Dynamic Balancing of U-Shaped Robotic Disassembly Lines Using an Effective Deep Reinforcement Learning Approach

Kaipu Wang, Yibing Li, Jun Guo, Liang Gao, *Senior Member, IEEE*, and Xinyu Li, *Member, IEEE*

***Abstract*—Disassembly line balancing (DLB) is used for efficient task planning of large-scale end-of-life products, which is a key issue to realize resource recycling and reuse. Robot disassembly and U-shaped station layout can effectively improve disassembly efficiency. To accurately characterize the problem, a mixed-integer linear programming model of U-shaped robotic DLB is proposed. The aim is to minimize the cycle time to shorten the offline time of the product. Since there are many dynamic disturbances in the actual disassembly line, and traditional optimization methods are suitable for dealing with static problems, this article develops a deep reinforcement learning approach based on problem characteristics, namely deep Q network (DQN), to achieve a dynamic balancing of disassembly lines. Eight state features and ten heuristic action rules are designed in the proposed DQN to describe the disassembly environment completely. The effectiveness and superiority of the proposed DQN are verified by numerical experiments. In the case of a laptop disassembly line, not only the cycle time of the robots is reduced, but also intelligent decision-making and dynamic planning of disassembly tasks are realized.**

***Index Terms*—Deep reinforcement learning (DRL), disassembly line balancing (DLB), dynamic balancing, mixed-integer linear programming (MILP).**

Manuscript received 15 September 2023; revised 10 December 2023; accepted 27 December 2023. Date of publication 25 January 2024; date of current version 4 April 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 52305552 and Grant U21B2029, in part by the Hubei Provincial Natural Science Foundation of China under Grant 2023AFB138, in part by the Knowledge Innovation Program of Wuhan-Shuguang Project under Grant 2023020201020322, and in part by the Fundamental Research Funds for the Central Universities under Grant WUT233104001. Paper no. TII-23-3561. *(Corresponding author: Xinyu Li.)*

Kaipu Wang, Yibing Li, and Jun Guo are with the Hubei Digital Manufacturing Key Laboratory, School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan 430070, China (e-mail: wangkaipu@whut.edu.cn; ahlyb@whut.edu.cn; junguo@whut.edu.cn).

Liang Gao and Xinyu Li are with the State Key Laboratory of Intelligent Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: gaoliang@mail.hust.edu.cn; lixinyu@mail.hust.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TII.2023.3348811.

Digital Object Identifier 10.1109/TII.2023.3348811

I. Introduction

THE disassembly line balancing (DLB) is one of the key issues to be solved in the process of disassembling large-scale end-of-life products by disassembly enterprises [1]. Reasonable disassembly task planning can not only improve disassembly efficiency but also realize resource recycling and reduce the impact of harmful substances on the environment [2]. Therefore, DLB has always been concerned and valued by governments, manufacturers, and environmentalists [3].

Most traditional disassembly lines adopt manual disassembly. Compared with manual disassembly, robotic disassembly has high precision and flexibility [4], which can reduce the cost of manual disassembly and improve the working conditions of workers [5]. Unlike the conventional linear layout, the operator in the U-shaped layout can disassemble the tasks on both the entrance and exit sides of the product line [6], [7], to shorten the product offline time and improve the disassembly efficiency [8]. Considering the flexibility of robot operation, a U-shaped station layout can be adopted in the disassembly line to improve the utilization rate of robots and production lines [9], [10]. The cycle time should be reduced as much as possible when the number of robots is certain. Therefore, this article will focus on U-shaped robotic DLB (URDLB).

DLB is an NP-hard optimization problem [11], while URDLB is more complex than DLB. The optimization methods of DLB mainly include exact methods [12], [13], [14], [15], [16], heuristic methods [17], [18], and meta-heuristic algorithms [19], [20]. Existing research mostly focuses on meta-heuristic algorithms, such as genetic algorithm (GA) [1], ant colony optimization (ACO) [8], and artificial bee colony (ABC) [21]. The prerequisite for applying the above methods is that the data information of the problem is known, that is, the problem is static. However, there are many dynamic disturbances in the actual disassembly process [22], resulting in uncertainty in the disassembly time in the disassembly line. Conventional disassembly lines are static, that is, the disassembly time is determined. When the disassembly time changes, the original disassembly scheme is no longer applicable, and the disassembly tasks need to be replanned to achieve the dynamic balancing of the disassembly line. If an exact algorithm or meta-heuristic algorithm is used in this case, reiteration is required, and the running time is long.

TABLE I
COMPARISON OF DIFFERENT RL ALGORITHMS

Refs.	Problem	Method	State	Action
Tuncel et al. [24]	DLB	QL	Disassembly task	Disassembly task
Allagui et al. [25]	DSP	QL	Disassembly sequence	Disassembly part
Liu et al. [26]	DLB	QL	Disassembly task	Disassembly task
Mete and Serin [27]	DLB	QL	Disassembly task	Disassembly task
Xia et al. [28]	DSP	QL	Disassembly plan	Disassembly operation
Mao et al.[29]	DSP	DQN	Disassembly part	Disassembly operation
Zhao et al.[30]	DSP	DQN	{Precedence, contact, and level}	Disassembly part
Liu et al.[31]	DSP	DQN	Three component states	Disassembly component
Mei and Fang [32]	DLB	DQN	{Robot, resource, task, and execution}	Operation of robot
Zhang et al. [33]	HRC	DDPG	{Human, robot, and execution}	Assembly task
Cai et al.[34]	DLB	A2C	{Subassembly and station}	Disassembly task
Zhong et al. [35]	DLB	PPO	{Subassembly and station}	Disassembly subassembly
This article	DLB	DQN	{Precedence, station, robot, task, time, and execution}	Disassembly task based on heuristic rules

Therefore, it is essential to propose an efficient algorithm that can adapt to the dynamic disassembly environment.

URDLB is a sequential decision problem. The disassembly system can be divided into multiple decision moments according to the number of disassembly tasks, and decisions are made according to the current state of the disassembly system. Therefore, the URDLB problem can be regarded as a Markov decision process (MDP). Reinforcement learning (RL) has become an effective method to deal with MDP [23], which can explain the interaction between agent and environment. For example, Tuncel et al. [24] proposed a Q-learning (QL) algorithm for DLB and used the Q table to record the state-action value, which could obtain a better disassembly scheme. Allagui et al. [25] and Liu et al. [26] used disassembly tasks to represent actions and states in the QL. However, the dimension of the Q table cannot be increased infinitely, and the state space is limited [27], [28].

Deep Q-network (DQN) is a RL algorithm based on deep learning, namely deep RL (DRL), which uses neural networks to approximate the Q-function and solves the limitations of QL in dealing with high-dimensional state space problems. Moreover, DQN can utilize the trained neural network to make fast decisions in dynamic environments. For example, Mao et al. [29] proposed a DQN to implement adaptive disassembly sequence planning (DSP) for virtual reality maintenance training. Zhao et al. [30] introduced a DQN to adapt the optimal disassembly sequence for selective DSP. Similarly, Liu et al. [31] proposed a DQN to handle uncertain missing conditions. Mei and Fang [32] constructed a DQN with four states to solve the multirobotic DLB, which directly regarded the assignment of disassembly tasks as the action of DQN. In addition, RL algorithms with different neural networks are applied to optimize production planning problems, such as deep deterministic policy gradient (DDPG) [33] for human-robot collaborative (HRC), advantage actor-critic (A2C) [34], and proximal policy optimization (PPO) [35].

Although these methods can optimize the disassembly planning problem, the actions and states do not fully characterize the disassembly environment, and there are fewer dynamic balancing strategies. The comparison between the above algorithms is given in Table I. The status and actions of the algorithm will directly affect the performance of the algorithm. Thus, it is crucial to design an efficient DQN according to the disassembly environment characteristics.

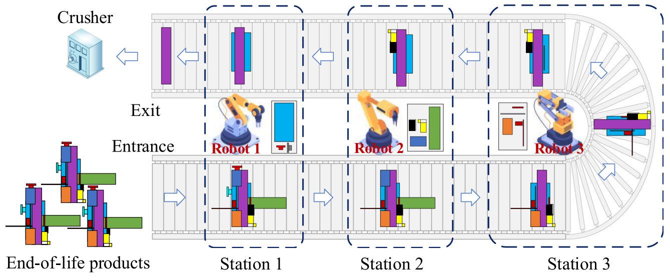

Fig. 1. Layout of the U-shaped robotic disassembly line.

Based on the above motivations, the main contributions of this article are as follows.

1) A mixed-integer linear programming (MILP) model of URDLB is proposed, and an exact solver is used to verify the correctness of the model.
2) A DRL approach combined with disassembly characteristics is developed, including eight state features and ten heuristic actions, to deal with the dynamic disassembly disturbances.
3) The effectiveness and efficiency of the proposed model and method are verified by numerical experiments and an engineering case.

The rest of this article is organized as follows. Section II describes the mathematical model of URDLB. Section III presents the designed DQN. Section IV is the numerical experiments. Section V is an engineering application. Finally, Section VI concludes this article.

II. PROBLEM STATEMENT

A. Problem Description

In the U-shaped robotic disassembly line, the number of robots is fixed, and different robots have different working time on different disassembly tasks. Each robot can be assigned to any station, and only one robot can be assigned to each station at most. To shorten the product offline time, the optimization objective is to minimize the cycle time of the disassembly line. The disassembly line transmits the tasks in the stations according to the cycle time, which is divided into multiple stations, and the maximum working time of each station is the cycle time. The layout of the U-shaped robotic disassembly line is shown in Fig. 1. The three robots are distributed in the middle of the U-shaped line, which can disassemble the tasks on the entrance and exit sides.

There are two challenges to be solved in the URDLB. First, how to reasonably assign the disassembly tasks to the entrance and exit sides of the U-shaped stations, so that all stations have the same time as much as possible. Second, how to reasonably select robots to be assigned to each station, and the working time of the station is different when different types of robots are selected. Thus, it is necessary to reasonably combine robot assignments with task assignments. Moreover, the dynamic uncertainties of end-of-life products in the disassembly line include corrosion, deformation, damage, missing parts, etc. These uncertainties will make the disassembly time change. In this study, the dynamic uncertainty of the URDLB model is mainly concerned with the disassembly time. The planning principle remains unchanged in the dynamic environment.

To facilitate the construction of the mathematical model, the following assumptions are given in this article.

1) The disassembly information is known, including the precedence relationships between disassembly tasks, and the number of robots.
2) The type of disassembly product is single and the quantity of disassembly product is sufficient.
3) Line failures and some parameters, such as setup times of robots and conveyor belt speeds, are not considered.

B. Mathematical Model

The symbols in the mathematical model are shown below.

i	Disassembly task, $i \in I$, where I is the disassembly task set, and $\lvert I \rvert$ is the maximum number of tasks.
w	Station, $w \in W$, where W is the station set, and $\lvert W \rvert$ is the maximum number of stations, $\lvert W \rvert \leq \lvert I \rvert$.
m	Robot or mechanical arm, $m \in M$, where M is the robot set, and $\lvert M \rvert$ is the maximum number of robots.
p_{ij}	The precedence relationship between tasks: 1, task i is the immediate predecessor of task j; 0, otherwise.
t_{im}	Disassembly time of task i in robot m.

The decision variables in the mathematical model are shown below.

x_{iw}	Task assignment variable: 1, task i is assigned to station w; 0, otherwise.
y_w	Station state variable: 1, station w is opened; 0, otherwise.
T_C	Cycle time of the disassembly line (non-negative variable)
z_{mw}	1, robot m is assigned to station w; 0, otherwise.
u_i	1, task i is assigned to the entrance side of the U-line; 0, task i is assigned to the exit side of the U-line.

The mathematical model of the URDLB is as follows:

$$\min \quad f = T_C \tag{1}$$

$$\text{s.t.} \quad \sum_{w\in W} x_{iw} = 1 \ \ \forall i \in I \tag{2}$$

$$\sum_{m\in M} z_{mw} = y_w \ \ \forall w \in W \tag{3}$$

$$\sum_{w\in W} z_{mw} \leq 1 \ \forall m \in M \tag{4}$$

$$\sum_{m\in M}\sum_{i\in I} x_{iw} z_{mw} t_{im} \leq T_C \ \ \forall w \in W \tag{5}$$

$$y_w \leq \sum_{i\in I} x_{iw} \leq |I|\, y_w \ \ \forall w \in W \tag{6}$$

$$y_w \geq y_{w+1} \ \ \forall w \in \{1, \ldots, |W|-1\} \tag{7}$$

$$\sum_{w\in W} w\cdot x_{iw} - \sum_{w\in W} w\cdot x_{jw} - |W|\,(1-p_{ij}) \leq |W|\,(1-u_j) \ \ \forall i,j \in I,\ i \neq j \tag{8}$$

$$\sum_{w\in W} w\cdot x_{jw} - \sum_{w\in W} w\cdot x_{iw} - |W|\,(1-p_{ij}) \leq |W|\, u_i \ \ \forall i,j \in I,\ i \neq j \tag{9}$$

$$u_i \geq u_j - (1-p_{ij}) \ \ \forall i,j \in I,\ i \neq j \tag{10}$$

$$x_{iw},\, y_w,\, z_{mw},\, u_i \in \{0,\, 1\}\,,\, T_C \geq 0 \ \ \forall i \in I\, \forall w \in W\, \forall m \in M. \tag{11}$$

Equation (1) shows that the optimization objective is to minimize the cycle time of the disassembly line, and its purpose is to shorten the product off-line time. Equation (2) shows that all tasks are disassembled. Equation (3) indicates that at most one robot is assigned to each station. Equation (4) indicates that not all robots are assigned. Equation (5) is the cycle time constraint, that is, the disassembly time of any one station should not exceed the cycle time. Equation (6) indicates that the number of tasks in the opened station is not less than one, and does not exceed the total number of tasks. Equation (7) shows that the stations are opened in sequence. Equations (8)–(10) ensure that the disassembly sequence satisfies precedence relationships. Equation (11) gives the decision variables.

Equation (5) is a nonlinear expression. To linearize it, a 0-1 variable χ_{imw} is introduced, and $\chi_{imw} = x_{iw}{\cdot}z_{mw}$. Equation (5) can be rewritten as

$$\sum_{m\in M}\sum_{i\in I} \chi_{imw} t_{im} \leq T_C \quad \forall w \in W. \tag{12}$$

The constraints of variable χ_{imw} are shown in (13)–(15).

$$2\chi_{imw} \leq x_{iw} + z_{mw} \ \ \forall i \in I\, \forall m \in M\, \forall w \in W \tag{13}$$

$$x_{iw} + z_{mw} - \chi_{imw} \leq 1 \ \ \forall i \in I\, \forall m \in M \ \ \forall w \in W \tag{14}$$

$$\chi_{imw} \in \{0,\ 1\} \ \ \forall i \in I\, \forall m \in M\, \forall w \in W. \tag{15}$$

To sum up, the MILP model of the URDLB problem can be expressed as follows.

Objective: (1);

Constraints: (2)–(4), (6)–(15).

C. Model Validation and Examples

To verify the correctness of the MILP model, the CPLEX solver is used to calculate the model, and the characteristics of the problem are explained through the disassembly scheme. There are four different types of robots, and the disassembly time and precedence relationships are shown in Appendix.

To compare the performance of straight and U-shaped lines, CPLEX is used to optimize the two models respectively. In the straight robotic DLB (SRDLB) model, the task assignment constraints on the entrance and exit sides are not considered. In this case, only constraints (8)–(10) need to be changed to constraint (16), and other constraints remain unchanged

$$\sum_{w\in W} w\cdot x_{iw} - |W|\,(1-p_{ij}) \leq \sum_{w\in W} w\cdot x_{jw},\ \forall i,j \in I,\ i \neq j. \tag{16}$$

The cycle time of the SRDLB is 11, and the running time is 0.56 s; the cycle time of the URDLB is 10, and the running time is 0.30 s. The Gantt chart of the disassembly scheme of the two layout forms is shown in Fig. 2. The numbers in the rectangles

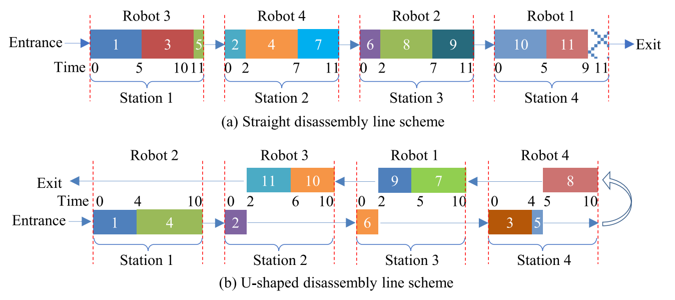

Fig. 2. Gantt chart of the disassembly scheme of the two layout forms.

are the task numbers, and the numbers outside the rectangles are the task operation time.

In Fig. 2(b), robot 3 in the second station can disassemble task 2 on the entrance side and tasks 10 and 11 on the exit side, and the same for the third and fourth stations, while the robot can only disassemble task on one side in Fig. 2(a). Comparing the two disassembly schemes in Fig. 2, it can be seen that the cycle time of the U-shaped disassembly line is shorter than that of the straight disassembly line, indicating that the U-shaped layout can shorten the offline time of products. In addition, the disassembly scheme satisfies all constraints, indicating that the MILP model in Section II-B is correct.

III. Proposed DRL for URDLB

The data-driven intelligent disassembly system can collect the disassembly states and data of the disassembly line. The optimal task decision of the disassembly line can be obtained by analyzing the states and data and combining with the existing historical data. The disassembly system is regarded as an agent, and the MDP is made on task allocation to realize the intelligent decision and dynamic balancing of the URDLB problem.

A. Reinforcement Learning

The mathematical basis of RL is the MDP, which is usually represented by a tuple $\{S, A, P, \gamma, R\}$ [23], where: S represents the state space of the disassembly system, and the disassembly state $s \in S$; A represents the action space of the disassembly system, and disassembly action $a \in A$; P represents the state transition probability function, that is, the probability of selecting the action a from the state s; γ represents the reward discount factor; R represents the reward function, that is, the reward obtained after performing the disassembly action a.

The interaction between the agent and the environment is that the agent observes the disassembly state s of the disassembly environment and then makes a disassembly action a. The action a changes the state s of the environment, and the environment feeds back to the agent a reward r and a new state s'. The RL aims to find the optimal policy π between the state s and the action a in a given MDP, to maximize the expected value of the accumulated reward.

The discounted return, the state value function, and the action-value function at time t under the policy π are shown in (17)–(19) [23], respectively,

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{17}$$

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \sum_a \pi(a|s) \sum_{s'} p(s'|s, a)[r + \gamma V_\pi(s')] \tag{18}$$

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \sum_{s'} p(s'|s, a)[r + \gamma Q_\pi(s', a')] \tag{19}$$

where s' is the subsequent new state of the current state s, and a' is the subsequent new action of the current action s.

The expression of the optimal strategy π^* is shown in (20), and the optimal state-value function $V^*(s)$ and action-value function $Q^*(s, a)$ are shown in (21) and (22) [23]

$$\pi^*(a|s) = \arg\max_\pi Q_\pi(s, a) \tag{20}$$

$$V^*(s) = \max_a \sum_{s'} p(s'|s, a)[r + \gamma V^*(s')] \ \forall s \tag{21}$$

$$Q^*(s, a) = \sum_{s'} p(s'|s, a)\left[r + \gamma \max_{a'} Q^*(s', a')\right] \forall s \,\forall a. \tag{22}$$

B. Deep Q-Network

DQN combines DL and RL, uses a deep neural network to predict $Q(s, a; \theta)$ to approximate $Q^*(s, a)$, and uses experience replay to train the RL, where θ is the parameter in the neural network. Equation (23) is used in RL to update the action-value function. In DQN, the network $Q(s, a; \theta)$ is used to replace $Q(s, a)$ in RL, and updating $Q(s, a; \theta)$ is essentially updating the parameter θ. The action-value function $Q(s, a; \theta)$ of DQN is shown in (24) [36]

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a)\right] \tag{23}$$

$$Q(s, a; \theta) \approx r + \gamma \max_a Q(s', a; \theta). \tag{24}$$

The training of the parameter θ is achieved by minimizing the loss function between the target Q value and the predicted Q value, using gradient descent and error backpropagation to update the neural network parameter θ. There are two neural networks with similar structures and different parameters in DQN, namely the target network $Q(s', a'; \theta^-)$ and the prediction network $Q(s, a; \theta)$. The parameters of $Q(s, a; \theta)$ are updated every episode, and the parameters of $Q(s, a; \theta)$ are assigned to $Q(s', a'; \theta^-)$ after every C steps of training. The expression of parameter update is shown in (25) [36]. The framework of the proposed

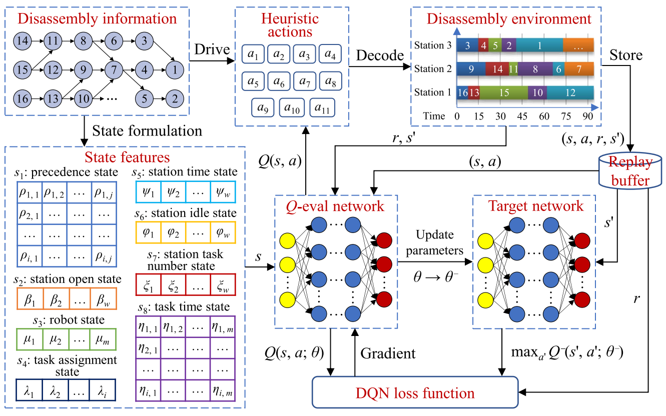

Fig. 3. Framework of the proposed DQN.

TABLE II
STATE FEATURES OF THE DISASSEMBLY ENVIRONMENT

No.	Expression	State description
s_1	$\rho_{ij} = \sum_{w\in W}(1-x_{iw})p_{ij}, \forall i,j \in I$	Precedence relationship state
s_2	$\beta_w = y_w, \forall w \in W$	Station open state
s_3	$\mu_m = \sum_{w\in W} z_{mw}, \forall m \in M$	Robot state
s_4	$\lambda_i = \sum_{w\in W} x_{iw}, \forall i \in I$	Task assignment state
s_5	$\psi_w = \sum_{m\in M}\sum_{i\in I} x_{iw} z_{mw} t_{im} \Big/ \sum_{m\in M}\sum_{i\in I} t_{im}, \forall w \in W$	Station time state
s_6	$\varphi_w = 1 - \sum_{m\in M}\sum_{i\in I} x_{iw} z_{mw} t_{im} \Big/ \sum_{m\in M}\sum_{i\in I} t_{im}, \forall w \in W$	Station idle state
s_7	$\xi_w = \sum_{i\in I} x_{iw} \big/ \lvert I \rvert, \forall w \in W$	Station task number state
s_8	$\eta_{im} = \sum_{w\in W}(1-x_{iw}) t_{im} \Big/ \sum_{m\in M}\sum_{i\in I} t_{im}, \forall i \in I, m \in M$	Task time state

DQN is shown in Fig. 3

$$\theta_{t+1} = \theta_t + \alpha \left[r + \gamma \max_{a'} Q\left(s',\ a';\ \theta^-\right) - Q\left(s,\ a;\ \theta\right) \right]$$
$$\nabla Q\left(s,\ a;\ \theta\right). \tag{25}$$

C. State Features

The state features of the disassembly system are the numerical representation of the state variables, reflecting the main characteristics of the disassembly environment. The state features cover eight kinds of state information, including station state, precedence relationship state, robot state, and task state. The dimension of the state matrix is related to the number of disassembly tasks and the parts of the robot state and station state matrix that are less than the number of disassembly tasks are filled with 0. All state element values are between [0, 1]. The precedence relationship state reflects the precedence constraints between tasks. When a task is assigned, the precedence constraints between it and other tasks are automatically released, and the precedence relationship is updated at the same time, that is, the precedence relationship state changes. The descriptions of the eight state features of the disassembly environment are given in Table II.

D. Action

Since the disassembly task information is not directly related to the state features, if the task assignment is directly used as the action of the disassembly system, it is difficult to improve the learning efficiency of the agent. Referring to the action design method in RL for solving scheduling problems [37] and heuristic algorithms in operations research, various heuristic rules in the disassembly line are designed as actions. The efficiency of machine learning can be improved by adaptively selecting the appropriate heuristic actions according to the disassembly states. When the disassembly environment changes, the disassembly states and actions will also change. The specific actions performed by the robot are obtained through RL training.

TABLE III
HEURISTIC ACTIONS FOR THE DISASSEMBLY SYSTEM

No.	Expression	Action description	Symbol
a_1	$i = \arg\max_{i\in Set,\, m\in M} \{t_{im}\}$	Select the task with the longest disassembly time	TLT
a_2	$i = \arg\min_{i\in Set,\, m\in M} \{t_{im}\}$	Select the task with the shortest disassembly time	TST
a_3	$i = \arg\max_{i\in Set} \left\{t_{im} + \sum_{m\in M}\sum_{j\in I} x_{jw} z_{mw} t_{jm}\right\}$	Select the task that maximizes the station time	TXT
a_4	$i = \arg\min_{i\in Set} \left\{t_{im} + \sum_{m\in M}\sum_{j\in I} x_{jw} z_{mw} t_{jm}\right\}$	Select the task that minimizes the station time	TNT
a_5	$i = \arg\max_{i\in Set} \{n_i^p\}$	Select the task with the maximum number of predecessors	TXP
a_6	$i = \arg\min_{i\in Set} \{n_i^p\}$	Select the task with the minimum number of predecessors	TNP
a_7	$i = \arg\max_{i\in Set} \{n_i^s\}$	Select the task with the maximum number of successors	TXS
a_8	$i = \arg\min_{i\in Set} \{n_i^s\}$	Select the task with the minimum number of successors	TNS
a_9	$i = \arg\max_{i\in Set} \left\{T_C^* - t_{im} - \sum_{m\in M}\sum_{j\in I} x_{jw} z_{mw} t_{jm}\right\}$	Select the task that keeps the station time away from the T_C^*	TAT
a_{10}	$i = \arg\min_{i\in Set} \left\{T_C^* - t_{im} - \sum_{m\in M}\sum_{j\in I} x_{jw} z_{mw} t_{jm}\right\}$	Select the task that keeps the station time close to the T_C^*	TCT
a_{11}	$i,\ \forall i \in Set$	Select a task randomly	RAT

There are ten heuristic actions for candidate tasks in the disassembly environment. The selection of candidate tasks is guided from the perspectives of task time, station operation time, number of precedence tasks, and distance from theoretical cycle time. The detailed action descriptions are given in Table III. To compare the heuristic performance, the 11th action in the table does not use any heuristic rules, that is, a candidate task is selected randomly. In the table, *Set* represents the candidate task set, n^p represents the number of immediate predecessors, n^s represents the number of immediate successors, and T_C^* represents the theoretical cycle time, and its expression is shown as

$$T_C^* = \sum_{i\in I} \max_{m\in M} t_{im} \Big/ |M|. \tag{26}$$

It should be pointed out that to ensure the feasibility of the results and improve the learning efficiency, the candidate tasks are not random, but are selected according to the precedence matrix. The precedence matrix is updated in time after a task assignment, and the candidate task set is determined to ensure that all actions are feasible, which is one of the keys in DQN.

E. Reward Function

Once the action is determined, the task can be assigned, and the heuristic rule is used when the task is assigned in the station, that is, the task is assigned to the station with the shortest total

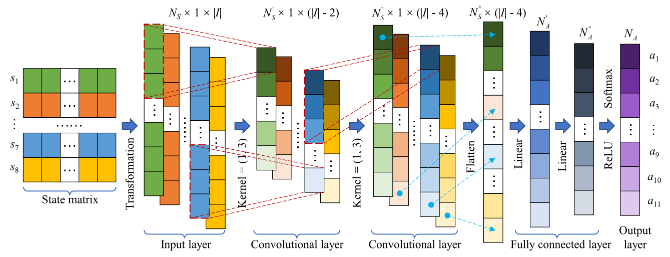

Fig. 4. Neural network of DQN.

working time of the robot. Minimizing the cycle time in the URDLB is equivalent to the distance between the maximum station time and the theoretical cycle time. The expressions of the reward function are shown in (27) and (28)

$$r_k = \int_{\tau = t_{k-1}}^{t_k} \delta_k(\tau)\, \mathrm{d}\tau \tag{27}$$

$$\delta_k(\tau) = \left(T_C^* - \min_{m \in M} \left\{ \sum_{i \in I} x_{iw} z_{mw} t_{im} \right\} \right) \Big/ T_C^* \tag{28}$$

where t_k is the time of state s_k, and r_k is the reward at t_k, that is, the increase of station time in time $[t_{k\text{-}1}, t_k]$.

F. Neural Network Structure

The input layer of the DQN neural network is the system state $\{s_k\}$, and the output layer is the action value $Q(s, a_k; \theta)$. Since DQN only focuses on input and output and does not need to pay attention to the timing of disassembly decisions, it is more appropriate to use convolutional neural networks (CNNs). CNN is used to extract the disassembly state features well and perform translation invariance processing on the disassembly data. The structure of the neural network is shown in Fig. 4, including the input layer, convolutional layer, fully connected layer, and output layer. The input layer contains eight state features, and the output layer contains 11 heuristic actions. One dimension of the kernel in the convolutional layer is 1. The ReLU function is adopted in the activation layer.

G. Procedures of DQN

Referring to the DQN structure in [36], the procedures of the proposed DQN are given in Table IV. The proposed DQN fully considers the characteristics of URDLB in the state, action design, and reward function calculation.

IV. Numerical Experiment and Comparison

There is no research on DRL to solve the URDLB. In this section, two different types of DLB problems in the literature are used as test cases to analyze the performance of the proposed DQN. The running environment is Intel Core i5-8400 CPU, 2.80 GHz, 16 GB RAM.

Referring to the parameter settings in [32], the main parameters of DQN are set as discount factor $\gamma = 0.9$, learning rate $\alpha = 0.0001$, greedy strategy $\varepsilon = 0.9$, training episodes $M = 1000$, replay memory $D = 2000$, batch sampling size $b = 32$, and parameter update frequency $v = 100$.

TABLE IV
Procedures of the Proposed DQN

```
No. Description
1:  Input disassembly information, state features, heuristic action rules
2:  Initialize the replay memory $D$ and capacity $N$, the parameter $\theta$ of the network $Q$, and the parameter $\theta^-$ of the target network $Q^-$, let $\theta^- = \theta$
3:  for episode = 1 to $M$ do
4:      Initialize the state $s_t$, and calculate the candidate task set ($Set$)
5:      if $Set \neq \varnothing$ then // The disassembly task has not been assigned
6:          Randomly choose the action $a_t$ or $a_t = \mathrm{argmax}_a\, Q(s_t, a; \theta)$ according to the $\varepsilon$ greedy rule, determine the robot and the task
7:          Execute the action $a_t$ in the environment, calculate the reward $r_t$, and determine the next state $s_{t+1}$
8:          Store $(s_t, a_t, r_t, s_{t+1})$ in $D$, update $Set$, and let $s_t = s_{t+1}$
9:          Randomly sample a small batch $(s_j, a_j, r_j, s_{j+1})$ from D when there are enough samples in $D$
10:         Set $y_j = r_j$ when $s_{j+1}$ is the terminal state, otherwise $y_j = r_j + \gamma \max_{a'} Q^-(s_{j+1}, a'; \theta^-)$
11:         Take $(y_j - Q(s_j, a_j; \theta))^2$ as the loss function, and use the gradient descent to train the parameter $\theta$ of the network $Q$
12:         Assign the parameters of the network $Q$ to the target network $Q^-$ every $C$ steps, that is, $\theta^- \leftarrow \theta$
13:     end if
14: end for
```

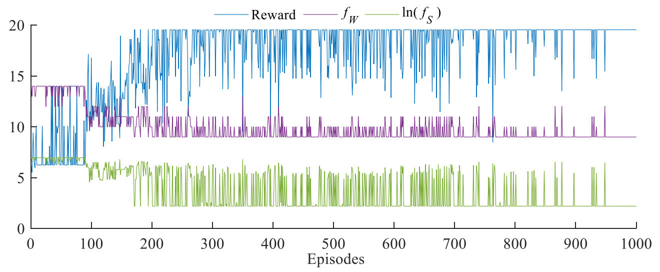

Fig. 5. Training process of DQN on the cell phone disassembly case.

A. Cell Phone Disassembly Case

The cell phone disassembly case with 25 tasks in [38] is classic. This case takes the minimum number of workstations (f_W) and the smoothness index (f_S) as the main objectives, where $f_S = \Sigma_{w \in W} (T_C - T_w)^2$ and T_w represents the disassembly time of workstation w. The optimal values of f_W and f_S are both 9. The proposed DQN is used to solve this case, $\{s_1, s_2, s_4 - s_8\}$ are selected as states, $\{a_1 - a_8, a_{11}\}$ are selected as actions, and the inverse of the square of idle time in the station is used as the reward function. The values of reward, f_W, and $\ln(f_S)$ during the training process are shown in Fig. 5.

Fig. 5 shows that DQN can obtain the optimal value 9 of f_W, and finally converges to 9. The values of f_W include 9, 10, 11, 12, 13, and 14, and the number of these values is 595, 168, 117, 33, 17, and 70 in 1000 episodes. Meanwhile, the number of solutions with the optimal value on f_S is 576, and the number of solutions with better values of 11 and 13 is 10 and 9, indicating that DQN can obtain optimal solutions. The reward is low and fluctuates at the early stage of training. As the training progresses, the reward tends to be stable and the number of fluctuations decreases and tends to converge at the end of the training, indicating that the trained neural network has better parameters. The results show

TABLE V
RESULTS OF THE COMPARISON ALGORITHMS ON THE BENCHMARK CASES

No.	$\|I\|$	$\|M\|$	TLT	TST	TXT	TNT	TXP	TNP	TXS	TNS	TAT	TCT	RAT	GA	ACO	ABC	QL	DRL	D2QN	DDPG	A2C	PPO	DQN
1	25	3	526	524	519	532	534	511	528	512	527	510	518	475	469	475	483	475	479	469	469	469	**468**
2	25	4	371	361	352	351	363	347	353	373	370	372	360	314	**312**	**312**	**312**	315	**312**	314	314	314	**312**
3	25	6	246	258	260	255	248	247	256	262	240	266	257	**207**	211	211	213	212	211	213	211	**207**	**207**
4	25	9	148	151	159	163	160	171	144	172	153	168	146	**121**	122	122	129	**121**	122	**121**	129	129	**121**
5	35	4	479	481	486	484	489	478	501	490	475	493	485	**431**	**431**	**431**	439	445	445	445	**431**	445	**431**
6	35	5	410	398	415	419	411	412	413	409	396	414	406	354	354	**352**	353	353	**352**	356	356	354	**352**
7	35	7	300	289	278	301	290	283	302	292	285	276	294	**236**	240	**236**	240	240	**236**	247	239	**236**	**236**
8	35	12	158	165	157	162	166	171	167	155	153	149	160	125	**124**	125	127	**124**	125	127	**124**	125	126
9	53	5	589	608	599	594	592	596	603	610	601	605	595	529	529	532	530	529	530	**528**	529	530	**528**
10	53	7	396	372	391	382	379	370	393	373	380	378	386	**315**	**315**	316	**315**	317	317	316	317	317	**315**
11	53	10	306	299	309	312	322	295	300	311	320	313	308	242	**241**	251	244	244	242	244	251	242	**241**
12	53	14	234	222	233	226	235	237	232	216	213	214	227	**180**	183	183	**180**	**180**	183	**180**	**180**	182	**180**
13	70	7	575	554	558	559	569	561	549	564	572	570	553	471	471	**464**	472	**464**	467	471	**464**	467	**464**
14	70	10	366	345	365	367	350	360	355	354	358	359	347	**282**	**282**	283	288	284	284	**282**	288	283	283
15	70	14	297	279	291	280	293	286	278	304	301	302	288	235	235	238	236	235	**234**	238	235	238	**234**
16	70	19	222	219	238	231	225	212	241	233	227	230	229	**185**	**185**	199	189	187	**185**	**185**	199	187	187
17	89	8	581	563	561	567	560	570	573	558	559	574	579	**489**	492	492	493	492	493	**489**	493	495	**489**
18	89	12	472	456	459	454	460	461	462	473	450	467	468	386	**385**	**385**	386	**385**	**385**	386	386	**385**	**385**
19	89	16	308	307	328	303	318	305	320	309	315	323	302	**265**	**265**	270	**265**	267	266	266	266	**265**	**265**
20	89	21	283	285	295	280	271	278	288	281	273	297	290	241	242	**238**	242	243	241	243	241	241	**238**
21	111	9	740	725	736	731	727	732	749	739	726	730	723	634	633	636	634	636	634	634	**632**	633	**632**
22	111	13	447	449	451	435	443	446	441	440	431	439	433	388	**387**	388	392	392	**387**	392	**387**	**387**	388
23	111	17	351	345	362	371	360	352	364	368	359	355	347	**308**	317	317	319	315	**308**	**308**	317	315	**308**
24	111	22	301	288	297	280	274	298	293	279	289	292	281	244	244	240	244	238	240	245	245	245	**238**
25	148	10	822	821	802	818	803	817	812	811	820	804	815	**716**	718	**716**	723	718	718	718	**716**	718	**716**
26	148	14	579	572	585	595	596	571	590	593	582	587	594	534	522	**518**	536	522	522	534	536	**518**	**518**
27	148	21	423	410	434	421	433	428	411	408	416	417	432	**369**	**369**	**369**	372	374	372	372	376	376	**369**
28	148	29	308	303	310	301	306	290	291	292	307	298	282	265	265	258	269	258	258	265	**257**	265	**257**
29	297	19	799	804	811	827	815	816	802	800	801	809	814	**735**	747	747	747	747	**735**	749	747	747	**735**
30	297	29	572	590	582	589	574	596	591	587	586	585	578	528	528	529	529	529	529	**526**	531	**526**	**526**
31	297	38	455	477	483	473	459	467	463	468	465	466	478	421	421	428	422	424	424	424	421	421	**419**
32	297	50	362	348	356	367	365	351	359	370	369	376	357	329	326	329	333	331	329	326	329	326	**325**
	N_{best}		0	0	0	0	0	0	0	0	0	0	0	14	11	10	4	5	9	8	8	7	27

that the proposed DQN can effectively optimize the small-scale case.

B. 32 Benchmark Cases

32 URDLB benchmark cases are constructed according to [39]. To verify the performance of the designed heuristic actions, the proposed DQN with ten mixed heuristic actions is compared with the DQN with one heuristic action, i.e., ten algorithms with a single heuristic, represented by the symbols in Table III, respectively. The state-of-the-art algorithms in the literature also include GA [1], ACO [8], ABC [21], QL [27], DRL [31], double DQN (D2QN) [32], DDPG [33], A2C [34], and PPO [35] which are used as comparison algorithms. The number of iterations of the comparison algorithms is the same as the training episodes of DQN. Each algorithm is run independently five times, and its best-so-far solutions are given in Table V, where $|I|$ represents the number of disassembly tasks, $|M|$ represents the number of robots, and black bold numbers represent the best-so-far solutions of all algorithms.

The comparison results show that the best-so-far solutions of the 32 benchmark cases are obtained by advanced meta-heuristics and complex RL algorithms, while the single heuristic algorithm does not obtain any best-so-far solutions, and there are advantages and disadvantages between the ten heuristic algorithms and no heuristic algorithm (namely RAT), indicating that a single heuristic rule cannot interpret state information. Moreover, the analysis of advanced meta-heuristics (namely GA, ACO, and ABC) and complex RL algorithms (namely QL, DRL, D2QN, DDPG, A2C, PPO, and DQN) shows that the number of best-so-far solutions (N_{best}) obtained by these ten algorithms is 14, 11, 10, 4, 5, 9, 8, 8, 7, and 27, respectively, which verifies that the performance of the proposed DQN is superior to that of meta-heuristics and general heuristic RL algorithms in different scale cases.

It should be pointed out that DQN needs to be trained separately for different scale problems, and the training time is long, which is significantly greater than the running time of the meta-heuristics, but the trained neural network can quickly obtain better solutions in dynamic disturbance environments, and the relevant verification will be given in Section V.

In the first case, the number of tasks is 25, the number of robots is 3, and the obtained cycle time is 468. Fig. 6 is the Gantt chart of the disassembly scheme. The disassembly scheme is the optimal strategy π^*, the robot sequence is [1, 3, 2], the first action result

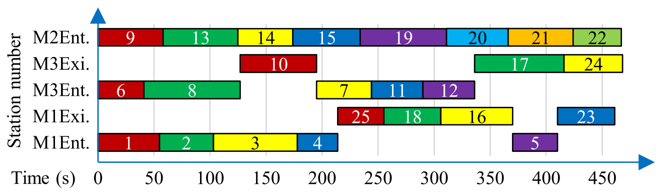

Fig. 6. Gantt chart of the disassembly scheme of case 1.

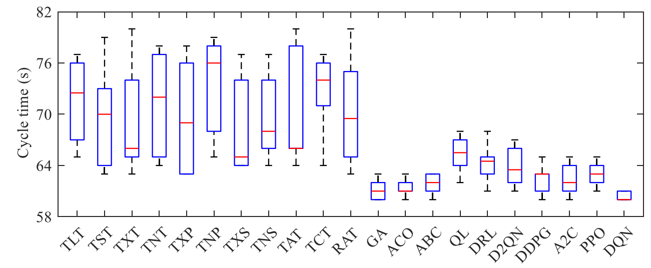

Fig. 10. Boxplot of the 21 algorithms.

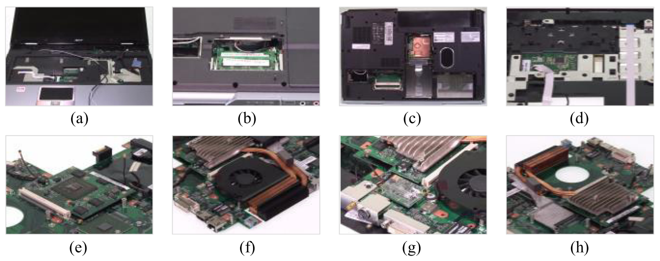

Fig. 7. Some components of the laptop.

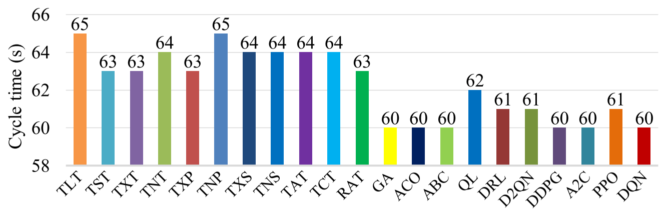

Fig. 11. Best cycle time of each algorithm.

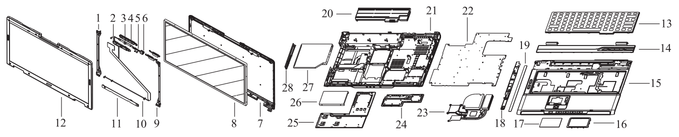

Fig. 8. Exploded view of the laptop.

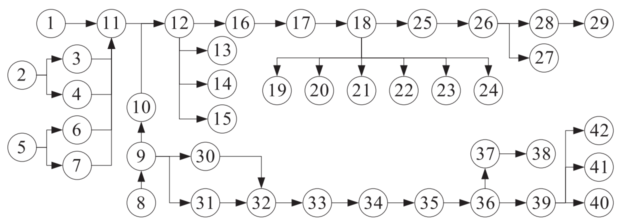

Fig. 9. Precedence relationships among the 42 disassembly tasks.

is to assign task 1 to the entrance side of robot 1, and so on, the whole disassembly system performs 25 steps of the decision process.

V. Application to Laptop Disassembly Line

Since there is no case of a robotic disassembly line in the industry, this section will construct URDLB based on the laptop disassembly data of a disassembly enterprise, and analyze the application performance of the proposed model and algorithm in the practical engineering case. The physical pictures of some components and the exploded view of the laptop are shown in Figs. 7 and 8. The laptop contains 28 main parts, which are divided into 42 disassembly tasks, and 4 robots are provided. The precedence relationships are shown in Fig. 9. The other information of the laptop case is shown in Appendix.

A. Algorithm Performance Comparison

Similar to Section IV-B, the proposed DQN is used to solve the application case. The comparison algorithm includes ten single heuristic algorithms (see Table III), three meta-heuristic algorithms, and six complex RL algorithms. Each algorithm is run ten times independently, and the boxplot of the smaller cycle time obtained by the 21 algorithms is drawn, as shown in Fig. 10. A comparison of the best cycle time achieved by each algorithm is shown in Fig. 11.

Analysis of Fig. 10 shows that the average level of meta-heuristic algorithms and RL algorithms is smaller than that of the ten single heuristic algorithms, and the average level of the proposed DQN is also better than that of three meta-heuristic algorithms and six complex RL algorithms, indicating that the convergence of the proposed DQN is better than that of 20 comparison algorithms. All algorithms have no outliers, and the stability of the proposed algorithm is better than that of ten single heuristic algorithms significantly. It can be seen from Fig. 11 that both meta-heuristic algorithms and four complex RL algorithms can obtain the best-so-far cycle time of 60 s, indicating that the proposed DQN has superior performance in the actual engineering case.

B. Analysis and Discussion of Disassembly Scheme

The training process of the proposed DQN is shown in Fig. 12 when it obtains the best-so-far cycle time. In the early stage of training, the cycle time fluctuates greatly, and the neural network is unstable; then, the cycle time continuously tends to a near-optimal value in the middle stage of training, especially after 300 episodes of training; the number of best-so-far cycle time is increased in the later stage of training, and the fluctuation is small. The change of reward value corresponds to the change of cycle time. The results show that the proposed DQN can effectively optimize the URDLB problem.

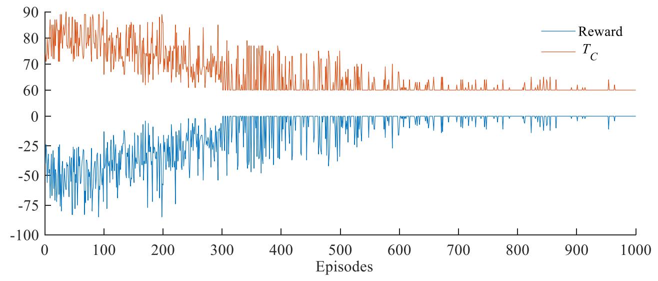

Fig. 12. Training process of cycle time and reward.

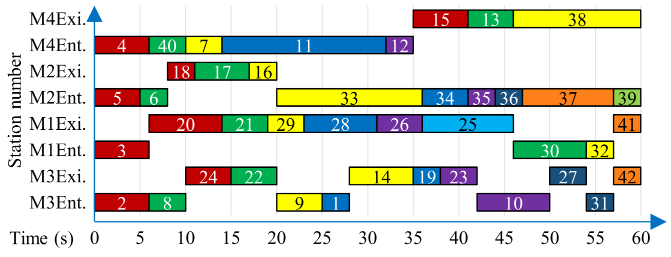

Fig. 13. Gantt chart of the optimal strategy π^*.

The disassembly scheme with a cycle time of 60 s is the optimal strategy π^* of the disassembly system, and the Gantt chart of this strategy is shown in Fig. 13.

The decision-making process of DQN only performs one action per episode, that is, assigning a task, and the entire system needs to perform a total of 42 decision-making steps. The robot sequence of this scheme is [3, 1, 2, 4], the entrance and exit sides of each robot are assigned tasks, and the operation time of all stations is 60 s, that is, there is no idle time in the station, indicating that all robots work at full capacity. There are two reasons for obtaining a better disassembly scheme: First, the U-shaped layout can increase the possibility of optimal assignment of tasks. Second, the proposed DQN has efficient optimization performance, which increases the possibility of the algorithm exploring the optimal solution.

C. Dynamic Balancing Analysis

The quality of end-of-life products is uncertain, such as corrosion and deformation of parts, damage and missing parts due to maintenance, etc. These uncertain factors will make it difficult to determine the disassembly time. For example, the disassembly time of corroded parts will be longer than the normal disassembly time, and the disassembly time of missing parts will become zero. When the disassembly time changes, the disassembly scheme in Section V-B is no longer applicable. It is necessary to replan the disassembly line scheme in this case. In this section, the trained DQN is used to replan the disassembly tasks in a dynamic environment.

The disassembly time in a dynamic environment is given in the Appendix, and the precedence relationships among tasks remain unchanged. The proposed DQN and the nine advanced comparison algorithms in Section V-A are used to solve the problem, and the comparison results are given in Table VI.

TABLE VI
RESULTS OF THE NEW PROBLEM

Algorithm	GA	ACO	ABC	QL	DRL	D2QN	DDPG	A2C	PPO	DQN
T_C	62	63	62	67	67	66	65	66	66	64
Running time	83.52	96.35	77.89	0.039	0.048	0.053	0.059	0.051	0.054	0.042

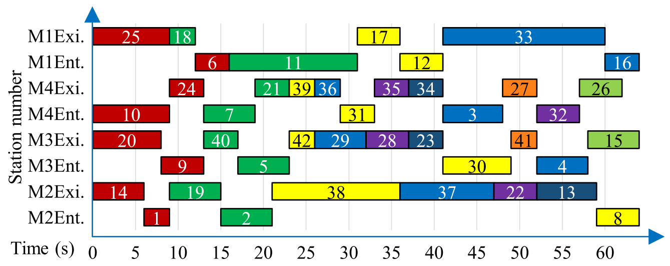

Fig. 14. Gantt chart of the adaptive disassembly scheme.

The results in Table VI show that the proposed DQN can obtain an acceptable disassembly scheme in 0.042 s, and can quickly respond to the dynamic changes of the disassembly environment. Although the meta-heuristics can achieve better results after many iterations, the running time is longer and the response to the dynamic changes of the environment is slower. The results indicate that the proposed DQN has generalization ability and can achieve a dynamic balancing of disassembly lines in a dynamic environment.

The cycle time of the new disassembly scheme is 64 s, and the Gantt chart of this scheme is shown in Fig. 14. Comparing Figs. 13 and 14, it can be seen that there are many re-planning tasks in the disassembly line, which shows that the proposed algorithm has better engineering application performance.

VI. Conclusion

In this article, a MILP model of URDLB was established, and a DQN approach based on problem features was developed. The designed eight states cover the complete information of tasks, robots, stations, working time, and idle time in the disassembly environment, and ten heuristic actions can realize reasonable multidimensional planning of tasks. The effectiveness of the algorithm was verified by a cell phone disassembly case, and 32 benchmark cases were used to verify that the proposed multi-heuristic DQN approach was superior to the ten single heuristic algorithms, three meta-heuristics, and six complex RL methods. In an engineering case of laptop disassembly, the disassembly scheme with the minimum cycle time was obtained, and the optimal decision and planning of tasks in the robots are realized. In addition, the proposed algorithm can use the trained neural network to quickly provide an adaptive disassembly scheme in a dynamic environment.

The proposed DRL can be applied to other combinatorial optimization problems in future research, such as straight, two-sided, parallel, mixed-model assembly/disassembly line balancing, or sequence planning problems.

References

[1] F. Pistolesi, B. Lazzerini, M. D. Mura, and G. Dini, "EMOGA: A hybrid genetic algorithm with extremal optimization core for multiobjective disassembly line balancing," *IEEE Trans. Ind. Inform.*, vol. 14, no. 3, pp. 1089–1098, Mar. 2018.

[2] Y. Feng, Y. Gao, G. Tian, Z. Li, H. Hu, and H. Zheng, "Flexible process planning and end-of-life decision-making for product recovery optimization based on hybrid disassembly," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 1, pp. 311–326, Jan. 2019.

[3] Y. Laili, Y. Li, Y. Fang, D. T. Pham, and L. Zhang, "Model review and algorithm comparison on multi-objective disassembly line balancing," *J. Manuf. Syst.*, vol. 56, pp. 484–500, Jul. 2020.

[4] W. Xu, J. Cui, B. Liu, J. Liu, B. Yao, and Z. Zhou, "Human-robot collaborative disassembly line balancing considering the safe strategy in remanufacturing," *J. Clean Prod.*, vol. 324, Nov. 2021, Art. no. 129158.

[5] J. Liu et al., "An improved multi-objective discrete bees algorithm for robotic disassembly line balancing problem in remanufacturing," *Int. J. Adv. Manuf. Technol.*, vol. 97, no. 9-12, pp. 3937–3962, Aug. 2018.

[6] Z. Li, M. Janardhanan, Q. Tang, and Z. Zhang, "Models and algorithms for U-shaped assembly line balancing problem with collaborative robots," *Soft Comput.*, vol. 27, no. 14, pp. 9639–9659, Jul. 2023.

[7] Z. Zhang, Q. Tang, Z. Li, and L. Zhang, "Modelling and optimisation of energy-efficient U-shaped robotic assembly line balancing problems," *Int. J. Prod. Res.*, vol. 57, no. 17, pp. 5520–5537, Sep. 2019.

[8] S. Agrawal and M. K. Tiwari, "A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem," *Int. J. Prod. Res.*, vol. 46, no. 6, pp. 1405–1429, Mar. 2008.

[9] Z. Li and M. N. Janardhanan, "Modelling and solving profit-oriented U-shaped partial disassembly line balancing problem," *Expert Syst. Appl.*, vol. 183, Nov. 2021, Art. no. 115431.

[10] S. Qin, S. Zhang, J. Wang, S. Liu, X. Guo, and L. Qi, "Multi-objective multi-verse optimizer for multi-robotic U-shaped disassembly line balancing problems," *IEEE Trans. Artif. Intell.*, to be published, doi: 10.1109/TAI.2023.3266187.

[11] S. M. McGovern and S. M. Gupta, "A balancing method and genetic algorithm for disassembly line balancing," *Eur. J. Oper. Res.*, vol. 179, no. 3, pp. 692–708, Jun. 2007.

[12] M. Liu, X. Liu, F. Chu, F. Zheng, and C. Chu, "An exact method for disassembly line balancing problem with limited distributional information," *Int. J. Prod. Res.*, vol. 59, no. 3, pp. 665–682, Feb. 2021.

[13] J. He, F. Chu, F. Zheng, M. Liu, and C. Chu, "A multi-objective distribution-free model and method for stochastic disassembly line balancing problem," *Int. J. Prod. Res.*, vol. 58, no. 18, pp. 5721–5737, Sep. 2020.

[14] F. T. Altekin, L. Kandiller, and N. E. Ozdemirel, "Profit-oriented disassembly-line balancing," *Int. J. Prod. Res.*, vol. 46, no. 10, pp. 2675–2693, May 2008.

[15] Y. Fang, Q. Liu, M. Li, Y. Laili, and P. D. Truong, "Evolutionary many-objective optimization for mixed-model disassembly line balancing with multi-robotic workstations," *Eur. J. Oper. Res.*, vol. 276, no. 1, pp. 160–174, Jul. 2019.

[16] E. B. Edis, R. Sancar Edis, and M. A. Ilgin, "Mixed integer programming approaches to partial disassembly line balancing and sequencing problem," *Comput. Oper. Res.*, vol. 138, Feb. 2022, Art. no. 105559.

[17] Y. Kazancoglu and Y. Ozturkoglu, "Integrated framework of disassembly line balancing with green and business objectives using a mixed MCDM," *J. Clean Prod.*, vol. 191, pp. 179–191, Aug. 2018.

[18] S. Avikal, R. Jain, and P. K. Mishra, "A Kano model, AHP and M-TOPSIS method-based technique for disassembly line balancing under fuzzy environment," *Appl. Soft. Comput.*, vol. 25, pp. 519–529, Dec. 2014.

[19] X. Guo, M. Zhou, S. Liu, and L. Qi, "Lexicographic multiobjective scatter search for the optimization of sequence-dependent selective disassembly subject to multiresource constraints," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3307–3317, Jul. 2020.

[20] G. Tian, C. Zhang, A. M. Fathollahi-Fard, Z. Li, C. Zhang, and Z. Jiang, "An enhanced social engineering optimizer for solving an energy-efficient disassembly line balancing problem based on bucket brigades and cloud theory," *IEEE Trans. Ind. Inform.*, vol. 19, no. 5, pp. 7148–7159, Jul. 2023.

[21] C. B. Kalayci, A. Hancilar, A. Gungor, and S. M. Gupta, "Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm," *J. Manuf. Syst.*, vol. 37, pp. 672–682, Oct. 2015.

[22] Y. Ren, H. Guo, Y. Li, J. Li, and L. Meng, "A self-adaptive learning approach for uncertain disassembly planning based on extended Petri net," *IEEE Trans. Ind. Inform.*, vol. 19, no. 12, pp. 11889–11897, Dec. 2023.

[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An introduction*, Cambridge, U.K., MA, USA: MIT Press, 2018.

[24] E. Tuncel, A. Zeid, and S. Kamarthi, "Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning," *J. Intell. Manuf.*, vol. 25, no. 4, pp. 647–659, Aug. 2014.

[25] A. Allagui, I. Belhadj, R. Plateaux, M. Hammadi, O. Penas, and N. Aifaoui, "Reinforcement learning for disassembly sequence planning optimization," *Comput. Ind.*, vol. 151, Oct. 2023, Art. no. 103992.

[26] Y. Liu, M. Zhou, and X. Guo, "An improved Q-learning algorithm for human-robot collaboration two-sided disassembly line balancing problems," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2022, pp. 568–573.

[27] S. Mete and F. Serin, "A reinforcement learning approach for disassembly line balancing problem," in *Proc. Int. Conf. Inf. Technol.*, 2021, pp. 424–427.

[28] K. Xia, L. Gao, W. Li, L. Wang, and K. - M. Chao, "A Q-learning based selective disassembly planning service in the cloud based remanufacturing system for WEEE," in *Proc. ASME 2014 Int. Manuf. Sci. Eng. Conf.*, 2014, Paper MSEC2014-4008.

[29] H. Mao, Z. Liu, and C. Qiu, "Adaptive disassembly sequence planning for VR maintenance training via deep reinforcement learning," *Int. J. Adv. Manuf. Technol.*, vol. 124, no. 9, pp. 3039–3048, Feb. 2023.

[30] X. Zhao, C. Li, Y. Tang, and J. Cui, "Reinforcement learning-based selective disassembly sequence planning for the end-of-life products with structure uncertainty," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7807–7814, Oct. 2021.

[31] J. Liu, Z. Xu, H. Xiong, Q. Lin, W. Xu, and Z. Zhou, "Digital twin-driven robotic disassembly sequence dynamic planning under uncertain missing condition," *IEEE Trans. Ind. Inform.*, vol. 19, no. 12, pp. 11846–11855, Dec. 2023.

[32] K. Mei and Y. Fang, "Multi-robotic disassembly line balancing using deep reinforcement learning," in *Proc. ASME 2021 16th Int. Manuf. Sci. Eng. Conf.*, 2021, Paper MSEC2021-63522.

[33] R. Zhang, Q. Lv, J. Li, J. Bao, T. Liu, and S. Liu, "A reinforcement learning method for human-robot collaboration in assembly tasks," *Robot. Comput.-Integr. Manuf.*, vol. 73, Feb. 2022, Art. no. 102227.

[34] W. Cai, X. Guo, J. Wang, S. Qin, J. Zhao, and Y. Tan, "An improved advantage actor-critic algorithm for disassembly line balancing problems considering tools deterioration," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2022, pp. 3336–3341.

[35] Z. Zhong, X. Guo, M. Zhou, J. Wang, S. Qin, and L. Qi, "Proximal policy optimization algorithm for multi-objective disassembly line balancing problems," in *Proc. Australian New Zealand Control Conf.*, 2022, pp. 207–212.

[36] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[37] Z. Zhang, L. Zheng, N. Li, W. Wang, S. Zhong, and K. Hu, "Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning," *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1315–1324, Jul. 2012.

[38] S. M. Gupta, E. Erbis, and S. M. McGovern, "Disassembly sequencing problem: A case study of a cell phone," in *Proc. SPIE Int. Conf. Environ. Conscious Manuf.*, 2004, pp. 43–52.

[39] J. Gao, L. Sun, L. Wang, and M. Gen, "An efficient approach for type II robotic assembly line balancing problems," *Comput. Ind. Eng.*, vol. 56, no. 3, pp. 1065–1080, Apr. 2009.

Kaipu Wang received the Ph.D. degree in mechanical engineering from Huazhong University of Science and Technology, Wuhan, China, in 2022.

He was a Visiting Scholar with Eindhoven University of Technology, Eindhoven, The Netherlands, in 2021. He is currently an Associate Research Fellow with the School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan, China. His research interests include production planning and scheduling, big data, and intelligent optimization.

Yibing Li received the M.S. and Ph.D. degrees in mechanical engineering from Wuhan University of Technology (WUT), Wuhan, China, in 2003 and 2008, respectively.

He is currently a Professor with the School of Mechanical and Electronic Engineering, WUT. His research interests include manufacturing informatization and intelligent manufacturing.

Jun Guo received the M.S. and Ph.D. degrees in mechanical engineering from Wuhan University of Technology (WUT), Wuhan, China, in 2009 and 2012, respectively.

He is currently an Associate Professor with the School of Mechanical and Electronic Engineering, WUT. His research interests include production scheduling and optimization.

Liang Gao (Senior Member, IEEE) received the Ph.D. degree in mechatronic engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2002.

He is currently a Professor with the State Key Laboratory of Intelligent Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, HUST. He has authored or coauthored more than 620 publications, and authored 15 monographs. His research interests include intelligent optimization, big data, and machine learning with their applications in design and manufacturing.

Dr. Gao is currently a Co-Editor-in-Chief for *IET Collaborative Intelligent Manufacturing* and *Light: Advanced Manufacturing*, and an Associate Editor for *Chinese Journal of Mechanical Engineering*.

Xinyu Li (Member, IEEE) received the Ph.D. degree in industrial engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2009.

He is currently a Professor with the State Key Laboratory of Intelligent Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, HUST. He has authored or coauthored more than 100 publications. His research interests include intelligent algorithm and scheduling, big data, and machine learning.