j.jclepro.2023.138758. Oct.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034. Accessed: Jan. 31, 2025. [Online]. Available: http://openaccess.thecvf.com/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html.

Hu, Y., Liu, C., Zhang, M., Jia, Y., Xu, Y., 1652, Feb. 2023. A novel simulated annealing-based Hyper-Heuristic Algorithm for stochastic parallel disassembly line balancing in smart remanufacturing. Sensors 23 (3). https://doi.org/10.3390/s23031652.

Junior, M.L., Filho, M.G., 2012. Production planning and control for remanufacturing: literature review and analysis. Prod. Plann. Control 23 (6), 419–435. https://doi.org/10.1080/09537287.2011.561815.

Karaboga, D., 2005. An idea based on honey bee swarm for numerical optimization, [Online]. Available: http://abc.erciyes.edu.tr/pub/tr06_2005.pdf (Accessed 15 March 2025).

Kim, H.-J., Harms, R., Seliger, G., Apr. 2007. Automatic control sequence generation for a hybrid disassembly system. IEEE Trans. Autom. Sci. Eng. 4 (2), 194–205. https://doi.org/10.1109/TASE.2006.880538.

Laud, A.D., 2004. Theory and Application of Reward Shaping in Reinforcement Learning. University of Illinois at Urbana-Champaign [Online]. Available: https://search.proquest.com/openview/bb29dc3d66eccbe7ab65560dd2c4147f/1?pq-origsite=gscholar&cbl=18750&diss=y. (Accessed 25 October 2024).

Liang, W., Zhang, Z., Yin, T., Zeng, Y., Zhang, Y., 2024. Multi-parallel disassembly line balancing problem and improved ant lion optimizer for mixed-waste electrical and electronic equipment. Int. J. of Precis. Eng. and Manuf.-Green Tech. 11 (1), 243–258. https://doi.org/10.1007/s40684-023-00525-4.

Liu, R., 2022. Deep Reinforcement Learning-Based Dynamic Scheduling. https://doi.org/10.32657/10356/158353.

Luo, S., 2020. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. Appl. Soft Comput. 91, 106208. https://doi.org/10.1016/j.asoc.2020.106208.

Poschmann, H., Brüggemann, H., Goldmann, D., 2020. Disassembly 4.0: a review on using robotics in disassembly tasks as a way of automation. Chem. Ing. Tech. 92 (4), 341–359. https://doi.org/10.1002/cite.201900107.

Rizova, M.I., Wong, T.C., Ijomah, W., 2020. A systematic review of decision-making in remanufacturing. Comput. Ind. Eng. 147, 106681. https://doi.org/10.1016/j.cie.2020.106681.

Sampson, J.R., 1976. Adaptation in natural and artificial systems (john H. Holland). SIAM Rev. 18 (3), 529–530. https://doi.org/10.1137/1018105.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms, arXiv:1707.06347. doi: 10.48550/arXiv.1707.06347.

Tian, G., Zhou, M., Li, P., 2018. Disassembly sequence planning considering fuzzy component quality and varying operational cost. IEEE Trans. Autom. Sci. Eng. 15 (2), 748–760. https://doi.org/10.1109/TASE.2017.2690802.

Ullah, M., Asghar, I., Zahid, M., Omair, M., AlArjani, A., Sarkar, B., 2021. Ramification of remanufacturing in a sustainable three-echelon closed-loop supply chain management for returnable products. J. Clean. Prod. 290, 125609. https://doi.org/10.1016/j.jclepro.2020.125609.

Wang, K., Li, X., Gao, L., Li, P., 2021. Modeling and balancing for green disassembly line using associated parts precedence graph and multi-objective genetic simulated annealing. Int. J. of Precis. Eng. and Manuf.-Green Tech. 8 (5), 1597–1613. https://doi.org/10.1007/s40684-020-00259-7.

Wang, K., Li, X., Gao, L., Li, P., Sutherland, J.W., 2022. A discrete artificial bee Colony algorithm for multiobjective disassembly line balancing of end-of-life products. IEEE Trans. Cybern. 52 (8), 7415–7426. https://doi.org/10.1109/TCYB.2020.3042896.

Wang, K., Li, Y., Guo, J., Gao, L., Li, X., 2024a. Dynamic balancing of U-shaped robotic disassembly lines using an effective deep reinforcement learning approach. IEEE Trans. Ind. Inf. 20 (4), 6855–6865. https://doi.org/10.1109/TII.2023.3348811.

Wang, J., Xi, G., Guo, X., Liu, S., Qin, S., Han, H., 2024b. Reinforcement learning for hybrid disassembly line balancing problems. Neurocomputing 569, 127145. https://doi.org/10.1016/j.neucom.2023.127145.

Wurster, M., Michel, M., May, M.C., Kuhnle, A., Stricker, N., Lanza, G., 2022. Modelling and condition-based control of a flexible and hybrid disassembly system with manual and autonomous workstations using reinforcement learning. J. Intell. Manuf. 33 (2), 575–591. https://doi.org/10.1007/s10845-021-01863-3.

Yuan, E., Wang, L., Cheng, S., Song, S., Fan, W., Li, Y., 2024. Solving flexible job shop scheduling problems via deep reinforcement learning. Expert Syst. Appl. 245, 123019. https://doi.org/10.1016/j.eswa.2023.123019.

Zhang, L., et al., 2023. Deep reinforcement learning for dynamic flexible job shop scheduling problem considering variable processing times. J. Manuf. Syst. 71, 257–273. https://doi.org/10.1016/j.jmsy.2023.09.009.

Zhao, X., Li, C., Tang, Y., Cui, J., 2021. Reinforcement learning-based selective disassembly sequence planning for the end-of-life products with structure uncertainty. IEEE Rob. Autom. Lett. 6 (4), 7807–7814. https://doi.org/10.1109/LRA.2021.3098248.