



Disassembly line optimization with reinforcement learning

Tamás Kegyes^{1,2} · Zoltán Süle^{1,2} · János Abonyi¹

Accepted: 12 January 2024
© The Author(s) 2024

Abstract

As the environmental aspects become increasingly important, the disassembly problems have become the researcher's focus. Multiple criteria do not enable finding a general optimization method for the topic, but some heuristics and classical formulations provide effective solutions. By highlighting that disassembly problems are not the straight inverses of assembly problems and the conditions are not standard, disassembly optimization solutions require human control and supervision. Considering that Reinforcement learning (RL) methods can successfully solve complex optimization problems, we developed an RL-based solution for a fully formalized disassembly problem. There were known successful implementations of RL-based optimizers. But we integrated a novel heuristic to target a dynamically pre-filtered action space for the RL agent (DLOPTRL algorithm) and hence significantly raise the efficiency of the learning path. Our algorithm belongs to the Heuristically Accelerated Reinforcement Learning (HARL) method class. We demonstrated its applicability in two use cases, but our approach can also be easily adapted for other problem types. Our article gives a detailed overview of disassembly problems and their formulation, the general RL framework and especially Q-learning techniques, and a perfect example of extending RL learning with a built-in heuristic.

Keywords Disassembly line optimization · Reinforcement learning · Q-learning · Mixed-integer quadratic programming · Heuristically accelerated reinforcement learning · DLOPTRL algorithm

✉ János Abonyi
janos@abonyilab.com

¹ HUN-REN-PE Complex Systems Monitoring Research Group, Egyetem utca 10., Veszprém 8200, Hungary

² Department of Computer Science and Systems Technology, University of Pannonia, Egyetem utca 10., Veszprém 8200, Hungary

1 Introduction

The increasing environmental pressures caused by human activities drive governments and organizations to limit or decrease their footprints. As an outcome, the European Commission declared this a strategic goal. A new Circular Economy Action Plan for a cleaner and more competitive Europe has been announced to sustainably transform the economy and society. There is a wide range of recommendations and proposals on how these kinds of issues should be handled, and the key point is to reduce consumption and increase the recycling rate. The efforts led to the creation of the concept of circular economy, which is more complex than just a manufacturing optimization because it also covers the supply chain, disassembly, and recycling optimization steps (Loiseau et al. 2016; Camacho-Otero et al. 2018). A process flow overview diagram of the circular economy is shown in Fig. 1 (Kalmykova et al. 2018).

Every step in the circular process has practices and methods to optimize the operation. Still, the primary goal of the circular economy is to optimize the whole supply chain up to recycling. The classical problem of manufacturing supply chain optimization is a deeply researched topic (Beamon 1999; de Koster et al. 2007; Hervani et al. 2005). The circular economy optimization goal also covers the recollection and recycling processes that can be a bit more complex because these are more stochastic and less controlled processes. Disassembly line design and balancing problems describe the methodological background of the relevant segment of the whole circle (Jovane et al. 1993; Dufflou et al. 2008; Sasi-kumar and Kannan 2008). The most basic models assumed deterministic inputs but the advanced models started to handle the uncertainty observed in real problems. Among others, the source materials and their distributions are described

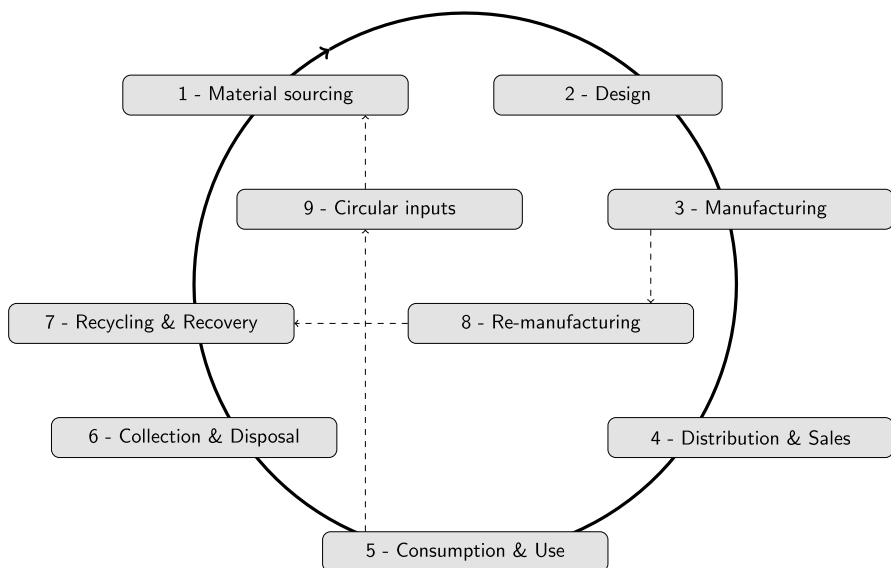


Fig. 1 Schematic flow of circular economy Kalmykova et al. (2018)

by stochastic variables. Similarly, the disassembly tasks, their required process times, and the demand values of recycling steps can also be stochastic attributes in some models.

The supply chain optimization and assembly line balancing topics have been intensively studied issues since the 1950s years, while the focused analysis of disassembly lines started almost 40 years later (Gupta and McLean 1996). A detailed historical overview of disassembly methods (Kim et al. 2007) delivered a summary table of the different solutions, which was extended by using a comprehensive classification (Chand and Ravi 2023) in Table 1. It shows a wide range of machine-learning methods for solving the disassembly line balancing problem. There are already successful RL-based solutions with attractive learning performance, but it needs to be explained in detail how to construct an effective implementation for a given problem.

The above-collected facts strengthened our motivation to prepare an effective self-learning RL-based solution that is easy to adapt to different disassembly problems. We present a general guide on how to define the reward function from problem parameters. Furthermore, we highlight the importance of customizing the action-taking method, which will be relevant whenever a general RL framework is parameterized. According to the current outlook, the number of disassembly lines and their optimization requirements will increase significantly, and such a self-learning solution will explain this adequately.

Our first experiences with an RL-based solution show that the most basic version of training an RL agent is not obviously efficient: after declaring the state- and action spaces the central task is to define the reward function. Most of the RL frameworks support this approach without the option of customizing the action-taking method. But a huge portion of infeasible state-action pairs should not be learned. For disassembly line balancing problems, it is an obvious option to dynamically filter for the possible uncompleted actions by considering the precedence graph. We found that integrating such an efficient state-dependent action-restriction method can radically reduce the learning path. The same conclusion has been found (Woo and Sung 2020). The major idea was to apply the identical constraints that were set up in mixed-integer quadratic problem (MIQP) formulation, and hence to deliver a systematic method to define an appropriate restriction for action-selection step. The key result of our article is to present a Q-learning solution with an integrated heuristic for dynamic state-dependent action restriction. The `DLOptRL` algorithm is described and explained in detail. From a higher aspect, our algorithm belongs to the class of Heuristically Accelerated Reinforcement Learning (HARL) methods, which is highlighted by an appropriate formulation as well.

Our article stands for the following major parts:

- First, in Sect. 2.1, we will give a problem formulation of disassembly line optimization problems. We will summarize the notations, declare the objective function components, and present an MIQP formulation with fewer decision variables. It enables solving larger-size problems with the widely used solvers by keeping the limitations of decision variables.

Table 1 Overview of disassembly line balancing solutions

Solution method	Author(s)	Year	Ref	Speciality
Reverse MRP	S. Gupta, K. Taleb	1994	Gupta and Taleb (1994)	
Integer programming	D-H Lee, H-J Kim, G Choi, P Xirouchakis	2004	Lee et al. (2004)	
Branch and bound	H-J Kim, D-H Lee, P. Xirouchakis, OK Kwon	2009	Kim et al. (2009)	
Heuristic	K-N Taleb, S-M Gupta, L. Brennan	1997	Taleb et al. (1997)	Single product
Heuristic	K-N Taleb, S-M Gupta	1997	Taleb and Gupta (1997)	Multi-product
Petri-net	K-P Neundorff, D-H Lee, D. Kiritsis, P. Xirouchakis	2001	Neundorff et al. (2001)	
Ant colony optimization	Wang, Y. Rong, D. Xiang	2014	Wang et al. (2014)	
Ant colony optimization	J. Wang, X. Wu, X. Fan	2015	Wang et al. (2015)	
Ant colony optimization	Y. Luo, Q. Peng, P. Gu	2016	Luo et al. (2016)	Two-stage
Ant colony optimization	H-E Tseng, C-C Chang, S-C Lee, Y-M Huang	2019	Tseng et al. (2019)	Multi-layer
Artificial bee colony	Y. Ren, G. Tian, F. Zhao, D. Yu, C. Zhang	2017	Ren et al. (2017)	Hybrid bidirectional
Artificial bee colony	G. Tian, Y. Ren, Y. Feng, M. Zhou, H. Zhang, J. Tan	2018	Tian et al. (2018)	Multi-objective
Discrete bees algorithm	W. Xu, Q. Tang, J. Liu, Z. Liu, Z. Zhou, D-T Pham	2020	Xu et al. (2020)	Dual-objective
Genetic algorithm	T-F Go, D-A Wahab, M-A Rahman, R. Ramli	2010	Go et al. (2010)	Human-robot collaboration
Genetic algorithm heuristic	J. L. Rickli, J. A. Camelio	2013	Rickli and Camelio (2013)	Multi-objective
Genetic algorithm	H-E Tseng, S-C Lee	2018	Tseng and Lee (2018)	Interactive
Genetic algorithm	Y. Ren, C. Zhang, F. Zhao, H. Xiao, G. Tian	2018	Ren et al. (2018)	Asynchronous
Genetic algorithm	Y. Tian, X. Zhang, Z. Liu, X. Jiang, J. Xue	2019	Tian et al. (2019)	Cooperative
Genetic algorithm	J-C Chen, Y-Y Chen, T-L Chen, Y-C Yang	2022	Chen et al. (2022)	Adaptive
Particle swarm opt	W-D Li, K. Xia, L. Gao, K-M Chao	2013	Li et al. (2013)	Selective disassembly
Artificial fish swarm algo	J. Guo, J. Zhong, Y. Li, B. Du, S. Guo	2018	Guo et al. (2018)	Hybrid model
Teaching-learning-based opt	K. Xia, L. Gao, L. Wang, W. Li	2013	Xia et al. (2013)	Simplified method
Teaching-learning-based opt	K. Xia, L. Gao, L. Wang, W. Li, X. Li, W. Ijomah	2016	Xia et al. (2016)	Modified method
Graph-based method	S. Smith, G. Smith, W-H Chen	2012	Smith et al. (2012)	Sequence structure graph

Table 1 (continued)

Solution method	Author(s)	Year	Ref	Speciality
Graph-based method	P. Mitrouchev, C-G Wang, L-X Lu, G-Q Li	2015	Mitrouchev et al. (2015)	Geometry contacting graph
Fuzzy model	X-F Zhang, G. Yu, Z-Y Hu, C-H Pei, G-Q Ma	2014	Zhang et al. (2014)	Fuzzy-rough set mapping
Fuzzy model	H-P Hsu	2016	Hsu (2016)	Fuzzy knowledge-based
Reinforcement learning	E. Tuncel, A. Zeid, S. Kamarthi	2014	Tuncel et al. (2014)	
Reinforcement learning	S. Mete, F. Serin	2021	Mete and Serin (2021)	

- Sect. 3 will present the general framework of Reinforcement Learning solutions. We will describe the Q-learning method in detail that was applied in our development. We will also summarize all the necessary steps that should be prepared for an RL solution to the previously defined disassembly problem.
- Then, in Sect. 3.2, we will describe a novel algorithm called `DLOptRL`. It contains a built-in heuristic to minimize applicable action space and speed up the learning of an RL agent. Our algorithm highlights how RL methods can be combined with problem-specific heuristics to get an efficient self-learning solution. We will also point out that our algorithm belongs to the Heuristically Accelerated Reinforcement Learning class.
- Sect. 4 will describe two commonly used use cases for which we will summarize a MIQP solution as a reference and the results of our `DLOptRL` learning path. We will show that our solution approximates the optimal solution effectively without any prior preparations.
- Finally, in Sect. 5, we will take an overview of our results and summarize some potential directions for further research.

2 Optimization of disassembly line balancing

In this section, we will summarise the general notations of disassembly problems. Then we will overview the solution methods, including the problem formulation as a linear programming task.

2.1 Problem formulation

We consider a disassembly line balancing model for a single product with a finite supply. There are N^c elementary components in each product to remove. The task of eliminating component i is specified by its processing time T_i^{rm} , while a boolean flag of h_i indicates its hazardousness. The general problem is to assign every task to workstations of the disassembly line to optimize the objective function. We will make the following additional assumptions:

- There are N_a^{ws} available workstations that are preliminarily prepared.
- All workstations are identical, and they are capable of performing any component removal tasks.
- The cycle time is denoted by T^c . Each workstation should finish its allocated removal tasks on the current product in the disassembly line. The cycle time is preliminarily defined.
- A precedence graph describes the logical dependencies of component removal tasks. The vertices represent the components to be removed. The edges are directed, and there are two types of edges: AND type and OR type edges. The removal process of a component i can be started, only if all the components from which a directed AND-type path goes to the vertex of i are already removed, and at least one of the components from which a directed OR-type path goes to the

Table 2 Notations of disassembly line balancing problems

N^c	Number of disassembly tasks
N_a^{ws}	Number of available workstations
N_u^{ws}	Number of used workstations
T_i^{rm}	Part removal time of component i
T^c	Cycle time
T_j^{ws}	Total time for workstation j to perform all assigned tasks
C_k	k th components in disassembly sequence
h_i	boolean flag indicates the hazardousness of component i
d_i	demand of of the component i

vertex of i is removed. Typically the precedence graph is used in its transitive reduced form.

- A solution is described by a sequence of the component removal tasks, where C_k denotes the k th component in the disassembly sequence.
- A workstation will perform a continuous range of component removal tasks.
- The total time for workstation j to perform all of its assigned tasks is denoted by T_j^{ws} .
- N_u^{ws} denotes the number of workstations used.

The major attributes of the disassembly line balancing problem are summarised in Table 2.

2.2 Objective function

There are several ways to measure how well is a disassembly line balanced. Based on the different disassembly optimization solutions collected in Table 1, there are two major approaches for the objective: one is cost-benefit-based, and the other one is based on the processed quantities. By following the mixed objective approach (Tuncel et al. 2014), we used a combination of three components in our analysis.

- $F_1 = \min\{\sum_{j=1}^{N_u^{ws}} (T^c - T_j^{ws})^2\}$ minimizes the total idle time of workstations used,
- $F_2 = \min\{\sum_{k=1}^{N^c} (k \cdot h_{C_k})\}$ forces to remove hazardous components as early as it can,
- $F_3 = \min\{\sum_{k=1}^{N^c} (k \cdot d_{C_k})\}$ supports removing components with higher demand earlier.

The preliminary defined circle time and the total sum of components' removal times and the number of used workstations determine the total idle time: $N_u^{ws} \cdot T^c - \sum_{i=1}^{N^c} T_i^{rm}$. The shorter idle time results a higher processed quantity. The objective function F_1 amplifies the imbalance and contains the corresponding items in a quadratic term. Hence it supports not only minimizing the idle times but also decreasing the imbalance. Objective functions F_2 and F_3 depend on the component property and disassembly sequence. The component's hazardousness requires additional care or causes extra risk, which motivates removal as early as possible. A

binary boolean indicator describes the Hazardousness property. F_2 prefers to remove hazardous components earlier than non-hazardous ones. F_3 is similar to F_2 construction except that demand values are not binary but non-negative figures, and they represent the disassembled components' importance in remanufacturing due to the benefit values. It was reviewed and shown (Laili et al. 2020) that using the multi-objective approach for disassembly optimization problems is quite general. Further aspects can also be taken into account by adding other components to the objective for minimizing the disassembly cost, maximizing the profit obtained from disassembly, or minimizing environmental pollution.

In our article, according to the analyzed use cases, we will use a linear combination of the three selected objectives: $w_1F_1 + w_2F_2 + w_3F_3$. In this context, weights of w_1 , w_2 , and w_3 have double roles. These should compensate for the scaling discrepancies of the objectives, and the weights can determine the relative importance of the objectives based on external preferences. The first role could be substituted by normalizing or standardizing the objectives. However the second aspect cannot be replaced with an autonomous solution, although the relative importance constantly changes in real-world problems. A dynamic weighting optimization that reflects the external conditions is out of the scope of our article. Hence we assume that weights of w_1 , w_2 , and w_3 are preliminarily defined as external parameters.

2.3 Linear programming problem formulation

There are already described formulations of the disassembly problem in the literature (Kalaycilar et al. 2016), but we will present a new formulation with a decreased size of decision variables. It stands for six significant types of decision variables:

- Type 1 decision variables describe which of the removable component is assigned to a disassembly sequence order number:

$$x_{i,j}^{Type1} = \begin{cases} 1 & \text{if component } i \text{ will be removed} \\ & \text{as } j\text{th task in the removal sequence} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

- Type 2 decision variables determine the process times of every step in the disassembly sequence:

$$x_j^{Type2} = T_{C_i}^{rm}. \quad (2)$$

- Type 3 decision variables describe the workstation on which the component will be removed:

$$x_{i,k}^{Type3} = \begin{cases} 1 & \text{if component } i \text{ will be removed on workstation } k \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

- Type 4 decision variables determine whether a workstation will be in use or not:

$$x_k^{Type4} = \begin{cases} 1 & \text{if workstation } k \text{ will be in use} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

- Type 5 decision variables determine the total idle time of each workstation if it is in use:

$$x_k^{Type5} = T^c x_k^{Type4} \sum_i x_{i,k}^{Type3} C_i. \quad (5)$$

- Type 6 decision variables describe the workstation assignments in the disassembly sequence order:

$$x_j^{Type6} = \begin{cases} k & \text{if workstation } k \text{ will be assigned the } j\text{th component in the disassembly sequence} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Then we need to set up constraints to satisfy all the requirements collected in Sect. 2.1:

- Constraints guarantee that each component is listed exactly once in the removal sequence:

$$\sum_{j=1}^{N^{ws}} x_{i,j}^{Type1} = 1 \quad \forall i \in \{1, \dots, N^c\} \quad (7)$$

- The workstation's process time should not exceed the cycle time limit:

$$\sum_{i=1}^{N^c} T_i^{rm} \cdot x_{i,k}^{Type3} \leq T^c \quad \forall k \in \{1, \dots, N^{ws}\} \quad (8)$$

Before declaring the constraints of the precedence graph, we need to declare the two types of predecessor relations:

- Predecessor AND relation ($P_{AND}(i)$) declares a set of predecessor tasks that all need to be finished before starting task i .
- Predecessor OR relation ($P_{OR}(i)$) declares a set of predecessor tasks of which at least one needs to be finished before starting task i .

Figure 2 shows examples of predecessor AND and OR relations.

In this context, we can continue to define the required constraints:

- All the predecessor tasks with AND relation should be assigned earlier in the sequence than the particular one:

$$\sum_{j=1}^{N^c} j \cdot x_{i,j}^{Type1} \leq \sum_{j=1}^{N^c} j \cdot x_{i,j}^{Type1} \quad \forall i \in \{1, \dots, N^c\}; \forall l \in P_{AND}(i) \quad (9)$$

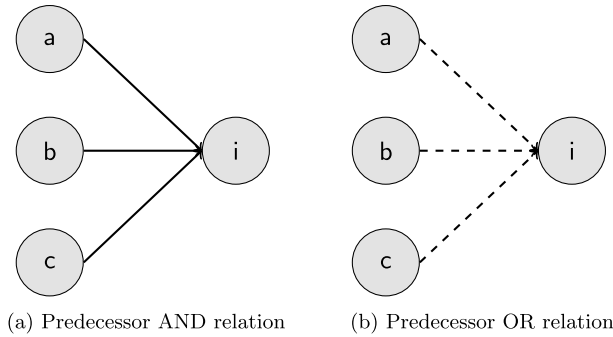


Fig. 2 Predecessor relation types

- At least one of the predecessor tasks with OR relation should be assigned earlier in the sequence than the particular one:

$$x_{i,j}^{Type1} \leq \sum_{h=1}^j \sum_{l \in P_{OR}(i)} j \cdot x_{l,h}^{Type1} \quad \forall i \in \{1, \dots, N^c\}; \forall j \in \{1, \dots, N^c\} \quad (10)$$

- Assuring that a disassembly task is assigned to exactly one workstation:

$$\sum_{k=1}^{N^{ws}} x_{i,k}^{Type3} = 1 \quad \forall i \in \{1, \dots, N^c\} \quad (11)$$

- The workstation assignments should be in a monotone sequence:

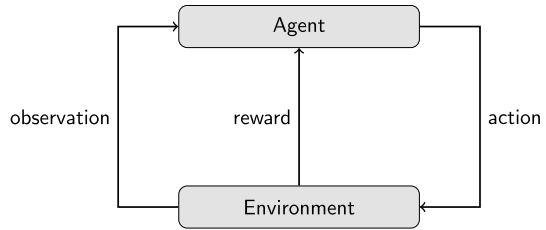
$$0 \leq x_j^{Type6} - x_{j-1}^{Type6} \leq 1 \quad \forall j \in \{2, \dots, N^c\} \quad (12)$$

- Integer and non-negative properties:

$$\begin{aligned} x_{i,j}^{Type1} &\in \{0, 1\} & \forall i, j \in \{1, \dots, N^c\} \\ 0 \leq x_j^{Type2} &\leq T^c & \forall j \in \{1, \dots, N^c\} \\ x_{i,k}^{Type3} &\in \{0, 1\} & \forall i \in \{1, \dots, N^c\} \\ & & \forall k \in \{1, \dots, N^{ws}\} \\ 0 \leq x_k^{Type4} &\leq T^c & \forall k \in \{1, \dots, N^{ws}\} \\ x_k^{Type5} &\in \{0, 1\} & \forall k \in \{1, \dots, N^{ws}\} \\ x_j^{Type6} &\in \{1, \dots, N^{ws}\} & \forall j \in \{1, \dots, N^c\} \end{aligned} \quad (13)$$

By solving the MIQP problem, we can conduct an optimal solution, but in practice, it can be a heavily resource-intensive process for mid and large-scale problems. A former analysis of a profit-oriented linear objective shows that an exact MILP solution cannot be reached in a reasonable time limit for disassembly problems over 60 components (Kalaycilar et al. 2016). In this formulation, the number of decision variables was quadratic to the number of components. We used a modified formulation with $2(N^c)^2 + 4N^c$ decision variables for an N^c -component disassembly problem. Moreover, our weighted objective leads to a quadratic optimization problem.

Fig. 3 Reinforcement learning framework



We used a Gurobi-based solver from Matlab, and experienced the same issue: no solution was found within the same time limit.¹ This fact may bring alternative solutions to the fore, especially reinforcement learning methods.

3 Formulation of the disassembly line balancing problem as an RL-based optimization task

Reinforcement learning (RL) refers to learning problems. An agent takes observations of the environment, and based on that, it executes an action (A_t). As a result of the action in the environment, the agent will get a reward (R_t), and it can take a new observation (O_t) from the environment, and the cycle is repeated. The problem is to let the agent learn to maximize the total reward. Figure 3 shows the general reinforcement learning framework.

Reinforcement learning is based on the reward hypothesis which states that all goals can be described by the maximization of expected cumulative rewards.

Formally the history is the sequence of observations, actions, and rewards: $H_t = O_1, T_1, A_1, \dots, A_{t-1}, O_t, R_t$. The state is the information used to determine what happens next. Formally, state is a function of the history: $S_t = f(H_t)$. A state is Markov if and only if $\mathbb{P}[S_{t+1} | S_t] = \mathbb{P}[S_{t+1} | S_1, \dots, S_t]$. Markov property is fundamental to the theoretical basis of RL methods. G_t denotes the total discounted reward from time-step t : $G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$.

The state value function $v(s)$ gives the expected total discounted return if starting from state s : $v(s) = \mathbb{E}[G_t | S_t = s]$. The Bellman Equation practically states that the state value function (VF) can be decomposed into two parts: immediate reward (R_{t+1}) and the discounted value of successors states $\gamma v(S_{t+1})$.

The policy covers the agent's behavior in all possible cases, so it is essentially a map from states to actions. There are two major categories in it: deterministic policy ($a = \pi(s)$) and stochastic policy ($\pi(a | s) = \mathbb{P}[A_t = a | S_t = s]$).

We will focus on using an action-value function to determine the current optimal action. However, for large state- and/or action spaces it can be a very slow process to keep the value function updated (and hence optimal).

There are several situations when the learning process is not based on just own experience. Formally this means that action-value function $q_{\pi}(s; a)$ is determined by observing results of an external behavior policy $\mu(a|s)$.

A possible way to handle the difference between target and behavior policy is to modify the value-function update logic as Q-learning does (Sutton and Barto

¹ Similarly to (Kalaycilar et al. 2016), we applied 3.600 sec. the time limit for executions.

2018, Section 6.5). Assume that in state S_t the very next action is derived by using behavior policy: $A_{t+1} \sim \mu(\cdot|S_t)$. By taking action A_{t+1} immediate reward R_{t+1} and the next state S_{t+1} will be determined. But for value-function update let's consider an alternative successor action based on target policy: $A' \sim \pi(\cdot|S_t)$. Then the Q-learning value-function update will look like: $Q(S_t;A_t) \leftarrow Q(S_t;A_t) + Q(S_t;A_t) \leftarrow Q(S_t;A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1};A') - Q(S_t;A_t))$.

In a special case if target policy π is chosen as a pure greedy policy and behavior policy μ follows ϵ -greedy policy then so-called SarsMAX update can be defined as follows: $Q(S;A) \leftarrow Q(S;A) + \alpha(R + \gamma \max_{a'} Q(S';a') - Q(S;A))$. Last, but not least it was proven that Q-learning control converges to the optimal action-value function: $Q(s;a) \rightarrow q_*(s;a)$.

3.1 Design of reinforcement learning solution

In this section, we will present a reinforcement learning-based solution design.

- *State space*: A state needs to contain all the relevant information from the past and be identical for equivalent situations. Hence, the current state should contain the set of removed and remaining components as well as the utilization of the active workstation. So we can declare the state vector in a similar way to the multi-type decision variable in the MIQP formulation:

- the performed removal steps and the remaining ones:

$$\hat{x}_i^{Type1} = \begin{cases} 1 & \text{if component } i \text{ has been removed} \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

- the current utilization of the active workstation:

$$\hat{x}^{Type2} = \text{total assigned removal time of last active workstation.} \quad (15)$$

- *Action space*: The next action is determined by considering both the current status and the constraints defined by the precedence graph. Formally, it will be described by the component's identifier which one needs to be removed next. It is essential to highlight that a built-in heuristic to limit the potential actions to the feasible ones can significantly speed up the solution.
- *Reward*: Reward function is defined as the reciprocal of the weighted sum of objective components:

$$\frac{1}{w_1 F_1 + w_2 F_2 + w_3 F_3}. \quad (16)$$

- *Reinforcement Learning method*: Considering that both state- and action spaces are discrete, we decided to use the Q-learning method. Triplets of the single state vector, the next action, and cumulative discounted rewards will determine Q-table rows (practically Q-table structure).

- *Q-table growth*: Some approaches suggest declaring the Q-table structure initially, and during the learning phase its rows need to be updated. By having a precedence graph determining the total number of feasible states is not trivial. Hence, we applied a dynamic Q-table growth mechanism (Viharos and Jakab 2021): the learning phase starts with an empty Q-table, and whenever a new state-action pair is observed, it should be inserted into the Q-table. Therefore, the Q-table contains only visited rows.
- *Knowledge transition*: Whenever the RL agent experiences a better reward from a visited state than the former best one, the Q-table needs to be updated accordingly. The learning process can be sped up by using the knowledge transition process. In this case, the Q-table updates are made backward from the latter visited states of the episode to the former ones. The key idea is to update not only the visited state-action pairs of the episode but all further state-action pairs that lead to the visited route. In other words, the rewards of those state-action pairs, which partially overlap with the visited episode, can also be updated.
- If the agent should make the optimal action and not a random one, but it is not known (not listed in Q-table yet) because the current state has never been visited before, then a random decision will be made as a fallback action.
- A disabling discount factor can simplify the Q-learning method. The reason is that the state defines how many actions are required to finish the episode, so the discount factor value could be easily calculated from the state. On the other hand, its ability to differentiate between potential routes by considering their lengths also breaks off.
- Q-learning method works with ϵ -Greedy decisions: the RL agent takes a random action with ϵ probability or the best-known action based on the Q-table with $(1 - \epsilon)$ probability. There are different ϵ -strategies, from which we tested the following four:
 1. *Pure ϵ -Greedy approach*: during the whole simulation, the value of ϵ is constant in all the episodes.
 2. *Two-step ϵ -Greedy approach*: in the first phase of the simulation ϵ has a value of 100%, and hence the RL agent takes only random actions, while in the second phase of the simulation ϵ switches to a lower reasonable constant.
 3. *Linearly decreasing ϵ approach*: the value of ϵ starts from 100% at the beginning and linearly decreases to 0% proportionally to the progress of the simulation.
 4. *Sigmoid-shape ϵ approach*: ϵ value goes from 100% to 0%, but in contrast to the linear version, it follows a sigmoid-shape curve.

Figure 4 shows the tested ϵ -functions by the progress of the simulation (in proportion of scheduled episodes).

3.2 Disassembly optimization algorithm with reinforcement learning

This section will describe a reinforcement learning-based method for disassembly optimization with a built-in heuristic to determine the following action.

Algorithm 1 Procedure for disassembly line optimization with reinforcement learning

```

1: function DLOPTRL( $P, \mathbf{t}, \mathbf{h}, \mathbf{d}, c, \mathbf{o}$ )
2: Input:
3:    $P$   $n \times n$  matrix represents the precedence graph       $\triangleright n = |P|$  is the number of parts to remove
4:    $\mathbf{t}$   $n$ -length vector describes part removal times
5:    $\mathbf{h}$   $n$ -length vector indicates hazardousness of components
6:    $\mathbf{d}$   $n$ -length vector determines demand values
7:    $c$  contains cycle time
8:    $\mathbf{o}$  3-length vector declares the objective components' weights
9:
10:   $l \leftarrow 1500$                                            $\triangleright$  initialize learning parameters
11:   $Q((\mathbf{s}_1, \mathbf{s}_2), a, r) \leftarrow []$                        $\triangleright$  learning length: number of episodes
12:  for  $i = 1$  to  $l$  do                                      $\triangleright$  initialize Q-table
13:     $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow \mathbf{0}$                               $\triangleright$  execute  $l$ -episode simulation
14:     $\triangleright n$ -length  $\mathbf{s}_1$  describes the processed steps, while scalar  $s_2$  shows current workstation utilization
15:     $r \leftarrow 0$                                           $\triangleright$  reset state vector components:
16:     $k \leftarrow 1$                                           $\triangleright$  reset cumulative reward
17:     $\mathbf{s} \leftarrow \mathbf{0}$                                         $\triangleright$  set workstation assignment pointer
18:     $\mathbf{w} \leftarrow \mathbf{0}$                                         $\triangleright$  reset  $n$ -length component ordering vector
19:    for  $j = 1$  to  $n$  do                                      $\triangleright$  reset  $n$ -length workstation assignment vector
20:       $\mathbf{b} \leftarrow (\mathbf{1} - (((s_1 P_{AND}) < (\mathbf{1} P_{AND})) \cdot (s_1 P_{OR} == \mathbf{0}))) \cdot (\mathbf{1} - s_1)$   $\triangleright$  determine applicable steps
21:       $\xi_1 \leftarrow \mathcal{U}(0, 1)$                               $\triangleright$  generate a random number by standard uniform distribution
22:      if  $\xi_1 > \frac{1}{i}$  then                                    $\triangleright$  Q-table based optimal action
23:         $m \leftarrow \max(r | q \in \{Q(\mathbf{s}_1, \mathbf{s}_2, a, r) | \mathbf{s}_1, \mathbf{s}_2\})$   $\triangleright$  get maximal expected reward from Q-table
24:         $\mathcal{M} \leftarrow \{a | q \in \{Q(\mathbf{s}_1, \mathbf{s}_2, a, r) | \mathbf{s}_1, \mathbf{s}_2, r = m\}\}$   $\triangleright$  get maximal-reward steps from Q-table
25:         $\mathbf{b} \leftarrow \mathbf{b} \cdot \mathbf{a}_{\mathcal{M}}$                           $\triangleright$  update applicable steps by intersecting maximal-reward steps
26:        if  $|\mathbf{b}| == 0$  then
27:          goto else branch in line 29  $\triangleright$  if no applicable step then fallback to random action
28:        end if
29:      else
30:         $\xi_2 \leftarrow \mathcal{U}(0, 1)$                               $\triangleright$  generate a random number by standard uniform distribution
31:         $\mathbf{z}(x) \leftarrow \frac{1}{\|\mathbf{b}\|} \mathbf{b} U$                         $\triangleright$  calculate cumulative distribution
32:         $a \leftarrow \arg\max_x \mathbf{z}(x) < \xi_2$                   $\triangleright$  determine random action
33:      end if
34:       $\mathbf{o}(j) \leftarrow a$                                       $\triangleright$  store  $j$ th action in episode ordering vector
35:       $\mathbf{s}_1(a) \leftarrow 1$                                     $\triangleright$  register component  $a$  as processed step
36:      if  $c - s_2 > \mathbf{t}(a)$  then
37:         $k \leftarrow k + 1, s_2 \leftarrow 0$   $\triangleright$  if workstation utilization requires then move workstation pointer on
38:      end if
39:       $\mathbf{w}(j) \leftarrow k$                                       $\triangleright$  assign workstation  $k$  to the  $j$ th component removal action
40:       $s_2 \leftarrow s_2 + \mathbf{t}(a)$                               $\triangleright$  add component  $a$  removal time to workstation utilization
41:    end for
42:     $\mathbf{r} \leftarrow \text{reward}(\mathbf{h}, \mathbf{d}, c, \mathbf{o}, \mathbf{w})$                   $\triangleright$  get rewards
43:    for  $j = 1$  to  $n - 1$  do
44:       $(\hat{\mathbf{s}}_1, \hat{s}_2) \leftarrow (\mathbf{s}_1, s_2)|_{\text{processed to step } j}$   $\triangleright$  get state vectors restricted/reconstructed until step  $j$ 
45:      if  $\exists (\hat{\mathbf{s}}_1, \hat{s}_2, a, \cdot) \in Q$  then                    $\triangleright$  the state-action pair exists in the Q-table
46:        if  $r > \max_q Q((\hat{\mathbf{s}}_1, \hat{s}_2), \mathbf{s}(j+1), q)$  then     $\triangleright$  current reward > best in Q-table
47:           $Q((\hat{\mathbf{s}}_1, \hat{s}_2), \mathbf{s}(j+1), q) \leftarrow Q((\hat{\mathbf{s}}_1, \hat{s}_2), \mathbf{s}(j+1), r)$   $\triangleright$  update Q-table record
48:        end if
49:      else                                                  $\triangleright$  no appropriate record in Q-table
50:         $Q \leftarrow Q \cup ((\hat{\mathbf{s}}_1, \hat{s}_2), a, r)$             $\triangleright$  insert Q-table record
51:      end if
52:    end for
53:  end for
54: end function

```

Generally, in reinforcement learning applications, all the feedback arrives in the reward, which means that the agent takes a sequence of actions and will experience whether it works fine or not. In this approach, the agent is not restricted to preserving itself from an easily foreseeable bad action. Instead, it will realize the badness of the actions only afterward by getting the low reward values. Our algorithm implemented a heuristic within the action-choosing process to significantly decrease the potential action space's size. The principal idea was that all the restrictions we declared in the MIQP formulation could be used to pre-filter possible actions before the agent chooses the final one. Such an approach helps the agent discover only the feasible part of the action space and not waste time exploring irrelevant paths, which significantly speeds up the learning phase.

DLOPTRL algorithm requires the following inputs:

- P_{AND} is an $n \times n$ matrix representing the predecessor AND relations of the disassembly problem's precedence graph, where $n = |P|$ is the total number of parts to remove. According to the edge types of a precedence graph, the matrix elements have a binary applicable value set. The value of $P(i, j)$ is defined by the type of $e(v_i, v_j)$ as follows: 0 means no direct predecessor AND dependencies between part i and j (represented by v_i and v_j in the precedence graph) during the disassembly process; 1 describes an existing predecessor AND relation between part i and j .
- P_{OR} is an $n \times n$ matrix representing the predecessor OR relations of the disassembly problem's precedence graph similar to P_{AND} construction.
- \mathbf{t} is an n -length vector that describes part removal times
- \mathbf{h} is an n -length binary vector that indicates hazardousness of each component
- \mathbf{d} is an n -length vector that determines the demand values of parts
- c is a constant that describes cycle-time
- \mathbf{o} is a 3-length vector that contains the objective components' weights

After entering into an outer loop that iterates the episodes, we should initialize some technical variables (counters and pointers), and an inner loop is started that represents a single episode. We should first determine the applicable steps by considering the current disassembly status. Then a decision should be made about how will be determined the next action. The agent will try to follow the known best option that will be successful in the case of a double condition will be met, namely:

- we are on a known track, and hence at least one applicable step is known, and
- the generated standard uniform random number is over the uniformly decreasing threshold

If both the conditions above are satisfied, we will exploit our cumulative knowledge and the optimal action will be chosen that will result in the highest reward for it. Otherwise, the next action will be chosen randomly out of the applicable actions. The chosen action will be registered into the short-term episode history.

After the episode ends, we will retrieve the reward. Then, we need to check whether the visited state-action pairs are already registered in the long-term Q-table.

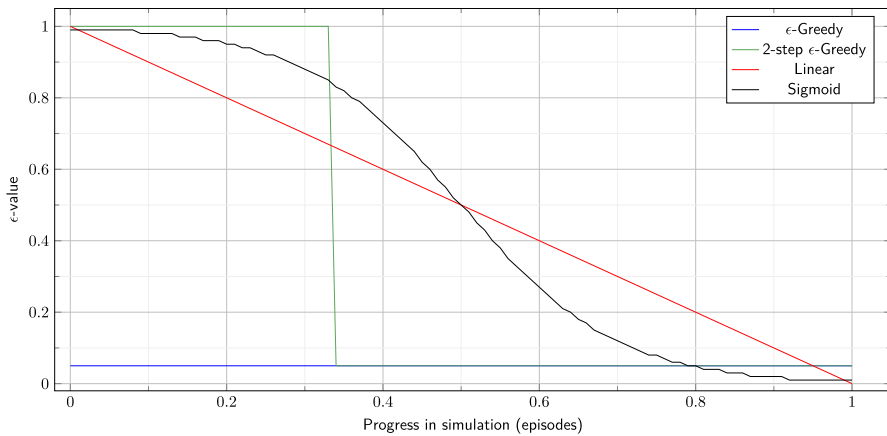


Fig. 4 ϵ functions tested in ϵ strategies

If an appropriate row is available in the Q-table, then the discounted reward value will be compared to the one stored in the Q-table. If we realize that the newly experienced path provides a greater reward than the known best one, it needs to be updated. This is also a minor modification to the original Q-learning method, where the Q-table rows contain the average reward values. For a deterministic disassembly problem, we can use a MAX aggregation function and let the agent immediately learn whenever a new best route is visited. Finally, if the observed state-action pair is not listed yet in the Q-table, then we should add it to the table.

3.3 Formulating dLOptRL algorithm as a HARL method

Since finding an optimal solution by using RL methods can be very time-consuming, in recent years many researchers made efforts to speed up the learning process by improving the action selection method (Bianchi et al. 2012). There are successful references for extracting domain knowledge by integrating special heuristics into an RL method (Cheng et al. 2021). It was shown (Bianchi et al. 2015) that the value function can be mathematically combined with a heuristic function if $[F_t(s_t, a_t) \bowtie \xi H_t(s_t, a_t)^\beta]$, where $\mathcal{F} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is an estimate of a value function, while $\mathcal{H} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the heuristic function. It can be easily seen that a sufficiently large negative ξ value can practically avoid selecting inappropriate actions. Regarding dLOptRL algorithm, the heuristic function can be defined as 0 when the component is disassembled according to the precedence graph and $-\Omega$ otherwise, where Ω is larger than the theoretical maximum objective value of the concrete disassembly problem.

4 Application examples

There are several benchmark problems analyzed in the literature. In this section, we will present two of them and summarise the performance of our reinforcement learning-based solution by comparing them to classical methods. Then we will highlight the main advantages of RL-based optimizations on further problems.

4.1 Small scale benchmark problem–personal computer disassembly

There is a small-scale problem in the literature (Tuncel et al. 2014; Lambert and Gupta 2004) regarding disassembling personal computers. There are identified 8 salvageable components of a PC. The parts themselves, their removal times, demand values, and hazardousness indicators are collected in Table 3.

A precedence graph describes the logical dependencies of the disassembly task's order in Fig. 5.

By choosing a combined objective function of $F = \frac{1}{w_1 F_1 + w_2 F_2 + w_3 F_3}$, where the components are the same as defined in Sect. 2.2, the optimal global solution can be determined by using a MIQP solver. We want to highlight that the weights allow for prioritizing the objective components to align with the user's needs. Therefore, the concrete weighting values are less important from a scientific perspective, and hence the researchers often set them equally. Although there are multiple reasons not to follow this, to let the results compare, we also applied equal weights in this use case. The published global optimum to remove the components is presented in Table 4.

First, by solving the MIQP problem, we found that the optimal objective value is $F_{opt} = (3^2 + 2^2 + 4^2 + 2^2) + 7 + 19025 = 19065$. Now we can evaluate the result of DLOPTRL by comparing it to the known optimal objective as a reference value.

As a first approach, we implemented a simple Q-learning-based algorithm controlled purely by the reward function and has only a single restriction to choosing the upcoming part for removal. Every component needs to be selected exactly once in the disassembly sequence. We realized that this approach practically results in very low efficiency, like testing a random permutation of the parts whether it meets the criterion of the precedence graph or not.

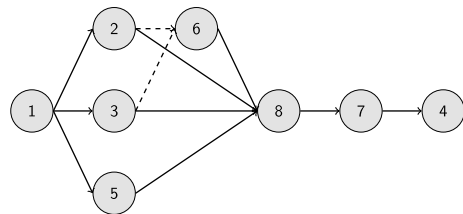
This kind of experience motivated us to integrate the constraints (that were collected at MIQP formulation) into the next action determination step by restricting the set of potential actions only to the applicable ones, which is practically the intersection of three sets:

- parts that are not removed yet,
- parts of which all AND type predecessor parts are already removed,
- parts of which at least one OR type predecessor part is removed.

Such a limitation of action space indicates a significant change in the learning speed: the agent found the optimal solution after very few steps in our small-scale use case.

Table 3 Personal computer disassembly tasks and parameters

Task no	Disassembly task	Removal time	Demand	Hazardousness
1	PC top cover	14	360	No
2	Floppy drive	10	500	No
3	Hard drive	12	620	No
4	Backplane	18	480	No
5	PCI cards	23	540	No
6	RAM modules (2)	16	750	No
7	Power supply	20	295	Yes
8	Motherboard	36	720	No

Fig. 5 Precedence graph of personal computer disassembly problem**Table 4** Optimal solution for personal computer disassembly problem

Sequence order	1	2	3	4	5	6	7	8
Component to remove	1	5	3	6	2	8	7	4
Part removal time	14	23	12	16	10	36	20	18
Assigned workstation	1	1	2	2	2	3	4	4
Workstation idle time [s]		3			2	4		2

4.2 Mid scale benchmark problem–cell phone disassembly

Another case study problem is about disassembling cell phones. There are identified 25 salvageable components of a cell phone. The parts themselves, their removal times, the demand values, and hazardousness indicators are collected in Table 5.

A precedence graph describes the logical dependencies of the disassembly order in Fig. 6. This drives to a less trivial solution than in the previous use case (Tuncel et al. 2014). This led us to highlight our new formulation that works with significantly smaller decision variable vector size as it was described in Sect. 2.3.

The degraded MIQP problem solution is presented in Table 6. In this case, the optimal objective value is: $F_{opt} = 15 + 75 + 815 = 905$. The pure Q-learning algorithm could not deliver a feasible solution without limiting the action space to the applicable actions. In contrast, the DLOPTRL algorithm provides a feasible solution from the very first episode. Of course, this fact does not mean that the early solutions are efficient enough. By performing multiple simulations, we found that the

Table 5 Cell phone disassembly tasks and parameters

Task no	Disassembly task	Removal time	Demand	Hazardousness
1	Antenna	3	4	Yes
2	Battery	2	7	Yes
3	Antenna guide	3	1	No
4	Bolt (type 1) a	10	1	No
5	Bolt (type 1) b	10	1	No
6	Bolt (type 2) 1	15	1	No
7	Bolt (type 2) 2	15	1	No
8	Bolt (type 2) 3	15	1	No
9	Bolt (type 2) 4	15	1	No
10	Clip	2	2	No
11	Rubber seal	2	1	No
12	Speaker	2	4	Yes
13	White cable	2	1	No
14	Red/blue cable	2	1	No
15	Orange cable	2	1	No
16	Metal top	2	1	No
17	Front cover	2	2	No
18	Back cover	3	2	No
19	Circuit board	18	8	Yes
20	Plastic screen	5	1	No
21	Keyboard	1	4	No
22	Liquid crystal display	15	6	No
23	Sub-keyboard	15	7	Yes
24	Internal circuit	2	1	No
25	Microphone	2	4	Yes

DLOPTRL algorithm could find a reasonably good approximation for the optimal solution. As we described in Sect. 3.1, we tested four different static ϵ -strategies for the RL agent's decision. Figure 7 shows the learning performance results by following the different approaches.

For easier interpretation, the weighted total objective values are plotted instead of the cumulative rewards. Although the continuous ϵ -Greedy approach presents the best performance with the lowest objective values in the first phase of the simulation, it reaches the worst solution at the end of the simulation. The other methods start with a high ϵ -value to discover the search space more intensively. The linearly decreasing ϵ approach gets the best overall objective value, and hence we will use that one further. Another advantage is that it does not need a custom parameter for its operation, which simplifies the DLOPTRL algorithm.

In contrast to the small-scale use case presented in Sect. 4.1, the DLOPTRL algorithm does not reach the global optimal solution of the mid-scale benchmark

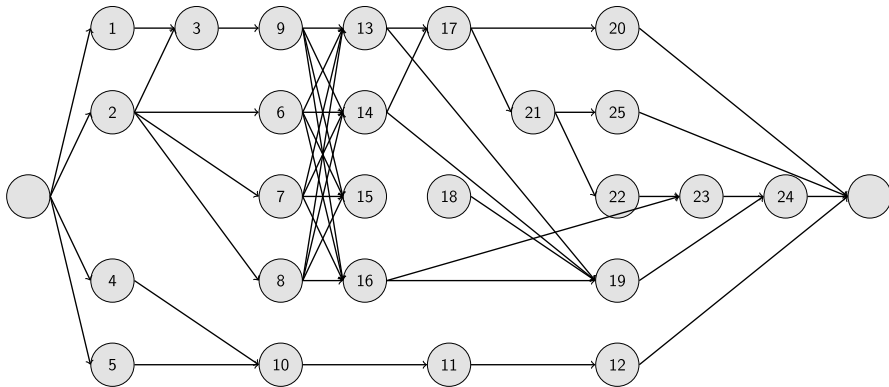


Fig. 6 Precedence graph of cell phone disassembly problem

problem. We executed 100 identically parameterized simulations to provide cross-validated results. Figure 8 shows the empirical results by presenting:

- the range of the observed objectives,
- 50-episode moving averages of median objectives,
- 50-episode moving averages of upper/lower quartiles of objectives,
- best objective of learned routes by episodes.

Although the global optimum was not found, all the simulations show a stable convergence in objective values. The median value of the objectives is 985, which is 8.8% worse than the global optimum. The best solution of DL_{OPT}RL method has an objective of 917, and it is presented in Table 7 in detail. A widely used indicator comparing two sequences is the concordance ratio. It is calculated by counting all of the item pairs that are in the same order in both sequences and dividing by the total number of item pairs.

Out of the $\binom{25}{2} = 300$ different item pairs 293 have concordant orders and 7 have discordant orders, and hence the concordance ratio is 97.67%. So the RL solution delivered a strongly similar solution to the optimal one. As the multiple steps of the best objective curve show, the learning process is continuous. There is a significant difference between the moving average values and the known best reward value, which highlights the “cost-of-learning”: if we decide to explore an unknown path instead of the known best one, it will cause some loss in the overall performance, but it leaves open the chance to find a better path than the current best one.

4.3 Comparison of RL-based solution to the mixed-integer solution and further research directions

In this section, we will summarize the major experiences of the two described solution methods for disassembly line balancing problems, and further research

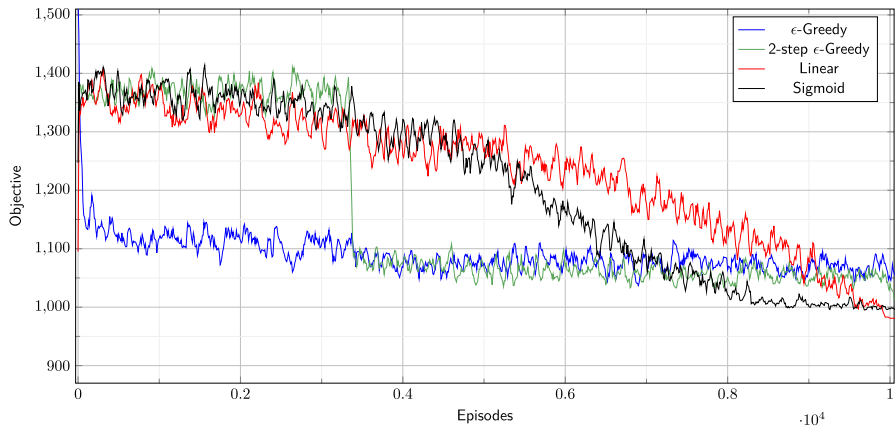


Fig. 7 Learning performance of the tested ϵ -strategies

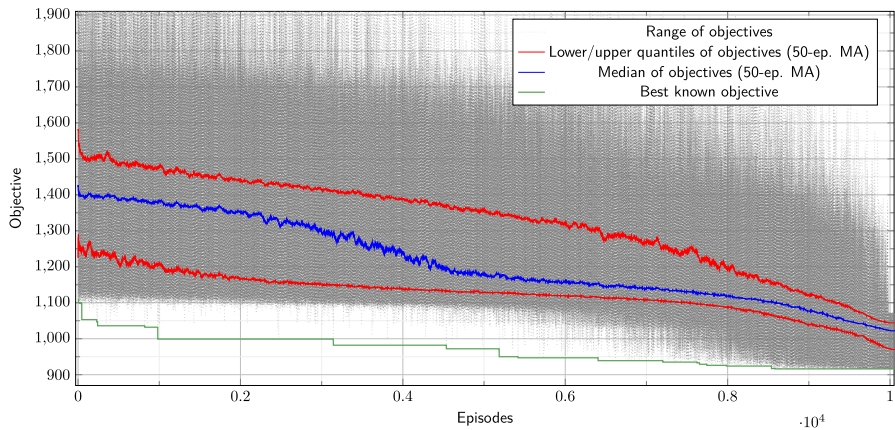


Fig. 8 Objective values (reciprocal reward values) by iterations

directions, which can have significant potential to improve the solution's robustness and adaptivity.

As the disassembly line balancing problem is NP-hard (Chand and Ravi 2023), it cannot be guaranteed, that an optimal solution will be found in polynomial calculation time. In our formulation, the number of decision variables in the MIQP problem is quadratic to the number of components. We validated the formulation on the small-size PC disassembly problem, and the MIQP solver provided the optimal solution within a second. However, for the mid-size cell phone disassembly problem the MIQP solver processed 983,557 branches in 3,600 s. The first feasible solution was found after 539,992 iterations. In contrast, the RL-based solution reached a complete Q-table in 2 s for the small size problem. Furthermore, it performed 10,000

Table 6 Optimal solution for cell phone disassembly problem

Sequence order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Component to remove	2	7	1	8	3	6	9	14	13	17	21	22	25	16	15	23	18	19	20	4	5	10	11	12	24
Part removal time	2	15	3	15	3	15	15	2	2	2	1	5	2	2	2	15	3	18	5	10	10	2	2	2	2
Assigned workstation	1	1	2	2	3	3	4	4	5	5	5	5	5	5	5	6	6	7	8	8	9	9	9	9	9
Workstation idle time [s]	1			0		0		1							2		0	0		3					0

Table 7 Best RL solution for cell phone disassembly problem

Sequence order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Component to remove	2	7	1	6	8	3	9	13	14	17	15	21	22	25	16	23	18	19	20	4	5	10	11	12	24
Part removal time	2	15	3	15	15	3	15	2	2	2	2	1	5	2	2	15	3	18	5	10	10	2	2	2	2
Assigned workstation	1	1	2	2	3	3	4	4	5	5	5	5	5	5	5	6	6	7	8	8	9	9	9	9	9
Workstation idle time [s]	1		0	0	0	0	0	1							2	0	0	0	3						0

Table 8 Summary of use case solutions

Solution type	PC problem		CP problem	
	MIQP	RL	MIQP	RL
Execution time (s)	<1	2	3,600	182
Final objective	19,065	19,065	1269 ^a	917 ^b
Gap to global optimum	0	0	331	12
Relative gap to global optimum (%)	0%	0%	40.22%	1.33%

^a Terminated because of the execution time limit

^b Median objective of 100 simulations

episodes of the mid-size problem in 382 s, and it grew a Q-table with 8,845 rows. Table 8 summarizes the key performance measures of the simulations.

The difference between the two solutions is that the MIQP solver requires a preliminary training process, while the RL-based solution learns online. As our MIQP formulation shows the component removal times, the hazardous indicators, the demands, the cycle time and the objective component weights are all necessary to start the MIQP solver. In practice, it is often easier to set up an empirical reward function by measuring idle times and component removal orders and then letting the RL agent start learning. Furthermore, it is a complex task to implement an efficient MIQP solver, and costly to buy one, while our RL solution is easy to implement. In case of having multiple identical disassembly lines or if there is a virtual twin of it, the RL learning process is easy to parallel. Our results show that the RL agent reliably converges to the optimal solution. The DLOptRL algorithm delivers feasible solutions from the beginning and finds a competitive disassembly setup within a reasonable training time limit.

We identified further research directions, which can have significant potential to improve the solution's robustness and adaptivity.

- The DLOptRL algorithm can be extended to a multi-agent approach for parallelizing the learning process by updating a central Q-table.
- A new indicator for measuring the proportion of undiscovered routes (actions) would be worth introducing, and an adaptive episode length determination process could help to approach better the global optimum.
- Disassembly components' removal times have higher uncertainty than assembly process times because the condition of a used product is more heterogeneous than a uniform new one. This implies applying stochastic removal times instead of deterministic ones.
- Using a moving time window or resetting the Q-table periodically can raise the adaptivity of the RL solution.
- The RL-based solution is less sensitive to measurement inconsistencies and one-time issues. Therefore, even if these observations are involved in the Q-value aggregations, their effects will be marginal in the long term.

- By measuring the rewards directly, the cumulative measurement errors can decrease compared to an MIQP formulation, where the errors will be accumulated.

We want to analyze the RL-based solution's behavior in the above contexts in the next phase of our ongoing research to verify them. We also recommend such analysis for other researchers in the disassembly domain.

5 Summary and conclusions

Disassembly line optimization problems become more important, leading researchers to pay more attention to developing dedicated solutions. The optimization challenges have many problem formulations, objectives, and restrictions, and a wide range of problem sizes. We presented a compact formulation for the disassembly optimization problem that requires fewer decision variables to solve larger problems with the same solver limitations.

We showed that the standard approach of reinforcement learning application when only the reward function must be declared, has a low convergence rate in the learning path. We described a Q-learning-based solution with an integrated heuristic named `DLOPTRL` algorithm that lets the reinforcement learning agent learn the solution very effectively. We demonstrated the learning capability of our algorithm in two selected use cases that proved the real-life applicability of our approach.

We believe that the presented solution shows a possible way to fine-tune reinforcement learning algorithms to increase their learning performance for disassembly problems and other fields.

Furthermore, we have shown that our algorithm formally belongs to the Heuristically Accelerated Reinforcement Learning class. It delivers a working example of translating an MIQP problem into a heuristic function.

Our algorithm has further potential for adapting to slowly changing disassembly environments or completely stochastic problems. The presented method can be used for other problem classes that need ordering complex actions into a sequence, such as the Travelling Salesman Problem (TSP), network/map discovery, or Vehicle Routing Problem (VRP), especially in stochastic cases when the state space is mixed continuous-discrete.

Acknowledgements This work has been implemented by the TKP2021-NVA-10 project with the support provided by the Ministry for Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the 2021 Thematic Excellence Programme funding scheme.

Author Contributions TK—Writing—original draft preparation, Visualization; ZS—Methodology, Validation, Reviewing; JA—Conceptualization, Methodology, Supervision, Project administration, Funding acquisition. All authors read and approved the manuscript.

Funding Open access funding provided by University of Pannonia. Innovációs és Technológiai Minisztérium (TKP2021-NVA-10).

Declarations

Conflict of interest The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Beamon BM (1999) Measuring supply chain performance. *Int J Oper Prod Manag* 19(3):275–292
- Bianchi RA, Ribeiro CH, Costa AHR (2012) Heuristically accelerated reinforcement learning: theoretical and experimental results. In: *ECAI*, pp 169–174
- Bianchi RAC, Celiberto JLA, Santos PE, Matsuura JP, Lopez De Mantaras R (2015) Transferring knowledge as heuristics in reinforcement learning: a case-based approach. *Artif Intell* 226:102–121
- Camacho-Otero J, Boks C, Pettersen IN (2018) Consumption in the circular economy: a literature review. *Sustainability (Switzerland)* 10(8):2758
- Chand M, Ravi C (2023) A state-of-the-art literature survey on artificial intelligence techniques for disassembly sequence planning. *CIRP J Manuf Sci Technol* 41:292–310
- Chen JC, Chen Y-Y, Chen T-L, Yang Y-C (2022) An adaptive genetic algorithm-based and and/or graph approach for the disassembly line balancing problem. *Eng Optim* 54(9):1583–1599
- Cheng C-A, Kolobov A, Swaminathan A (2021) Heuristic-guided reinforcement learning. *Adv Neural Inf Process Syst* 34:13550–13563
- de Koster R, Le-Duc T, Roodbergen KJ (2007) Design and control of warehouse order picking: a literature review. *Eur J Oper Res* 182(2):481–501
- Dufloy JR, Seliger G, Kara S, Umeda Y, Ometto A, Willems B (2008) Efficiency and feasibility of product disassembly: a case-based study. *CIRP Ann Manuf Technol* 57(2):583–600
- Go T, Wahab D, Rahman MA, Ramli R et al (2010) A design framework for end-of-life vehicles recovery: optimization of disassembly sequence using genetic algorithms. *Am J Environ Sci* 6(4):350
- Guo J, Zhong J, Li Y, Du B, Guo S (2018) A hybrid artificial fish swarm algorithm for disassembly sequence planning considering setup time. *Assem Autom* 39(1):140–153
- Gupta SM, McLean CR (1996) Disassembly of products. *Comput Ind Eng* 31(1–2):225–228
- Gupta S, Taleb K (1994) Scheduling disassembly. *Int J Prod Res* 32(8):1857–1866
- Hervani AA, Helms MM, Sarkis J (2005) Performance measurement for green supply chain management. *Benchmarking* 12(4):330–353
- Hsu H-P (2016) A fuzzy knowledge-based disassembly process planning system based on fuzzy attributed and timed predicate/transition net. *IEEE Trans Syst Man Cybern Syst* 47(8):1800–1813
- Jovane F, Altling L, Armillotta A, Eversheim W, Feldmann K, Seliger G, Roth N (1993) A key issue in product life cycle: disassembly. *CIRP Ann Manuf Technol* 42(2):651–658
- Kalaycilar EG, Azizoglu M, Yeralan S (2016) A disassembly line balancing problem with fixed number of workstations. *Eur J Oper Res* 249(2):592–604
- Kalmykova Y, Sadagopan M, Rosado L (2018) Circular economy—from review of theories and practices to development of implementation tools. *Resour Conserv Recycl* 135:190–201
- Kim H-J, Lee D-H, Xirouchakis P (2007) Disassembly scheduling: literature review and future research directions. *Int J Prod Res* 45(18–19):4465–4484
- Kim H-J, Lee D-H, Xirouchakis P, Kwon O (2009) A branch and bound algorithm for disassembly scheduling with assembly product structure. *J Oper Res Soc* 60(3):419–430

- Laili Y, Li Y, Fang Y, Pham DT, Zhang L (2020) Model review and algorithm comparison on multi-objective disassembly line balancing. *J Manuf Syst* 56:484–500
- Lambert AJD, Gupta SM (2004) Disassembly modeling for assembly, maintenance reuse and recycling. CRC Press, Boca Raton, FL, USA
- Lee D, Kim H, Choi G, Xirouchakis P (2004) Disassembly scheduling: integer programming models. *Proc Inst Mech Eng Part B J Eng Manuf* 218(10):1357–1372
- Li W, 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(4):248–260
- Loiseau E, Saikku L, Antikainen R, Droste N, Hansjürgens B, Pitkänen K, Leskinen P, Kuikman P, Thomsen M (2016) Green economy and related concepts: an overview. *J Clean Prod* 139:361–371
- 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
- Mete S, Serin F (2021) A reinforcement learning approach for disassembly line balancing problem. In: 2021 international conference on information technology (ICIT), IEEE, pp 424–427
- Mitrouchev P, Wang C, Lu L, Li G (2015) Selective disassembly sequence generation based on lowest level disassembly graph method. *Int J Adv Manuf Technol* 80:141–159
- Neuendorf K-P, Lee D-H, Kiritsis D, Xirouchakis P (2001) Disassembly scheduling with parts commonality using petri nets with timestamps. *Fundam Inform* 47(3–4):295–306
- 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
- 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(2):647–660
- Rickli JL, Camelio JA (2013) Multi-objective partial disassembly optimization based on sequence feasibility. *J Manuf Syst* 32(1):281–293
- Sasikumar P, Kannan G (2008) Issues in reverse supply chains, part I: end-of-life product recovery and inventory management—an overview. *Int J Sustain Eng* 1(3):154–172
- Smith S, Smith G, Chen W-H (2012) Disassembly sequence structure graphs: an optimal approach for multiple-target selective disassembly sequence planning. *Adv Eng Inform* 26(2):306–316
- Sutton RS, Barto AG (2018) Reinforcement learning: an introduction, 2nd edn. The MIT Press, Cambridge, MA, USA
- Taleb KN, Gupta SM (1997) Disassembly of multiple product structures. *Comput Ind Eng* 32(4):949–961
- Taleb KN, Gupta SM, Brennan L (1997) Disassembly of complex product structures with parts and materials commonality. *Prod Plan Control* 8(3):255–269
- Tian G, Ren Y, Feng Y, Zhou M, Zhang H, Tan J (2018) Modeling and planning for dual-objective selective disassembly using and/or graph and discrete artificial bee colony. *IEEE Trans Ind Inform* 15(4):2456–2468
- Tian Y, Zhang X, Liu Z, Jiang X, Xue J (2019) Product cooperative disassembly sequence and task planning based on genetic algorithm. *Int J Adv Manuf Technol* 105(5):2103–2120
- Tseng H-E, Lee S-C (2018) Disassembly sequence planning using interactive genetic algorithms. In: 2018 14th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD), IEEE, pp 77–84
- 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
- Tuncel E, Zeid A, Kamarthi S (2014) Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. *J Intell Manuf* 25(4):647–659
- Viharos ZJ, Jakab R (2021) Reinforcement learning for statistical process control in manufacturing. *Measurement* 182:109616
- Wang H, Rong Y, Xiang D (2014) Mechanical assembly planning using ant colony optimization. *Comput Aided Des* 47:59–71
- Wang J, Wu X, Fan X (2015) A two-stage ant colony optimization approach based on a directed graph for process planning. *Int J Adv Manuf Technol* 80(5):839–850
- Woo S, Sung Y (2020) Dynamic action space handling method for reinforcement learning models. *J Inf Process Syst* 16(5):1223–1230
- Xia K, Gao L, Wang L, Li W, Li X, Ijomah W (2016) Service-oriented disassembly sequence planning for electrical and electronic equipment waste. *Electron Commer Res Appl* 20:59–68
- Xia K, Gao L, Wang L, Li W, Chao K-M (2013) A simplified teaching-learning-based optimization algorithm for disassembly sequence planning. In: 2013 IEEE 10th international conference on e-business engineering, IEEE, pp 393–398

- Xu W, Tang Q, Liu J, Liu Z, Zhou Z, Pham DT (2020) Disassembly sequence planning using discrete bees algorithm for human-robot collaboration in remanufacturing. *Robot Comput Integr Manuf* 62:101860
- Zhang XF, Yu G, Hu ZY, Pei CH, Ma GQ (2014) Parallel disassembly sequence planning for complex products based on fuzzy-rough sets. *Int J Adv Manuf Technol* 72:231–239

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