




## Article

# An Improved Q-Learning Algorithm for Optimizing Sustainable Remanufacturing Systems

Shujin Qin <sup>1</sup> , Xiaofei Zhang <sup>2</sup>, Jiacun Wang <sup>3</sup> , Xiwang Guo <sup>2</sup>, Liang Qi <sup>4,\*</sup> , Jinrui Cao <sup>5</sup> and Yizhi Liu <sup>2</sup>

<sup>1</sup> College of Economics and Management, Shangqiu Normal University, Shangqiu 476000, China; qinshujin@sqnu.edu.cn

<sup>2</sup> College of Information and Control Engineering, Liaoning Petrochemical University, Fushun 113001, China; edward\_fei1128@163.com (X.Z.); x.w.guo@163.com (X.G.); liuyz061800@163.com (Y.L.)

<sup>3</sup> Department of Computer Science and Software Engineering, Monmouth University, West Long Branch, NJ 07764, USA; jwang@monmouth.edu

<sup>4</sup> Department of Computer Science and Technology, Shandong University of Science and Technology, Qingdao 266590, China

<sup>5</sup> Computer Science Department, New Jersey City University, Jersey City, NJ 07102, USA; jcao@njcu.edu

\* Correspondence: qiliang@sdust.edu.cn

**Abstract:** In our modern society, there has been a noticeable increase in pollution due to the trend of post-use handling of items. This necessitates the adoption of recycling and remanufacturing processes, advocating for sustainable resource management. This paper aims to address the issue of disassembly line balancing. Existing disassembly methods largely rely on manual labor, raising concerns regarding safety and sustainability. This paper proposes a human–machine collaborative disassembly approach to enhance safety and optimize resource utilization, aligning with sustainable development goals. A mixed-integer programming model is established, considering various disassembly techniques for hazardous and delicate parts, with the objective of minimizing the total disassembly time. The CPLEX solver is employed to enhance model accuracy. An improvement is made to the Q-learning algorithm in reinforcement learning to tackle the bilateral disassembly line balancing problem in human–machine collaboration. This approach outperforms CPLEX in both solution efficiency and quality, especially for large-scale problems. A comparative analysis with the original Q-learning algorithm and SARSA algorithm validates the superiority of the proposed algorithm in terms of convergence speed and solution quality.

**Keywords:** reinforcement learning; improved Q-learning algorithm; two-sided disassembly line balancing; human–robot collaboration



**Citation:** Qin, S.; Zhang, X.; Wang, J.; Guo, X.; Qi, L.; Cao, J.; Liu, Y. An Improved Q-Learning Algorithm for Optimizing Sustainable Remanufacturing Systems. *Sustainability* **2024**, *16*, 4180. <https://doi.org/10.3390/su16104180>

Academic Editor: Ripon Kumar Chakraborty

Received: 27 March 2024

Revised: 11 May 2024

Accepted: 13 May 2024

Published: 16 May 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

As society advances, the frequent disposal of both premium and everyday products poses a significant environmental threat, highlighting the urgency of incorporating sustainable practices into product lifecycle management. The direct disposal of such products not only contributes significantly to environmental pollution but also depletes finite resources, exacerbating environmental issues that disrupt people’s daily lives. Dismantling these products becomes imperative, with a focus on extracting valuable end-of-life (EOL) components that contribute to mitigating environmental pollution [1]. However, to truly address the environmental impact of product disposal, a holistic approach rooted in sustainable principles is essential.

Embracing sustainable principles in product disassembly involves more than just extracting valuable components. It requires adopting strategies that prioritize resource efficiency, waste reduction, and environmental conservation throughout the entire product lifecycle. By optimizing disassembly processes and promoting the reuse of EOL components, not only can environmental pollution be minimized, but valuable resources can also be conserved [2]. This

shift toward sustainable disassembly practices aligns with the overarching goal of sustainable resource management and environmental conservation, reflecting a commitment to preserving ecosystems and promoting long-term societal well-being.

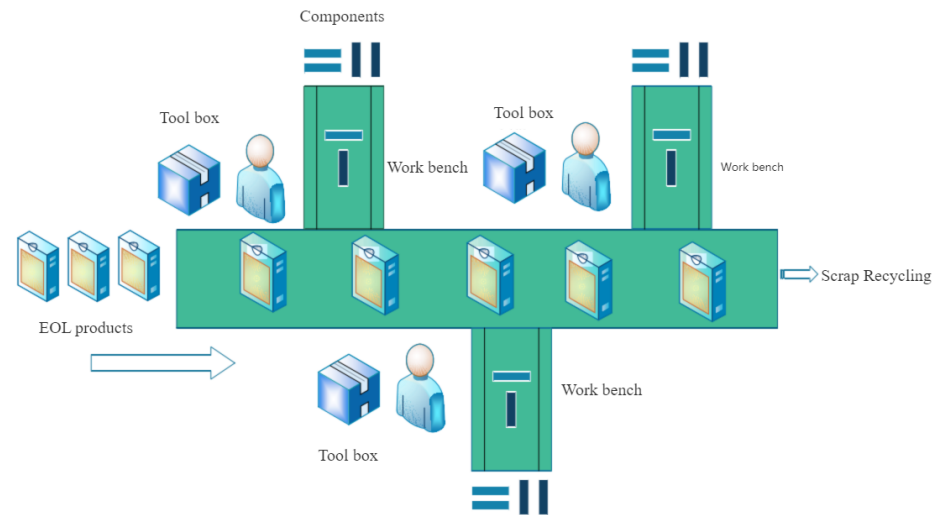
Incorporating sustainable considerations into the disassembly line balancing problem is essential for achieving lasting environmental benefits. By optimizing disassembly processes to maximize resource recovery and minimize waste generation, manufacturers can contribute to a circular economy model where resources are used more efficiently and environmental impacts are minimized. Moreover, by extending the lifespan of products through effective disassembly and component reuse, the need for raw material extraction and the production of new goods can be reduced, further advancing sustainable practices and mitigating environmental degradation [3].

In the contemporary era, escalating concerns about pollution have prompted the adoption of effective measures to alleviate its impact, aligning with the principles of sustainability. One such approach involves the disassembly and reprocessing of recyclable products, which not only mitigates pollution but also conserves valuable resources and promotes environmental sustainability. This necessitates addressing the disassembly line balancing problem, which exhibits variations across diverse real-world scenarios. Various metrics, including revenue maximization, minimization of the disassembly time, and optimization of the workstation smoothness index, among others, come into play in sustainable disassembly practices. Formulating a tailored mathematical model based on specific situations allows for a targeted resolution of the associated disassembly line balancing problem, integrating sustainability considerations into decision-making processes.

An array of disassembly methods exists, encompassing complete and partial disassembly, along with both destructive and non-destructive approaches [4]. Different disassembly lines, such as linear, bilateral, parallel, and U-shaped configurations, offer distinct approaches, each with its own implications for sustainability. Designing specific disassembly lines for different facilities facilitates the examination of balancing problems while ensuring that sustainable practices are upheld throughout the process. Currently, there is widespread exploration of mixed lines across various sectors of society, reflecting the growing recognition of the importance of sustainability in industrial practices. Given the diversity of recyclable products and the imperative to enhance disassembly and recycling practices, the implementation of mixed disassembly line methods, integrating linear and U-shaped configurations, has gained traction. This endeavor aims to better serve society, purify the environment, and contribute to the establishment of a green ecosystem, aligning with the overarching goals of sustainable development and promoting a circular economy model.

In this paper, our focus is directed toward tackling the two-sided disassembly line balancing problem (DLBP) [5,6]. The conventional layout of a disassembly line is depicted in Figure 1. The literature addressing two-sided disassembly lines, particularly in the realm of human–robot collaboration, remains sparse. Furthermore, there is a scarcity of research applying reinforcement learning to tackle this specific problem. Previous work by Xie et al. [7] considered the two-sided DLBP with constraints on workstations and energy consumption, utilizing a differential evolutionary algorithm for solving the multi-objective two-sided DLBP. However, this method is more suitable for continuous problems, while DLBP is inherently discrete. As a result, transformation into a discrete algorithm is necessary for effective problem resolution. Wang et al. [8] explored modeling based on two-sided disassembly lines and proposed a solution using a genetic algorithm with variable neighborhood search, enhancing the algorithm's convergence speed. Another work by Wang et al. [9] delved into DLBP modeling and solutions, investigating cases involving uncertain disassembly times, unknown beat times, two-sided pattern layouts, and the presence of disassembly obstructions between parts. They employed a hybrid artificial bee colony algorithm along with a variable neighborhood-depth search strategy. Zhao et al. [10] studied the parallel DLBP considering carbon emission, establishing a hybrid graph-based disassembly model, and used a genetic algorithm to solve the problem. However, these studies do not account for human–robot collaboration, primarily focus on

multi-objectives that cannot be addressed with the CPLEX solver, and employ heuristic algorithms. Therefore, our model in this paper integrates human and robot collaboration to formulate a corresponding model, and the algorithm utilizes current popular reinforcement learning techniques to effectively address the problem.



**Figure 1.** Serial disassembly line layout.

Many existing articles resort to intelligent algorithms, such as the artificial bee colony algorithm, elite differential evolution algorithm, and others, to address the disassembly line balancing problem (DLBP) [11]. Although these algorithms can yield optimal solutions, they frequently require numerous iterations, potentially leading to instability in their final results. In this paper, we employ a reinforcement learning algorithm from machine learning to address disassembly line balancing across small, medium, and large scales. Reinforcement learning, currently recognized as a more efficient artificial intelligence method [12], emerges as the preferred choice for resolving the DLBP in selective disassembly, ensuring quicker attainment of optimal solutions and thereby enhancing efficiency. Given the escalating societal attention to the DLBP, a plethora of algorithms is being applied in this domain, with a growing emphasis on the efficiency of delivering optimal solutions. Reinforcement learning has become widely utilized in path planning problems [13,14]. Simultaneously, the integration of reinforcement learning and disassembly lines has garnered growing interest from academia and industry alike [15,16]. In this paper, we utilize the Q-learning algorithm to address the DLBP and validate the algorithmic results by solving the corresponding mathematical model with CPLEX. The outcomes demonstrate that the Q-learning algorithm exhibits a rapid solution speed and yields optimal solutions.

The contributions of this paper are primarily encapsulated in the following aspects:

(1) A corresponding mixed-integer programming model is established. Existing solutions to the disassembly line balancing problem often involve multi-product and multi-objective formulations, posing challenges for solvers like CPLEX. This paper adopts a single-objective linear mathematical model, facilitating a solution with CPLEX, thus ensuring accurate verification and mitigating potential errors.

(2) A reinforcement learning approach is introduced to solve the DLBP, a method less commonly utilized in current practices. A novel Q-table is defined, enabling the algorithm to learn the environment based on the objective function.

(3) The model's accuracy is initially validated with CPLEX, followed by a comparison of the algorithmic results with those obtained from CPLEX. This two-step validation process ensures the reliability of the proposed algorithm and guards against the possibility of failing to identify the optimal solution.

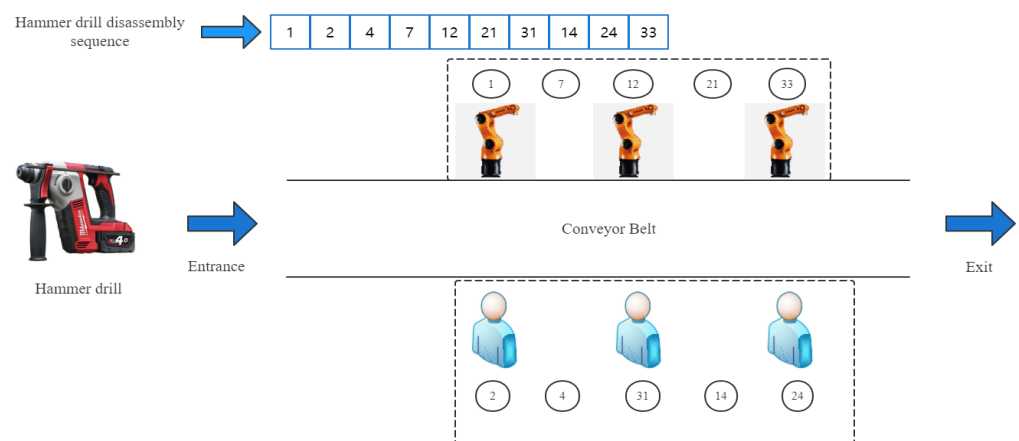
The subsequent sections of this paper are structured to provide a comprehensive understanding of our research. In Section 2, we offer a detailed description of the problem,

accompanied by the formulation of the mixed-integer programming model. Furthermore, we provide relevant illustrations for a detailed explanation of the process. Moving on to Section 3 we delve into the principles of the Q-learning algorithm and elucidate its algorithmic procedure. This section particularly emphasizes the application of the Q-learning algorithm to effectively address the disassembly line balancing problem (DLBP). In Section 4, we thoroughly analyze the results obtained using our algorithm and compare them with the outcomes derived using the CPLEX solver. This comparative analysis aims to showcase the performance of our algorithm. Finally, Section 5 serves as the concluding segment of this paper, summarizing the key findings and contributions presented in the preceding sections.

## 2. Problem Description

### 2.1. Problem Description

This paper investigates the two-sided disassembly line balancing problem, taking into account the categorization of parts into fine and hazardous classifications. Specifically, fine parts are designated for human disassembly, while hazardous parts are reserved for robot disassembly. With the increasing prevalence of intelligent factories, there is a growing trend toward leveraging robots for certain tasks to enhance efficiency, achieve better yields, and reduce disassembly times. Consequently, this paper addresses the challenge of human–robot collaboration. Figure 2 illustrates a strategically designed layout for disassembly sequences based on the types of disassembly lines discussed in this paper. This layout ensures compliance with priority relationships and effectively manages conflicting relationships within the disassembly process.



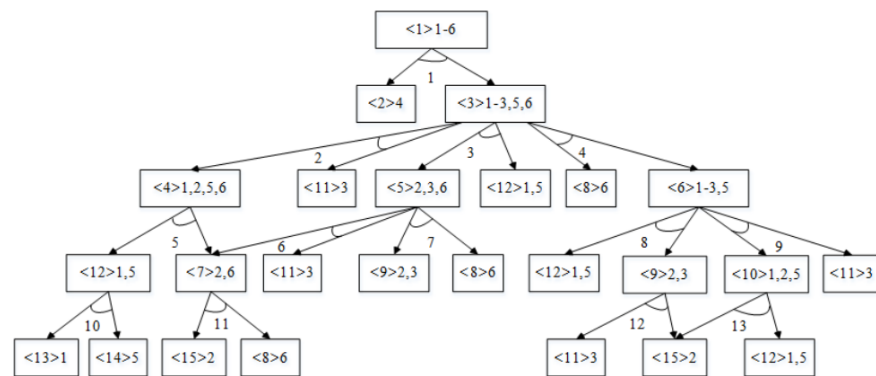
**Figure 2.** Two-sided disassembly line layout based on human–robot combination.

From Figure 2, it is evident that robots and human workers are deployed on either side of the disassembly line. Based on the prioritized sequence for component disassembly and task assignment, different components are allocated to robots and human workers for disassembly work. It can be observed that components 1, 7, 12, 21, and 33 are assigned to robots with varying skills for disassembly, while components 2, 4, 31, 14, and 24 are assigned to different human workers. Ultimately, the optimal disassembly sequence for obtaining the hammer drill is determined to be 1–2–4–7–12–21–31–14–24–33.

In this paper, we elucidate the interrelationships among individual parts and tasks by employing the with-or-without diagram of the disassembled product [17,18]. By utilizing the physical diagram, we create an AND/OR graph that effectively encapsulates the hierarchical relationships among components. From this AND/OR graph, we derive both the priority relationship matrix and the conflict relationship matrix. The AND/OR graph proves to be a valuable representation, providing a visual depiction of various item relationships. This visual representation serves as a preparatory step for developing algorithmic code later. Considering the practicality of the problem, our chosen disassembly

strategy, as studied in this paper, aims to avoid unnecessary disassembly and mitigate the overall disassembly time, thereby enhancing efficiency. This study focuses on the single-objective two-sided disassembly line balancing problem, utilizing the Q-learning algorithm in reinforcement learning. The objective is to minimize the total disassembly time while ensuring a certain disassembly profit. To illustrate the disassembly process, we present a small case of an AND/OR graph.

As shown in Figure 3, the parentheses in the small box  $\langle \rangle$  represent the subassembly number, and the example is from subassemblies 1–15, i.e., 15 subassemblies. The number after the parentheses represents the part numbers of the subassembly, e.g., subassembly 5 contains parts with serial numbers 2, 3, and 6. With arrows and brackets representing tasks, the example task's serial numbers range from 1 to 13, encompassing a total of 13 tasks.



**Figure 3.** An AND/OR graph of a washing machine.

As depicted in Figure 3, starting from task 1, subassembly  $\langle 1 \rangle$  can be disassembled into subassemblies  $\langle 2 \rangle$  and  $\langle 3 \rangle$ , representing an AND relationship. Subassembly  $\langle 3 \rangle$  can perform tasks 2, 3, and 4, which have an OR relationship with each other. That is, subassembly  $\langle 3 \rangle$  can only choose one of tasks 2, 3, or 4 for disassembly, so tasks 2, 3, and 4 are in conflict with each other. In the figure, we can see that task 1 is the immediate predecessor of tasks 2, 3, and 4, i.e., task 1 must be completed before tasks 2, 3, and 4 are performed. Hence, based on this figure, we can derive the correct disassembly sequence as  $1 \rightarrow 4 \rightarrow 8 \rightarrow 12$ . The disassembly sequence  $1 \rightarrow 2 \rightarrow 3 \rightarrow 6$  is incorrect because it does not satisfy the conflict relationship. The disassembly sequence  $1 \rightarrow 3 \rightarrow 11$  is also incorrect because it does not satisfy the prior order relationship of the tasks.

Based on the AND/OR graph, we can obtain the corresponding priority relationship matrix, conflict matrix, and relationship matrix. They are described as follows:

(1) To describe the priority relationships between tasks, we define the priority relationship matrix  $P = [p_{jk}]$ , where

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

(2) To describe the conflict relationship of each task, we add the conflict matrix  $M_p = [m_{jk}]$ , where

$$m_{j,k} = \begin{cases} 1, & \text{if task } j \text{ can be performed before task } k. \\ -1, & \text{if task } j \text{ and } k \text{ conflict with each other.} \\ 0, & \text{otherwise.} \end{cases}$$

(3) To describe the relationships between individual tasks and subassemblies, we build the task–subassembly relationship matrix  $D = [d_{ij}]$ , where

$$d_{i,j} = \begin{cases} 1, & \text{if subassembly } i \text{ is obtained by task } j. \\ -1, & \text{if subassembly } i \text{ is disassembled by task } j. \\ 0, & \text{otherwise.} \end{cases}$$

Based on the above definition, the following is the conflict matrix  $M_p$  for the washing machine case:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 0 & -1 & 0 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\ 0 & -1 & -1 & 0 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & 1 & 1 \\ 0 & 0 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 0 & -1 & 0 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & 1 & -1 & -1 \\ 0 & -1 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & 0 & -1 & 1 & -1 \\ 0 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & -1 & 1 & -1 & 1 & -1 \\ 0 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 1 & -1 & -1 & 0 \end{pmatrix}.$$

The following is the relationship matrix  $D$  for the washing machine:

$$\begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Given the complexity of the DLBP addressed in this paper, we make the following assumptions in the mathematical model:

- (1) Human–robot collaboration disassembly (two-sided disassembly line: robot on one side and human on the other).
- (2) Each robot can only disassemble one part at a time, and each worker can only disassemble one part at a time.
- (3) The priority relationship of disassembled parts must be met.
- (4) Dangerous parts must be disassembled by a robot; fine parts must be disassembled manually.
- (5) The time of each disassembly task is a known and determined value.
- (6) A disassembly task is assigned to only one robot or human.

## 2.2. Notations

$I$	Number of tasks.
$\mathbb{I}$	Task collection $\mathbb{I} = \{1, 2, \dots, I\}$ .
$III$	Number of subassemblies.
$\mathbb{K}$	Collection of task locations on the workstation $\mathbb{K} = \{1, 2, \dots, I\}$ .
$\mathbb{I}_i^C$	Matrix of tasks that conflict with task $i$ .
$\mathbb{I}_i^P$	Precedence task matrix for task $i$ .
$\mathbb{I}^R$	Matrix of hazardous tasks that are required to be disassembled by the robot.
$\mathbb{I}^H$	Matrix of delicate tasks that require manual disassembly.
$T_i^R$	Robot disassembly time for task $i$ .
$T_i^H$	Manual disassembly time for task $i$ .
$C_i^R$	Robot disassembly cost for task $i$ .
$C_i^H$	Manual disassembly costs for task $i$ .
$P_i$	Dismantling proceeds for task $i$ .
$M$	The maximum demolition gain of the case.

The decision variables are as follows:

- (1)  $x_{ik} = \begin{cases} 1, & \text{if task } i \text{ is the } k\text{-th executed task.} \\ 0, & \text{otherwise.} \end{cases}$
- (2)  $u_i = \begin{cases} 1, & \text{if task } i \text{ is the disassembly by the robot.} \\ 0, & \text{otherwise.} \end{cases}$
- (3)  $v_i = \begin{cases} 1, & \text{if task } i \text{ is manual disassembly.} \\ 0, & \text{otherwise.} \end{cases}$
- (4)  $t_i =$  start time of task  $i$  (real number).
- (5)  $t_{\max} =$  max completion time (real number).

## 2.3. Mathematical Model

$$\text{Min } f = t_{\max} \quad (1)$$

$$\sum_{i \in \mathbb{I}} \sum_{k \in \mathbb{I}} \sum_{j \in III} d_{ij} p_j x_{ik} - \sum_{i \in \mathbb{I}} T_i^R u_i - \sum_{i \in \mathbb{I}} T_i^H v_i \geq 0.5M \quad (2)$$

$$\sum_{k \in \mathbb{I}} x_{ik} \leq 1, \quad \forall i \in \mathbb{I} \quad (3)$$

$$\sum_{i \in \mathbb{I}} x_{ik} \leq 1, \quad \forall k \in \mathbb{I} \quad (4)$$

$$\sum_{i \in \mathbb{I}} x_{ik} \geq \sum_{i \in \mathbb{I}} x_{ik+1}, \quad \forall k \in \mathbb{I} - \{I\} \quad (5)$$

$$\sum_{k \in \mathbb{I}} x_{ik} + \sum_{i' \in \mathbb{I}_i^C} \sum_{k \in \mathbb{I}} x_{i'k} \leq 1, \quad \forall i \in \mathbb{I} \quad (6)$$

$$x_{ik} \leq \sum_{i' \in \mathbb{I}_i^P} \sum_{k'=1}^{k-1} x_{i'k'}, \quad \forall i \in \mathbb{I}, \forall k \in \mathbb{I} \quad (7)$$

$$t_{i'k+1} \geq t_{ik}, \quad \forall i \in \mathbb{I}, \forall i' \in \mathbb{I}, \forall k \in \mathbb{I} - \{1\} \quad (8)$$

$$u_i + v_i \leq 1, \quad \forall i \in \mathbb{I} \quad (9)$$

$$u_i \geq v_i, \quad \forall i \in \mathbb{I}^R \quad (10)$$

$$u_i \leq v_i, \quad \forall i \in \mathbb{I}^H \quad (11)$$

$$t_i \geq t_{i'} + T_i^R + M(u_i + u_{i'} + x_{ik} + x_{i'k'} - 4), \quad (12)$$

$$\forall i \in \mathbb{I}, \forall i' \in \mathbb{I}, \forall k \in \mathbb{I}, k' \leq k$$

$$t_i \geq t_{i'} + T_i^H + M(v_i + v_{i'} + x_{ik} + x_{i'k'} - 4), \quad (13)$$

$$\forall i \in \mathbb{I}, \forall i' \in \mathbb{I}, \forall k \in \mathbb{I}, k' \leq k$$

$$t_i \geq t_{i'} + T_i^H v' + T_{i'}^R v' + M(u_i + u_{i'} + v_i + v_{i'} - 2), \quad (14)$$

$$\forall i \in \mathbb{I}, \forall i' \in \mathbb{I}^P, \forall k \in \mathbb{I}, k' \leq k$$

$$t_{max} \geq \sum_{i \in \mathbb{I}} T_i^R u_i + T_i^H v_i, \quad \forall i \in \mathbb{I} \quad (15)$$

$$x_{ik} \in \{0, 1\}, \forall i \in \mathbb{I} \forall k \in \mathbb{I} u_i \geq 0, \forall i \in \mathbb{I} v_i \geq 0, t_{max} \geq 0 \quad (16)$$

The objective function aims to minimize the total disassembly time. This constraint on the total time is reflected in constraint (14). Constraint (1) ensures that the disassembly profit is greater than or equal to 0.5 times the maximum profit. Constraint (2) ensures that each task is performed at most once. Constraint (3) ensures that at most one task is executed per location. Constraint (4) ensures that the latter position has a task when the former position has a task. Constraint (5) ensures that conflicting tasks are executed at most once. Constraint (6) ensures that at least one task in the previous sequence is executed. Constraint (7) addresses the temporal relationship of the sequential execution of tasks. Constraint (8) specifies that each task is assigned to manual disassembly or robotic disassembly, but not both. Constraint (9) ensures that tasks that must be disassembled by a robot are assigned accordingly. Constraint (10) ensures that tasks that must be disassembled by a human are assigned accordingly. Constraint (11) specifies the tasks that must be disassembled by a robot. Constraint (12) defines the task sequence time relationship for human disassembly. Constraint (13) defines the time relationship for the execution of preceding tasks. Constraint (15) addresses the relationship between the maximum completion time and the disassembly task. Constraint (16) defines the range of values of the decision variables.

The model takes the maximum return as  $M$  in the above model, and the solution of the minimization disassembly time model is performed according to  $M$ .

### 3. Q-Learning Algorithm and Application

#### 3.1. Q-Learning Algorithm Description

This subsection introduces the principles and processes of the Q-learning algorithm [19–21]. The Q-learning algorithm is a widely used method in reinforcement learning, with a common example being its application to maze exploration to locate a treasure [22,23]. In the realm of reinforcement learning, the intelligent agent continuously interacts with an unknown environment. Two fundamental concepts, namely state and action, are introduced. The agent takes actions with corresponding values to continuously maximize rewards. The Q-learning algorithm is employed to store information learned about the environment using a Q-table. In each iteration, the action with the highest Q-value in the current state is chosen for state transition. The agent takes action  $A_t$  based on the current state  $S_t$ . An action value is associated with each corresponding action [24]. Different return values are obtained based on the state transition and the action taken by the agent. These return values are stored in the Q-table after comprehensive environment learning, allowing the agent to select the appropriate action for state transitions.

In reinforcement learning, the fundamental concept is to enable the agent to interact with the environment to gather information. When the agent reaches a particular state, the



information within that state is recorded. This facilitates a better selection of states and actions in the broader environment later on. Essentially, reinforcement learning involves the continuous process of allowing the intelligent agent to explore and make mistakes. As the agent accumulates enough experience, denoted as  $E$ , in the Q-learning algorithm, this translates to the constant updating of values in the Q-table [25,26].

The core idea of the learning algorithm is as follows:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

The essence of this formula lies in the Bellman equation, which represents the transformation relationship of the value–action pair.  $R_{t+1} + \gamma \max_a Q(S_{t+1}, A_{t+1})$  is referred to as the Temporal-Difference (TD) target, where  $\delta_t = [R_{t+1} + \gamma \max_a Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$  denotes the TD deviation, commonly known as the TD error. Here,  $\alpha$  represents the learning rate, and  $\gamma$  is the reward decay coefficient. This formula undergoes updates using the TD method, placing it under the category of TD algorithms.

The aforementioned equation constitutes the Q-learning update formula. It relies on the next state, where the largest Q-value for the action is selected, multiplied by the decay  $\gamma$ , and added to the actual return value corresponding to the most realistic Q-value. The update is based on the  $Q(s, a)$  from the previous Q-table, serving as a Q-value estimation. Through this formula, we continuously extract information from the environment, updating the Q-values in the corresponding positions of the Q-table based on the acquired information, whether positive or negative. The Bellman equation involves the state value function in applying this algorithm, which we denote as follows:

$$V_\pi(s) = E_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right]$$

The Bellman equation for this state function is expressed as

$$V(s) = E[R_{t+1} + \gamma v(S_{t+1} | S_t = s)]$$

The state function represents the value of the current state, taking into account not only its own state value but also the cumulative sum of values from all states that can be reached in subsequent steps. At each step, we multiply by a decay factor  $\gamma$ , signifying the proportion of the later state values. A larger  $\gamma$  implies a greater influence of later state values on the Q-value in our Q-table, emphasizing their significance. This allows us to derive the optimal state value function, denoted as  $V^*(s)$ , based on the state function.

Applying this definition of the state function to our disassembly line context involves subtracting the disassembly time from 100 for each position. This transformation ensures that smaller time values correspond to larger values inside the Q-table, following the learning method. Ultimately, the optimal disassembly sequence is obtained based on the greedy strategy.

The pseudo-code for the Q-learning algorithm is shown in Algorithm 1.

---

#### Algorithm 1: Q-learning Algorithm

---

Input:  $Q(s, a)$ , for all  $s \in S, a \in A$ , arbitrarily and  $Q(\text{terminal-state}, -) = 0, \alpha, \gamma$

Output: a new  $Q(s, a)$

Repeat (for each episode):

Initialize  $S$

Repeat (for each step of episode):

    Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)

    Take action  $A$ , observe  $R, S'$

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

Until  $S$  is terminal

---

The algorithm iterates the model with the goal of selecting the action with the maximum value for the next state.

The action selection for the next state is executed using the  $\epsilon$ -greedy algorithm, placing it within the category of offline learning algorithms. By choosing the maximum action value as the target value, the direction of the value update is directed toward the optimal  $q^*$ .

This approach relies on the value function, facilitating iteration calculations through the value function. Following the optimization strategy based on the value function described above, we obtain the action corresponding to the optimal value.

### 3.2. Q-Learning Combined with Disassembly Lines

From the aforementioned explanation, it is evident that the algorithm is well suited for solving discrete problems, and the disassembly line balancing problem (DLBP) is inherently a discrete problem. Therefore, the algorithm is indeed applicable to solve the DLBP. This paper specifically aims to minimize the total disassembly time. The mathematical transformation of small time values into relatively large values allows the algorithm to effectively select the state with a smaller time at each step, aligning with the objective function. To integrate the task AND/OR graph into reinforcement learning, we treat the set of selectable actions as the next set of states that the current state can reach. Subsequently, the algorithm proceeds to solve the disassembly line problem, and the steps are described in detail below based on the aforementioned algorithm flowchart.

#### Step 1:

Initialize the Q-table by constructing a two-dimensional array with  $n$  columns and  $m$  rows. The columns represent the number of actions, and the rows represent the number of states. The value of each position in this two-dimensional array is initially set to 0. According to the DLBP and AND/OR graph, we set the state to the demolition task and the action to the next task. So, the Q-table is a square array.

#### Step 2:

Select an action by choosing an action in state  $S$  according to the Q-table. However, at the very beginning, each Q-value is set to 0. The state values reachable from this state are set to the corresponding reward values according to the specificities of the DLBP. At the beginning, the agent explores the environment and randomly chooses actions. This is because the agent knows nothing about the environment at this time. As the agent continues to explore the environment, the probability of making a random choice decreases, i.e., the agent is able to learn about the environment.

#### Step 3:

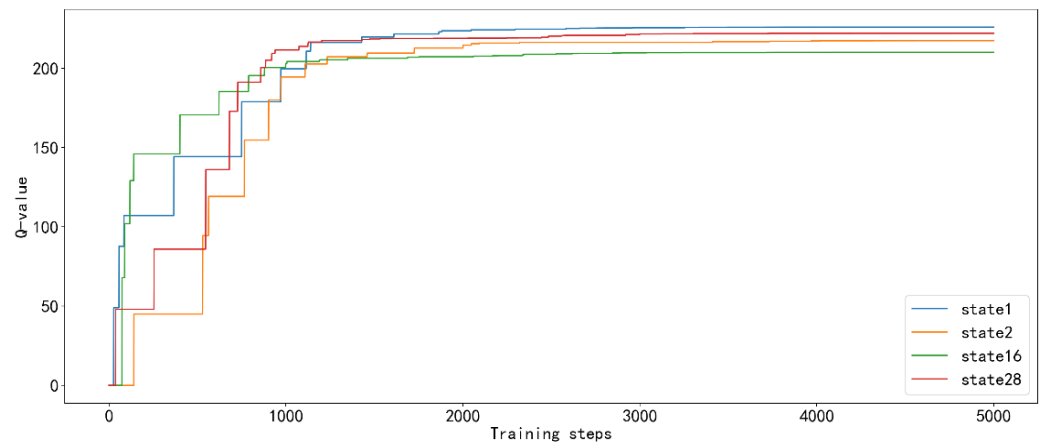
Execute an action. In the disassembly line balancing problem, this starts with all states being reachable, so a random task is chosen to perform. In the process of continuous exploration, the agent's estimation of the Q-value becomes more and more accurate, i.e., after a certain number of explorations, the algorithm can find the optimal solution.

#### Step 4:

Evaluate. Now that the action has been taken and the result and reward observed, the function  $Q(s, a)$  now needs to be updated according to the algorithm. Finally, after our pre-set exploration steps have been completed, the agent is now ready to exploit the environment. At this point, we obtain the Q-table, which represents the optimal Q-values. The agent now selects better actions, and at this point, the subscript corresponding to the largest value in a row of the Q-table is the optimal action for the current state. According to the model mentioned in this paper, the algorithm stops when the gain criterion is satisfied, which occurs when the agent completes a state transition.

Figure 4 depicts the evolution of Q-values for reachable states in the Q-table corresponding to state 0, using the copier assembly example solved in this paper. In accordance with the approach presented in this paper, the constructed task 0 can reach tasks 1, 2, 16, and 28 through the AND/OR graph. The iterative graph of Q-values reflects this connectivity. By observing the iterative graph, it becomes apparent that stability is almost reached

around 2500 training steps, and the final state 1 attains the highest Q-value. Consequently, we opt to begin the disassembly process with task 1.



**Figure 4.** Q-value convergence diagram for optional actions from state 0.

The following describes how to improve the Q-learning algorithm to solve the DLBP, using the example of the washing machine AND/OR diagram from the previously mentioned paper.

In the original Q-learning algorithm, the Q-table is initialized to guide the algorithm on how to reach the destination. The initial values are set to 1 for reachable states,  $-1$  for unreachable states, and 100 for reaching the destination. We construct an initial table based on this logical relationship and set the total number of learning steps according to the value iteration principle, ultimately obtaining the Q-table after learning. However, this approach assumes a clear destination, and for our problem, it is not suitable since the algorithm should not terminate only when a specific task is reached. Instead, we should use a constraint as a termination criterion. Given that Q-learning selects the action with the maximum value in each state, we can modify the approach for our problem as follows:

(1) Each state selects the action with the smallest value in the corresponding row of the Q-table, meaning the action with the smallest time consumed. The initial table stores the time consumed by each task, ensuring that the state transition selects an action with minimal time.

(2) According to the original Q-learning algorithm, each time in the Q-table corresponds to the maximum value in a row for an action. In this case, the initial table stores 100 minus the time consumed by that task. However, this corresponds to selecting the action with the largest value, which, in our model, represents the task that actually takes less time. This completes the state transition. In this paper, we adopt the second approach.

### 3.3. Definition of Reward Matrix $R$

According to the previously mentioned washing machine AND/OR diagram, the case involves 13 tasks [27], corresponding to a total of 13 states in the algorithm. Each state can have 13 actions, meaning that each task can choose from 13 tasks for state transition. Therefore, we construct a  $13 \times 13$  two-dimensional matrix. If a state cannot be reached, we set the value of the corresponding position to  $-1$ , and if it is the same as itself, we set it to 0. For states that can be reached, we set the value to 100 minus the disassembly time spent on the corresponding task. From the figure, it is apparent that task 1 can reach tasks 2, 3, and 4. Since the array's memory starts with a subscript of 0, we set the values of  $[0,1]$ ,  $[0,2]$ , and  $[0,3]$  in the array to values of 100 minus the disassembly times spent on tasks 2, 3, and 4, respectively. This ensures that the agent can learn effectively. In the end, at each step, the position with the largest value in the Q-table is selected, and the corresponding state is the task number to be performed next. This completes the learning of the example, representing the environment. From this arithmetic example, we can summarize the method of defining

the matrix  $R = [r_{ij}]$  to be learned using the following approach:

$$r_{i,j} = \begin{cases} 100 - \text{time spent on task } j, & \text{if task } i \text{ can reach task } j. \\ 0, & \text{if } i = j. \\ -1, & \text{otherwise.} \end{cases}$$

Based on this definition, the  $R$  matrix of the washing machine is as follows:

$$\begin{pmatrix} 0 & 96 & 93 & 94 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 & 94 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & -1 & -1 & 93 & 92 & -1 & -1 & 90 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & -1 & -1 & -1 & 88 & 91 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 90 & 94 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 94 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 87 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & 90 & -1 & 87 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & 91 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 90 & -1 & -1 & 0 \end{pmatrix}.$$

Based on the matrix obtained from the AND/OR graph, the agent interacts with the environment multiple times, and the environment provides feedback values to the agent. This process results in a Q-table, which stores the Q-values for the actions that can be selected in each state. Using the greedy algorithm, we select the action corresponding to the largest Q-value from the starting state to reach the next state, which represents the next task. This completes the state transition. The algorithm terminates when the gain constraint is satisfied. The optimal solution is then output based on the achieved results.

The algorithm's learning process based on the initial table is as follows: assuming 2000 learning steps, a learning rate of  $\alpha = 0.5$ , and a future return decay value of  $\gamma = 0.9$ . After training, the following matrix is obtained:

$$\begin{pmatrix} 231.1 & 256.7 & 252.8 & 249.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 160.7 & 0 & 0 & 178.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 159.8 & 0 & 0 & 177.6 & 170.3 & 0 & 0 & 90 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 155.6 & 0 & 0 & 0 & 169.0 & 172.9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 84.6 & 0 & 0 & 0 & 0 & 90 & 94 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 84.6 & 0 & 0 & 0 & 0 & 94 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 81.0 & 0 & 90 & 0 & 87 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 81.9 & 0 & 0 & 0 & 91.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

In the table, we can observe that the maximum value in the first row is 256.7, corresponding to action 2. Therefore, the next step is to select task 2 for disassembly. Moving to the second row, the maximum value is 178.6, so we select action 5 for state transition, corresponding to disassembly task 5. Moving to the fifth row, the maximum value is 94, so we select action 11 for state transition, corresponding to task 11. When we disassemble to task 5 and satisfy the disassembly profit, we meet the algorithm's stopping criterion and thus output the optimal solution. At this point, the corresponding disassembly sequence is  $1 \rightarrow 2 \rightarrow 5$ .

#### 4. Case Introduction and Results Analysis

This section introduces six different cases of AND/OR graphs and describes how they are incorporated into the Q-learning algorithm. The goal is to address cases of varying sizes to cater to the specificity of the problem and account for different scenarios. The results of the respective runs are analyzed based on the varying case sizes, and conclusions are drawn [28]. The model was solved using both the Q-learning algorithm and IBM ILOG CPLEX. The Q-learning algorithm was coded in Python and implemented on a computer with the configuration: Intel(R) Core(TM) i7-10870H @ 2.30 and Windows 10 operation system.

For the initial case, a small example featuring 13 tasks related to ballpoint pens is utilized for testing [29]. The AND/OR graph for this small ballpoint pen case is illustrated in the Figure 5 below, denoted as Case 1. Figure 6 shows the part diagram of the hammer drill, from which its main components can be seen.

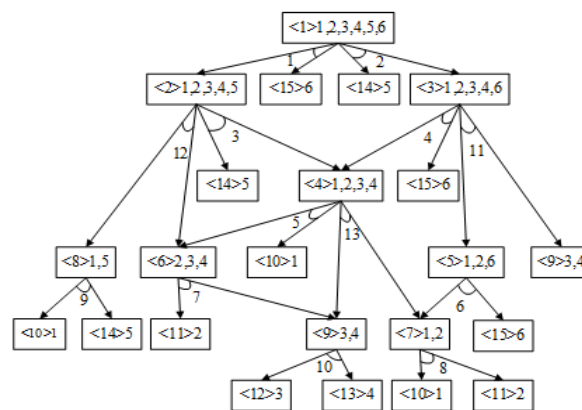


Figure 5. An AND/OR graph of a small ballpoint.

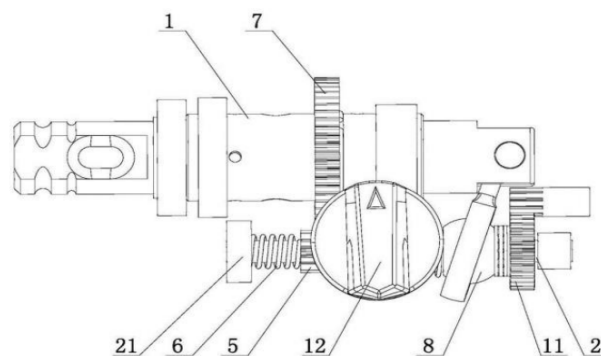


Figure 6. Hammer drill part diagram.

In the second case, a ballpoint pen with 20 tasks is considered. Since subassembly 1 can perform both task 1 and task 2, an additional task 0 is introduced into the algorithm. Task 0 can reach tasks 1 and 2. Consequently, there are 21 tasks and 24 subassemblies, leading to the construction of a  $21 \times 21$  two-dimensional matrix. This case is denoted as Case 2 [30,31]. For the third case, a slightly larger copier with 32 tasks is used. According to the AND/OR graph, subassembly 1 can handle tasks 1, 2, 16, or 28. Therefore, task 0 is constructed in the algorithm, allowing it to reach tasks 1, 2, 16, and 28. This results in the construction of a  $33 \times 33$  matrix for the algorithm to learn. This case is referred to as Case 3 [32,33]. In the fourth case, a rotor is considered, featuring 15 tasks and 17 subassemblies. A  $15 \times 15$  two-dimensional matrix is constructed based on the AND/OR graph for the algorithm to learn. This case is referred to as Case 4 [34,35]. The fifth case involves a slightly larger radio with 30 tasks and 29 subassemblies. To enable task 0 to reach tasks 1 and 2, a  $31 \times 31$

two-dimensional matrix is constructed for the algorithm to learn. This case is denoted as Case 5 [36,37]. Finally, a large example of a hammer drill with a partial structure is utilized for the sixth case. This case comprises 46 tasks and 63 subassemblies [38,39]. A  $46 \times 46$  matrix is constructed for the algorithm to learn. This case is referred to as Case 6.

Table 1 presents a detailed description of all the cases, including the various parameters for small, medium, and large products, respectively. Table 2 describes the algorithm parameter settings for each product. Table 3 shows the results of the algorithm for each case and the results obtained with the CPLEX solver. Table 4 presents a comparison of the results of the Q-learning algorithm and the SARSA algorithm for six cases. The latter part of Table 4 presents the comparison results between the improved Q-learning algorithm and the original Q-learning algorithm for six cases.

**Table 1.** Descriptions of the six products.

Case	Disassembled Product	Number of Disassembly Tasks	Number of Subassemblies
Case 1	Ballpoint pen I	13	15
Case 2	Ballpoint pen II	20	24
Case 3	Copier	32	29
Case 4	Rotor	15	17
Case 5	Radio	30	29
Case 6	Hammer drill	46	63

**Table 2.** Parameters for each case in Q-learning.

Case	Training Steps	Future Return Decay Value	Learning Rate	Action Space Size	Number of Rounds
Case 1	2000	0.9	0.5	13	4
Case 2	2000	0.9	0.5	21	4
Case 3	20,000	0.6	0.5	33	4
Case 4	2000	0.9	0.5	15	4
Case 5	10,000	0.7	0.5	31	4
Case 6	20,000	0.7	0.5	46	4

**Table 3.** Running times and CPLEX results for Q-learning algorithm for different examples.

Case	Improved Q-Learning			CPLEX			Time Ratio
	Disassembly Sequence	$f$	Running Time	Disassembly Sequence	$f$	Running Time	
1	1→3	6	0.05 s	1→3	6	0.32 s	14.33%
2	1→4→7→11	14	0.06 s	1→4→7→11	14	5.94 s	1.03%
3	1→17→12→5	16	0.56 s	1→17→12→5	16	6.88 s	8.18%
4	1→2→3→4	11	0.05 s	1→2→3→4	11	0.55 s	8.55%
5	1→3→4→15	16	0.28 s	1→3→4→15	16	6.13 s	4.58%
6	1→2→4→8→15→25→34→11	45	0.61 s	–	–	–	–

**Table 4.** Running time results of improved Q-learning and SARSA algorithms for different cases.

Case	Improved Q-Learning		SARSA		Gap	Original Q-Learning		Gap
	$f$	Running Time	$f$	Running Time		$f$	Running Time	
1	6	0.05 s	6	0.04 s	0%	6	0.04 s	0%
2	14	0.06 s	29	0.05 s	107.14%	14	1.28 s	0%
3	16	0.56 s	fail	–	–	fail	–	–
4	11	0.05 s	11	0.04 s	0%	11	0.04 s	0%
5	16	0.28 s	fail	–	–	fail	–	–
6	45	0.61 s	56	0.80 s	24.44%	fail	–	–

Figure 7 depicts the convergence of the objective function for Q-learning and Sarsa algorithms in Case 6. As the objective function of this chapter aims to minimize the maximum disassembly time while ensuring a certain disassembly profit, smaller values of the objective function are preferable. From the graph, it can be observed that after stabilization, the objective function value of the Q-learning algorithm is smaller, indicating that it obtains better feasible solutions. Additionally, the stability of the Q-learning algorithm is superior to that of the Sarsa algorithm.

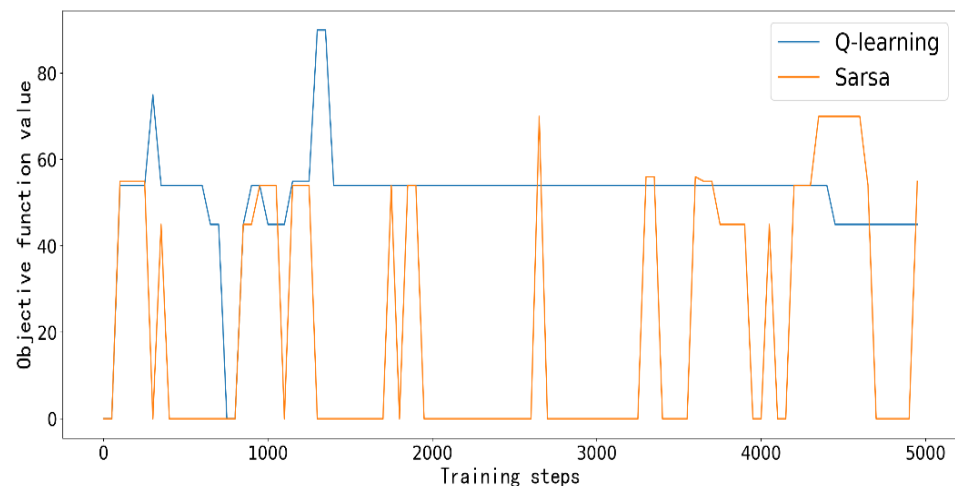
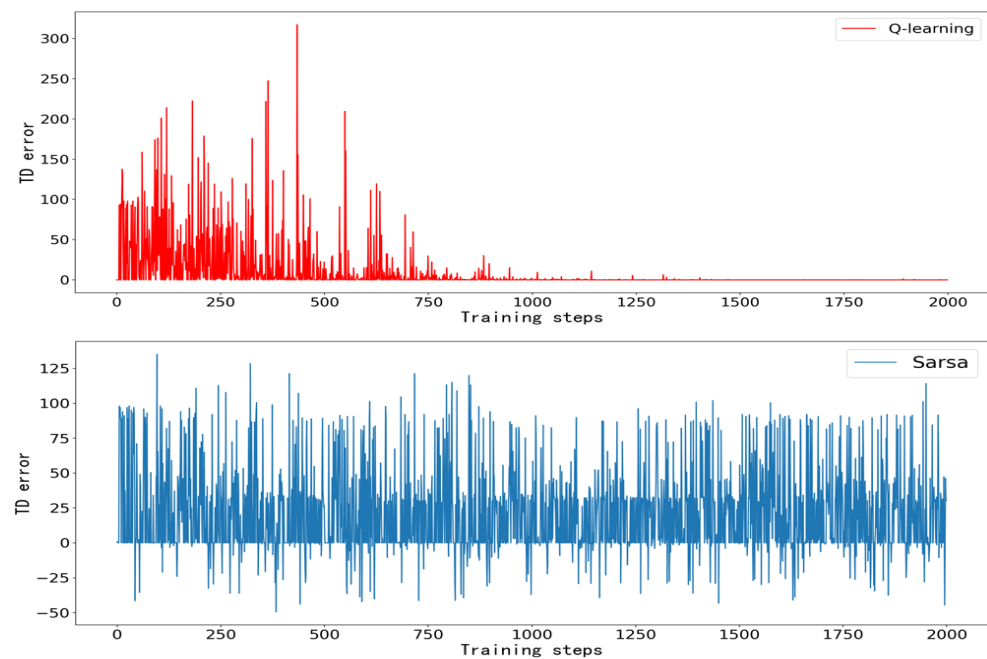
**Figure 7.** Change diagram of objective function value of Q-learning and SARSA algorithms for Case 6.

Figure 8 shows the convergence of TD error for the Q-learning and Sarsa algorithms. It can be observed that the Q-learning algorithm reaches convergence after approximately 1250 training steps, whereas Sarsa remains unstable even after 2000 steps.

From the above tables, it is evident that the improved Q-learning algorithm can indeed find the optimal solution, and its running time is much faster than that of CPLEX across various case scales. Additionally, we can see in the table that the running time of CPLEX grows linearly with the case scale, while the running time of the Q-learning algorithm increases with the case scale. The running time of the Q-learning algorithm does not change significantly as the case size increases, and the differences are negligible, ranging from 0.04 s to 0.61 s. Moreover, according to the results for the first five cases, the maximum running time of the algorithm is only 14% of the running time of CPLEX, which is a small percentage of time. Even in the second case, it is only 1%, showing that the algorithm can more efficiently find the optimal solution compared to CPLEX, which is crucial for companies or factories. Importantly, it can be seen from the experimental results

that in small- and medium-scale cases, CPLEX provides the optimal solution similar to the algorithm.



**Figure 8.** TD error variation diagram for Q-learning and SARSA algorithms.

However, when the last large-scale case with 46 tasks is carried out, CPLEX encounters a memory overflow issue. It can be seen that CPLEX has eliminated many rows and improved the boundary many times. However, because the scale of the calculation example is too large and the RAM in the running environment is insufficient, there is not enough memory to continue constructing the corresponding data in the future. The solution scale is too large for the case, and there are too many possible solutions, making it impossible to find a better solution, whereas the Q-learning algorithm can provide a better solution. This shows that the improved Q-learning algorithm can indeed solve the DLBP, and the solution it provides is optimal. In the future, the scale of our recycled products may become larger and may involve large products that need to be disassembled such as engines and used motorcycles. In this case, CPLEX may struggle to find a better solution, but the proposed algorithm can find a feasible solution in a very short time. This proves that it can address not only the recycling problem of small disassembled products but also the recycling problem of large used products. The proposed algorithm is not bounded by the size of the arithmetic case when solving the DLBP and can provide an optimal solution in a shorter time.

## 5. Conclusions

In this paper, reinforcement learning is applied to solve the DLBP, with the Q-learning algorithm used to solve the problem's discrete nature. A new Q-table is defined, and the algorithm learns to obtain the optimal solution. The Q-table for the corresponding arithmetic cases is built based on the relationship with the OR graph. Experimental demonstrations show that the size of the arithmetic case does not significantly impact the algorithm's execution time. Consequently, it is substantiated that the algorithm efficiently solves the disassembly line balancing problem.

When faced with more complex models or more complex disassembly line balancing problems, the complexity of solving the problem may increase. For three-dimensional and above problems, the performance of the Q-learning algorithm may be slightly worse. In such cases, we need to use deep reinforcement learning techniques, such as the A3C and DDQN algorithms. With such algorithms, we can handle more complex models and



consider more factors. Such algorithms are not limited by the complexity of the disassembly problem. They use neural networks to solve the problem, and the running time of these algorithms is mainly spent on building the neural network [40]. Furthermore, the dimensionality of the information being learned does not significantly impact the performance of these algorithms. Therefore, we can use these algorithms to solve disassembly line balancing problems for complex models more effectively in the future.

**Author Contributions:** Conceptualization, X.G. and S.Q.; methodology, X.Z.; validation, Y.L.; writing—original draft preparation, X.Z.; writing—review and editing, L.Q. and J.W.; visualization, S.Q.; supervision, J.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by the National Natural Science Foundation of China under Grant 61903229 and in part by the Natural Science Foundation of Shandong Province under Grant ZR2022MF270.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

- Xia, X.; Zhu, H.; Zhang, Z.; Liu, X.; Wang, L.; Cao, J. 3D-based multi-objective cooperative disassembly sequence planning method for remanufacturing. *Int. J. Adv. Manuf. Technol.* **2020**, *106*, 4611–4622. [\[CrossRef\]](#)
- Zhang, Z.; Cai, N.; Zeng, Y.; Li, L.; Zou, B. Review of modeling theory and solution method for disassembly line balancing problems for remanufacturing. *China Mech. Eng.* **2018**, *29*, 2636–2645.
- Guo, X.W.; Fan, C.Y.; Zhou, M.; Wang, J.; Liu, S.X.; Qin, S.; Tang, Y. Human–Robot collaborative disassembly line balancing problem with stochastic operation time and a solution via multi-objective shuffled frog leaping algorithm. *IEEE Trans. Autom. Sci. Eng.* **2023**. [\[CrossRef\]](#)
- Zeng, Y.; Zhang, Z.; Liu, J.; Zhang, Y. Modeling and optimization of partially destructive demolition line balancing problem. *Inf. Control* **2020**, *49*, 365–376.
- Liang, J.; Guo, S.; Du, B.; Li, Y.; Guo, J.; Yang, Z.; Pang, S. Minimizing energy consumption in multi-objective two-sided disassembly line balancing problem with complex execution constraints using dual-individual simulated annealing algorithm. *J. Clean. Prod.* **2021**, *284*, 125418. [\[CrossRef\]](#)
- Kucukkoc, I. Balancing of two-sided disassembly lines: Problem definition, MILP model and genetic algorithm approach. *Comput. Oper. Res.* **2020**, *124*, 105064. [\[CrossRef\]](#)
- Xie, M.; Zhang, Z.; Jiang, J. Modeling and optimization of bilateral disassembly line balancing problem considering workstation constraints and energy consumption. *Comput. Integr. Manuf. Syst.* **2021**, *27*, 701–715.
- Wang, S.W.; Guo, X. P.; Liu, J. Research on Optimization Model and Algorithm for Bilateral Disassembly Line Balancing Problem. *Ind. Eng. Manag.* **2018**, *23*, 8–15. [\[CrossRef\]](#)
- Wang, S. *Research on Modeling and Solution Method of Disassembly Line Balance Problem*; Southwest Jiaotong University: Chengdu, China, 2019.
- Zhao, X. *An Optimization Method for Low-Carbon and Efficient Parallel Demolition Line Balancing Problem*; Hefei University of Technology: Hefei, China, 2019.
- Zhang, Z.Q.; Liang, W.; Xie, M.K.; Zheng, H.B. An Elite Differential Evolution Algorithm for the Mixed Model Bilateral Disassembly Line Balancing Problem. *J. Jilin Univ. (Eng. Technol. Ed.)* **2022**, 1–14. [\[CrossRef\]](#)
- Li, R.Y.; Peng, H.M.; Li, R.G.; Zhao, K. A Survey on Reinforcement Learning Algorithms and Applications. *J. Comput. Syst. Appl.* **2020**, *29*, 13–25. [\[CrossRef\]](#)
- Li, R.; Peng, H.; Li, R.; Zhao, K. A multi-intelligent body path planning algorithm based on reinforcement learning. In Proceedings of the 32nd Chinese Process Control Conference (CPC2021), Taiyuan, China, 30 July–1 August 2021.
- Wang, W.; Mo, D.Y.; Wang, Y.; Tseng, M.M. Assessing the cost structure of component reuse in a product family for remanufacturing. *J. Intell. Manuf.* **2019**, *30*, 575–587. [\[CrossRef\]](#)
- Konar, A.; Chakraborty, I.G.; Singh, S.J.; Jain, L.C.; Nagar, A.K. A deterministic improved Q-learning for path planning of a mobile robot. *IEEE Trans. Syst. Man Cybern. Syst.* **2013**, *43*, 1141–1153. [\[CrossRef\]](#)
- Tian, G.; Ren, Y.; Feng, Y.; Zhou, M.; Zhang, H.; Tan, J. Modeling and planning for dual-objective selective disassembly using AND/OR graph and discrete artificial bee colony. *IEEE Trans. Ind. Inform.* **2018**, *15*, 2456–2468. [\[CrossRef\]](#)
- Liu, J.; Wang, S.W. A Dynamic Cooperative Evolutionary Algorithm for Solving Bilateral Sequentially Dependent Disassembly Line Balancing Problems. *J. Syst. Manag.* **2020**, *29*, 1197–1204.
- Tuncel, E.; Zeid, A.; Kamarthi, S. Inventory Management in Multi-Product, Multi-Demand Disassembly Line using Reinforcement Learning. In Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management, Istanbul, Turkey, 3–6 July 2012; pp. 1866–1873.

19. Er, M.J.; Deng, C. Online tuning of fuzzy inference systems using dynamic fuzzy Q-learning. *IEEE Trans. Syst. Man Cybern. Part (Cybern.)* **2004**, *34*, 1478–1489. [[CrossRef](#)] [[PubMed](#)]
20. Schilperoort, J.; Mak, I.; Drugan, M.M.; Wiering, M.A. Learning to play pac-xon with q-learning and two double q-learning variants. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, 18–21 November 2018; pp. 1151–1158.
21. Tan, F.; Yan, P.; Guan, X. Deep reinforcement learning: From Q-learning to deep Q-learning. In Proceedings of the 24th International Conference, ICONIP 2017, Guangzhou, China, 14–18 November 2017; pp. 475–483.
22. Wang, Y.H.; Li, T.H.S.; Lin, C.J. Backward Q-learning: The combination of Sarsa algorithm and Q-learning. *Eng. Appl. Artif. Intell.* **2013**, *26*, 2184–2193. [[CrossRef](#)]
23. Low, E.S.; Ong, P.; Cheah, K.C. Solving the optimal path planning of a mobile robot using improved Q-learning. *Robot. Auton. Syst.* **2019**, *115*, 143–161. [[CrossRef](#)]
24. Wang, H.; Sarker, B.R.; Li, J.; Li, J. Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q-learning. *Int. J. Prod. Res.* **2021**, *59*, 5867–5883. [[CrossRef](#)]
25. Wang, Y.; De Silva, C.W. Multi-robot box-pushing: Single-agent Q-learning vs. team Q-learning. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–13 October 2006; pp. 3694–3699.
26. Lee, D.; Defourny, B.; Powell, W.B. Bias-corrected Q-learning to control max-operator bias in Q-learning. In Proceedings of the 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), Singapore, 16–19 April 2013; pp. 93–99.
27. Wu, K.; Guo, X.; Liu, S.; Qi, L.; Zhao, J.; Zhao, Z.; Wang, X. Multi-objective discrete brainstorming optimizer for multiple-product partial u-shaped disassembly line balancing problem. In Proceedings of the 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 22–24 May 2021; pp. 305–310.
28. Guo, X.; Zhou, M.; Liu, S.; Qi, L. Multiresource-constrained selective disassembly with maximal profit and minimal energy consumption. *IEEE Trans. Autom. Sci. Eng.* **2020**, *18*, 804–816. [[CrossRef](#)]
29. Wu, K.; Guo, X.; Zhou, M.; Liu, S.; Qi, L. Multi-objective discrete brainstorming optimizer for stochastic disassembly line balancing problem subject to disassembly failure. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; pp. 1224–1229.
30. Chen, S.H.; Zhao, C.Y.; Wang, C.; Yan, Z.B. Multi-Agent Path Planning Algorithm Based on Reinforcement Learning. In Proceedings of the 32nd Chinese Process Control Conference (CPCC2021), Taiyuan, China, 30 July–1 August 2021; p. 1619. [[CrossRef](#)]
31. Wang, J.M. Multi-objective Selective Disassembly Sequence Planning Model under Uncertainty Conditions. *J. Remanufacturing Recycl.* **2023**, *16*, 22–26.
32. Chen, T. Improved Ant Colony Optimization Algorithm for Urban Water Supply Pipeline Path Dynamic Planning and Design. *Sci. Technol. Innov.* **2024**, 134–137.
33. Huang, W.J.; Cui, X.; Chen, J.; Rao, Y.J. Coordinated Scheduling of Energy Interconnected Park Based on Multi-Agent Q-Learning Algorithm. *J. Wuhan Univ. (Eng. Ed.)* **2022**, *55*, 1141–1148. [[CrossRef](#)]
34. Zhao, Z.; Liu, S.; Zhou, M.; You, D.; Guo, X. Heuristic scheduling of batch production processes based on petri nets and iterated greedy algorithms. *IEEE Trans. Autom. Sci. Eng.* **2020**, *19*, 251–261. [[CrossRef](#)]
35. Xia, J.P. Research on Balance Optimization of LCD-TV Disassembly Line of A Company Based on Improved NSGA-III Algorithm. Doctoral Dissertation, Southwest Jiaotong University, Chengdu, China, 2024.
36. Zhao, Z.; Zhou, M.; Liu, S. Iterated Greedy Algorithms for Flow-shop Scheduling Problems: A Tutorial. *IEEE Trans. Autom. Sci. Eng.* **2021**, *19*, 1941–1959. [[CrossRef](#)]
37. Zhao, Z.; Liu, S.; Zhou, M.; Abusorrah, A. Dual-objective mixed integer linear program and memetic algorithm for an industrial group scheduling problem. *IEEE/CAA J. Autom. Sin.* **2020**, *8*, 1199–1209. [[CrossRef](#)]
38. Zhang, L.; Geng, X.R.; Tao, K.B. Stochastic Parallel Disassembly Line Balancing Optimization Considering Carbon Emissions and Revenue. *J. Mech. Eng.* **2023**, *59*, 330–338.
39. Wang, Y.P.; Cui, Z.W.; Cao, K.; Yang, L.; Shen, W.J. Time Series Differential Phase Classification of Ground-based SAR Based on Attention Network. *J. Signal Process.* **2021**, *37*, 1207–1216. [[CrossRef](#)]
40. Zhang, Y.; Gao, B.; Guo, L.; Chen, H.; Zhao, J. Velocity control in a right-turn across traffic scenario for autonomous vehicles using kernel-based reinforcement learning. In Proceedings of the 2017 Chinese Automation Congress (CAC), Jinan, China, 20–22 October 2017; pp. 6211–6216.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.