

# Automating the assembly planning process to enable design for assembly using reinforcement learning

Rafael Parzeller <sup>1,2,✉</sup>, Dominik Koziol <sup>1</sup>, Tizian Dagner <sup>1</sup> and Detlef Gerhard <sup>2</sup>

<sup>1</sup> Siemens AG, Germany, <sup>2</sup> Ruhr-Universität Bochum, Germany

✉ raphael.parzeller@siemens.com

## Abstract

This paper introduces a new concept for the automation of the assembly planning process, to enable Design for Assembly (DfA). The approach involves the application of reinforcement learning (RL) to assembly sequence planning (ASP) based on a 3D-CAD model. The ASP algorithm determines assembly sequences through assembly by disassembly. The assembly sequence is then used for the generation of subassemblies by considering the product contact information. The approach aims to support the creation of the manufacturing bill of materials (MBOM) by automating the assembly planning process.

*Keywords: design for x (DfX), artificial intelligence (AI), assembly sequence planning, subassembly identification*

## 1. Introduction

A common problem in the manufacturing industry is the separation of product development and manufacturing. Products are often designed without fully considering the challenges of assembly. This results in inefficient manufacturing processes and repetitive design workflows. By applying Design for Assembly (DfA), companies can overcome this problem and develop products that are assembled faster and with higher quality (Rashid et al. 2012). This results in an accelerated and more efficient product creation process. DfA specifically examines the assemblability of a product during the engineering design process, thereby avoiding potential errors during assembly as well as creating an assembly-friendly solution.

As product lot sizes continue to shrink and the number of variants increases due to more customized products, automated or at least assisted consideration of assembly at the design stage is necessary to keep production costs low (Rashid et al. 2012). Rapid advances in the use of artificial intelligence methods are opening up new optimization opportunities for the assembly process. Using reinforcement learning (RL) methods to determine an optimal assembly sequence for a product is one possibility. This helps to make assembly planning more efficient.

RL is a subfield of machine learning where an agent learns to develop an optimal strategy by interacting with its environment to maximise the cumulative reward over time. The agent makes decisions, receives feedback in the form of rewards or punishments for its actions, and adapts its behaviour to achieve optimal results in the long term. (François-Lavet et al. 2018)

This paper presents an automated assembly planning approach based on a reinforcement learning algorithm interacting with a physics-based environment. The algorithm is trained to find a feasible assembly sequence through assembly by disassembly, which is then applied to identify subassemblies using a complementary method.

## 2. Related works

### 2.1. Assembly Sequence planning

Assembly Sequence planning (ASP) refers to one of the core tasks of the product assembly planning (Rashid et al. 2012). The concept of ASP is to automatically generate all possible assembly sequences based on the assembly design (Xing et al. 2007). Traditionally, the assembly sequence is determined depending on the knowledge of an engineer (Bahubalendruni and Biswal 2015). The assembly sequence is critical to the assembly plan and affects the layout of the assembly line, as well as the efficiency and cost of product design. (Abdullah et al. 2019) The identification of feasible assembly sequences can be achieved through graphical representation of the ASP problem, such as an AND/OR graph (Homem de Mello and Sanderson 1991). As the search space of ASP significantly expands with a growing number of product components to be assembled, it becomes an NP-hard combinatorial problem (Lv and Lu 2009; Yong-Fa and Zhi-Gang 2007). Several soft computing techniques have been presented in the literature to solve the ASP problem, including Genetic Algorithms (GA), Ant Colony Optimization (ACO) and Neural Networks (NN). Most research activities are currently limited or focused on assembly processes along the principal direction axes. (Deepak et al. 2018) A subset of ASP is Assembly Path Planning (APP), which is concerned with computing the physical collision-free path of components in three-dimensional space to assemble them into a product. Current research is using APP to obtain assembly sequences through assembly by disassembly (Tian et al. 2022).

### 2.2. Subassembly identification

Subassembly identification (SI) techniques enable the identification of components that can be assembled prior to the final assembly of a product (Dini and Santochi 1992; Trigui et al. 2017). In the context of ASP, current researchers use SI to reduce the complexity of a product, as well as to simplify the combinatorial problem (Belhadj et al. 2016; Munker et al. 2023). There are various possible outcomes of SI for a single product due to different assembly constraints, which include subassemblies that are not feasible for assembly. Like ASP, SI is also a combinatorial problem, whose search space grows with the increase in the number of components present in the assembly. In most of the previous research in the field of SI, a human intervention remains essential to ensure a feasible classification of subassemblies (Belhadj et al. 2016). In contrast to prior research, the presented method conducts the SI after obtaining a viable assembly sequence, guaranteeing feasible identified subassemblies.

## 3. Concept for an automated assembly planning process

Considering the features of assembly planning described in the prior sections, a generation concept for the Manufacturing Bill of Materials (MBOM) is derived (see Figure 1). This process is comparable to the traditional assembly planning method, with the addition of automatically processed information without the need for human intervention.

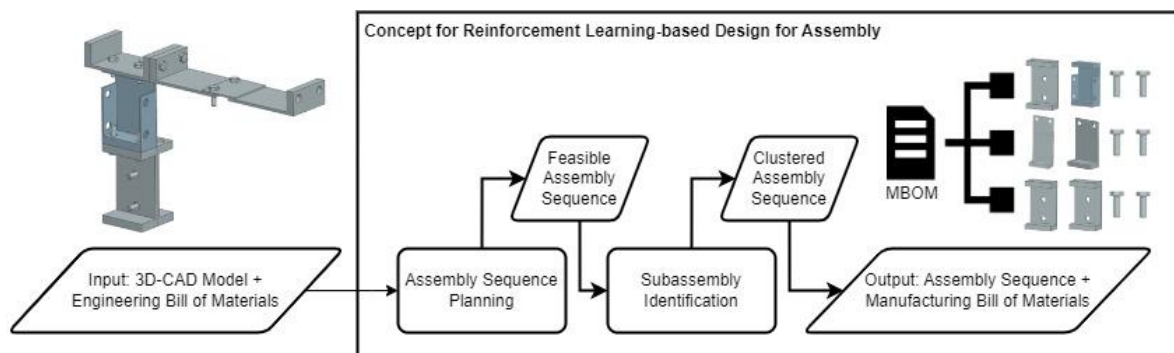


Figure 1. Concept for an automated assembly planning process

Automated assembly planning is part of the design phase, which allows the designer to evaluate the product design for assembly. The design phase involves finishing the assembly in the 3D-CAD system

and creating the Engineering Bill of Materials (EBOM). The automated assembly process extends the output of the design stage with the automated generation of the Manufacturing Bill of Material through the information of the identified subassemblies from the product. An algorithm is used to generate a feasible assembly sequence for the product, which is subsequently used, together with the contact matrix of the 3D-CAD model, to identify subassemblies. This leads to the automatic transformation of the EBOM into the MBOM.

### 3.1. Method for assembly sequence planning based on reinforcement learning

To enable an automated evaluation of the assembly process of a product, the determination of the assembly sequence is critical, given that it indicates whether a product can be assembled as designed or if the design needs to be modified. In this paper, a new assembly by disassembly approach for automating ASP in 3D-CAD by employing reinforcement learning is proposed.

#### 3.1.1. Assembly sequence planning as Partially Observable Markov Decision Process

To enable the utilization of RL in dynamic environments, it is essential to describe the combinatorial problem of the ASP as a finite-horizon Partially Observable Markov Decision Process (POMDP). The POMDP is characterized by a 7-Tuple  $[S, A, \Omega, T, O, R, \gamma]$ , where  $S$  stands for a set of partially observable states  $S=[s_1, s_2, \dots]$ ,  $A$  for a set of actions  $A=[a_1, a_2, \dots]$ ,  $\Omega$  for a set of observations  $\Omega=[o_1, o_2, \dots]$ ,  $T$  for a set of conditional probabilities  $T(s_{t+1}|s_t, a)$  for the transition from state  $s_t$  to the subsequent state  $s_{t+1}$  after taking action  $a$ . The observation function, denoted by  $O$ , determines the observation probability  $O(o|s_t, a)$  of obtaining a specific observation after taking an action in the current state  $s_t$ .  $R$  defines the reward function which determines the reward of the agent after taking an action in the current state. The discount factor  $\gamma$  is a value between 0 and 1 which balances the weight of future rewards compared to immediate rewards (François-Lavet et al. 2018).

The state space  $S$  of the POMDP is described by the set of partially observable states  $s=[x_b, y_b, z_b, o_{x_b}, o_{y_b}, o_{z_b}, m_b, \dots]$ , where  $x_b, y_b, z_b$  stands for the position and  $o_{x_b}, o_{y_b}, o_{z_b}$  for the orientation of the selected component  $b$  in the three-dimensional space. Whether the component  $b$  is disassembled or not is described by  $m_b$ . A component is disassembled by touching of a disassembly zone, which is defined by a box that contains the assembly inside.

The action space of the POMDP defines the force and torque which is applied on the  $x, y$  and  $z$  axes of a component. This leads to a description of an action in the action space  $A$  by  $a=[f_x, f_y, f_z, q_x, q_y, q_z]$  where  $f_x, f_y, f_z, q_x, q_y, q_z \in [-1, 1]$ .

An observation  $o$  of the Observation Space  $\Omega$  is described by  $o=[x_b, y_b, z_b, d_x, d_{xinv}, d_y, d_{yinv}, d_z, d_{zinv}, c, t]$  which contains partially information about the environment, where  $x_b, y_b, z_b$  represent the position of the selected component in the three-dimensional space. Meanwhile  $d_x, d_{xinv}, d_y, d_{yinv}, d_z, d_{zinv}$  stand for the distance to the disassembly zone for the selected component of the assembly,  $c$  for the number of collisions of the component and  $t$  for the current time step of the episode. The algorithm receives a reward for successfully moving a component to the disassembly zone and disassembling all components. In the same vein it incurs penalties for each time step and collision that occurs (see Equation 1).

$$R(o) = -0.01 + \begin{cases} 200, & \text{if component is disassembled} \\ 0, & \text{otherwise} \end{cases} + \begin{cases} 1500, & \text{if terminated} \\ 0, & \text{otherwise} \end{cases} + \begin{cases} -1, & \text{if collision} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

The algorithm operates with a defined time to disassemble a chosen component. When the allotted time has expired, the subsequent component on the assembly's part list is chosen. If all components have been processed, the operation restarts with the remaining components that have not yet been disassembled. The agent's objective is to maximise its reward over time by achieving its goal. The goal of optimisation is to disassemble the assembly in the shortest possible time with minimal collisions. Collisions during the disassembly of the selected component refer to the contact between the solid body of the component selected by the agent and the solid bodies of the other assembly components.

### 3.1.2. Method for automated subassembly identification through the assembly sequence

To automatically divide an assembly into subassemblies, the method requires a possible assembly sequence and information regarding the contact between the product components. The identification of subassemblies begins by verifying the contact relationships between consecutive components of the assembly sequence, specifically  $\tau_t$  and  $\tau_{t+1}$ . In the initial assembly, two components are considered to be in contact when their solids touch. If two components are in contact and are consecutive in the assembly sequence, a subassembly is formed with both components. The assembly process then proceeds by increasing the considered time  $t$  by one unit to evaluate the next component in the sequence. The investigation continues by assessing whether the newly considered component,  $\tau_t$ , is in contact with one of the components of the previously created subassembly, in which case, the component becomes part of the subassembly and the time step increases by one. If there is no contact with the components within the previously created subassembly, the initial check is started with the component of the current time step. The time step is increased by one if there is no contact between components during the initial check. This process is repeated until the time step exceeds the total length of the assembly sequence minus one. This systematic approach results in the division of the entire assembly into feasible subassemblies in accordance with the assembly sequence and contact information of a product (see Figure 2).

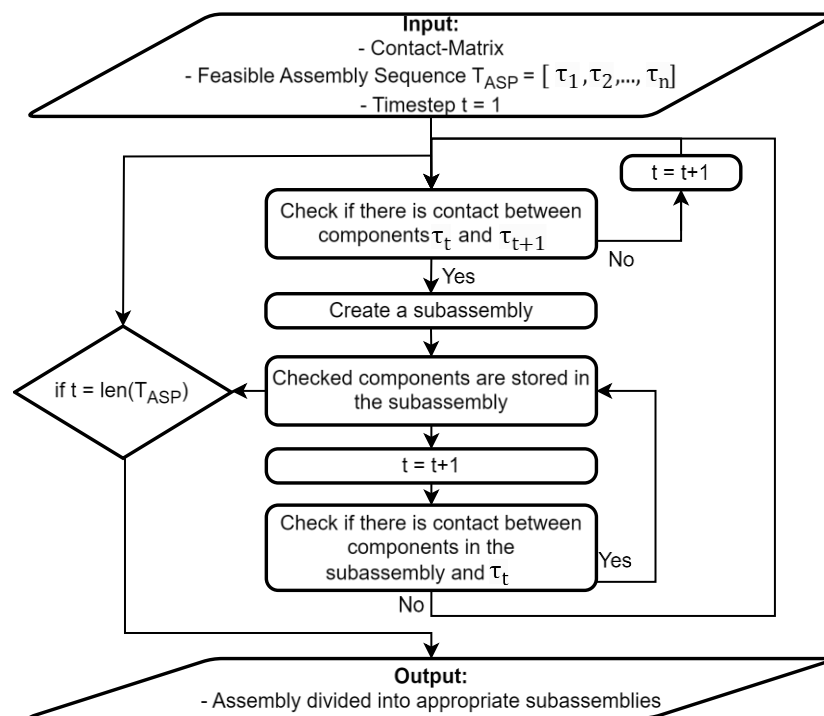


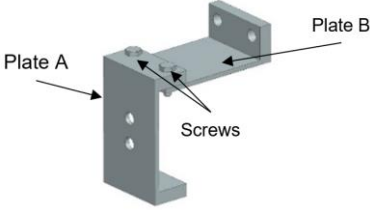
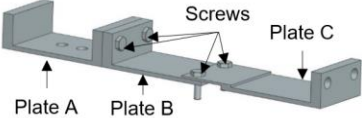
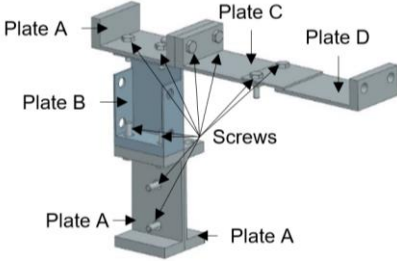
Figure 2. Proposed method for subassembly identification through the assembly sequence

## 4. Implementation and validation

The following section applies the proposed methods for assembly sequence planning and subassembly identification. 3D-CAD models of example assemblies are utilized to verify the proposed methods. The assemblies are based on an example from the literature (Giordano et al. 2010; Whitney et al. 1999). These assemblies are used to assess the methodology's ability to handle different degrees of complexity. The primary objective of the validation process is to determine the usability of reinforcement learning for ASP problem solving in accordance with the assembly by disassembly principle and the application of the subassembly identification approach based on feasible assembly sequences.

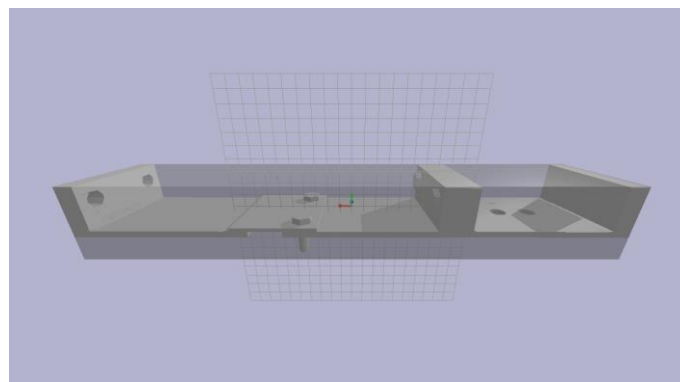
The learning of the RL algorithm is performed on a general-purpose system (Intel®Core(TM) i5-1245U CPU @ 1,6 GHz, 32.000 MB, Python 3.10) without the usage of an external GPU. All training environments are trained for 20000 timesteps each. As part of the validation process, three assemblies with up to 16 components were designed using commercially available 3D-CAD software. The components were joined together using touch constraints. Screw and hole threads are not accounted for in the assemblies. The assemblies consist of differently shaped plates with holes and screws (see Table 1).

**Table 1. Description of the assemblies used for the validation**

Assembly Information	Visualisation of the Assembly
Name: Assembly 1 Components: 1x Plate A 1x Plate B 2x Screws Total Components: 4	
Name: Assembly 2 Components: 1x Plate A 1x Plate B 1x Plate C 4x Screws Total Components: 7	
Name: Assembly 3 Components: 3x Plate A 1x Plate B 1x Plate C 1x Plate D 10x Screws Total Components: 16	

#### 4.1. Assembly sequence planning

The presented reinforcement learning method is validated for the ability to find at least one feasible assembly sequence for the respective assembly using the assemblies shown in Table 1. The validation of the reinforcement learning method is implemented by using a Proximal Policy Optimization (PPO) algorithm (Schulman et al. 2017). The planning of the assembly sequence through the RL algorithm is performed within the open source PyBullet physics-based simulation environment (Coumans and Bai 2021). Figure 3 displays assembly 2 within the PyBullet simulation environment. The dark grey rectangle surrounding the assembly corresponds to the designated disassembly area.



**Figure 3. Assembly and disassembly zone within the PyBullet simulation environment**

The algorithm is able to successfully disassemble all validation assemblies. During the training period of 20000 time steps, assembly 1 is disassembled 353 times, assembly 2 27 times and assembly 3 5 times. The training process is shown in Figure 4, where the red markers indicate the time step at which an assembly is completely disassembled. The green markers at the level of the maximum number of components in an assembly indicate the start of each training episode.

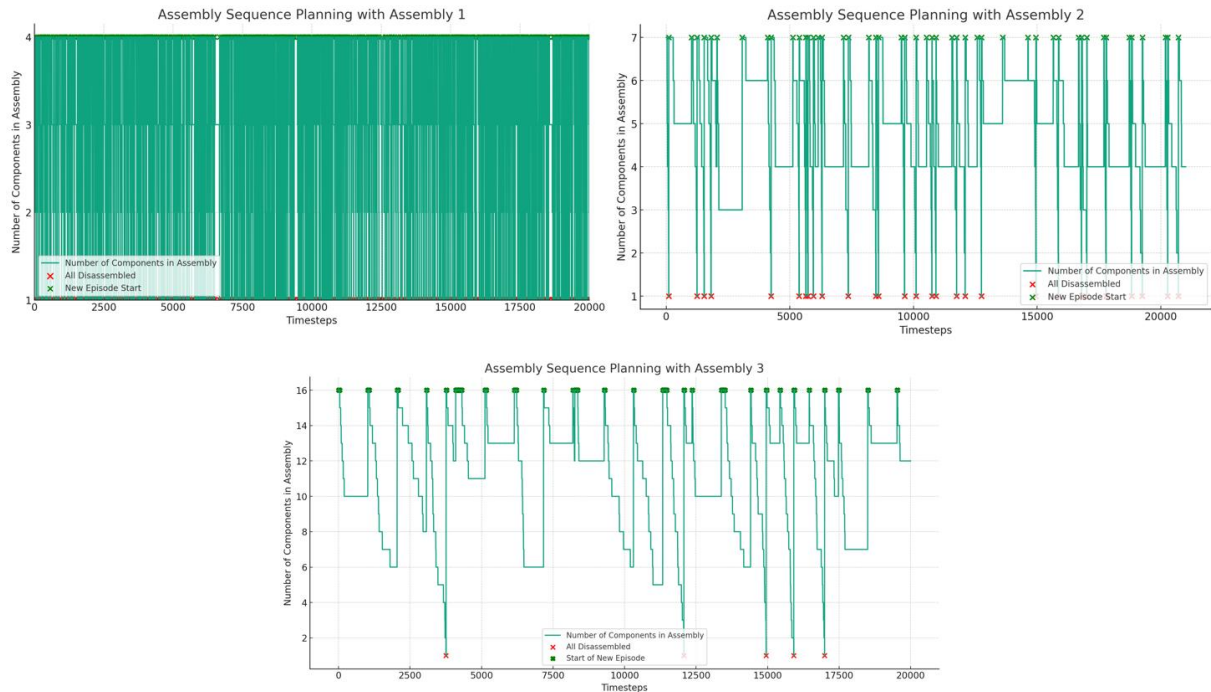


Figure 4. Validation results of the reinforcement learning algorithm for assembly sequence planning

## 4.2. Subassembly identification

To ascertain the applicability of the proposed subassembly identification method (see Figure 2), the contact matrix and the appropriate assembly sequence are required. The method is validated using two feasible assembly sequences of assembly 3 (see Table 1). The selected feasible assembly sequences are  $T1 = [P1, P2, S1, S2, P6, S7, S8, P5, P4, S5, S6, P3, S9, S10, S4, S3]$  and  $T2 = [P1, P2, S1, S2, P6, S7, S8, P3, S3, S4, P4, S9, S10, P5, S5, S6]$ . In the first assembly sequence, the assembly is divided into two distinct subassemblies using the previously outlined method. The first subassembly, labelled subassembly 1, is comprised of the following components: P1, P2, S1, S2, P6, S7, and S8. The components P5, P4, S5, S6, P3, S9, S10, S4, and S3 are part of the second subassembly, labeled subassembly 2. This is because in the eighth assembly step, component P5 is not in contact with any of the components from the previous seven assembly steps, namely P1, P2, S1, S2, P6, S7 or S8. In the second assembly sequence, a new subassembly is created that contains all the components of the product, corresponding to the entire assembly. The observed outcome for assembly sequence 2 stems from the fact that the assembly sequence forms a complete chain of contact. In this assembly sequence, component P3 follows component S8, which is different from assembly sequence 1. When assembled, component P3 comes into contact with component P6, continuing the contact chain and preventing the formation of a second subassembly. Consequently, the validation demonstrates that the outcome of the proposed subassembly identification method is highly reliant on the input assembly sequence. Figure 5 illustrates the structure of the identified subassemblies within the feasible assembly sequences.

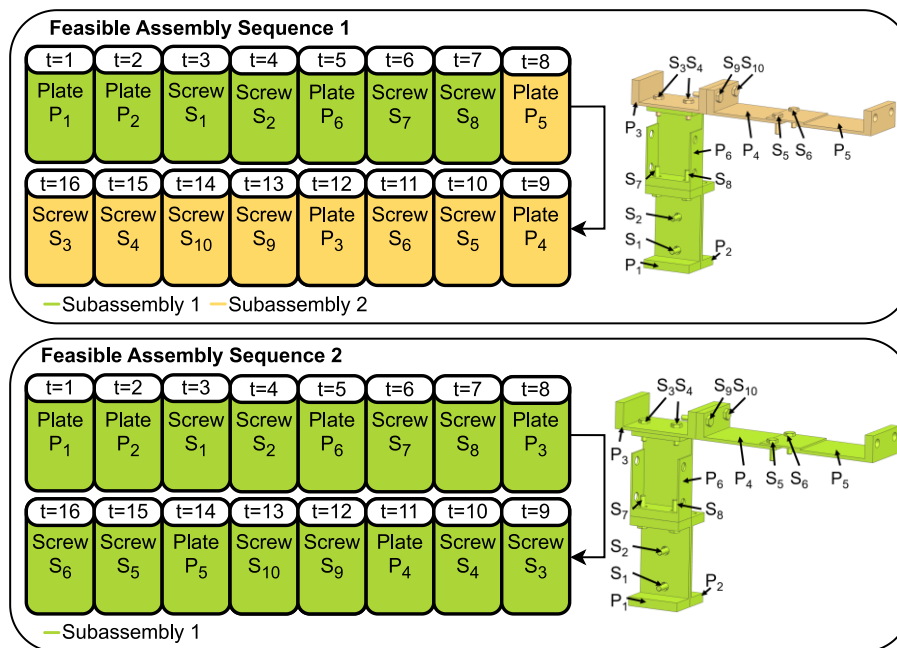


Figure 5. Validation results of the proposed subassembly identification method

## 5. Conclusion and outlook

The ability to integrate subassemblies into a unified whole greatly relies on the feasibility of their clustering. In order to guarantee this, a viable assembly sequence must be implemented. Therefore, a novel method that automates the identification of subassemblies by utilising assembly contact and assembly sequence data has been introduced. To validate this approach, example assemblies were utilised. The research demonstrates that the method identifies subassemblies suitable for assembly, without obstructing each other during assembly or disassembly. Therefore, assemblability is guaranteed through this method. The proposed automated assembly planning process concept allows the designer to obtain input on the assemblability of their product during the design phase, enhancing design for assembly and aiding in the development of the MBOM. Subsequent research should consider the individual features of factory workstations for assembly, such as the available assembly space. Additionally, the proposed extension would allow the automated planning process for assembly to cover the Bill of Process (BOP). Further research is needed to investigate the accessibility of components for assemblers and the space available for assembly tools during assembly.

## References

- Abdullah, M.A., Rashid, M.F. and Ghazalli, Z. (2019), "Optimization of Assembly Sequence Planning Using Soft Computing Approaches: A Review", *Archives of Computational Methods in Engineering*, Vol. 26 No. 2, pp. 461-474. <https://doi.org/10.1007/s11831-018-9250-y>
- Bahubalendruni, M.R., and Biswal, B.B. (2015), "A review on assembly sequence generation and its automation", *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 230 No. 5, pp. 824-838. <https://doi.org/10.1177/0954406215584633>
- Belhadj, I., Trigui, M. and Benamara, A. (2016), "Subassembly generation algorithm from a CAD model", *The International Journal of Advanced Manufacturing Technology*, Vol. 87, No. 9-12, pp. 2829-2840. <https://doi.org/10.1007/s00170-016-8637-x>
- Coumans, E. and Bai, Y. (2021), *PyBullet*, a Python module for physics simulation for games, robotics and machine learning. [online] *PyBullet*. Available at: <http://pybullet.org> (accessed 15.11.2023)
- Deepak, B., Bala Murali, G., Bahubalendruni, M.R. and Biswal, B.B. (2018), "Assembly sequence planning using soft computing methods: A review", *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*, Vol. 233 No. 3, pp. 653-683. <https://doi.org/10.1177/0954408918764459>
- Dini, G. and Santochi, M. (1992), "Automated Sequencing and Subassembly Detection in Assembly Planning", *CIRP Annals*, Vol. 41 No. 1, pp. 1-4. [https://doi.org/10.1016/S0007-8506\(07\)61140-8](https://doi.org/10.1016/S0007-8506(07)61140-8)

- François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G. and Pineau, J. (2018), "An Introduction to Deep Reinforcement Learning", *Foundations and Trends® in Machine Learning*, Vol. 11 No. 3-4, pp. 219-354. <https://doi.org/10.1561/22000000071>
- Giordano, M., Mathieu, L. and Villeneuve, F. (2010), *Product Lifecycle Management: Geometric Variations*, Wiley, Hoboken. <https://doi.org/10.1002/9781118557921>
- Homem de Mello, L.S. and Sanderson, A.C. (1991), "A correct and complete algorithm for the generation of mechanical assembly sequences", *IEEE Transactions on Robotics and Automation*, Vol. 7 No. 2, pp. 228–240. <https://doi.org/10.1109/70.75905>
- Lv, H. and Lu, C. (2009), "A discrete particle swarm optimization algorithm for assembly sequence planning", 2009 8th International Conference on Reliability, Maintainability and Safety. IEEE, Chengdu, pp. 1119–1122. <https://doi.org/10.1109/ICRMS.2009.5270057>
- Münker, S., Swoboda, D., El Zaatari, K., Malhotra, N., Manassés Pinheiro de Souza, L., Göppert, A., et al. (2023), "CAD-Based Product Partitioning for Automated Disassembly Sequence Planning with Community Detection", In: Galizia, F. G. and Bortolini, M. (Ed.), *Production Processes and Product Evolution in the Age of Disruption*, Springer, Cham, Bologna, pp. 570–577. [https://doi.org/10.1007/978-3-031-34821-1\\_62](https://doi.org/10.1007/978-3-031-34821-1_62)
- Rashid, M. F., Hutabarat, W. and Tiwari, A. (2012), "A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches", *The International Journal of Advanced Manufacturing Technology*, Vol. 59 No. 1-4, pp. 335–349. <https://doi.org/10.1007/s00170-011-3499-8>
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017). "Proximal Policy Optimization Algorithms", ArXiv preprint. <https://doi.org/10.48550/arXiv.1707.06347>
- Tian, Y., Xu, J., Li, Y., Luo, J., Sueda, S., Li, H. et al. (2022), "Assemble Them All", *ACM Transactions on Graphics*, Vol. 41 No. 6, pp. 1-11. <https://doi.org/10.1145/3550454.3555525>.
- Trigui, M., Belhadj, I. and Benamara, A. (2017), "Disassembly plan approach based on subassembly concept", *The International Journal of Advanced Manufacturing Technology*, Vol. 90 No. 1-4, pp. 219–231. <https://doi.org/10.1007/s00170-016-9363-0>
- Whitney, D.E., Mantripragada, R., Adams, J.D. and Rhee, S.J. (1999), "Designing Assemblies", *Research in Engineering Design*, Vol. 11 No. 4, pp. 229–253. <https://doi.org/10.1007/s001630050017>
- Xing, Y., Chen, G., Lai, X., Jin, S. and Zhou, J. (2007), "Assembly sequence planning of automobile body components based on liaison graph", *Assembly Automation*, Vol. 27 No. 2, pp. 157–164. <https://doi.org/10.1108/01445150710733423>
- Yong-Fa, Q. and Zhi-Gang, X. (2007), "Assembly Process Planning Using a Multi-objective Optimization Method", 2007 International Conference on Mechatronics and Automation, IEEE, Harbin, pp. 593–598. <https://doi.org/10.1109/ICMA.2007.4303610>