International Journal of Advances in Production Research

Machine Learning Methods for (Dis-)Assembly Sequence Planning - A Systematic Literature Review

Detlef Gerhard[a,1*], Julian Rolf[a,2], Pascalis Trentsios[a,3], Jan Luca Siewert[a,4]

[a] *Digital Engineering Chair, Ruhr University Bochum, Germany*

{detlef.gerhard[1], julian.rolf[2], pascalis.trentsios[3], jan.siewert[4]} @ruhr-uni-bochum.de

Abstract— **This paper presents a systematic literature review on the application of reinforcement learning in the domain of assembly and disassembly sequence planning. The authors conduct a keyword search to identify scientific publica-tions in the desired field in three scientific databases. Web of Science, Scopus and IEEE-Xplore. The analysis covers two core aspects of reinforcement learning, namely the definition of the reward function and the representation of states. In total 23 publications are identified, and the content of the collected works is presented. An analysis of the selected publications is then carried out in relation to the questions posed in order to be able to make recommendations for the application of reinforcement learning methods for the generation of efficient assembly and demonstration se-quences.**

Keywords— **reinforcement learning; assembly sequence planning; disassembly sequence planning**

I. Introduction

Sequence planning for assembly tasks is a fundamental step in manufacturing and industrial automation, crucial for ensuring efficient and error-free production processes. The time of assembling takes up 20-50 percent of the total time of production, while the cost of assembling takes up about 20-30 percent of the total cost [1,2], while the optimization of the disassembly also offers time and cost savings [3]. Given the fact, that scheduling both an assembly and disassembly sequence can be modelled as a NP-hard problem, those solutions are limited in handling the increasing complexity of modern manufacturing without exceeding computational limitations. Therefore, many different approaches to find semi-optimal solutions have been considered in literature.

Modelling the assembly as a graph allows the use of graph search algorithms to find an optimal or sufficiently good solution, depending on the algorithm [4,5]. Heuristic methods can also be used to solve ASP/ DSP problems. These include genetic algorithms (GA) [6,7]. GAs represent (dis)assembly sequences as chromosomes, which are iteratively adapted by mutations and recombination. Afterwards solutions get selected with regard to an evaluation scheme. This iteration procedure is repeated until a sufficient solution is found. Alternatively, neural networks can be trained to decide for each step which component should be (dis)assembled next [8]. These can be trained using supervised learning methods, which requires the availability of a suitable data set. This requirement does not apply to reinforcement learning and is therefore particularly advantageous in the planning phase of a product when no useful data from a real (dis)assembly is available.

This literature review provides an overview of the state of the research regarding the application of reinforce-ment learning methods for the generation of assembly or disassembly sequences based on a systematic literature review. To this end, the basics of reinforcement learning are presented first. The research questions on which the literature review is based are then defined and the procedure for identifying relevant studies is described. The results of the literature review are then presented and analysed. Finally, recommendations for the definition of rewards and state descriptions in the field of reinforcement learning for the generation of (dis)assembly sequences are given and potential research.

Deep Reinforcement Learning (DRL) falls under the broader category of machine learning, a core component of artificial intelligence (AI). As outlined by Sutton and Barto, DRL involves three primary elements: an entity known as the agent, the environment in which it operates, and a specific objective or task [9]. Within this frame-work, DRL incorporates several critical aspects, including a policy, a reward function, a value function, and some-times a model representing the environment.

In DRL, the agent has the capability to perceive the state of its environment, interact with it through various actions, and receive rewards based on successful task completion. The policy, which determines how actions are chosen in response to the states perceived, is developed through training, where rewards or penalties are given according to the reward function. The value function predicts the total expected reward from a particular action in the current state. In certain DRL approaches, providing the agent with a model of the environment enables more effective action planning.

*Corresponding Author

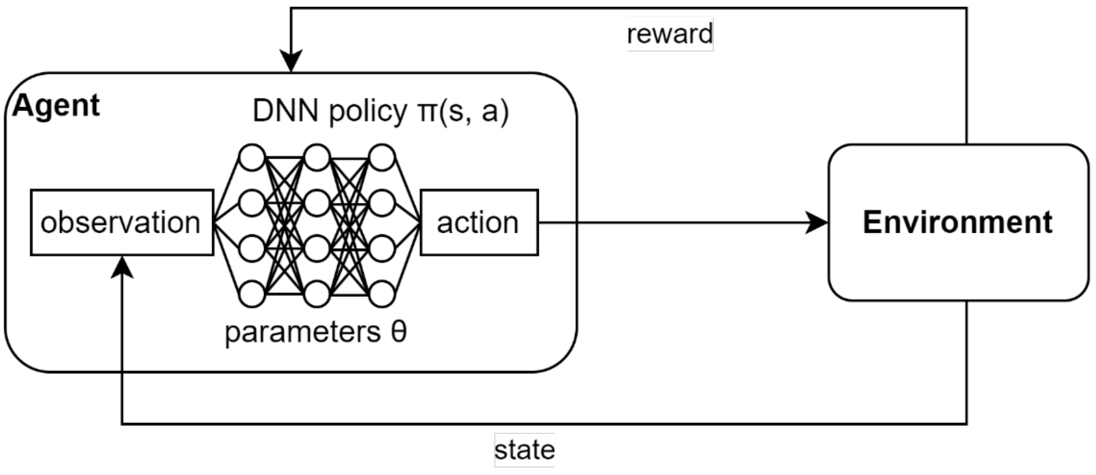


FIGURE 1. Implemented Components for the Assembly Support System

DRL is distinguished from traditional reinforcement learning by its incorporation of deep neural networks (DNNs) to formulate the agent's policy. The versatility of DNNs as universal function approximations renders them effective in various applications [10]. The structure of these networks can vary significantly, depending on both the task at hand and the architecture of the agent or system [11]. The training process in DRL involves repetitive agent-environment interactions, as depicted in Figure 1. Each interaction includes the perception of the current state, execution of an action, and receipt of a reward. The state encapsulates comprehensive information about the system, environment, and their ongoing interaction [12]. Training episodes begin at a start state and conclude upon meeting specific conditions, like successful task completion, failure, or reaching a predetermined time or step limit. These episodes and individual steps generate data used to adjust the DNN's parameters or the policy, aiming to maximize the agent's total reward. Training data should be diverse to effectively prepare for real-world application post-training. Randomizing the initial state in each episode contributes to this diversity. Training effectiveness is also influenced by various hyperparameters [13]. Policy updates depend on the application of rewards through the reward function, which directs the system's behaviour by assessing the states and interactions among the agent and its environment. These interactions are classified as either beneficial or not. The reward function sets the parameters for desired system behaviour, operating independently of the system's state or action space and not directly influencing its sensors or actuators [14].

II. MATERIALS AND METHODS

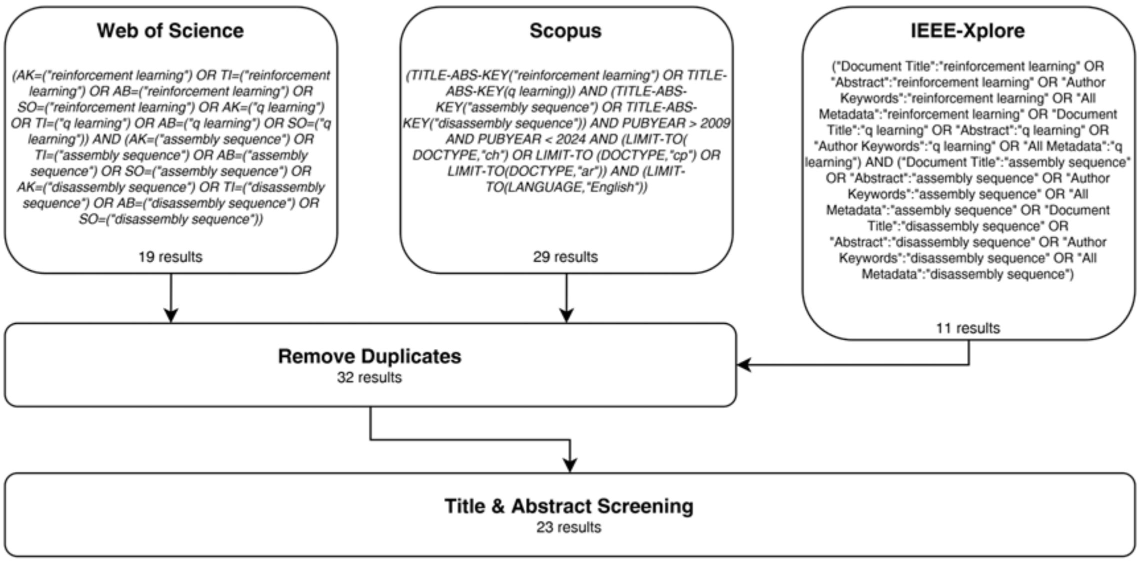
This section provides information about the research questions defined to analyse the scientific works. Additionally, the procedure for identifying relevant scientific work is presented.

A. Research Questions

The scientific analysis focuses on two research questions, aimed at illustrating key aspects for applying reinforcement learning in the context of assembly and disassembly sequence planning. The main difficulties lie particularly in the description of the state and the reward for the agent’s actions. The state must contain all relevant information that influences the agent’s decision. At the same time, high dimensional input requires more data samples to identify underlying correlations. This leads to an increased training effort in RL. Therefore, it is mandatory to choose a state description that is both complete and sufficiently extensive.

The reward leads the agent to the correct policy. Accordingly, a suitable value must be assigned to all good and bad actions to be assigned. The definition of a reward function is often associated with a trial-and-error procedure. Thus, the systematic literature review provides a starting point for defining the reward function to simplify the RL application. The research questions are as follows:
RQ 1: Which factors should be considered when defining an agent's reward function for reinforcement learning, and which conditions should be fulfilled to do so?
RQ 2: Which state representation for the reinforcement learning agent should be chosen and under what conditions?

B. Keyword Analysis



```
(AK=("reinforcement learning") OR TI=("reinforcement learning") OR AB=("reinforcement learning") OR SO=("reinforcement learning") OR AK=("q learning") OR TI=("q learning") OR AB=("q learning") OR SO=("q learning")) AND (AK=("assembly sequence") OR TI=("assembly sequence") OR AB=("assembly sequence") OR SO=("assembly sequence") OR AK=("disassembly sequence") OR TI=("disassembly sequence") OR AB=("disassembly sequence") OR SO=("disassembly sequence"))
```

```
(TITLE-ABS-KEY("reinforcement learning") OR TITLE-ABS-KEY(q learning)) AND (TITLE-ABS-KEY("assembly sequence") OR TITLE-ABS-KEY("disassembly sequence")) AND PUBYEAR > 2009 AND PUBYEAR < 2024 AND (LIMIT-TO( DOCTYPE,"ch") OR LIMIT-TO (DOCTYPE,"cp") OR LIMIT-TO(DOCTYPE,"ar")) AND (LIMIT-TO(LANGUAGE,"English"))
```

```
("Document Title":"reinforcement learning" OR "Abstract":"reinforcement learning" OR "Author Keywords":"reinforcement learning" OR "All Metadata":"reinforcement learning" OR "Document Title":"q learning" OR "Abstract":"q learning" OR "Author Keywords":"q learning" OR "All Metadata":"q learning") AND ("Document Title":"assembly sequence" OR "Abstract":"assembly sequence" OR "Author Keywords":"assembly sequence" OR "All Metadata":"assembly sequence" OR "Document Title":"disassembly sequence" OR "Abstract":"disassembly sequence" OR "Author Keywords":"disassembly sequence" OR "All Metadata":"disassembly sequence")
```

FIGURE 2. Process of the systematic literature search

The workflow for identifying relevant scientific papers for this structural review analysis involved a systematic keyword analysis across three reputable databases: Scopus, IEEE-Xplore, and Web of Science (WoS). The first step in this process is to define the research scope, and to establish the parameters that would guide the subsequent literature search. The search terms chosen are “reinforcement learning” or “q learning” to determine the methodology, combined with “assembly sequence” or “disassembly sequence” to define the field of application. Publications released before 2010 were not considered. Scientific papers are identified by the existence of the keyword combinations in the authors keywords, abstracts, or titles, ensuring a comprehensive representation of the targeted domain. The search in the IEEE-Xplore database also includes the metadata field, while WoS search also includes the "plus keywords".

After performing the search, publications are filtered for duplicates using their DOI. The remaining works are filtered by screening the title and abstract, checking their potential for answering the defined research questions. The process of the systematic literature search is visualized in figure 2.

III. Results

This section presents the scientific papers identified. These are categorized according to their focus on assembly sequence planning or disassembly sequence planning.

In the total, 23 scientific papers were identified that match the described criteria They span a publication period from 2013 to 2023, with 22 papers released in the last 4 years. This demonstrates the actuality of the topic as described in Figure 3.

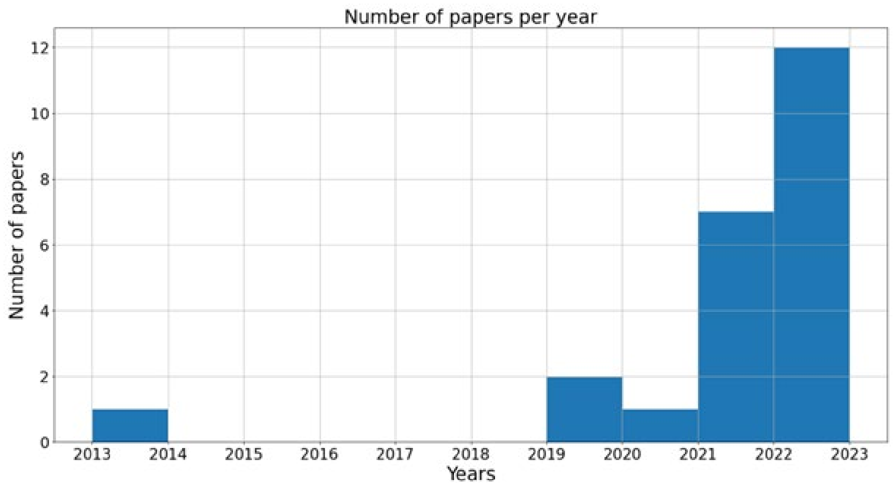


FIGURE 3. Annual Releases

An analysis of the collaboration network in Figure 4 revealed three dominant author clusters with multiple publications to the domain of applying reinforcement learning for assembly or disassembly sequence planning using bibliometrix [15].

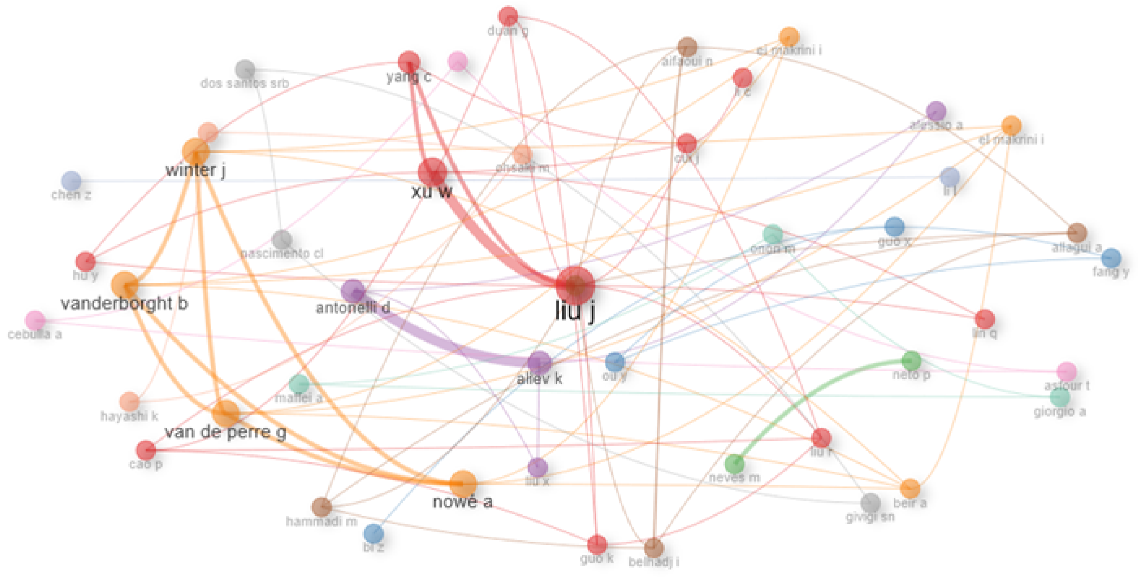


FIGURE 4. Collaboration Network

An overview of the selected works is given in Table 1. For each work, the selection of the algorithm used, the chosen approach, the properties to be optimized, the state description and the number of components in the assembly being studied are given.

Table1 1
Overview of the selected works

Publication	**Algorithm**	**Approach**	**Objective**	**State Desc.**	**Assembly Parts**
Neves [16]	Q-Learning	Assembly	Duration	Binary List + Selected Tool	9
Neves [17]	Tabular Q-Learning, A2C, Deep Q-Learning, Rainbow	Assembly	Duration, Assembler Feedback	Binary List + Selected Tool	9
Giorgio [18]	Online RL	Assembly	Duration, Assembler Feedback	-	21

Hayashi [19]	RL, GA	Assembly by Disassembly	Number of temporary support elements	Connectivity Matrix	-
Kitz [20]	Q-Learning	Assembly by Disassembly	Collision free Space	2 ½ D Collision Matrix	7, 16, 56
Zhao [21]	Deep Q-Learning	Assembly	Correctly Assembled	Part & Assembly Image	7
Antonelli [22]	Q-Learning	Assembly Human + Robot	Number Steps, Reaching Final States	Discrete Number	10
Antonelli [23]	RL	Assembly Human + Robot	Reaching Final States	Discrete Number	10
Yin [24]	RL	Robotic Assembly	Assembly Time, Energy Consumption	Connectivity Matrix	8
Alessio [25]	Multi Agent RL	Assembly Human + Robot	Reaching Final States	Grid-Layout	10
Dos Santos [26]	Learning Automata	Robotic Assembly	Robotic Movement Adaptions	Grid-Layout	3 Floor Tower
Watanabe [27]	Q-Learning	Robotic Assembly, Assembly by Disassembly	Duration	Binary List	-
Winter [28]	Hierarchical RL	Assembly, Human + Robot	Number Steps	Binary List	Cranfield Benchmark
Winter [29]	Q-Learning	Robotic Assembly	Duration	Binary List	Cranfield Benchmark
Cebulla [30]	Monte Carlo Tree Search, Q-Learning	Assembly by Disassembly	Accessibility, Directional Changes	2 ½ D Collision Matrix	38, 58
Guo [31]	Deep RL	Assembly	Directional Changes	Connector Matrix	20, 30, 40, 50
Bi [32]	Q-Learning	Disassembly	Component wise Reward	AND/OR Graph	6, 7, 22
Chen [33]	Q-Learning	Disassembly	Component wise Reward, Duration	Binary List	11
Yang [34]	Deep RL, Bee Algorithm, GA	Robotic Disassembly	Tool switch, Directional Changes	Binary List	21
Zhao [35]	Deep Q-Learning	Disassembly	Component wise Reward, Duration	MSDHGM	26

Cui [36]	Deep Q-Learning	Robotic Disassembly	Tool switch, Directional Changes	Binary List	21
Allagui [37]	Q-Learning	Disassembly	Duration, Directional Changes	Collision Matrix	5
Liu [38]	Deep Q-Learning	Robotic Disassembly	Duration	Binary List	17, 23

A. Assembly Sequence Planning

Neves et al. [16,17] use the duration of the assembly to determine the reward. To determine the duration, the authors normalize the required time t of an assembly step between the minimum assembly time T_{min} and the maximum assembly time T_{max}. These two bounds are defined by empirical values from previous episodes. To reduce outliers in the value of the maximum assembly time, only values of the assembly time are considered if they lie within 2 standard deviations of the durations of the previous 100 episodes.

$$R_t = \frac{(T_{max} - t)}{(T_{max} - T_{min})}$$

The system also checks whether the tool required for the assembly step needs to be changed. The tool switch time is included in the reward function as well. The object used in a study case is an airplane toy, which consists of 9 structural parts and 2 types of fasteners. The assembly process is divided into 8 different tasks, resulting in a total of 40320 possible sequences, and 3360 feasible ones. The authors conduct multiple different training scenarios, altering the measurement of the task time (averaging over 10 repeated times) or restricting impossible actions. By averaging the task time and without the restriction of impossible actions, the authors state the optimal training parameters with 6500 max. training episodes.

In [17], it is also argued that, in addition to considering the assembly time, feedback from the assembler should also be considered when awarding the reward. Properties to be considered can be the difficulty of the assembly step or ergonomics. The status description of the assembly is described by a list with N_{a+1} elements. The first N_a entries refer to the possible tasks. If a task is fulfilled, the corresponding value in the list is set to 1. Otherwise, it has the value 0. The last additional position describes the selected tool, which is defined by a discrete number, whereby the value 0 stands for no selected tool. The selection of the agent's action is also described by a discrete value.

One focus of the work [17] is the comparison of different RL algorithms. These are: Tabular Q-learning, A2C, DQN and Rainbow. During training, the agent had to find an optimal assembly sequence using the same assembly as in [16]. The training results state, that the DQN algorithm presented the worst performance, reaching suboptimal assembly sequence time durations and experiencing a 4 to 5 times higher number of unwanted assembly sequences when compared to the other 3 algorithms. The algorithms tabular Q-Learning, A2C, and Rainbow had similar performances and were able to achieve near optimal assembly times in both deterministic and stochastic scenarios, after approximately 10.000 episodes.

The authors assume, that the Q-learning algorithm is known to suffer from the “curse of dimensionality”, so that by increasing the assembly complexity, an even worse result is to be expected. Giorgio et al. [18] argue that online RL helps to shorten the feedback path in assembly planning. Traditionally, the assembler can only give feedback on the design or arrangement of the components and assemblies after assembly, whereas with RL additional feedback can be given for each individual assembly step. Although this "fast feedback" only serves to optimize the assembly sequence and not to adapt the design, it can still improve the process in sufficient quantities. The authors consider training in a real environment instead of a virtual one. In the first publication [18], data is collected for training the RL approach. The assembly consists of 21 components, including the connecting elements. The exact number of possible assemblies is not specified. Giorgio et al. argue to divide the reward function into two categories: the satisfaction of the assembler and the numerical characteristics of the assembly, such as the duration. The definition of observations, actions and states is not specified.

Hayashi et al. [19] consider the use case of the assembly of truss systems. The work focuses on linking the RL approach for finding effective assembly sequences and the structural analysis of the assembly. The authors use the assembly by disassembly approach. Starting from the complete state of the assembly, one element of the assembly is removed per step until only one element remains. The agent is responsible for selecting the element to be removed in each step. At the end, the disassembly sequence is reversed and considered as the assembly sequence.

The aim of the authors is to minimize the number of support elements required to ensure the structural integrity of the assembly. The reward is determined accordingly from the number of temporary support elements multiplied by -1. The state of the assembly is represented by a connectivity matrix C and an input matrix $\hat{v}$, which represents the inputs for each node with binary flags to distinguish permanent pin-supports and locally unstable nodes. The agent's action consists of removing a component. For the training, validation and testing of the RL model, truss structures of different sizes are selected. However, the exact number of components is not specified. The training took 2.5h to complete the 5000-episode long training on multiple different spatial trusses. Afterwards, the trained model is validated on three different trusses, to show its general applicability.

Kitz et al. [20] also use the assembly by disassembly approach. Collision checks are carried out to determine the dismantlability of an element. In addition, it is ensured that the stability of the assembly is guaranteed after the disassembly of a component. The stereographic projection of the respective collision-free vectors of each component is stored in a 2 ½ D collision map and describes how far a component can be moved along a vector until a collision occurs. The collision-free space of an element is used to determine the dismantling costs. The greater the space, the lower the costs and the greater the reward for the agent. If the agent chooses an invalid action, it is penalized with a fixed negative reward. The state is described using a vector with binary entries for each element of the assembly. To validate the concept, the authors used three assemblies with 7, 16 and 56 parts. The total training episodes are 1322, 2000 and 3741 and the agent is reliably able to determine cost-minimal sequences.

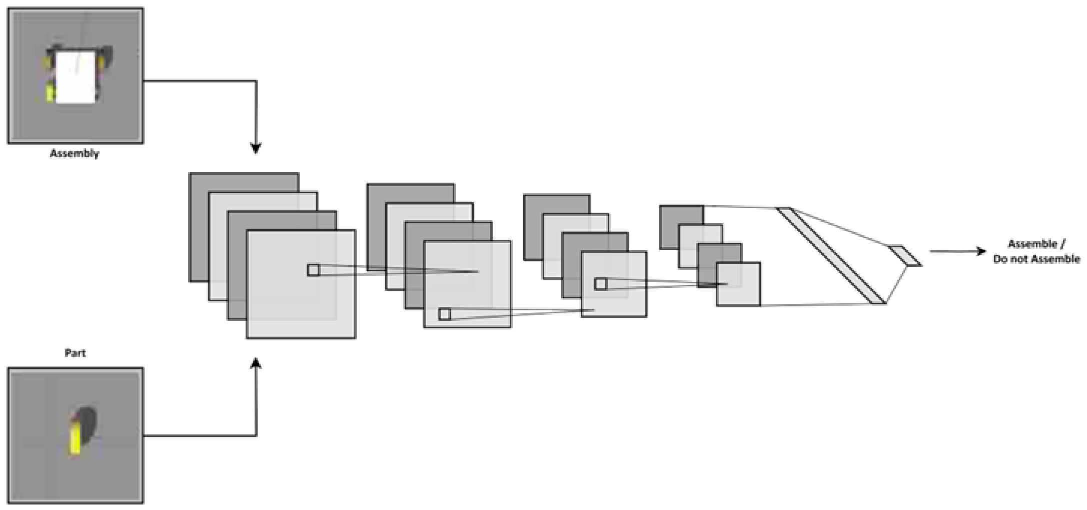


FIGURE 5. Structure of the neural network used by Zhao et al. [21]

Zhao et al. [21] consider the formulation and evaluation of a generally applicable concept for generating assembly sequences of components using RL. In each step, the agent decides for a selected element whether it should be assembled or not. Accordingly, the agent's action is a Boolean value. The description of the status consists of two parts. Firstly, the status of the assembly, represented by a camera image, and secondly, the component whose assembly must be decided, also represented by an image. Both images are feed into a convolutional neural network (CNN), as described in figure 5. The agent receives a positive reward of 1 if all elements of the assembly have been correctly selected from the set of available elements, otherwise, a negative reward of -1 is awarded. To shorten the training time and improve the performance of the CNN used, the authors apply Curriculum Learning and Transfer Learning. The concept is first validated in a virtual environment by assembling an assembly with seven elements. Using curriculum learning, the model can determine a correct assembly sequence in 85% of cases with random initialization of the environment. With parameter transfer, the training time is about twice as fast. To test the general applicability, the authors create further use cases. In each case, around 100 further training episodes are required to generate suitable assembly sequences.

Antonelli et al. [22,23] consider the cooperation between humans and robots. The authors focus on the agile adaptation of the sequence by the human actor in the system. Accordingly, the robot must react to this adaptation by planning a new assembly sequence. First, the assembly steps are grouped according to the actors (executable only by the human/robot, executable both by the human/robot).

In [22], the awarding of rewards is based on the achievement of states. In [23], on the other hand, a reward of -1 is awarded for each action performed. Only the achievement of final configurations leads to a positive reward of 2. A state is described by a discrete number. The case study includes 26 possible Operations to pick from, with 15 different feasible states of the assembly. Assembly is made from a base part, on which three flanges are mounted and joined by screwed bolts. An optimal assembly sequence was found after approx. 350 steps, conducting of 3 assembly operations.

The generation of an optimal assembly sequence for a scenario in which the cooperation of several two-armed assembly robots is necessary is considered by Yin et al. [24]. In this scenario, truss elements must be assembled by robots. These consist of bi-directional tubes and connecting elements at the ends. Each robot makes one of three possible decisions per step:

1. supporting a connecting element
2. the assembly of a pipe
3. adjusting the position

A state is described by four components. A truss matrix H_s. The number of rows and columns of the matrix corresponds to the number of connecting elements. An element in row i and column j describes the state of a pipe between the corresponding connecting elements. The state is described by a Boolean value. The actions currently performed by the robots as a nested list A_s. The current position of each robot is a list of vectors O_s. A list of Boolean values F_s describes whether a robot is currently carrying a pipe. The reward is based on the required assembly time, the energy consumption of the robots and the stability of the setup. Two scenarios, each with two robots, were considered to validate the concept. In the experiment, the sequence of two mobile dual arm robots assembling a triangular pyramid truss structure is optimized, with eight rods. The length of the training comprises around 3000 episodes. However, the reward between the episodes is still very variable towards the end and seems to converge only slightly.

As in [22,23], Alessio et al. [25] also consider the cooperation between a human agent and a robot as a multi-agent reinforcement learning scenario. However, the human agent is designed in such a way that errors occur in its behaviour. The aim of the work is to recognize human errors and to determine suitable reactions to these errors. The virtual environment is designed as a grid layout, with each cell representing possible states of the agents, based on the Markov Decision Process, which can be described using a graph (figure 6). The evaluation of the human agent and the robot is slightly different but follows the same principle. Positive rewards are awarded for achieving defined final states. Non-permissible actions or idling are punished with negative rewards. Any other permissible action is penalized with a reward of -1 to shorten the duration of the assembly. Since the robot agent's behaviour is to be improved as part of the work, 10% of the human agent's reward is deducted from the robot's reward. In this way, possible human misbehaviour is mapped, and the robot is forced to adapt its behaviour accordingly. Four properties are used to describe the state:

1. the structure of the grid
2. the previous path of the agent through the grid
3. the previous path of the other agent through the grid
4. information about final state

To reach one of the defined terminal conditions in training, the agent required 1363 episodes.

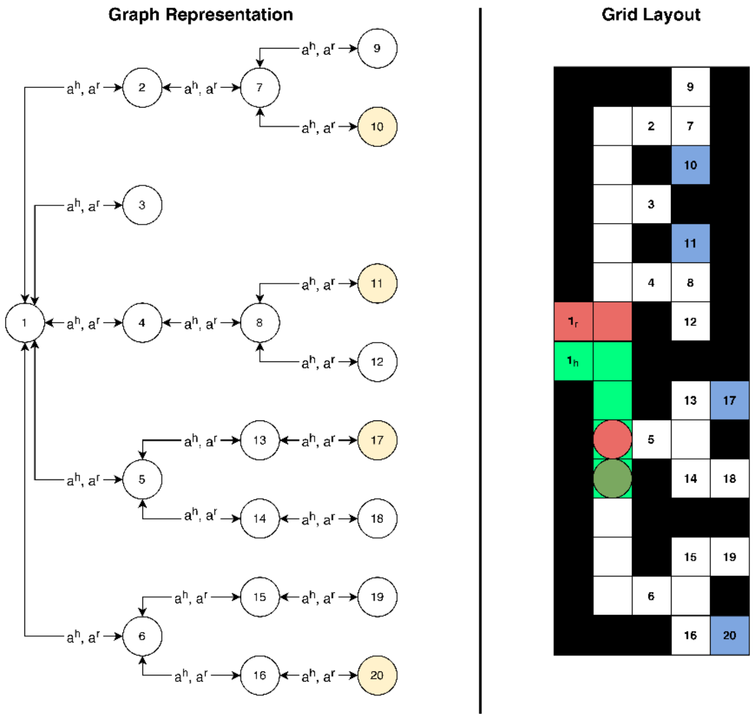


FIGURE 6. Grid Layout based on the graph representation used by Alessio et al. [25]]

Dos Santos et al. [26] consider task planning for mobile flying robots regarding the transportation and assembly of truss components. Since the authors consider not only the assembly planning but also the control of the robots, they divide their approach into two phases. First, the control of the robot is trained, followed by the efficient planning of the task steps. This includes the sequence of the individual control commands, the correct selection of components regarding the structural properties of the assembly, as well as the sequence of the assembly steps. For the analysis of the work, the first part of the approach is primarily considered. The sum of the duration for adjustments to the height, x-z position and orientation of the robot is used as the cost function. The more adjustments are necessary, the lower the reward for the agent. A lower and upper limit for the reward avoids outliers. The environment is represented by a uniform grid with a side length of 7 cm per element. As an action, the agent selects an entry from an action matrix. The entries in this matrix consist of position and rotation entries for each component. A construction algorithm checks the structural integrity of the construction to identify incorrect sequences. At the same time, a path planning algorithm based on the A* algorithm evaluates possible flight paths, as these adapt dynamically as the assembly is built. After training, the learned behaviour is successfully transferred to a real environment by building a 3-story tower, requiring approximately 900 episodes during training.

Watanabe et al. [27] consider the case of efficient assembly sequence generation for a two-armed robot. The correct assembly of rectangular building blocks serves as the use case. The authors assume, that an assembly step is always reversible. Therefore, they follow the assembly by disassembly approach. The efficiency of an assembly is determined by the total assembly time. Here, the duration of an assembly step is described by the duration of movement of the robot's hands. The description of the state also consists of a binary list, whereby the length of the list corresponds to the number of components and the value in the list describes the disassembly of a respective component. The action is described by a tuple with two integer values. Each value represents the disassembly of a component with the corresponding number. The 0 stands for the disassembly of no component. The reward is split into three parts:

1. the agent receives a reward if the disassembly is complete and based on the duration of the entire disassembly.
2. the agent receives a reward if the disassembly of a component was completed.
3. the agent receives a negative reward if the disassembly did not take place.

A collision check is carried out to check whether a disassembly is faulty. After training an assembly arrangement, transfer learning is used to train the model on further assembly arrangements.

In their work, Winter et al. [28] focus on interactive reinforcement learning (IRL) between humans and robots. The authors present an approach to train a RL Agent in a virtual environment in cooperation with an external human actor in a real environment (figure 7). The authors aim to accelerate the training performance. Communication from the human to the RL agent is based on spoken language. In the virtual environment, the robot's actions are evaluated in compliance with selected limitations. A Greedy Selector chooses the most suitable action, based on the evaluations received. The robot's real counterpart executes the action in the real environment, so the human can observe its behaviour and can verbally define new limitations for the robot to influence the selection of actions. The aim is to learn not only the sequence of components, but also the robot's behaviour during assembly. A binary list is used to represent the status of the components. The number of actions is 5 and these are defined by a discrete value. The length of the assembly should be minimized. Accordingly, a negative reward is awarded for each action. If a component is successfully assembled, a positive reward is awarded as well. The authors use the Cranfield benchmark problem [39], which consists of the assembly of a pendulum. After approximately 50 episodes of simulated training, the number of constrain violations is approximately 0. The execution of the real user study shows that the approach can respond to user feedback by adapting behaviour within a few episodes.

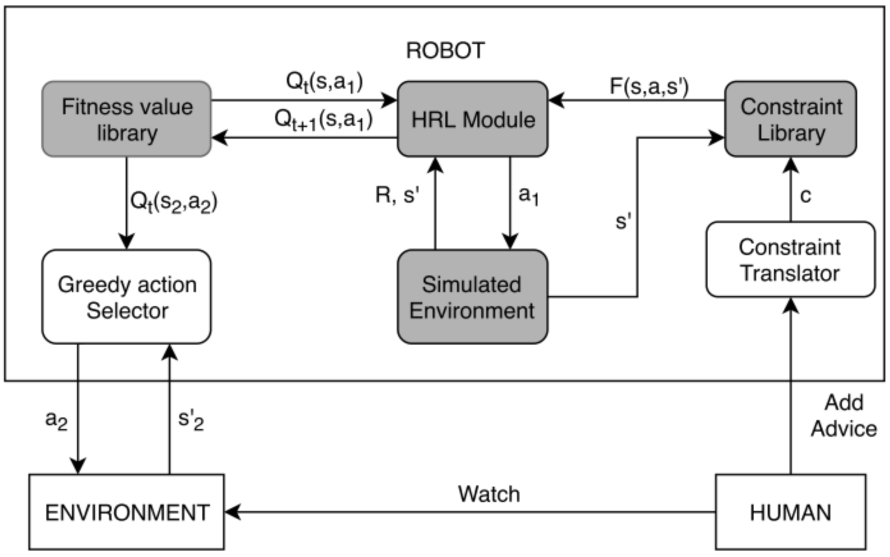


FIGURE 7. Environment setup used by Winter et al. [28]

In [29], Winter et al. focus on the optimization of a given assembly sequence. A virtual environment is used to train physically possible assembly sequences. Subsequently, the trained behaviour is transferred to a real environment to identify the fastest of the identified assembly sequences. The virtual environment is described as a digital counterpart. The authors compare different 3D engines in terms of their collision accuracy, performance, and flexibility. In the first phase, a negative reward is awarded for an error in the assembly sequence. Plausible assembly sequences have a corresponding reward of 0. The states of the components are saved in a binary list. For phase 2, the description of the status from phase 1 is adopted and expanded according to the last skill performed. The reward is assigned based on the execution time multiplied with -1. Carrying out the case study, the authors state that training the real robot and the digital twin in parallel speeds up the training. In fact, the parallel approach converges about 7 times faster and finds an optimum after about 100 episodes. The authors use the Cranfield benchmark problem [39] as well.

Cebulla et al. [30] aim to use the Monte Carlo tree search algorithm (MCTS) and Q-functions to solve the ASP problem, by leveraging the assembly by disassembly approach. The concept makes use of pre-training to create a model that is easily adaptable to specific use cases. The model is trained to minimize the number of directional changes needed to assemble the product and to maximize the accessibility of each component during the assembly process. A graph neural network is used as the model architecture. To model the collision-free space in the six axis directions for each component, a 2 ½ D matrix is used. The matrix is also used to check the stability of the components regarding gravity, by checking the collision-free space on the negative y-axis, and to determine the accessibility of the components. In addition to accessibility, the number of directional changes during assembly should also be reduced. To minimize the directional changes, the disassembly direction vector of each component is selected so that it is parallel to the direction vector from the previous step, if possible. The reward is the sum of the disassembly direction vectors of the selected component that are parallel to the direction vectors of the previous steps.

The approach is validated with two assemblies, with 38 and 58 components. The authors can show that the use of pre-trained Q-functions in combination with MCTS provides better results than the use of an unmodified MCTS, as well as the use of a Q-function specifically trained to the use case in combination with MCTS.

Guo et al. [31] use a connector-linked model, which is formulated as a graph, to model connector elements to be assembled. For training, the graph is formulated as a matrix for each connector. Each row contains the ID of the connector, the ID of the connector, which must be assembled beforehand, the connector type, the currently selected tool, and the current direction of the assembly. As the last two properties change during assembly, a new row with the corresponding values is added to the matrix for each plausible combinatorial possibility. Accordingly, the first three entries of each row are identical. Additionally, a binary value for each connector describes if the corresponding part can be selected. The List of matrices and binary values represent the state of the assembly.

The objective function aims to minimize the changes of the connector type, the assembly tool and assembly direction. The Reward is a weighted sum of these factors. The authors used 8 different connector and tool types in the study case. The total number of connectors is 20, 30, 40 and 50. The authors trained for 30.000 steps, which took 20-40h, depending on the number of connectors. After training, the model finds near optimal solutions in 0.6s in comparison to heuristic algorithms, which require 2-8 minutes.

B. Disassembly Sequence Planning

In [32], Bi et al. use Q-learning to search for a disassembly sequence of an assembly. To describe the assembly, the authors use an AND/OR graph that represents the relationships between the individual components in the assembly. A separate reward is defined for each component, which is awarded when the corresponding component is disassembled. To describe the status, the AND/OR graph is represented by a matrix M. Each row in M is a single state and has the following properties:

1. A start and a final state are defined to describe the part ranking.
2. An identifier of a disassembled component at the corresponding position.

The authors compare three use cases. A washing machine with 6 components and 15 states. A rotor assembly with 7 components and 17 states. And a hammer drill with 22 components and 63 states. The authors also compare Q-learning and GA in the use cases, whereby the RL approach not only converges significantly faster but also offers more room for improvement, according to the authors. The Q-learning approach requires about 8000 steps to find an optimal solution, for use case 3.

Chen et al. [33] also use Q-learning to find a disassembly sequence of selected smartphone components. The authors form the reward from the disassembly duration $R(t)$ and the disassembly of a component $R(m)$, whereby a corresponding reward is defined for each component. In addition to the actual execution time, the time required to change a tool is also used to determine the disassembly time. The resulting value t_i is finally offset against the value factor Z, which in the application case is 100.

$$R = \frac{Z}{t_i} + R(m)$$

If the agent performs an invalid disassembly, it is penalized with a negative reward. A list with n elements for all n components is used to describe the status. The same applies to the description of the action. In this way, the authors create a state-action-reward matrix. In the study case, the disassembly of a smartphone with 11 components and 5 tools is considered. The number of iterations performed is 50 until an optimum disassembly sequence was found. Yang et al. [34] consider the robot DSP problem, i.e. finding the optimal disassembly sequence using a robot. The reward is awarded according to the following equation:

$$r_i = T_{mx} - T_{tool\ switch} - T_{direction\ switch} - T_{moving}$$

T_{mx} represents the maximum duration for each disassembly step, T_{moving} is the time the robot needs to move between two disassembly points. $T_{tool\ switch}$ describes the duration of the tool switch and $T_{direction\ switch}$ is the duration of the robot to adjust its direction of movement. The state description consists of two lists D^k and C^k, where k corresponds to the number of components to be disassembled. D^k is a binary list for representing the components that have already been dismantled. The list C^k is one-hot encoded and describes the component currently being disassembled by the agent/robot. A discrete number from 1 to k is used as the action for the disassembly of the corresponding component. The lists are concatenated in a predefined order to describe the status. The case study is based on a double coupling shaft with 21 components. The authors compare the performance of the DRL approach with a bee algorithm and a genetic algorithm. For all three approaches, 300 iterations were performed in training. The DRL approach finds the fastest disassembly sequence, while the genetic algorithm determines the slowest of all three sequences.

Zhao et al. [35] argue that the commonly used data structures, such as an AND/OR graph or a disassembly tree, are not capable of mapping the constraints of the components to each other. The authors therefore present a new data structure, based on a hybrid graph model (HGM) data structure. The authors extend the data structure with three matrices to achieve a higher degree of detail regarding dynamic changes to the disassembly of individual instances:

1. The precedence matrix describes the precedence relationships among the components. A distinction is made among AND & OR relationships. The AND relationship describes that a part can only be dismantled once all previous dismantling parts have been removed. The OR relationship means that a part can be removed after one of its previous disassembly parts has been removed.
2. The contact matrix maps the contact relationships among the components. It includes both direct contact constraints and indirect contacts among parts.
3. The level matrix comprises the disassembly level of each component, which is determined by the leading edges of the components. Components that can be disassembled simultaneously are located on the same level.

If components are missing, an algorithm adjusts the data structure accordingly. A state is described using the three matrices and the previous action. An action is a discrete value that represents the disassembly of a component that is still present. The authors refer to this as a multi-level selective disassembly hybrid graph model (MSDHGM). The reward is determined by the required disassembly time and the profit of the disassembled component. In addition, a negative reward is awarded if the disassembly direction or the disassembly tool is changed. If components are disassembled that minimize the total number of disassembly steps due to their dependencies, this has a positive influence on the reward. To validate the approach, the authors use an assembly with 26 components. The duration of the training is not specified. The authors compare the performance of a DQN with that of a non-dominated sorting genetic algorithm and an artificial bee colony algorithm. The evaluation of the generated sequences shows that the DQN determines solutions that are closer to the optimal ones.

Cui et al. [36] also argue that common data structures for mapping disassembly sequences and similar dependencies of components are not sufficient to adequately cover the dynamic problems for EOL products. The authors suggest that the wear, deformation, corrosion, and possible fractures of the components should be assessed by experts before disassembly. The assessments are weighted and used to determine the disassembly time. The assessments are also used to check whether possible disassembly sequences can now be hindered. The description of the status is made up of the information about the disassembly status (binary list) of each element and the component currently to be disassembled. To calculate the reward, the basic disassembly duration of a component, the duration of the tool change, the duration of the agent's movements and the extra duration determined from the evaluations are added together. The resulting total is subtracted from a specified maximum value. To verify the effectiveness of proposed method, a double coupling shaft with 21 parts is used in a case study. Components 5 and 20 may have faulty properties which have been evaluated by an expert. In addition, the authors compared the DQN approach with other algorithms, such as an improved GA or a duelling DQN. The normal DQN finds the fastest disassembly sequence with the shortest training duration of 814 seconds. The time to calculate the sequence is less than one hundredth of a second after the training is complete.

Allagui et al. [37] present an approach for generating disassembly sequences using RL as well. The concept is based on two main steps. Firstly, the processing of a collision matrix. This is based on CAD data of the components and describes the possible collision of each component in six axis directions. In contrast to Guo et al. [31], the values of the collision matrix are binary coded and therefore do not represent the distance to other components. The second step is the execution of the RL algorithm.

To determine the fitness function, the authors consider the following properties:

1. the normalized volume of the selected component (smaller is better)
2. the normalized duration of a disassembly step (smaller is better)
3. the change in the disassembly direction (1: no change, 0.5: 90°, 0: 180°)
4. the change of the tool, binary encoded
5. whether the selected component is a wear part, also binary encoded

The factors are expressed as a weighted sum. To determine the reward, the value is multiplied by a factor of 100 if a component has been disassembled. Otherwise, the reward is -1, if it’s not physically possible to remove the selected part. As an experiment, the authors choose an assembly with five components. There are a total of 22 possible disassembly sequences.

In [38], Liu et al. aim to disassemble an EOL product with a robotic arm, under the assumption that some components of the product are missing. To determine the missing parts, several depth cameras are used and evaluated with a deep learning model. Afterwards, a deep Q-learning model is used to select the optimum action for disassembly, based on the condition of the product. In addition, a digital twin of the disassembly robot is connected to the system and linked to the real scene. The status of the assembly is represented as a vector consisting of three elements:

1. disassembled components (binary encoded)
2. the currently disassembled component (one hot encoded)
3. missing components (binary coded)

The reward is calculated using a constant value minus the required time to assemble the selected product. The authors used two assemblies with 17 and 23 components to carry out the experiment. The training includes approximately 11.000 and 14.000 episodes. Afterwards, a near optimal solution can be found in half a second for the first assembly and 1.5 seconds for the second assembly.

IV. Discussion

In this section, the presented works are analysed regarding the defined research questions to give recommendations on how to apply reinforcement learning to the domain of (dis-)assembly sequence planning.

A. Reward Definition

The most frequently considered factor for the design of the reward function is the required duration of executing the proposed (dis)assembly sequence. This is usually done directly, measuring the required time [16,17,24,27,33–36], or indirectly taking into account the number of steps [22,25,28]. In the latter case, a negative reward is awarded for each planned step, while the determination of the time usually considers multiple factors, such as the time required to execute a step, the time required for any tool changes or, if available, the additional time required to align the robot between the individual steps. In addition, some authors recommend offsetting the accumulated time against a maximum possible time. This can be done either by normalization [16,17] or a simple subtraction [34].

Another property for awarding both negative and positive rewards is the state of the assembly. The achievement of (final) valid states is rewarded with a positive reward [21–23,25,29], while the achievement of invalid states is penalized with a negative reward [25,29,33]. The determination of the validity of a state is derived from previously defined conditions for the assembly. For problems in disassembly sequence planning, some authors award the disassembly of certain components [32,33,35]. The amount of the reward depends on the selection of the component.

More specific rewards, which relate to the respective configurations of the assembly, are also presented. For example, the number of required support structures is minimized [19], the collision-free space around the selected component is maximized [20] or the stability of the assembly is maximized [24]. Robotic use cases consider additional application-specific rewards. On the one hand, the power consumption of the robot is minimized [24]. In a second work, the number of adjustments in the motion sequence of an autonomous flying robot is minimized [26]. For use cases in the field of human-centric approaches, two studies suggest incorporating feedback from the technician into the agent's reward [17,18]. This enables the human to give feedback according to the ergonomics of the individual construction steps to optimize them accordingly.

Based on this analysis, we recommend considering three different factors when defining the reward function:

1. The duration required for executing the (dis)assembly, either determined via the accumulated time or via the number of required steps. The latter is independent of the number of elements in the assembly and can therefore be used for assemblies of any size without additional adjustments. It is advisable to consider as many factors as possible, namely the time required executing a step, the time for any tool changes and, if applicable, the additional time required for the alignment of the robot between the individual steps. If these times are known, it can be recommended to choose this approach, due to stating the execution time is more precise. To increase the robustness during training, the time required should be normalized with a maximum duration.
2. Awarding achieving desired assembly states. On the one hand, desired configuration can be rewarded, or invalid states can be penalized. In this way, the conditions imposed on the assembly can also be mapped in the reward. In the context of disassembly, this also includes the awarding of a specific reward for the disassembly of certain components. Alternatively, desired properties

of the assembly, such as free space, stability, or human feedback, can be used. These can be determined, for example, by a simulation environment or in the field, while the determination of the reward for desired conditions must be defined in advance.

3. A reward adapted to the robot, such as the required energy or number of actions.

The final reward can be defined as the weighted sum of the three parts. The weighting of the individual parts should be adapted to the use case.

B. State Definition

The representation of the state is essential for the agent to understand its environment and be able to select suitable actions. Based on the action, the identified works can be divided into two approaches affecting the state representation. The agent can be trained to select the next part to be (dis)assembled on a set of possible parts, or it can be trained to decide if a given part should be (dis)assembled. If the latter approach is chosen, the state representation must cover the current state of the assembly, as well as the selected part, which is achieved by a discrete identifier for each part [36] or a one-hot encoded list [34].

The simplest representation of the assembly state is the subdivision into possible configuration states. The state description can thus be expressed as a discrete number [22,23]. A more comprehensive and proven description of the state of the assembly is a binary list. This list contains an entry for each component of the module, which describes the status of the component. This entry can either have the value 0 or 1. This simplistic representation is used in most publications [16,17,33,34,36]. However, it also leaves out some information. Therefore, some authors extend the list with more information like the currently selected tool expressed as a discrete number [16,17].

Another proven data structure is a connectivity matrix. It contains a row and column for each component of the assembly. The value at the position (i,j) describes the connection between components i and j. This representation is particularly useful for components with several possible connections, such as truss systems [19,24]. Spanning a grid to map the surroundings is also presented. Here, the space around the module is divided into a grid. Each grid field can assume a value that describes the state of the field [26]. Alessio et al. use the flexibility of this approach to represent configuration states as grid fields [25]. Valid, invalid, and final configurations are represented with different values.

Especially in disassembly sequence planning, the representation of the assembly is often represented as an AND/OR graph [32,35]. Accordingly, some authors deal with the processing of the data structure into a suitable state description of the assembly for the agent. Zhao et al. propose their MSDHGM data structure for this purpose [35]. The use of camera images as a status description of the assembly and the component to be assembled is also considered [21]. This approach is particularly flexible in its application, but also time-consuming for training the agent. The authors therefore suggest initializing the network with a pre-trained network and using transfer learning. In general, it can be said that when working in the field of robotics, the state description is divided into two parts. On the one hand, the state of the assembly is described, and on the other, the state of the robot [24,34].

The analysis of the state descriptions shows that the use of a binary list describing the state description of an assembly is sufficient in many cases. Due its simplistic one-dimensionality, the training time is limited because of the fewer combinatorial possibilities to be considered. However, this representation should not be used if components can have several possible connections, e.g. in truss systems. In this case, the use of a connectivity matrix is recommended instead. It is also possible to represent the state of the assembly as a grid. This approach is particularly flexible to use. However, when mapping a large space around the assembly, the size of the grid also tends to become very large, which therefore increases the number of combinatorial possibilities and corresponds in an increased training time.

Although the use of an image to describe the status of the assembly and, if applicable, the component to be installed was also used, this approach is proving to be time-consuming for training the agent. Additionally, it must be ensured that all aspects relevant to the agent's decision are covered in the images, so it may be necessary to take several images of the assembly from different perspectives. If this approach is chosen, it is advisable to train the network with an image data set in advance to shorten the training.

The descriptions mentioned so far refer exclusively to the state of the parts or assembly. Depending on the use case, it may also be necessary to consider the state of a robot. General statements are difficult to define. However, factors that are included in the determination of the reward must also be taken into account in the state description of the agent.

C. Potential Research Opportunities

Some studies compare the performance of RL with genetic algorithms in the respective application [17,31,32,34–36]. RL methods are often faster in determining the sequences as well as in the performance of the generated sequences regarding the defined costs. However, it should be noted that, depending on the complexity of the use case, training is very time-consuming. Therefore, many of the presented works limit the number of combinatorial possibilities in the studied use cases [22,23,25].

However, the application of concepts such as curriculum learning, parameter transfer or fine-tuning for specific use cases seems promising, but are not yet very widespread [19,35]. For this reason, we argue that research into highly general models through training on many different use cases with subsequent fine-tuning on the respective use case has great potential. Central to the development of such a model will be concepts such as parameter transfer and curriculum learning. Accordingly, more research is needed into the application of such a model in a variety of different industrial application areas.

V. Conclusion

In conclusion, the analysis of reinforcement learning applications in (dis-)assembly sequence planning provides valuable insights for the effective implementation of this technology. The recommended approach for defining reward functions involves considering three key factors: the duration of (dis)assembly execution, the achievement of desired assembly states, and rewards adapted to specific robotic parameters. By incorporating these factors into a weighted sum, tailored to the use case, a robust reward function can be devised. The examination of state representations highlights the suitability of a binary list for many cases due to its simplicity and efficient training time. However, the use of a connectivity matrix is advised for scenarios involving components with multiple possible connections, such as truss systems. Alternatively, a grid-based representation offers flexibility but requires careful consideration of the combinatorial possibilities to avoid increased training times.

While the use of images to describe assembly status is flexible, it proves time-consuming and necessitates comprehensive coverage of decision-relevant aspects. Overall, the state description must not only encompass the assembly but also consider factors relevant to the robot's state, aligning with the reward determination. This comprehensive understanding of reward functions and state representations provides a solid foundation for the successful integration of reinforcement learning in (dis-)assembly sequence planning across diverse applications and scenarios. The limited number of identified publications indicate, that the utilization of reinforcement learning methods for the generation of assembly and disassembly sequences is an underexplored subject. Therefore, possible areas of interest for future works were outlined, to guide future research in the described field.

Acknowledgment

This research received no external funding. The authors declare no conflict of interest.

References

[1] Boothroyd, G.; Alting, L. Design for Assembly and Disassembly. CIRP Annals 1992, 41, 625–636, doi:10.1016/S0007-8506(07)63249-1.

[2] Fazio, T.L. de; Edsall, A.C.; Gustavson, R.E.; Hernandez, J.; Hutchins, P.M.; Leung, H.-W.; Luby, S.C.; Metzinger, R.W.; Nevins, J.L.; Tung, K.; et al. A Prototype of Feature-Based Design for Assembly. Journal of Mechanical Design 1993, 115, 723–734, doi:10.1115/1.2919261.

[3] Wang, K.; Li, X.; Gao, L. Modeling and optimization of multi-objective partial disassembly line balancing problem considering hazard and profit. Journal of Cleaner Production 2019, 211, 115–133, doi:10.1016/j.jclepro.2018.11.114.

[4] Homem de Mello, L.S.; Sanderson, A.C. A correct and complete algorithm for the generation of mechanical assembly sequences. In Proceedings / 1989 IEEE International Conference on Robotics and Automation, [May 14 - 19, 1989, Scottsdale, Arizona]. Proceedings, 1989 International Conference on Robotics and Automation, Scottsdale, AZ, USA, 14-19 May 1989; IEEE Computer Society Press: Washington, DC, 1989; pp 56–61, ISBN 0-8186-1938-4.

[5] Jiménez, P.; Torras, C. An efficient algorithm for searching implicit AND/OR graphs with cycles. Artificial Intelligence 2000, 124, 1–30, doi:10.1016/S0004-3702(00)00063-1.

[6] Wang, D.; Shao, X.; Ge, X.; Liu, S. A hybrid technology for assembly sequence planning of reflector panels. AA 2017, 37, 442–451, doi:10.1108/AA-12-2016-171.

[7] Tseng, H.-E.; Chang, C.-C.; Lee, S.-C.; Huang, Y.-M. A Block-based genetic algorithm for disassembly sequence planning. Expert Systems with Applications 2018, 96, 492–505, doi:10.1016/j.eswa.2017.11.004.

[8] Hong, D.S.; Cho, H.S. A neural-network-based computational scheme for generating optimized robotic assembly sequences. Engineering Applications of Artificial Intelligence 1995, 8, 129–145, doi:10.1016/0952-1976(94)00068-X.

[9] Sutton, R.S.; Barto, A. Reinforcement learning: An introduction, Second edition; The MIT Press: Cambridge, Massachusetts, London, England, 2018, ISBN 9780262352703.

[10] Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep Reinforcement Learning: A Brief Survey. IEEE Signal Process. Mag. 2017, 34, 26–38, doi:10.1109/msp.2017.2743240.

[11] Goodfellow, I.; Courville, A.; Bengio, Y. Deep learning; The MIT Press: Cambridge, Massachusetts, 2016, ISBN 9780262337373.

[12] D. Silver. Lectures on reinforcement learning, 2015.
[13] Feurer, M.; Hutter, F. Hyperparameter Optimization. In Automated machine learning: Methods, systems, challenges; Hutter, F., Kotthoff, L., Vanschoren, J., Eds.; Springer: Cham, Switzerland, 2019; pp 3–33, ISBN 9783030053178.
[14] A. D. Laud. Theory and application of reward shaping in reinforcement learning, 2004.
[15] Aria, M.; Cuccurullo, C. bibliometrix : An R-tool for comprehensive science mapping analysis. Journal of Informetrics 2017, 11, 959–975, doi:10.1016/j.joi.2017.08.007.
[16] Neves, M.; Vieira, M.; Neto, P. A study on a Q-Learning algorithm application to a manufacturing assembly problem. Journal of Manufacturing Systems 2021, 59, 426–440, doi:10.1016/j.jmsy.2021.02.014.
[17] Neves, M.; Neto, P. Deep reinforcement learning applied to an assembly sequence planning problem with user preferences. Int J Adv Manuf Technol 2022, 122, 4235–4245, doi:10.1007/s00170-022-09877-8.
[18] Giorgio, A. de; Maffei, A.; Onori, M.; Wang, L. Towards online reinforced learning of assembly sequence planning with interactive guidance systems for industry 4.0 adaptive manufacturing. Journal of Manufacturing Systems 2021, 60, 22–34, doi:10.1016/j.jmsy.2021.05.001.
[19] Hayashi, K.; Ohsaki, M.; Kotera, M. Assembly Sequence Optimization of Spatial Trusses Using Graph Embedding and Reinforcement Learning. Journal of the International Association for Shell and Spatial Structures 2022, 63, 232–240, doi:10.20898/j.iass.2022.016.
[20] Kitz, K.; Thomas, U. Neural dynamic assembly sequence planning. In 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE). 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), Lyon, France, 23–27 Aug. 2021; IEEE, 2021; pp 2063–2068, ISBN 978-1-6654-1873-7.
[21] . Zhao, M.; Guo, X.; Zhang, X.; Fang, Y.; Ou, Y. ASPW-DRL: assembly sequence planning for workpieces via a deep reinforcement learning approach. AA 2019, 40, 65–75, doi:10.1108/AA-11-2018-0211.
[22] Antonelli, D.; Zeng, Q.; Aliev, K.; Liu, X. Robust assembly sequence generation in a Human-Robot Collaborative workcell by reinforcement learning. FME Transactions 2021, 49, 851–858, doi:10.5937/FME2104851A.
[23] Antonelli, D.; Aliev, K. Robust assembly task assignment in Human Robot Collaboration as a Markov Decision Process problem. Procedia CIRP 2022, 112, 174–179, doi:10.1016/j.procir.2022.09.068.
[24] Yin, J.; Chen, M.; Zhang, T. Optimization Algorithm for Cooperative Assembly Sequence of Truss Structure Based on Reinforcement Learning. In Intelligent Robotics and Applications; Liu, X.-J., Nie, Z., Yu, J., Xie, F., Song, R., Eds.; Springer International Publishing: Cham, 2021; pp 474–484, ISBN 978-3-030-89091-9.
[25] Alessio, A.; Aliev, K.; Antonelli, D. Robust Adversarial Reinforcement Learning for Optimal Assembly Sequence Definition in a Cobot Workcell. In Advances in Manufacturing III; Trojanowska, J., Kujawińska, A., Machado, J., Pavlenko, I., Eds.; Springer International Publishing: Cham, 2022; pp 25–34, ISBN 978-3-030-99309-2.
[26] dos Santos, S.R.B.; Givigi, S.N.; Nascimento, C.L. Autonomous construction of structures in a dynamic environment using Reinforcement Learning. In 2013 IEEE International Systems Conference (SysCon). 2013 7th Annual IEEE Systems Conference (SysCon), Orlando, FL, 15–18 Apr. 2013; IEEE, 2013; pp 452–459, ISBN 978-1-4673-3107-4.
[27] Watanabe, K.; Inada, S. Search algorithm of the assembly sequence of products by using past learning results. International Journal of Production Economics 2020, 226, 107615, doi:10.1016/j.ijpe.2020.107615.
[28] Winter, J. de; Beir, A. de; El Makrini, I.; van de Perre, G.; Nowé, A.; Vanderborght, B. Accelerating Interactive Reinforcement Learning by Human Advice for an Assembly Task by a Cobot. Robotics 2019, 8, 104, doi:10.3390/ROBOTICS8040104.
[29] Winter, J. de; EI Makrini, I.; van de Perre, G.; Nowé, A.; Verstraten, T.; Vanderborght, B. Autonomous assembly planning of demonstrated skills with reinforcement learning in simulation. Auton Robot 2021, 45, 1097–1110, doi:10.1007/s10514-021-10020-x.
[30] Cebulla, A.; Asfour, T.; Kröger, T. Efficient Multi-Objective Assembly Sequence Planning via Knowledge Transfer between Similar Assemblies. In 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE). 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), Auckland, New Zealand, 26–30 Aug. 2023; IEEE, 2023; pp 1–7, ISBN 979-8-3503-2069-5.
[31] Guo, K.; Liu, R.; Duan, G.; Liu, J.; Cao, P. Research on dynamic decision-making for product assembly sequence based on Connector-Linked Model and deep reinforcement learning. Journal of Manufacturing Systems 2023, 71, 451–473, doi:10.1016/j.jmsy.2023.09.015.
[32] Bi, Z.; Guo, X.; Wang, J.; Qin, S.; Qi, L.; Zhao, J. A Q-Learning-based Selective Disassembly Sequence Planning Method. In 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC). 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Prague, Czech Republic, 09–12 Oct. 2022; IEEE, 2022; pp 3216–3221, ISBN 978-1-6654-5258-8.
[33] Chen, Z.; Li, L.; Zhao, F.; Sutherland, J.W.; Yin, F. Disassembly sequence planning for target parts of end-of-life smartphones using Q-learning algorithm. Procedia CIRP 2023, 116, 684–689, doi:10.1016/j.procir.2023.02.115.
[34] Yang, C.; Xu, W.; Liu, J.; Yao, B.; Hu, Y. Robotic Disassembly Sequence Planning Considering Robotic Movement State Based on Deep Reinforcement Learning. In 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD). 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Hangzhou, China, 04–06 May 2022; IEEE, 2022; pp 183–189, ISBN 978-1-6654-0527-0.
[35] Zhao, X.; Li, C.; Tang, Y.; Cui, J. Reinforcement Learning-Based Selective Disassembly Sequence Planning for the End-of-Life Products With Structure Uncertainty. IEEE Robot. Autom. Lett. 2021, 6, 7807–7814, doi:10.1109/LRA.2021.3098248.
[36] Cui, J.; Yang, C.; Zhang, J.; Tian, S.; Liu, J.; Xu, W. Robotic disassembly sequence planning considering parts failure features. IET Collab Intel Manufact 2023, 5, doi:10.1049/cim2.12074.
[37] Allagui, A.; Belhadj, I.; Plateaux, R.; Hammadi, M.; Penas, O.; Aifaoui, N. Reinforcement learning for disassembly sequence planning optimization. Computers in Industry 2023, 151, 103992, doi:10.1016/j.compind.2023.103992.
[38] Liu, J.; Xu, Z.; Xiong, H.; Lin, Q.; Xu, W.; Zhou, Z. Digital Twin-Driven Robotic Disassembly Sequence Dynamic Planning Under Uncertain Missing Condition. IEEE Trans. Ind. Inf. 2023, 19, 11846–11855, doi:10.1109/TII.2023.3253187.
[39] Collins, K.; Palmer, A.J.; Rathmill, K. The Development of a European Benchmark for the Comparison of Assembly Robot Programming Systems. In Robot Technology and Applications: Proceedings of the 1st Robotics Europe Conference Brussels, June 27-28, 1984; Rathmill, K., MacConaill, P., O'Leary, P., Browne, J., Eds.; Springer: Berlin, Heidelberg, 1985; pp 187–199, ISBN 978-3-662-02442-3.