Adaptive disassembly sequence planning for VR maintenance training via deep reinforcement learning

Haoyang Mao1 · Zhenyu Liu1 · Chan Qiu1

Received: 5 May 2021 / Accepted: 25 October 2021 / Published online: 17 November 2021
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2021

Abstract
VR training equipped with meta-heuristic disassembly planning algorithms has been widely applied in pre-employment training in recent years. However, these algorithms are usually authored for specific sequences of a single product, and it remains a challenge to generalize them to maintenance training with unpredictable disassembly targets. As a promising method for settling dynamic and stochastic problems, deep reinforcement learning (DRL) provides a new insight to dynamically generate optimal sequences. This study introduces the deep Q-network (DQN), a successful DRL method, to fulfill adaptive disassembly sequence planning (DSP) for the VR maintenance training. Disassembly Petri net is established to describe the disassembly process, and then the DSP problem is defined as a Markov decision process that can be solved by DQN. Two neural networks are designed and updated asynchronously, and the training of DQN is further achieved by backpropagation of errors. Especially, we replace the long-term return in DQN with the fitness function of the genetic algorithm to avoid dependence on the immediate reward. Several experiments have been carried out to exhibit great potentials of our method in on-site maintenance where the fault is uncertain.

Keywords Disassembly sequence planning · Deep reinforcement learning · Genetic algorithm · VR maintenance training

1 Introduction
Over the past decade, great quantities of meta-heuristic methods have been developed for DSP problems [7], including genetic algorithm (GA) [8–11], artificial bee colony (ABC) [12–14], ant colony optimization (ACO) [15, 16], particle swarm optimization (PSO) [17, 18], simulated annealing (SA) [14], and tabu search (TS) [19]. These methods are independent on specific problems and expose acceptable performance in searching optimal or near-optimal solutions and generalization. Recently, several works were proposed to integrate advantages of different algorithms, and tried to make up for the limitations of classic algorithms in local search ability and convergence speed. Tao et al. [20] presented a TS-based hyper-heuristic algorithm with an exponentially decreasing diversity management strategy, which was proved more efficient for complex DSP problems. Inspired by the regenerative properties of the flatworm, Tseng et al. [21] proposed a novel flatworm algorithm that has great mechanisms to avoid trapping into a local optimum of DSP problems. Moreover, other fantastic algorithms such as artificial fish swarm (AFS) [22], fireworks algorithm (FWA) [23], firefly algorithm (FA) [24], and flower pollination algorithm (FPA) [25] were successively proposed for the solutions of DSP problems. Most of the swarm intelligence and evolutionary
algorithms above are inspired by some random phenomena in nature and widely applied for the goal-determined DSP tasks. However, these traditional meta-heuristics seem incompetent at on-site maintenance tasks, owing to the unpredictability of damaged targets.

In view of this, Tuncel et al. [26] applied a reinforcement learning method based on Monte Carlo for settling flexible disassembly problems, which extends new approaches to solve DSP problems in stochastic environments. Nonetheless, similar to Xia et al. [27] who employed the Q-learning method for the selective disassembly of waste electrical and electronic equipment, it is inevitable to cause a memory explosion since the complexity of products enlarges. By introducing deep neural networks to achieve the approximation of action-value function [28], deep reinforcement learning (DRL) methods have witnessed significant breakthroughs in several fields such as games [29–32], path planning [33–36], and dynamic scheduling [37–39]. Even so, when applied to DSP problems, it remains quite difficult to associate actions with returns since the reward is only observed after a series of disassembly operations, which is referred to as temporal credit assignment problem on sparse rewards.

To address all the issues mentioned above, this paper presents an improved deep Q-network (DQN) optimized by GA, namely GO-DQN, for adaptive disassembly sequence planning (ADSP) in VR training system. Firstly, disassembly Petri net (DPN) is utilized to model the disassembly process. Then, an improved DQN is employed for the solution of sequences by defining the DSP problem as a Markov decision process (MDP). Specifically, the fitness function of GA takes over from the long-term return in DQN to avoid dependence on the immediate reward. Finally, a VR training system is established to verify the effectiveness of the proposed method within a laboratory environment. Experimental results will hopefully serve as potential value for improvements in on-site maintenance work.

\[ \text{DPN} = (P, T, I_{\text{success}}, O_{\text{success}}, u_0, G, C, D, W) \]  
where

1. \( P \) is the set of places, which denotes the disassembly compositions at each step, i.e., it represents the set of products when as root, the subassemblies when as intermediate nodes, and the components when as leaves.
2. \( T \) is the set of transitions, each transition means a disassembly operation of a component, \( P \cap T = \emptyset \), and \( P \cup T \neq \emptyset \).
3. \( I_{\text{success}} : P \times T \rightarrow N \) is the matrix of \( n \times m \) dimensions that indexes all the directed arcs from places to transitions, where \( n \) denotes the number of places and \( m \) denotes the number of transitions.
4. \( O_{\text{success}} : P \times T \rightarrow N \) is the matrix of \( m \times n \) dimensions which indexes all the directed arcs from transitions to places.
5. \( u_0 \) is a vector of \( n \) dimensions representing the initial state of disassembly, which means the product keeps intact and no parts have been disassembled.
6. \( G \) is the time gradient representing the time spent on each disassembly operation. To make the process easy to calculate, we divide it into 16 grades from 30 s to 30 min. \( G \) is defined as follows:
   \[ G = \{g_1, g_2, ..., g_{16}\}, g_i \in \{0, 1, 2, ..., 15\} \]
7. \( C \) is the disassembly cost, including tools to use and change of directions. Similar to the time, the costs are divided into 4 gradients.
   \[ C = \{c_1, c_2, ..., c_4\}, c_i \in \{0, 1, 2, 3\} \]
8. \( D \) is the degree of difficulty for disassembly. For convenience, we divided the difficulty into 5 levels, including very easy, easy, normal, hard, and very hard.
   \[ D = \{d_1, d_2, ..., d_5\}, d_i \in \{1, 2, 3, 4, 5\} \]
9. \( W \) is the set of weights on the above three indexes.

Let \( E \) be the evaluation of the disassembly process, then it can be computed as
\[ E = \sum_{i=1}^{n} (w_1 \cdot g_i + w_2 \cdot c_i + w_3 \cdot d_i) / i, w_1 + w_2 + w_3 = 1, w_i \in W \]  

(2)

If there are transitions selected and fired, the next disassembly state \( u \) can be defined as
\[ u = u_0 + f \cdot (O_{\text{success}} - I_{\text{success}}^T) \]  

(3)

where \( f \) is the binary vector of \( m \) dimensions indicating whether the transition is fired.

2 DRL method for adaptive disassembly planning

Taking into account the uncertainty of the targets to be disassembled at the maintenance site, this section represents an approach of ADSP based on DQN.

2.1 Modeling for the disassembly process

To describe the disassembly process during training, we erect a nine-tuple DPN as follows:
A simple example of a product is shown in Fig. 1, and the DPN can be expressed as follows:

If we disassemble the product in the order \( A \rightarrow D \rightarrow B \rightarrow C \), that is \( f = (0, 1, 0, 1, 1) \), then the new disassembly state can be computed by Eq. (3).

\[
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 
\end{pmatrix},
\]

\( u_0 = (1, 0, 0, 0, 0, 0, 0) \)

### 2.2 Markov decision process

Constructing an environment in which a DRL agent can take actions and gain rewards is of great importance when applying the DRL method to ADSP. Furthermore, the newly established environment must meet the MDP framework. The MDP framework describes the environment with a 5-tuple \( \langle S, A, P, R, \gamma \rangle \), where \( S \) indicates the finite state space, \( A \) indicates the finite action space, \( P \) indicates the probability of a state transition, \( R \) indicates the reward, and \( \gamma \) indicates the discount factor.

As to the disassembly problem for VR maintenance training, DPN can be used as a bridge to convert it into an MDP. As mentioned above, the disassembly states can be expressed by the set of places \( P \) in DPN, whether there is a token in the place indicates whether the corresponding part could be disassembled. By establishing a mapping from the places \( P \) to the state space \( S \), the disassembled states are further transformed into finite state space in the MDP framework. Similarly, the disassembly operations can be conveyed by the transitions \( T \) in DPN as well as the action space \( A \) in MDP. Describe all the possible disassembly states of the agent’s environment as a set \( S \), where the current state is represented by \( S_t \in S \). During the interaction process, the action \( A_t \) is selected and executed from the set of all the possible disassembly operations \( \mathcal{A}(S_t) \). After that, the environment will return a reward value \( R_t \in R \) and reach a new state \( S_{t+1} \). Since the probability and reward of transforming to the next disassembly state only depend on the current disassembly operation and state, having nothing to do with historical ones, the disassembly task fulfills the MDP condition.

Further, we define the cumulative reward as the return \( G_t \), which represents the cumulative sum of the reward \( R \) from the current moment until the end of the task. Since the impact of reward on the action will gradually weaken over time, \( G_t \) is defined as follows:

\[
G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}
\]  

(4)

The discount rate \( 0 < \gamma \leq 1 \) denotes the present value of future rewards.
2.3 Sequence planning using DQN

To effectively make a trade-off between exploration and exploitation during the disassembly process, ε - greedy algorithm is adopted to select the action \( a \). After executed in the state \( s \), a new state \( s' \) will be explored, which can be expressed as

\[
s' = s + (O_m \cdot a - I_m \cdot a) \quad (5)
\]

where \( O_m \cdot a \) denotes the line \( a \) of the matrix \( O_m \) with \( m \times n \) dimensions.

And, the reward \( R_i \) of the action \( a \) that represents the \( i \)th operation in the state \( s \) can be computed as

\[
R_i = w_1 \cdot g_i + w_2 \cdot c_i + w_3 \cdot d_i \quad (6)
\]

Then, the state-action value function \( Q(s, a) \) can be updated by the reward \( R_i \):

\[
Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha \left[ R_i + \gamma \max_{a' \in A(s')} Q(s', a') \right] \quad (7)
\]

where \( \alpha (0 < \alpha \leq 1) \) is the learning rate, and \( \gamma (0 \leq \gamma \leq 1) \) is the discount factor.

Then, we fabricate two neural networks to complete error backpropagation and update the weights, one of which generates the value function approximation of the current state, called \( Q_{\text{evaluate}_1} \), while the other generates that of the next state, called \( Q_{\text{target}_1} \). Both networks can be expressed as

\[
Q(s, a, \theta) \approx Q(s, a) \quad (8)
\]

where \( \theta \) is the parameter of the fitting function, representing the weight of the connection between two layers of neurons in the network.

As shown in Fig. 2, the \( Q_{\text{evaluate}_1} \) outputs \( Q_{\text{evaluate}_2}(s, a, \theta) \), while the \( Q_{\text{target}_1} \) chooses the action for the next state that yields \( \arg \max_a Q(s', a', \theta_1) \) in the \( Q_{\text{evaluate}_1} \), i.e., \( Q_{\text{target}_2}(s', a', \theta_2) \). Based on this, the action \( a \) is selected by \( \varepsilon - \text{greedy} \) algorithm and executed in the state \( s \), receiving the reward \( R \) and then transferred to the next state \( s' \), all of which are stored in the replay memory \( D \) as a tuple \((s, a, R, s')\). During this process, \( T \) mini-batch samples are randomly extracted from \( D \) to update \( \theta_1 \). After that, the fitting action value \( y_i = r + \gamma \max_{a'} Q(s', a', \theta_2) \) can be calculated, and the loss is further computed as

\[
L(\theta) = E \left[ (R + \gamma \max_{a'} Q(s', a', \theta_2) - Q(s, a, \theta_1))^2 \right] \quad (9)
\]

Then, the gradient descent method is applied to update \( \theta_1 \) of the \( Q_{\text{evaluate}_1} \). Further, \( \theta_2 \) is updated as \( \theta_2 = \theta_1 \) after a fixed interval.

### 3 Optimization of DQN based on GA

In DSP problems, only when the entire disassembly process is finished will the agent receive a reward, which makes it quite difficult to associate each action with rewards. Benefiting from GA’s powerful ability to address temporal credit assignment, the use of fitness indicators that merge the returns of the entire episode makes it indifferent for GA to the sparsity of reward distribution and robust to long-term horizons [40]. Inspired by this, we propose an improved DQN optimized by GA named GO-DQN for the solution of optimal sequence in a VR training system, which uses the return in RL as the fitness function

\[
F_{\text{fitness}} = E_N(G_i) = E_N \left( \sum_{k=1}^{l} y^k R_{t+k+1} \right) \quad (10)
\]

where \( N \) denotes the number of episodes, \( l \) denotes the length of an episode, and \( G_i \) denotes the total return of each episode.

Figure 3 clearly illustrates the workflow of GO-DQN, which mainly includes the following four steps:

**Step 1: initialization.** A population of \( Q_{\text{evaluate}} \) is initialized as pop\(_{\text{init}}\), as well as an empty cyclic replay buffer \( D \). Moreover, one additional \( Q_{\text{evaluate}} \) alongside \( Q_{\text{target}} \) is initialized as \( Q_{\text{eva}} \) and \( Q_{\text{lag}} \) with weight \( \theta^Q_{\text{eva}} \) and \( \theta^Q_{\text{lag}} \), respectively.

**Step 2: elite selection.** Recycle the interactions between individuals in the population and the environment, and use the cumulative reward as the fitness of individuals to select elites.
Step 3: crossover and mutation. Select individuals from the population by segmented weight selection as parents with elites for crossover and mutation to create the next generation of $Q_{evaluate\_nets}$.

Step 4: network update. Periodically, we sample a random mini-batch of $T$ transitions $(s_i, a_i, R_i, s'_i)$ from $D$ and acquire the action-value function

$$y_i = R_i + \gamma Q_{tag}(s'_i, a'_i, \theta_{Q_{eva}})$$ (11)

Then, the loss function is computed as follows:

$$L = \frac{1}{T} \sum_i (y_i - Q_{eva}(s_i, a_i, \theta_{Q_{eva}}))^2$$ (12)

Thereby $Q_{eva}$ can be updated by minimizing the loss, and the weights can be updated as well.

Finally, we copy the $Q_{evaluate\_net}Q_{eva}$ into the population $pop_k$.

For optimum $\theta_{Q_{eva}} \in pop_k$: $\theta_{Q_{eva}} \Rightarrow \theta_{Q_{eva}}$

4 System implementation and experimental demonstration

4.1 System design and implementation

Embedding the GO-DQN method, the overall architecture of the training system consists of three major layers: data layer, function layer, and interaction layer, as illustrated in Fig. 4. Three-dimensional models and the interference matrix of industrial products to be disassembled are prepared in advance and stored in the data layer, which are used for training RL models. Once the sequences are generated in the function layer, the system will create disassembly animations for demonstration automatically and store them in the data layer. VR guidance instructions for training and evaluation will be created as well. The interaction layer supplies an interactive interface that allows users to specify their input preferences and accept corresponding feedback. Other VR modules, such as collision detection, haptic feedback, tracking, and rendering, have been developed to realize immersive training.

To conduct the data transmission between function layers, a server-client communication module has been established to integrate the disassembly sequence planning module with the VR application. A local server with Python installed is set up to handle requests from Unity.

When Unity receives inputs from the user, it will send requests to the server, which relies on the UnityWebRequest system. Then, the server will call the Python scripts for sequence calculation and return the results to Unity in JSON format with the help of a web framework called Flask. The results are further presented through the user interface (UI) and communicate with HTC VIVE at the same time to achieve multi-channel information feedback, as shown in Fig. 5.

Given the poor adaptability and waste of space resources caused by storing disassembly animations locally in most VR systems, this study has achieved automatic generation of disassembly animations by obtaining prefabricated templates...
associated with the parts to be disassembled from the server. Specifically, the 3D model of each part to be disassembled in the input product is stored as a Unity template and matched with the corresponding interference matrix. After the optimal sequence is generated, the system will successively call the templates stored in the Unity resource folder according to the order of the parts in the sequence and determine the removal directions (one of the six directions: $+x$, $+y$, $+z$, $-x$, $-y$, $-z$) of the parts according to the interference matrix. Finally, by combining all the called templates, the automatic synthesis of disassembly animations is achieved.

For obtaining the best experience, multi-sensory feedback is realized in this system integrating visual, auditory, and haptic feedback. Visual feedback includes the feedback of part status and text information; the former is acquired by selected part highlighting algorithm. When the virtual hand touches the part to be disassembled, its outline will be highlighted to indicate that the selection is correct. After the disassembly operation is completed, the part will return to its original state, and the panel will prompt the user for the next operation. If the user makes a wrong choice, the handle will vibrate to warn, and the helmet will give voice prompts at the same time, as shown in Fig. 6.

### 4.2 Experimental demonstration on aircraft engine

The model of the aircraft engine used in this demonstration is shown in Fig. 7, which consists of 25 parts. Since the
complete aircraft engine involves lots of parts, to facilitate subsequent calculations, we have omitted the connectors between parts such as bolts and nuts. Similarly, some trivial parts that do not affect the overall disassembly sequence are ignored as well.

To train the RL model based on the proposed method, a modified MountainCar-v0 environment has been developed based on OpenAI Gym. Each step is defined as an instance where the agent takes an action and receives a reward back from the environment, and all the steps taken by \( Q_{\text{evaluate}_nets} \) in the population are cumulative. During each training generation, the \( Q_{\text{evaluate}_net} \) with the highest fitness is selected as the best and then tested on 5 task instances, and the average score is logged as its performance.

Then, three experiments have been designed to verify the effectiveness of our method. In experiment I, we select the rear main bearing (#9) as the only disassembly target. For experiment II, we select the nose cone (#2), the rear main bearing (#9), and tapped covers (#12) as the disassembly targets with no additional preferences. As to experiment III, we choose additional preferences which are to exclude cam pulleys (#22) and the exhaust pipes (#11).

### 4.3 Results and discussion

As for the on-site maintenance, it is usually more desirable to use a shorter time and less cost to complete the training, thereby fixing the fault as soon as possible. Therefore, we set the weights of each index in the formula (6) as follows: \( w_1 = 0.5, w_2 = 0.4, w_3 = 0.1 \). The size of the population is set to 10, and the elite fraction is set to 0.1. The mutation probability is set to 0.5 with the expectation for more searches in the solving space, and the mutation fraction is set to 0.1 as well as the mutation strength corresponding to a 10% Gaussian noise. Adam optimizer with a learning rate of 0.0001 is used while the discount rate is set to 0.99. The training process is shown in Fig. 8, and the results for the above three experiments are represented in Table 1 and Fig. 9.

As indicated in Table 1, the developed system is able to automatically generate the optimal disassembly sequence according to different inputs of the trainee. Compared with the previous RL method [26, 27], the proposed GO-DQN algorithm can arrange the temporal credit assignment in DSP tasks quite well, which attributes to the following two improvements: (1) replace the return in DQN with the fitness function of GA and (2) update the optimal individuals of the population in each iteration by gradient descent. Since the states experienced by individuals with high fitness have a higher long-term return, their experience will be more likely to be stored in the replay buffer, which is a form of implicit prioritization effective for domains with long-term horizons and sparse rewards.

In experiment I, the system calculates the optimal disassembly sequence 25-24-23-22-21-11-10-9 and converts it into operable instruction after the user selects the target part #9, the whole process of which takes 0.7 s on average for 50 attempts. When the targets increase to three, as set in experiment II, the system is still able to immediately calculate the
optimal sequence, which only takes 0.2 s longer than experiment I. In experiment III, after the user chooses additional preferences which are to exclude #22 and #11, the system is unable to acquire any feasible sequence, this is because they are the parts that must be removed before disassembling parts #2, #9, and #12, which is determined by the constraint relationships between parts.

As a summary of the above analysis, GO-DQN succeeds GA’s invariance to sparse rewards with long-term horizons as well as DQN’s ability to leverage gradients for higher sample efficiency. Benefitting from these advantages, the built system can generate optimal disassembly sequences within a short time after receiving user commands.

For most VR training systems, the adoption of traditional meta-heuristics such as GA makes them only suitable for specific sequences of a single product [5, 41, 42]. Once the disassembly targets are not included in the sequences designed in advance, these algorithms will work abnormally and need to be modified or even rewritten. By contrast, the proposed RL-based method can flexibly output the corresponding optimal sequences according to different input targets, which is more satisfactory for on-site maintenance where the disassembly targets are often fickle due to the randomness of the faults. It is noted that this study is based on a real engineering issue of our cooperative enterprise. Yet due to its complex production line involving a large number of products and components as well as complicated environmental restrictions including illumination, radiation, and electromagnetic compatibility, it will take some time to launch this training system. Presently, it is still in the design and simulation stage and will be online and put into use after all the production line tests are completed in the later phase. Admittedly, with the increase in the complexity of the input products, the training cycle of the RL model will extend. How to shorten the training process is one direction of our future exploration. In addition, the system can also be used in different production lines and maintenance workshops by importing the models and interference matrices of other new products.

5 Conclusion and future work

To deal with the high cost and giant threshold of equipment maintenance caused by the haphazardness of faults in the maintenance task, this paper puts forward a method of GO-DQN coupled with GA optimization, which has been confirmed to be applicable for ADSP in on-site VR maintenance training system. The disassembly process is modeled by DPN, and then the DQN method is adopted for solving the DSP problem defined by the MDP. After that, two networks with different parameters are constructed to perform the backpropagation of errors for training. To minimize the impact of

![Running results of the three experiments in the training system](image)

**Table 1** Experimental results

| Experiment   | Target(s) | Sequence                        |
|--------------|-----------|---------------------------------|
| Experiment I | #9        | 25-24-23-22-21-11-10-9          |
| Experiment II| #2, #9, #12| 1-2-25-24-23-22-21-11-10-9-12  |
| Experiment III| #2, #9, #12| No sequence available           |

(avoid #11, #22)
immediate reward, the return in DQN is substituted by the fitness function of GA. Especially, through a series of experiments on the aircraft engine, we demonstrated for the first time the system equipped with the proposed method holds great potentials at on-site maintenance tasks.

In summary, the main contributions of our work are as follows:

1. From the method perspective, GO-DQN inherits GA’s competence to handle temporal credit assignment by its adoption of a fitness metric that integrates the return of an entire episode, as well as DQN’s update method based on gradient descent for speeding up the iteration.

2. From the technical application, the VR maintenance training system equipped with the GO-DQN method can dynamically generate optimal sequences based on different input targets, thus meeting the needs for cost-saving and efficient repair under the on-site maintenance environment with uncertain fault.

For future work, we will explore the following aspects:

1. Combine Petri nets with graph neural networks to seek more efficient expressions for the disassembly process.

2. Introduce heuristic knowledge for training such as expert experience, and cloud server if possible, so as to accelerate the training process.

3. Verify the robustness and efficiency of the system in much more complex and changeable environments.

References

1. Guo Z, Zhou D, Zhou Q, Zhang X, Geng J, Zeng S, Lv C, Hao A (2020) Applications of virtual reality in maintenance and repairing the industrial product lifecycle: a systematic review. J Manuf Syst 56:525–538. https://doi.org/10.1016/j.jmsy.2020.07.007

2. Berg LP, Vance JM (2017) Industry use of virtual reality in product design and manufacturing: a survey. Virtual Real 21:1–17. https://doi.org/10.1007/s10055-016-0293-9

3. Roldán JJ, Crespo E, Martín-Barrio A, Peña-Tapia E, Barrientos A (2019) A training system for Industry 4.0 operators in complex assemblies based on virtual reality and process mining. Robot Comput Integr Manuf 59:305–316. https://doi.org/10.1016/j.rcim.2019.05.004

4. Abidi MH, Al-Ahmari A, Ahmad A, Ameen W, Alkhalefah H (2019) Assessment of virtual reality-based manufacturing assembly training system. Int J Adv Manuf Technol 105:3743–3759. https://doi.org/10.1007/s00170-019-03801-3

5. Yang Y, Yang P, Li J, Zeng F, Yang M, Wang R, Bai Q (2019) Research on virtual haptic disassembly platform considering disassembly process. Neurocomputing 348:74–81. https://doi.org/10.1016/j.neucom.2018.05.120

6. Guo X, Zhou M, Abusorrah A, Alsokhry F, Sedraoui K (2020) Disassembly sequence planning: a survey. IEEECAA J Autom Sin 1–17. https://doi.org/10.1109/JAS.2020.1003515

7. Gao KZ, He ZM, Huang Y, Duan PY, Suganthan PN (2020) A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing. Swarm Evol Comput 57:100719. https://doi.org/10.1016/j.swevo.2020.100719

8. Giudice F, Fargione G (2007) Disassembly planning of mechanical systems for service and recovery: a genetic algorithms based approach. J Intell Manuf 18:313–329. https://doi.org/10.1007/s10845-007-0025-9

9. Tseng Y-J, Kao H-T, Huang F-Y (2010) Integrated assembly and disassembly sequence planning using a GA approach. Int J Prod Res 48:5991–6013. https://doi.org/10.1080/00207540903229173

10. Tseng H-E, Chang C-C, Lee S-C, Huang Y-M (2018) A block-based genetic algorithm for disassembly sequence planning. Expert Syst Appl 96:492–505. https://doi.org/10.1016/j.eswa.2017.11.004

11. Ren Y, Zhang C, Zhao F, Xiao H, Tian G (2018) An asynchronous parallel disassembly planning based on genetic algorithm. Eur J Oper Res 269:647–660. https://doi.org/10.1016/j.ejor.2018.01.055

12. Ren Y, Tian G, Zhao F, Yu D, Zhang C (2017) Selective cooperative disassembly planning based on multi-objective discrete artificial bee colony algorithm. Eng Appl Artif Intell 64:415–431. https://doi.org/10.1016/j.engappai.2017.06.025

13. Liu J, Zhou Z, Pham DT, Xu W, Yan J, Liu A, Ji C, Liu Q (2018) An improved multi-objective discrete bees algorithm for robotic disassembly line balancing problem in remanufacturing. Int J Adv Manuf Technol 97:3937–3962. https://doi.org/10.1007/s00170-018-2183-7

14. Tian G, Ren Y, Feng Y, Zhou M, Zhang H, Tan J (2019) Modeling and planning for dual-objective selective disassembly using and/or graph and discrete artificial bee colony. IEEE Trans Ind Inform 15:2456–2468. https://doi.org/10.1109/TII.2018.2884845

15. Luo Y, Peng Q, Gu P (2016) Integrated multi-layer representation and ant colony search for product selective disassembly planning. Comput Ind 75:13–26. https://doi.org/10.1016/j.compind.2015.10.011

16. Tseng H-E, Chang C-C, Lee S-C, Huang Y-M (2019) Hybrid bidirectional ant colony optimization (hybrid BACO): an algorithm for disassembly sequence planning. Eng Appl Artif Intell 83:45–56. https://doi.org/10.1016/j.engappai.2019.04.015

Author contribution Haoyang Mao: conceptualization, methodology, investigation, data curation, software, formal analysis, experiment, and writing of the original draft. Zhenyu Liu: conceptualization, resources, funding acquisition, supervision, project administration, and writing including review. Chan Qiu: methodology, software, equipment, visualization, validation, supervision, and writing including review and editing.

Funding This study is supported by the National Natural Science Foundation of China (52075480, 51875517), Key Research and Development Program of Zhejiang Province (2021C01008), and High-level Talent Special Support Plan of Zhejiang Province (2020RS2004).

Data availability The data that support the findings of this study are available upon reasonable request.

Declarations

Ethics approval This manuscript has not been published elsewhere in part or entirety and is not under consideration by any other journals.

Conflict of interest The authors declare no competing interests.
17. Tseng Y-J, Yu F-Y, Huang F-Y (2011) A green assembly sequence planning model with a closed-loop assembly and disassembly sequence planning using a particle swarm optimization method. Int J Adv Manuf Technol 57:1183–1197. https://doi.org/10.1007/s00170-011-3339-x

18. Li WD, Xia K, Gao L, Chao K-M (2013) Selective disassembly planning for waste electrical and electronic equipment with case studies on liquid crystal displays. Robot Comput-Integr Manuf 29:248–260. https://doi.org/10.1016/j.rcim.2013.01.006

19. Alshibli M, El Sayed A, Kongar E, Sobh TM, Gupta SM (2016) Disassembly sequencing using tabu search. J Intell Robot Syst 82:69–79. https://doi.org/10.1007/s10864-015-0289-9

20. Tao F, Bi L, Zuo Y, Nee AYC (2018) Partial/parallel disassembly sequence planning for complex products. J Manuf Sci Eng 140:011016. https://doi.org/10.1115/1.4037608

21. Tseng H-E, Huang Y-M, Chang C-C, Lee S-C (2020) Disassembly sequence planning using a flatworm algorithm. J Manuf Syst 57:416–428. https://doi.org/10.1016/j.jmsy.2020.10.014

22. Zhang Z, Wang K, Zhu L, Wang Y (2017) A Pareto improved artificial fish swarm algorithm for solving a multi-objective fuzzy disassembly line balancing problem. Expert Syst Appl 86:165–176. https://doi.org/10.1016/j.eswa.2017.05.053

23. Lu C, Li J-Y (2017) Assembly sequence planning considering the effect of assembly resources with a discrete fireworks algorithm. Int J Adv Manuf Technol 93:3297–3314. https://doi.org/10.1007/s00170-017-0663-9

24. Zhu L, Zhang Z, Wang Y (2018) A Pareto firefly algorithm for multi-objective disassembly line balancing problems with hazard evaluation. Int J Prod Res 56:7354–7374. https://doi.org/10.1080/00207543.2018.1471238

25. Wang K, Li X, Gao L (2019) A multi-objective discrete flower pollination algorithm for stochastic two-sided partial disassembly line balancing problem. Comput Ind Eng 130:634–649. https://doi.org/10.1016/j.cie.2019.03.017

26. Tuncel E, Zeid A, Kamarthi S (2014) Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. J Intell Manuf 25:647–659. https://doi.org/10.1007/s10806-012-0711-0

27. Xia K, Gao L, Li W, Wang L, Chao K-M (2014) A Q-learning based selective disassembly planning service in the cloud based remanufacturing system for WEEE. In: Volume 1: Materials; micro and nano technologies; properties, applications and systems; sustainable manufacturing. American Society of Mechanical Engineers, Detroit, Michigan, USA, p V001T04A012

28. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing Atari with deep reinforcement learning. Comput Sci

29. Moravcik M, Schmid M, Burch N, Lisy V, Morrill D, Bard N, Davis T, Waugh K, Johanson M, Bowling M (2017) DeepStack: expert-level artificial intelligence in heads-up no-limit poker. Science 356:508–513. https://doi.org/10.1126/science.aam6960

30. Jaderberg M, Czarnecki WM, Dunning I, Marris L, Lever G, Castañeda AG, Beattie C, Rabinowitz NC, Morcos AS, Ruderman A, Sonnerat N, Green T, Deason L, Leibo JZ, Silver D, Hassabis D, Kavukcuoglu K, Graepel T (2019) Human-level performance in 3D multiplayer games with population-based reinforcement learning. Science 364:859–865. https://doi.org/10.1126/science. aau6249

31. Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T, Lillicrap T, Simonyan K, Hassabis D (2017) Mastering chess and shogi by self-play with a general reinforcement learning algorithm. https://arxiv.org/abs/1712.01815

32. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction (2nd edition). The MIT Press

33. Fan T, Long P, Liu W, Pan J (2020) Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. Int J Robot Res 39:856–892. https://doi.org/10.1177/0278364920916531

34. Kim M, Han D-K, Park J-H, Kim J-S (2020) Motion planning of robot manipulators for a smoother path using a twin delayed deep deterministic policy gradient with hindsight experience replay. Appl Sci Basel 10:575. https://doi.org/10.3390/app10020575

35. Jiang L, Huang H, Ding Z (2020) Path planning for intelligent robots based on deep Q-learning with experience replay and heuristic knowledge. IEEE-CAA J Autom Sin 7:1179–1189. https://doi.org/10.1109/JAS.2019.9117132

36. Liu Q, Shi L, Sun L, Li J, Ding M, Shu FS (2020) Path planning for UAV-mounted mobile edge computing with deep reinforcement learning. IEEE Trans Veh Technol 69:5723–5728. https://doi.org/10.1109/TVT.2020.2982508

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

38. Shi D, Fan W, Xiao Y, Lin T, Xing C (2020) Intelligent scheduling of discrete automated production line via deep reinforcement learning. Int J Prod Res 58:3362–3380. https://doi.org/10.1080/00207543.2020.1717008

39. Kim YG, Lee S, Son J, Bae H, Chung BD (2020) Multi-agent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system. J Manuf Syst 57:440–450. https://doi.org/10.1016/j.jmsy.2020.11.004

40. Khadka S, Tumer K (2018) Evolution-guided policy gradient in reinforcement learning. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, pp 1196–1208

41. Ahmad A, Al-Ahmari AM, Aslam MU, Abidi MH, Darmoul S (2015) Virtual assembly of an airplane turbine engine. IFAC-Pap 48:1726–1731. https://doi.org/10.1016/j.ifacol.2015.06.335

42. Jiang W, Zheng J, Zhou H, Zhang B (2016) A new constraint-based virtual environment for haptic assembly training. Adv Eng Softw 98:58–68. https://doi.org/10.1016/j.advengsoft.2016.03.004

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.