Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-025-02622-4

Reinforcement learning approach for decision making in disassembly processes

Felix Paschko[1] · **Abderrahim Krini**[2] · **Markus Kemke**[3] · **Steffi Knorn**[1]

Received: 19 October 2024 / Accepted: 1 May 2025

Abstract

Remanufacturing represents a key strategy for the management of end-of-life products, with the potential to promote a circular economy. However, its implementation has been limited due to the labour-intensive and time-consuming nature of the disassembly processes required for component remanufacturing. The disassembly process has to cope with further uncertainties. The quality of the cores is often unknown, which can result in fluctuating processing times and random failures of cores or components during disassembly, due to damage. These uncertainties can have a significant impact on both on-time delivery and component service levels, both of which are costly and difficult to optimise simultaneously. In pursuit of multi-objective optimization, a novel reinforcement learning (RL) framework based on the Proximal Policy Optimization (PPO) algorithm has been formulated to inform decision-making processes during the disassembly process. Two distinct categories of RL agents have been developed to facilitate collaborative decision-making, namely one type for core decision and a second type for component decision. Furthermore, diverse configurations for merging these two types of RL agents have been explored. The approach aims to enable real-time decision-making during disassembly, potentially providing companies with an economic advantage. We compare our approach to heuristic control methods and the Deep-Q-Network (DQN) algorithm and prove the potential of the RL approach using the PPO algorithm through various test cases. Based on the reward function developed, the approach using the PPO algorithm received an approximately 12% higher reward than the DQN algorithm. In comparison to the Basic heuristic, it received an 22% higher reward and a 4% higher reward than the One Disassembly heuristic. This significantly increases adherence to on-time delivery and the service level. Simulations also demonstrate that the RL agent types can adapt to changing state values, resulting in high adaptivity and scalability.

Keywords Remanufacturing · Disassembly · Reinforcement learning · Decision making · Stock management · Inventory management · Disassembly decision making · Autonomous decision making

Highlights

- Decision making in disassembly process is challenging due to uncertainties in used products
- Current literature considers deterministic conditions.
- Our work proposes a Reinforcement Learning based decision-making for complete disassembly process considering uncertainties.
- The proposed technique considers conflicting optimisation goals and doesn't need any detailed disassembly system models.

✉ Felix Paschko
felixpaschko@web.de

Abderrahim Krini
Abderrahim.Krini@bosch.com

Markus Kemke
Markus.Kemke@bosch.com

Steffi Knorn
knorn@tu-berlin.de

[1] Technische Universität Berlin, Chair of Control, Straße Des 17. Juni 135, 10623 Berlin, Germany

[2] Engineering Safety/Reliability Related Remanufacturing and Diagnostic Software, Robert Bosch Automotive Steering GmbH, Richard-Bullinger-Straße 77, 73527 Schwäbisch Gmünd, Germany

[3] IT in Manufacturing, Robert Bosch Automotive Steering GmbH, Richard-Bullinger-Straße 77, 73527 Schwäbisch Gmünd, Germany

Published online: 02 June 2025

- Our approach receives an 22% higher reward than basic heuristic, 12% higher than with the DQN algorithm, and a 4% higher reward than the One Disassembly heuristic.

Introduction

Due to the world's increasing population and environmental concerns, many countries have regulations that require original equipment manufacturers (OEMs) to take back used products from consumers at the end of their lifecycle. Remanufacturing or reusing materials and functional components from used products can be a better alternative than disposal and can also provide financial benefits (Tuncel et al., 2014). Other business objectives that may be pursued from the company's perspective include reducing production costs, securing the aftermarket, establishing an environmentally conscious corporate image, meeting increasing customer demand for eco-friendly products, or simply reducing environmental impact (Fleischmann et al., 1997; Toffel, 2004).

Motivation

Remanufacturing is an industrial practice that aims to restore worn-out products to their original or near-original condition. This is achieved through a series of industrial processes. The initial stage is a pre-cleaning and analysis, which is followed by disassembly. Subsequently, the disassembled components are subjected to an inspection, after which those that are fit for rework are passed on to that process. Subsequently, the components are subjected to reworking, cleaning and storage in inventory. Finally, a combination of remanufactured and new components are used to reassemble the product, resulting in a unit that matches or exceeds the performance and expected life of the original new product (Lund, 1984).

Nevertheless, remanufacturing has only been partially implemented, primarily due to the time-consuming disassembly processes that are required in order to retrieve the core components from the returned products. The current dominance of manual labour within the remanufacturing value chain serves to undermine its economic feasibility (Cong et al., 2019; Ong et al., 2021). In the context of an increasing diversification of product variants, the role of human operators remains crucial, particularly in operations that are predominantly manual (Vongbunyong, 2015). The disassembly process has a significant influence on the reuse rate of components, the quality of recovered parts, and the cost of remanufacturing (Tuncel et al., 2014).

Recovery profit can decrease because the disassembly process is highly unpredictable. This is caused by the variability in product type, quantity and quality, as well as the condition of the cores and the lack of robust decision-making at the operational level. This unpredictability gives rise to a considerable degree of volatility in the remanufacturing systems. The quality of cores is uncertain due to the effects of wear and tear that occur over the lifetime of the cores, which can result in degradation, aging, and other forms of depreciation (Wurster et al., 2022). The functionality and physical properties of retired products often vary significantly and are subject to uncertainty. These variations may manifest as corrosion, looseness, or the unexpected presence of defective components (Meng et al., 2022). The act of disassembling these items may give rise to a number of difficulties, many of which are related to quality issues. These are discussed in further detail below:

(1) **Disassembly processing time variation influencing on-time delivery (OTD)** The quality of the cores collected for disassembly represents a significant challenge in the disassembly process. This has a direct impact on the type and duration of disassembly operations that are required (Wurster et al., 2022). The processing time for each component per quality category at the disassembly station has an impact on the ability to deliver on time.
(2) **Ensuring an optimal service level (SL) despite uncertain core quality** A well-designed stock management system is crucial for a disassembly system to balance the service levels of remanufactured components. This requires a thorough understanding of component demand patterns, which can inform decisions about the stock's component quantities (Wurster et al., 2022). The unpredictable quality of components introduces an additional layer of complexity to stock management, as they are subject to failure at random.

The disassembly process is complex due to conflicting objectives and influences the productivity of the remanufacturing production (Paschko et al., 2023). To illustrate, a low-quality core will require a greater processing time and may have an adverse effect on on-time delivery, whereas disassembling other components may have a beneficial impact on service levels. The complexity of this process is further compounded by the requirement to make decisions under time constraints and the uncertain nature of used products. This makes the automation of decision-making a challenging task.

Background on disassembly control strategies

In recent years, there has been a notable increase in research activity concerning the disassembly process and its economic benefits for material recovery (Tuncel et al., 2014). To address the complex nature of returned products in the disassembly process, researchers have developed various

decision-making methods, as conventional approaches may prove ineffective in such cases (Rizova et al., 2020). Despite the extensive research that has been conducted in this field, studies frequently concentrate on specific aspects of disassembly decision-making. The decision-making methods that have been developed can be applied to a number of different disassembly problems:

- Disassembly line balancing problem (DLBP): DLBP was originally introduced by Gungor and Gupta (1999), who used a heuristic approach with a priority function to solve simple DLBPs, where the goal is to balance the disassembly line by determining the most efficient sequence of disassembly tasks with the minimum number of workstations possible and minimising the idle time between them (Gao et al., 2020; Guo et al., 2021, 2022; Liang et al., 2022; Wang et al., 2020). Tuncel et al. (2014) introduced a Monte-Carlo based RL approach to solve the DLBP. The study aimed to apply RL techniques to DLBP, investigate their efficiency and effectiveness compared to existing methods, and incorporate stochastic elements to handle dynamic environments. The results showed that the proposed method performed well on a large complex problem and outperformed benchmark methods reported in the literature. Several studies have investigated inventory KPIs such as time targets, inventory levels and component salvage values to manage disassembly uncertainties. For example, Reveliotis (2007) explores the use of neuro-dynamic programming for uncertainty management and modelling, where the agent determines the level of disassembly. Paprocka and Skolud (2022) studied the disassembly system to balance line efficiency and profit by predicting disassembly time and product quality based on historical data. Similarly, Riggs et al. (2015) proposed a method with multiple quality classes for end-of-life products to accurately account for varying task times in the disassembly line. The paper by Mei and Fang (2021) presents a study on using DRL to solve the DLBP in multi-robot disassembly lines, which involves optimising multiple objectives such as minimising idle time, prioritising high demand components and reducing energy consumption.
- Disassembly planning and control problems: Wenzel and Peter (2017) mentioned, that decisions should be made dynamically and individually for each operation. The authors introduce a dynamic control system for a manual disassembly line, which enables a simulation-based optimisation of operations. The system allows for the adjustment of disassembly operations or methods at each station, as well as the transfer of operations to subsequent stations. In Stecke et al. (1981) and Xanthopoulos et al. (2016), the stations are afforded the ability to select the subsequent order to be processed from a set of pending orders. The selection is made in accordance with priority rules. Stecke et al. (1981) conclude that the effectiveness of priority rules is contingent upon the system configuration, while Xanthoplous et al. (2016) emphasize that a combination of rules yields optimal results. In Wurster et al. (2022), a dynamic control logic is proposed for agile hybrid disassembly systems. The proposed methodology aims to achieve a balanced allocation of disassembly tasks between a flexible robot and a human operator, taking into account the varying quality conditions of discarded products. The approach is based on Deep Q-Learning, which has demonstrated the potential to reduce operational failures and operational costs, and system make span when compared with a priority rule heuristic. Zhang et al. (2023) proposed an improved artificial bee colony algorithm to jointly optimise process planning and scheduling in flexible job-shop remanufacturing systems, demonstrating enhanced solution quality and robustness compared to conventional metaheuristics. Di Wang et al. (2024) applied a deep reinforcement learning approach to energy-aware disassembly planning by combining manual and stimuli-triggered self-disassembly. The study showed promising results in terms of balancing profitability and energy efficiency, particularly under uncertain conditions.
- Disassembly Sequencing Planning (DSP): There has been a significant focus on DSP (Feng et al., 2019; Ren et al., 2020; Xu et al., 2020), which aim to determine the optimal disassembly sequences for a give product. Due to the complexity of determining the correct sequence, it is considered an NP-hard problem (Liu & Zhang, 2021). Prioritising the early disassembly of hazardous and high-demand components has been a common strategy in various solution approaches to the DSP, as discussed by Tuncel et al. (2014). In their study, Liu and Zhang (2021) observe the use of different techniques, such as genetic algorithms and heuristics, to find optimal solutions for the DSP (Liu & Zhang, 2021). Several studies have investigated the use of machine learning and neural networks for disassembly sequence generation. Bi et al. (2022) proposed a Q-learning technique to optimise selective disassembly sequences and maximise disassembly profit. Mao et al. (2023) proposed an adaptive disassembly sequence planning method using DRL and Deep Q-network (DQN) methods. The DRL method was found to be effective in solving dynamic and stochastic problems. The proposed method was tested with experiments and showed great potential in use cases where the failure is uncertain.

In terms of implementation, an assembly sequence planning system for workpieces (ASPW) was proposed in Zhao et al. (2019), where Deep Reinforcement Learning (DRL) was adopted in the case of sparsity rewards and lack of

training environments. Altenmüller et al. (2020) developed a self-learning agent for order dispatching in a complex job shop with time constraints using a Q-learning algorithm and discrete event simulation. They combined RL in production control and applied it to a real use case in semiconductor manufacturing. Kuhnle and Lanza (2019) discuss the potential use of RL in production planning and control. They note that production complexity has increased due to increasing product variety, decreasing quantities, and higher quality requirements. Reinforcement learning has also been applied in remanufacturing for adaptive visual inspection planning, where Kaiser et al. (2024) demonstrated that RL agents can effectively determine optimal viewpoints without relying on complete CAD models, achieving high surface coverage with minimal scans There is increasing interest in using RL to control and schedule conventional production facilities (Cunha et al., 2020; Kuhnle et al., 2021). Paschko et al. (2023) highlights that static control strategies for material release, such as CONWIP, lead to efficiency losses when applied to highly uncertain disassembly environments. They propose using an adaptive control logic based on RL that incorporates real-time system information for better performance. Remanufacturing lacks a robust decision-making strategy that can significantly improve performance under such uncertainties (Hoffmann et al., 2025). Hoffmann and Knorn (2024) proposed a dynamic optimisation model for resource allocation that accounts for failure occurrences, system versatility, and adaptive learning behaviour.

While progress has been made in disassembly decision making, significant research gaps and challenges remain. Existing research often focuses only on specific aspects. Rizova et al. (2020) identified disassembly decision making that deals with conflicting objectives and multi-objective optimisation, as well as an integrated approach that considers the uncertain characteristics of returned products, as important research gaps.

Due to the large number of different disassembly system configurations, a wide variety of challenges can be addressed and analysed accordingly. In the works, where core quality was considered, it was known in advance, which is not always the case in practice. It has traditionally been the case that researchers have utilised dynamic programming for the purpose of solving control problems. However, the applicability of such techniques is limited due to their inherent inability to analyse large state spaces. Consequently, these techniques are not recommended for use in the context of complex optimisation problems, which often involve a significant number of states and variables. In order to address this issue, Sutton and Barto (2020) have proposed the implementation of RL methods that are capable of solving multi-dimensional optimisation problems without the need for detailed manufacturing system models.

In summary, while traditional optimization algorithms like Gravitational Search Algorithm (GSA) and Improved Particle Swarm Optimisation (IPSO) struggle with the dynamic nature of our use case, RL stands out by effectively adapting to real-time data and optimising key performance metrics, making it a robust and scalable solution for modern supply chain and production challenges (Heuillet et al., 2021). However, the use of RL in the field of disassembly is limited by several difficulties: (1) no uniform base environment, resulting in significant variations for the same problem; (2) the division of basic elements such as states and actions can lead to a loss of information. The disassembly line system involves complex variables and constraints, making the selection of an appropriate algorithm a serious challenge. There is currently no standardized comparison index, as is used in deep learning for RL, which makes it difficult to compare different studies (Guo et al., 2023).

Contribution

The manuscript introduces a novel approach to autonomous decision-making within the disassembly process. The system has been designed to accommodate a range of core qualities, product variations and inherent uncertainties that can affect production outcomes. At the centre of this approach lies the utilisation of RL agents, which facilitate autonomous decision-making. The contribution of the paper can be summarised as follows:

- Novel RL agent framework: We present a RL agent framework where separate RL agents independently make decisions at both the core and component levels. This architecture enhances decision granularity, enabling better alignment with operational objectives.
- Optimisation of critical performance metrics: By employing the PPO algorithm, the proposed framework prioritizes optimisation of OTD and SL metrics. These outcomes are critical for the efficient functioning of dynamic disassembly environments.
- Scalability and adaptability: The approach demonstrates scalability to larger and more complex product configurations through simulation tests. The framework's adaptability allows it to handle uncertainties related to core quality, variable disassembly processing times, and fluctuating demand profiles.
- Comparison with baselines: We validate the performance of the proposed system through comparative studies with heuristics and DQN approaches, demonstrating superior results in OTD and SL metrics.
- Practical application: The methodology aligns closely with real-world remanufacturing constraints, making it feasible

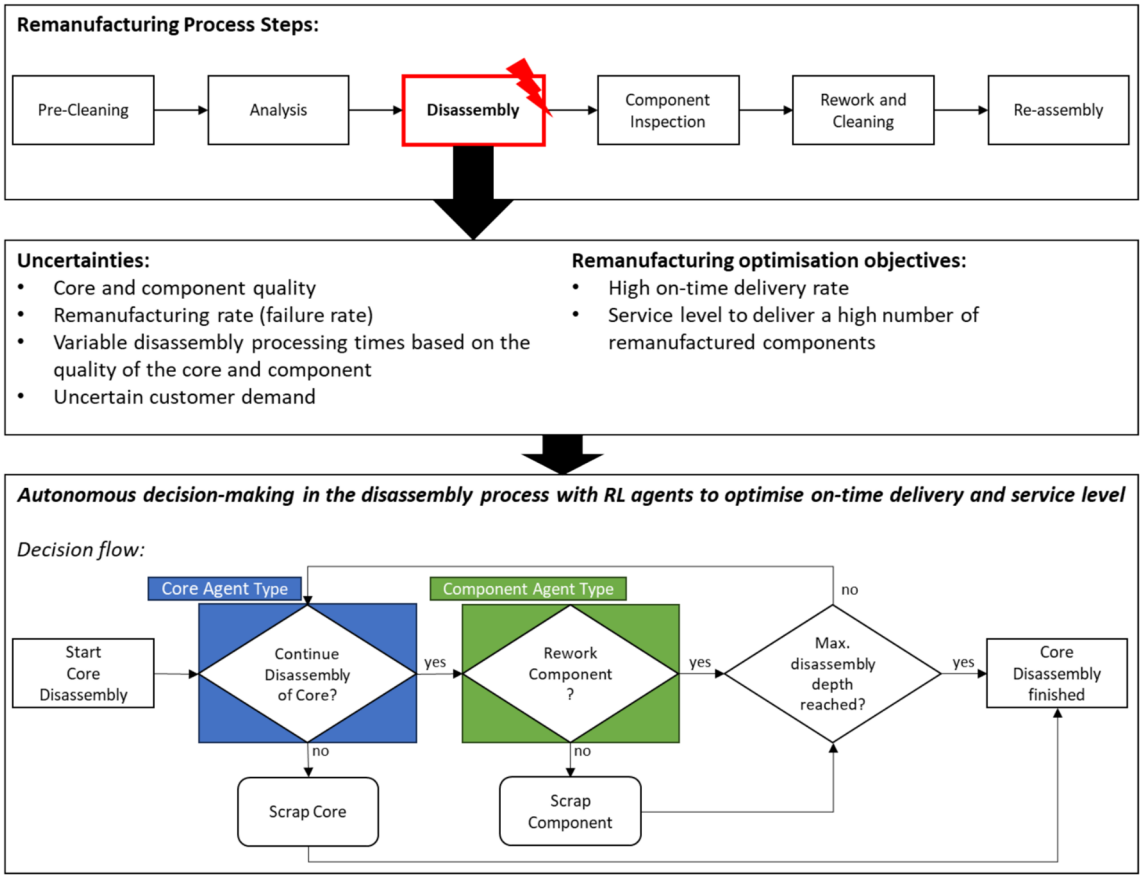

Fig. 1 Remanufacturing process steps, uncertainties and production objectives including decision flow of one core disassembly process including the core agent type and the component agent type

for practical deployment. Detailed case studies illustrate its potential for industrial-scale implementation.

The following different types of RL agents have been developed: the core agent type and the component agent type. The core agent decides whether to proceed with disassembling the entire core or stop after processing specific components. This has an impact on all the remaining components of the core. The component agent is solely responsible for deciding whether to disassemble a component and rework it or to disassemble the component and proceed with its disposal. It should be noted that the decision made by the component agent type has no impact on the other remaining components of the core. Both RL agent types are only activated upon the request for a decision and subsequently deactivated.

The parameters applied were derived from an empirically grounded simulation model that closely reflects real-world conditions. This simulation, which is based on real-world conditions, includes autonomous decision-making functionalities.

Figure 1 illustrates the remanufacturing process steps including the critical step of disassembly, the uncertainties and remanufacturing optimisation objectives and the decision flow of one core, taking into account both the core agent type and the component agent type.

The flow represents a general decision flow in the core disassembly process and is not specific to the number of components that will be used for remanufacturing. The process will be repeated for each core that is intended to be disassembled, with the objective of achieving an order quantity.

This study aims to investigate the impact of different combinations of RL agent types on the outcome. A simulation study is used to determine the best performing RL combination, and to evaluate it in comparison with alternative methodologies, thereby demonstrating the efficiency of the RL approach.

Manuscript structure

The paper is structured as follows: Sect. "Problem formulation and system model" presents the problem formulation and system model. Sect. "RL Approach" provides an in-depth

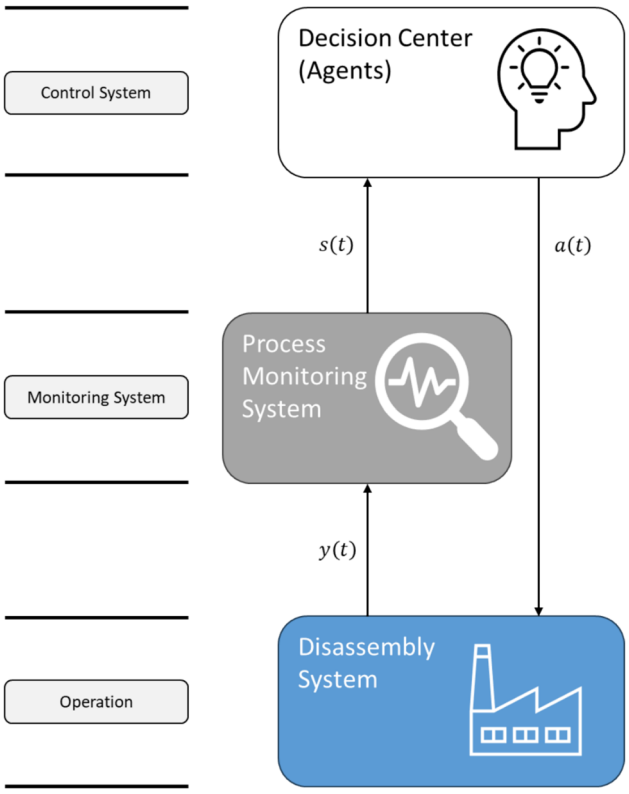

Fig. 2 Disassembly control system

explanation of the RL approach. Sect. "Practical Case Study and Validation of the RL approach" explains the scenarios in detail. Sect. "Results and discussion" presents the results and discussion. The paper finishes with the conclusion in Sect. "Conclusion".

Problem formulation and system model

Dynamic production systems are characterised by complex interactions between various operational factors. We assume that the output is measured directly in the disassembly system via sensors or other devices. Figure 2 illustrates the developed disassembly control system, which captures the time-dependent behaviour of the system. This includes the state of the system, represented by $s(t)$, the control inputs, $a(t)$, and the outputs of the operational system, $y(t)$.

The levels illustrated in Fig. 2 can be described as follows: The operation level represents the disassembly system, wherein the disassembly process is performed. The process monitoring system represents the monitoring system level, wherein the states of the disassembly system are collected, interpreted and can be visualised for the operators. The highest level of the decision instance contains the decision-making or action selection $a(t)$, which is made based on the transferred status $s(t)$ from the process monitoring system. This top-level entity is designated as 'control system'. The illustration is intended to provide a general overview, which can subsequently be transferred to the RL approach. Additional assumptions regarding the approach are explained later in this section.

Figure 3 shows a scheme of the disassembly system with n components and disassembly operations (OP_n) as well as the component inspection processes (IN_n) and the rework processes (RW_n).

The disassembly system receives an order j containing the following information: the demand for remanufactured products (D_j) and the maximum acceptable delivery time for completing the order j (DT_j). Based on the provided order information, the disassembly planning process determines the requisite quantity of cores necessary to fulfil the demand.

The production system comprises loosely connected disassembly stations, the number of which depends on the specific disassembly environment under consideration. Each station is tasked with handling N incoming orders, denoted as $j = \{1,2, \ldots, N\}$, with an order representing a predefined quantity of a cores planned for disassembly. This disassembly process follows a predetermined sequence of operations. The repertoire of potential disassembly operations, denoted as $OP = \{OP_1, OP_2, ..., OP_n\}$, depends on the number of components foreseen for remanufacturing. After disassembly, each component undergoes inspection process $IN = \{IN_1, IN_2, ..., IN_n\}$ to identify any damage that was not visible during the analysis. This may be due to internal damage or damage caused during disassembly. If there is any damage visible, the component must be scrapped. It is important to note that the RL agent has already made the decision before this step, which only determines whether the component is to be disassemble and passed on to the rework, denoted as $RW = \{RW_1, RW_2, \ldots, RW_n\}$, or if the component shouldn't be reworked. It is not feasible to perform an inspection of the components unless they are disassembled from the core, as this is not possible in the assembled state. Consequently, modification of the OP_n and IN_n steps is not feasible.

We assumed that the core's disassembly is limited to a single sequence due to the inability to specifically disassemble the most valuable (e.g., based on the monetary value) components in a different order.

The assumptions are as follows:

- Availability of the calculated cores to satisfy the demand (see Eq. (5))
- Order j must be fully processed. The demands for components are known in advance.
- The core disassembly sequence is given.
- Only one core can be selected for disassembly at a time.
- Service at disassembly workstation follows a first-come, first-serve policy.

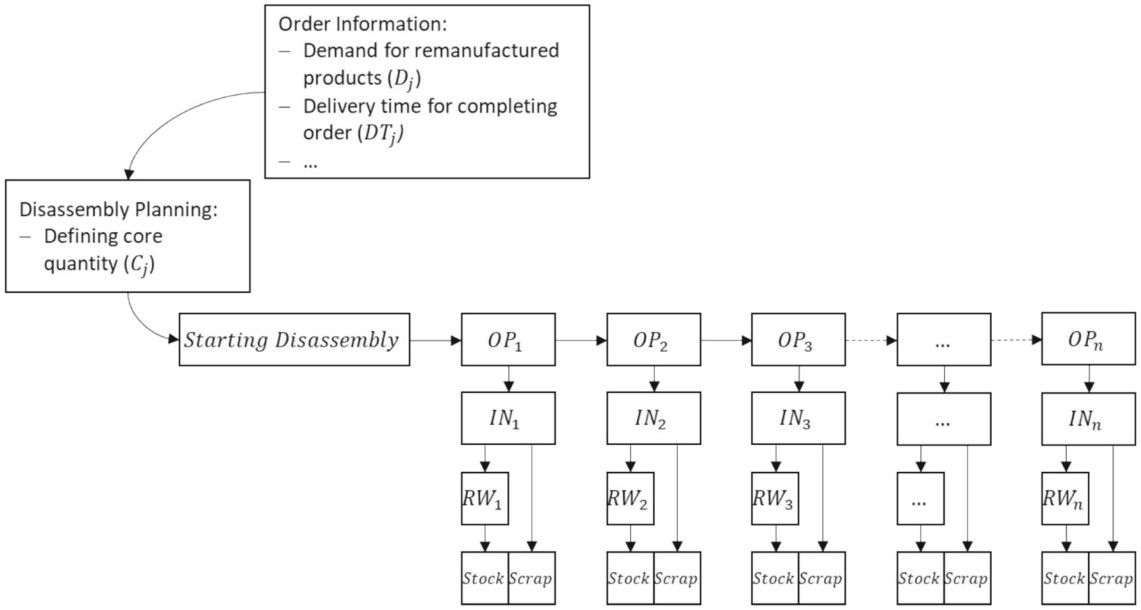

Fig. 3 Disassembly Process scheme for an example of n components

- Core conditions are taken into account. The condition is identified during the analysis process and is not known in advance.
- Damaged components are identified during the inspection process IN, e.g., inner damage or damages during the disassembly process.
- Processing times are made led as normal distributions.

After explaining the disassembly process scheme and the assumptions, we will explain the relevant variables of the remanufacturing system.

It is crucial to consider the stock value of an individual component i in the context of both inflows and outflows. In context of material management in remanufacturing, the stock value at any given time, denoted as $S_i(t)$, is a crucial metric that reflects the quantity of component i in stock at time t. The dynamic nature of stock value is governed by the rates of material inflows and outflows. Mathematically, this relationship is described by a differential equation that captures the net change in stock value over time. The differential equation representing this relationship is:

$$\frac{dS_i(t)}{dt} = I_i(t) - O_i(t) \tag{1}$$

where $I_i(t)$ is the rate of inflow of component i at time t and $O_i(t)$ is the rate of outflow of component i at time t. In order to ascertain the stock value at a specific time, given an initial stock value $S_i(0)$, it is necessary to integrate this differential equation over time. The solution to the differential equation is given by:

$$S_i(t) = S_i(0) + \int_0^t I_i(t) - O_i(t)dt \tag{2}$$

In the remanufacturing sector, determining the target stock value for an order j, denoted as $\overline{S_{i_j}}$, in a make-to-order production is essential to ensure that customer orders are fulfilled efficiently and on time. $\overline{S_{i_j}}$ can be expressed as:

$$\overline{S_{i_j}} = D_j \tag{3}$$

where D_j is the customer demand.

Since demand for order j D_j is known, the required quantity of cores for order j (C_j) can be calculated using the Injection Rate (IR) as a guiding parameter. IR represents a retrospective metric devoid of predictive insights or trends concerning the history of core quality at rework station. The IR can be computed utilizing historical data. The calculation entails dividing the number of disassembled cores by the count of good cores:

$$IR = \frac{Number of disassembled Cores}{Number of Good Cores} \tag{4}$$

The quantity of cores for order j (C_j) can be estimated using IR:

$$C_j = IR * D_j \tag{5}$$

Beside the IR, measuring the quality retrospective, there is an important variable in the remanufacturing system that

affects the whole process. In the analysis station we assign a quality class q to the core or component based on the estimated probability of failure $\overline{\lambda}$ (Klügel, 2024).

The probability of failure $\overline{\lambda}$ is derived from various damage parameters and historical data. The data obtained from field operations is of great value in providing insights into the extent of wear and tear, operational stresses and environmental conditions to which cores/components have been subjected. These parameters help in predicting the likelihood of failure. Key damage parameter, denoted as F_d, include, for example, the following elements:

- Wear and Tear (F_1): The extent of physical degradation due to friction, corrosion, or fatigue.
- Operational Stresses (F_2): The levels of mechanical, thermal, or electrical stress experienced by the object during its utilisation.
- Environmental conditions (F_3): Exposure to extreme temperatures, humidity or chemical environments.
- Usage Patterns (F_4): Frequency and intensity of use (e.g., mileage), including any overloads or improper handling.

$\overline{\lambda}$ can be expressed as a function of these damage parameters F_d, where F_d represents individual damage parameters from field operations:

$$\overline{\lambda} = g(F_1,\ F_2,\ \ldots,\ F_d) \tag{6}$$

This expression indicates that $\overline{\lambda}$ is a composite measure derived from multiple damage factors affecting the core/component. The relationship between the probability of failure and the actual core/component failures in the remanufacturing process is fundamentally rooted in damage parameters from field operations. By systematically analysing these parameters, remanufacturers can better predict and mitigate failures, leading to improved reliability and efficiency.

The responsibility of the analysis station is to estimate the probability of failure of the core and based on this estimation to assign a quality class to the core. This classification helps in optimising the decision-making in the disassembly process and helps to understand the rework and random failures of components at inspection better. Example for specifying quality classes:

- Quality Class 1: $0 \leq \overline{\lambda} \leq 0.1$
- Quality Class 2: $0.1 < \overline{\lambda} \leq 0.4$
- Quality Class 3: $0.4 < \overline{\lambda} \leq 0.8$
- Quality Class 4: $0.8 < \overline{\lambda} \leq 1$

The value q related to quality can be expressed as follows:

$$q \in \{1,2,\ \ldots,\ l\} \tag{7}$$

An additional variable that is calculated directly during disassembly and is not based on historical data regarding damage parameters is the failure rate $\lambda(t)$. In the context of disassembly, the failure rate $\lambda(t)$ in general can be calculated as the ratio of the number of failed components $F(t)$ to the total number of components disassembled $N(t)$ within a specified time period or process step:

$$\lambda(t) = \frac{F(t)}{N(t)} \tag{8}$$

The failure rate $\lambda(t)$ during disassembly is a critical metric in decision-making. $\lambda(t)$ can be calculated at the core level or at the component level, depending on the specific interest.

In remanufacturing systems, understanding and managing process times is critical for optimising the overall lead time T_{total}. T_{total} is the total time required to complete the remanufacturing process. This includes: (1) pre-cleaning, (2) analysing, (3) disassembling, (4) component inspection and (5) rework. T_{total} can be expressed as the sum of the average process times at each workstation ws ($\overline{T_{ws}}$):

$$T_{total} = \sum\nolimits_{ws=1}^{m} \overline{T_{ws}} \tag{9}$$

where m is the total number of workstations in the remanufacturing system. Variability in process times due to factors such as the condition of returned products, equipment efficiency, and workforce skill levels must be considered to provide accurate lead time estimates.

The planned delivery time for an order j (DT_j) is important for ensuring customer satisfaction and maintaining efficient operations. DT_j is determined by T_{total} and C_j. Furthermore, delays in preceding orders can have a cascading effect on the delivery times of subsequent orders, potentially causing both negative and positive impacts. Given C_j and T_{total}, DT_j can be expressed as:

$$DT_j = \left(T_{total} * C_j\right) + \Delta T_{previous_j} \tag{10}$$

where $\Delta T_{previous_j}$ is the time, the order j spends waiting before processing starts. The formula presented is based on a sequential process. If parallel processing is feasible, the formula should be modified accordingly.

The cost structure of remanufacturing is distinct from traditional manufacturing due to the unique processes involved in rework used products to like-new condition. A comprehensive cost model for remanufacturing incorporates various elements, including the costs associated with each individual process step. The general remanufacturing costs for a core (C^r) based on the quality q are calculated using the additional disassembly times (T_{total}), the labour required (C_{labour}), the

failure costs (C_f) and estimated probability of failure ($\overline{\lambda}$).

$$C^r = (T_{total} * C_{labour}) + \left(\overline{\lambda} * C^f\right) \tag{11}$$

Furthermore, the costs can be detailed at the component level (C_i^r):

$$C_i^r = (T_i * C_{labour}) + \left(\overline{\lambda_i} * C_i^f\right) \tag{12}$$

where T_i is the processing time needed only for the component i, $\overline{\lambda_i}$ and C_i^f the failure probability and cost for the individual component i. The cost advantage for each a remanufactured component, which can be expressed as the value of the remanufactured component i (V_i^r), can be calculated with the costs for a new component (C_i^n):

$$V_i^r = C_i^n - C_i^r \tag{13}$$

By optimising the remanufacturing processes, you have the possibility to maximise the cost savings.

The total value of the core (V_c^r) can then be calculated as follows:

$$V_c^r = \sum_{i=1}^{n} V_i^r \tag{14}$$

The main goal of this paper is to identify and implement an optimisation strategy that will address two conflicting objectives, each of which has the potential to have a negative impact on the other.

The following part of the section will present the two principal key performance indicators (KPIs) and the respective optimisation objectives.

- On-time delivery (OTD) serves as a crucial metric for assessing the dependability of order processing and delivery. In general, it measures the percentage of orders produced within the agreed-upon or scheduled delivery time. It serves as a gauge of operational efficiency within production processes. OTD is calculated as follows:

$$OTD = \left(\frac{\text{Number of On-Time Deliveries}}{\text{Total Number of Deliveries}}\right) * 100\% \tag{15}$$

  The OTD is measured over all orders completed in the remanufacturing system. A high OTD rate indicates that the production consistently meets its delivery commitments, while a low OTD rate may suggest issues in the supply chain, such as production delays or inadequate stock management.
  To reach an overall high OTD rate, the optimisation goal is to minimise the delivery time for an order j (FDT_j), which is the time the order j needs through the complete disassembly system. The optimisation objective can be represented as follows:

$$\text{minimise} FDT_j \tag{16}$$

- The Service Level (SL) generally measures the ability of the production to meet the target stock level for an order j. The SL of component i for an order j at time t ($SL_{i_j}(t)$) is here defined as the division of the actual stock value for component i at time t ($S_i(t)$) by the target stock value of component i for an order j ($\overline{S_{i_j}}$). $\overline{S_{i_j}}$ does not change with time t (see formula (3)). $SL_{i_j}(t)$ can be calculated using the following formula:

$$SL_{i\,j}(t) = \frac{S_i(t)}{\overline{S_{i_j}}} \tag{17}$$

  Ensuring alignment between the $S_i(t)$ and $\overline{S_{i_j}}$ is essential. Due to the uncertain core quality and quantity, $S_i(t)$ may exceed $\overline{S_{i_j}}$ at any time t. The objective is to minimise the difference between $\overline{S_{i_j}}$ and $S_i(t)$:

$$\text{minimise}\left|\overline{S_{i_j}} - S_i(t)\right| \tag{18}$$

  The objective optimisation for SL is simplified in this representation, while an asymmetric consideration (e.g., overstock vs. understock) is appropriately accounted for in the reward function.

It is important to understand why these two KPIs are crucial in remanufacturing. The quality of the cores for an order j determines FDT_j, which has a significant impact on overall OTD. Furthermore, poor quality negatively affects SL_{i_j}, leading to the scrapping of random components due to damage.

All decisions made by the RL agent types affect FDT_j. In the disassembly process the steps are complex and variable. If a component or a core is not intended for remanufacturing, faster disassembly becomes possible as there is no need to carefully handle connections or account for potential damage during the process. It reduces FDT_j at the end, because less time is needed for disassembly.

Figure 4 shows the connection between FDT_j, SL and OTD.

The schematic figure helps to visualise and to understand the complex connection between the different KPIs. The uncertainties described in Sect. "Introduction", specifically core quality and variable processing times, significantly influence T_{total} and thus affecting FDT_j. The dashed red line illustrates the threshold at which the overall OTD will be negatively influenced by order j. This occurs when the FDT_j exceeds the DT_j, as shown in Fig. 4 ($FDT_j > DT_j$).

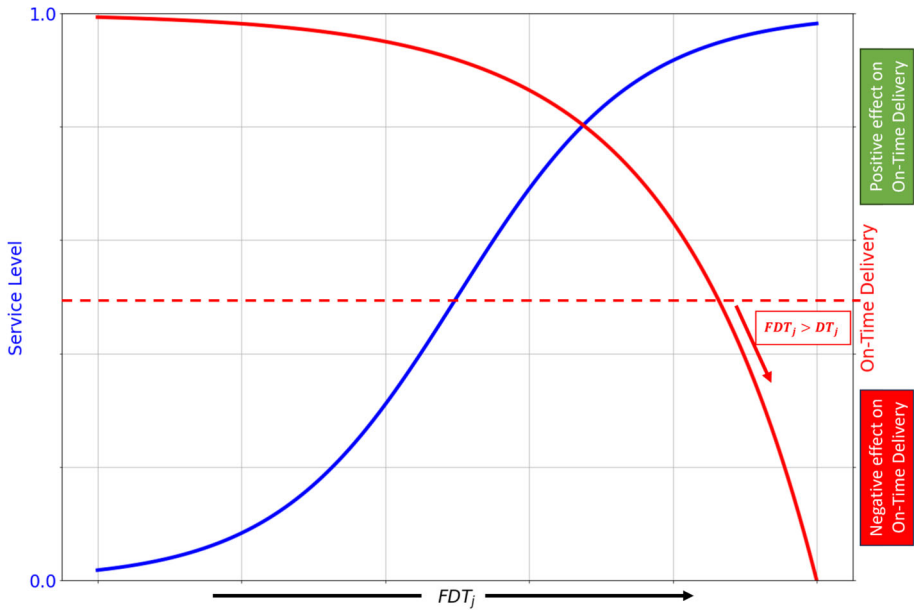

Fig. 4 Schematic figure showing the relationship between FDT_j, Service Level, and On-Time Delivery

To illustrate, in the scenario where the target SL_{ij} is not met with the planned C_j, the option to replan and disassemble additional cores is available in order to reach the target. However, this will have a negative impact on FDT_j, given the increased time required to complete the order j. Wherein a higher FDT_j will result in SL_{ij} approaching 1, while the OTD will be influenced negatively. In practice, the dashed red line in the illustration can be adjusted based on the specific circumstances of the individual or entity in question. It should be noted that this is simply a schematic representation of the threshold value beyond which the entire OTD is positively or negatively influenced by order j.

Influencing FDT_j is a key factor for overall OTD and, SL performance. Optimised FDT_j can reduce costs and increase efficiency and customer satisfaction. These factors are therefore critical in determining a cost-optimal disassembly system. As mentioned before, the RL agent influences FDT_j, and optimises the two objectives.

RL approach

The use of integer optimisation-based approaches is often limited by the simplifying assumptions they make, which are not suitable for real-time applications. Accordingly, we employ RL to address this issue approximately, thereby facilitating real-time decision-making. In the context of RL, the state vector $x(t)$ is replaced by the state $s(t)$ of the RL agent. The state vector $s(t)$ comprises all pertinent information that characterizes the present status of the production system.

In general, RL can obtain the optimal policy by learning from the interaction with the environment. The RL problem is modelled as a Markov Decision Process (MDP) with a tuple $< S, A, P, R, E >$ (Sutton & Barto, 2020):

- S: set of all states.
- A: set of executable actions of the agent.
- P: transition distribution, $P^a_{ss\prime} = P(S_{t+1} = s\prime | S_t = s, A_t = a)$.
- R: reward function, and r_t represents the reward obtained after taking an action at time t.
- E: set of states that have been already reached.

RL does not require a model of the dynamics of the environmental system. It is called model-free. The RL agent interacts with the environment in a closed-loop and learns to solve the underlying MDP optimally. Based on the state $s_t \in S$ of the environment observed at the time t, the RL agent chooses an action $a_t \in A$ according to its strategy π. Choosing this action transforms the environment into a subsequent state s_{t+1} of the environment. The RL agent receives a reward r_{t+1} that depends on the observed state s_t, the action a_t, and the next state s_{t+1}, according to the underlying MDP. Through repeated interaction with the environment, the RL agent's algorithm aims to maximise the cumulative reward. Based on the reward, the RL agent adjusts its strategy π. The strategy $\pi*$ that maximises the cumulative reward is called the optimal strategy, which optimally solves the MDP according to the given reward function $R(S_t, A_t, S_{t+1})$, defined by the reward r_{t+1} (Sutton & Barto, 2020).

In contrast to value-based methods, such as Deep-Q-Learning (DQN), there is a category of algorithms that learn the policy directly. For instance, Proximal Policy Optimization (PPO) extracts the policy from action-values. The use

Table 1 Selected hyperparameters for the PPO RL agents

Hyperparameter	Value
Minibatch size	128
Discount factor	0.99
Clip range	0.2
Train batch size	1000
Learning rate	1e-5
Neural network – hidden layers	512, 256, 128, 64, 32
Activation	ReLu
Optimizer	Adam

The pseudocode for the PPO algorithm is presented in Fig. 5

of the PPO algorithm results in enhanced sample efficiency and robustness when utilizing policy-based approaches. For decision making in the disassembly process, we have opted for PPO, as explained in detail in reference Schulman et al. (2017). Among the available reinforcement learning algorithms that are designed to process continuous state spaces while making discrete decisions, PPO stands out due to its computational simplicity and robustness in dynamic environments. Unlike Advantage Actor-Critic (A2C), which relies on synchronous updates and faces challenges with scalability, PPO employs a clipped surrogate objective function that stabilises training and prevents policy updates from diverging. Compared to Trust Region Policy Optimisation (TRPO), PPO achieves a similar level of stability without the computational overhead associated with constrained optimisation, making it more efficient for large-scale systems. Furthermore, PPO's balance between exploration and exploitation allows it to adapt effectively to uncertainties, such as varying core qualities and disassembly depths, which are characteristic of remanufacturing processes (Mnih et al., 2016; Schulman et al., 2017). Figure 10 presents the training performance comparison between PPO, DQN, and A2C. In this study, approaches based on purely continuous action spaces were not considered, as the disassembly decision-making process inherently requires discrete actions. While methods like Soft Actor-Critic (SAC) and Deep Deterministic Policy Gradient (DDPG) are well-suited for continuous control problems, our focus remains on reinforcement learning algorithms that efficiently handle discrete decision-making in dynamic environments.

Table 1 presents a brief overview of the selected hyperparameters. The hyperparameters were optimised by preliminary experiments. The agent modules implement the RL algorithms and their respective parameter configurations. While some RL algorithms require episodes for optimal strategy implementation, defining fixed episodes is not feasible for continuous manufacturing processes and represents a different modelling design option. This paper presents a departure from the established applications of RL in board or computer games by employing action-based definitions. The definition used for episodes is grounded in the number of performed actions. It is worth noting that the decisions of an RL agent ahead may impact the number of actions taken by an RL agent behind them.

Table 1 displays the selected hyperparameters of the RL agent types.

As mentioned earlier, we have created two types of RL agents. The corresponding RL agent type is activated (refer to Fig. 1 for more information), and a request for decision is made. Subsequently, the decision is executed, and the RL agent obtains a reward. Two RL agent types, namely the core agent and the component agent type, are considered:

- Core agent type: This type of RL agent makes all decisions that affect the entire core and thus all components that have not yet been disassembled. If the chosen action is to 'scrap', all remaining components within the core will not be reworked.
- Component agent type: This type of RL agent makes every decision that affects a single component. I.e., it does not impact the other components or the whole remaining core. The decision only affects the component being currently disassembled.

This part of the section presents the **state space** collected from the environment and provided as measurements to the corresponding RL agent types.

- Failure rate $\lambda(t)$ of each component i in order j at time t: $\lambda_{i_j}(t)$
- Service level of each component i for order j at time t: $SL_{i_j}(t)$
- Proportional monetary value of component i (V_i^p) is a metric used to compare each component to the total value of the core. The higher V_i^p, the greater the advantage gained from reusing the component i.

$$V_i^p = \frac{V_i^r}{V_c^r} \tag{19}$$

- Remaining delivery time for the current order j at time t ($TL_j(t)$) can be calculated using the values DT_j, the start time of processing for order j (t_j) and the current time t. The value t_j is e.g., minutes from the beginning of the shift the order j was started.

$$TL_j(t) = \frac{DT_j - (t - t_j)}{DT_j} \tag{20}$$

- Proportional remaining cores for order j at any time t ($CL_j(t)$) can be calculated with C_j and the number cores

Fig. 5 Pseudo code proximal policy optimization algorithm (Schulman et al., 2017)

Algorithm 1 PPO, Actor-Critic Style

```
for iteration=1, 2, ... do
    for actor=1, 2, ..., N do
        Run policy π_θold in environment for T timesteps
        Compute advantage estimates Â_1, ..., Â_T
    end for
    Optimize surrogate L wrt θ, with K epochs and minibatch size M ≤ NT
    θ_old ← θ
end for
```

that have already been disassembled for order j (C_j^d). This leads to the following formula:

$$CL_j(t) = \frac{C_j - C_j^d(t)}{C_j} \tag{21}$$

- Core(s) in buffer in front of disassembly station at time t ($C_{que}^{dis}(t)$) indicates that at time t, there is at least one core in the buffer before the disassembly station. The number of cores present in the buffer area in front of disassembly station at time t ($B_{dis}(t)$) is necessary for this state.

$$C_{que}^{dis}(t) = \begin{cases} 1, \, if \, B_{dis}(t) \neq 0 \\ 0, \, if \, B_{dis}(t) = 0 \end{cases} \tag{22}$$

This state is related to the remaining time for order j.

Relative values were chosen over absolute values for certain state values to facilitate interpretation by the RL agent types. This decision resulted in improved outcomes during the simulation. The monetary value is not limited to representing economic metrics such as purchase price. If additional information – such as energy consumption or environmental impact – is available, it can be included to refine the value further. This comprehensive approach allows the monetary value to reflect a broader range of factors, enhancing its role as an indicator of component importance.

We now present the **reward function**. We defined two general functions that address both optimisation goals SL_{i_j} and OTD. These two functions are then integrated into the reward function of the individual RL agent type separately. As a result, the optimisation goals are in conflict with one another, making the formulation of an appropriate reward structure a challenging task. Through a process of meticulous manual fine-tuning, guided by the performance of the models, we have developed the following reward structure.

Figure 6 shows the structure of the reward function for each RL agent type.

Key elements of the reward function include:

- Growth Rate k: Determines the steepness of the SL reward curve $R_{SL_{i_j}}(t)$, enabling sensitivity adjustments for various use cases. The inflection point (IP) aligns the function's growth with critical thresholds in the decision-making process.
- $R_{SL_{i_j}}(t)$ incorporates V_i^p, to gain a higher reward for components with a higher monetary value.
- k, IP, γ and β need to be adjusted to fit the specific use case. Their functions are responsible for either positively amplifying rewards or progressively increasing punishments.
- Selection of k, IP and γ also affects the response of the RL agent types to changes in the state.
- $R_{OTD_j}(t)$ varies depending on whether the scheduled time has already elapsed or whether the task is still being carried out within the time frame.

For the **component agent type**, the service level reward function $R_{SL_{i_j}}(t)$ evaluates only the component currently undergoing disassembly and requiring a decision. Other components are excluded from the decision-making process at this stage. The priority of an optimisation goal is determined individually using a weighting factor w_f. To prevent the perception of an objective being disproportionately affected by an order j with a high number of cores, the reward is normalised by dividing it by the defined $\overline{S_{i_j}}$.

In contrast, the **final reward function of the core agent type** differs as it does not consider individual component values. Instead, it measures the overall service level by averaging the remaining components – those not yet disassembled and still subject to future decisions. Here, $i(t)$ refers to the current stage of disassembly, where $i(t) = 2$ indicates the second component is being disassembled. The variable n represents the total number of components intended for remanufacturing. The final reward function$R_c(t)$, incorporates c, the number of decision points for the core agent, and $n(t)$, the number of components yet to be disassembled at time$t$. For instance, $n(t) = 3$ implies three components remain for disassembly after the first core decision.

The cumulative reward function $R_f(t)$ for the comparison outlined in Sect. "Results and discussion" is obtained by combining the reward functions for the component agent type ($R_i(t)$) and the core agent type ($R_c(t)$):

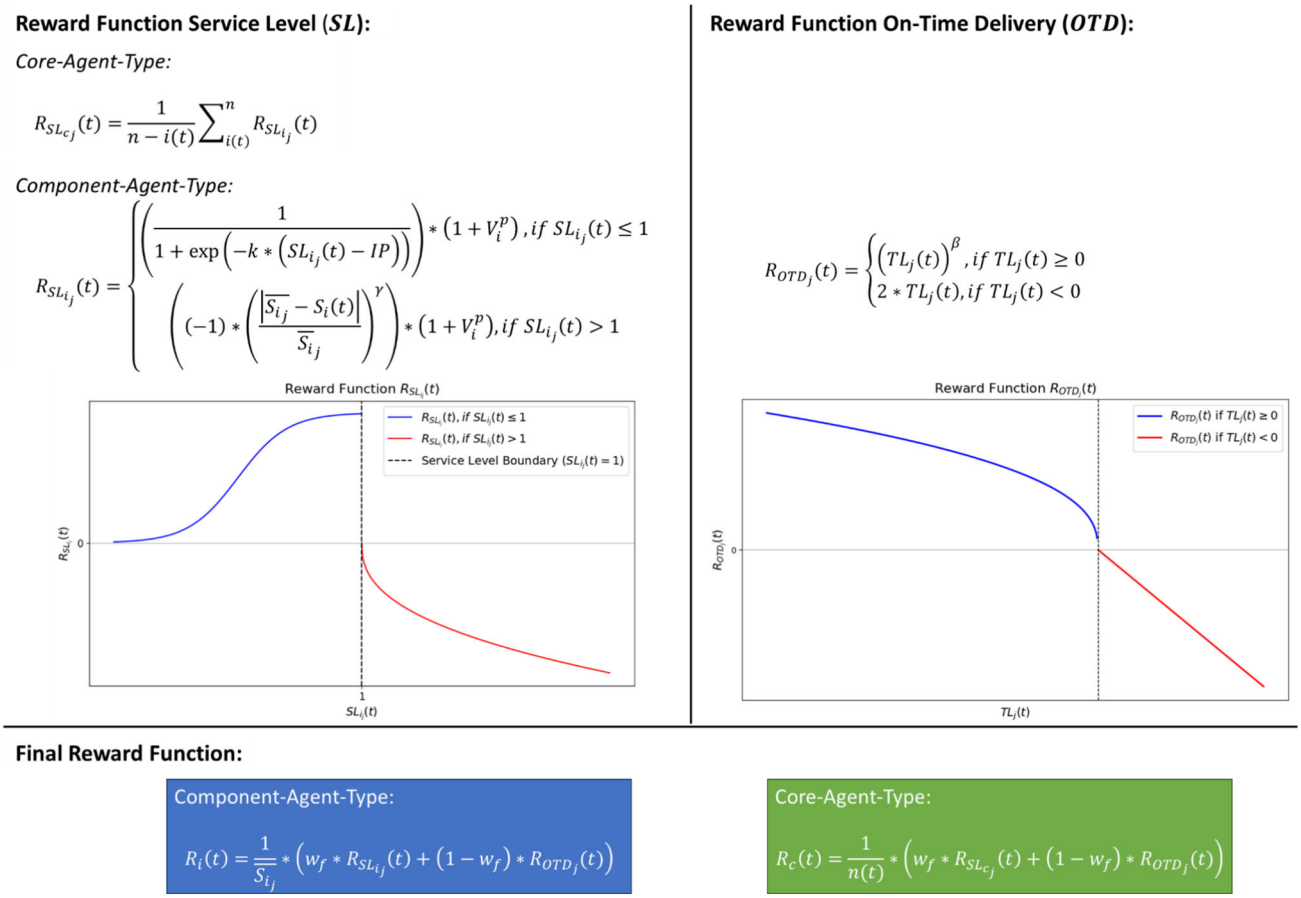

Fig. 6 Structure of the reward function for each RL agent type

$$R_f(t) = \sum_{i=1}^{n} R_i(t) + \sum_{c=1}^{n-1} R_c(t) \tag{23}$$

The variable $n-1$ represents the number of decision points that the core agent type must consider. As mentioned, the core agent type makes one decision less than the component agent type because the component agent type decides on the last component in the core, making a core decision unnecessary.

The decision regarding disassembly presents two options in the **action space**. Both core and component agent types have two available actions, resulting in the action space A.

$$A = \{disassemble, scrap\} \tag{24}$$

The actions of 'disassemble' and 'scrap' should be evaluated separately for both types of RL agents. If the core agent type selects the 'disassemble' option, it indicates a decision to continue considering the core for disassembly. However, if the chosen action is 'scrap', the entire core, including all remaining components, will not be reworked.

If the component agent type selects 'disassemble', the component is disassembled and inspected for damage at the next workstation. It is important to note that the selection of 'disassembly' does not guarantee that the component is free of damage. As previously stated, the components will be inspected after disassembly and may be scrapped due to damage. Consequently, the decision to label a component as 'scrap' solely applies to that specific component and not to other components for the same component agent type.

Practical case study and validation of the RL approach

The experiment models a disassembly system, which includes three test environments: (1) a single disassembly station, (2) parallel disassembly stations, and (3) a disassembly line. Additionally, the model represents the stations pre-cleaning and analysing D_0, disassembly, inspection and rework.

Figure 7 illustrates the structure of the disassembly test environments. The green arrow represents the decision 'disassemble', while the red arrow symbolizes the decision 'scrap'.

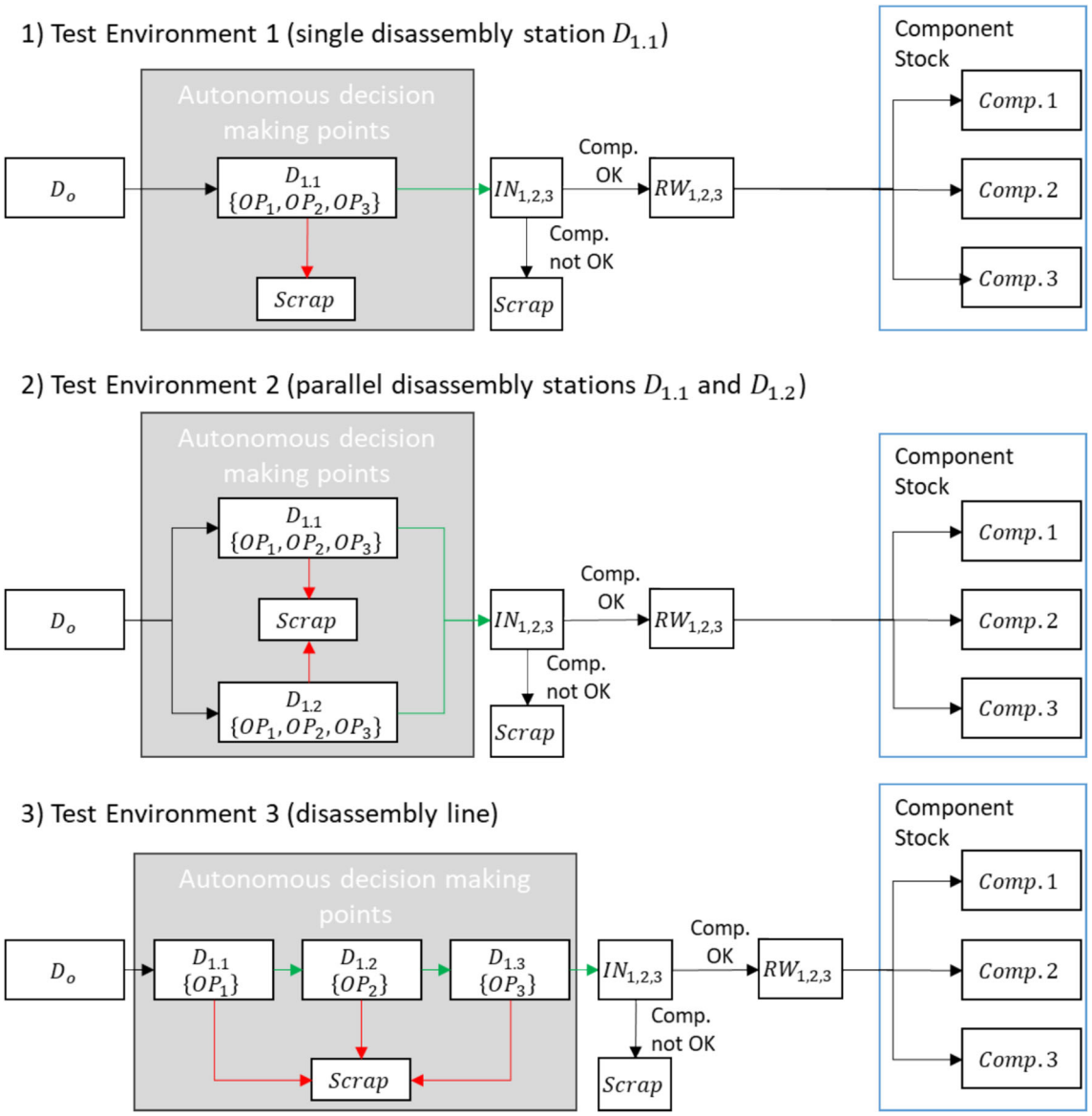

Fig. 7 Structure of test environment 1, 2 and 3

The details of the test environments will be provided subsequent to the discussion of the general topics. Within each disassembly station, e.g. $D_{1.1}$, operations denoted as OP_i are listed below, representing the tasks performed at the respective disassembly station. Preceding each operation OP_i are two decision points, see Fig. 1. Prior to the final operation (OP_3), only a decision point for the component agent type exists. It should be noted that at this decision point, no further components are to be considered for disassembly. D_0 represents the pre-cleaning and analysis station. The analysis station is responsible to estimate $\bar{\lambda}$ and assigns q to the core/component. And the arrows show the material flow as well as the decision by the RL agent types through the individual workstations. $IN_{1,2,3}$ is the inspection and $RW_{1,2,3}$ is the rework of the components, which are then transported to one of the component stocks or will be scrapped, if they are not reusable.

A detailed explanation of the test environments is provided below:

(1) Disassembly test environment with a single disassembly station $D_{1.1}$

- At disassembly station $D_{1.1}$, a single worker will completely disassemble the core, performing operations OP_1, OP_2 and OP_3. Consequently, other cores before $D_{1.1}$ must wait until the current core is fully disassembled, or until the RL agent decides to stop the operation. In this test environment, the worker

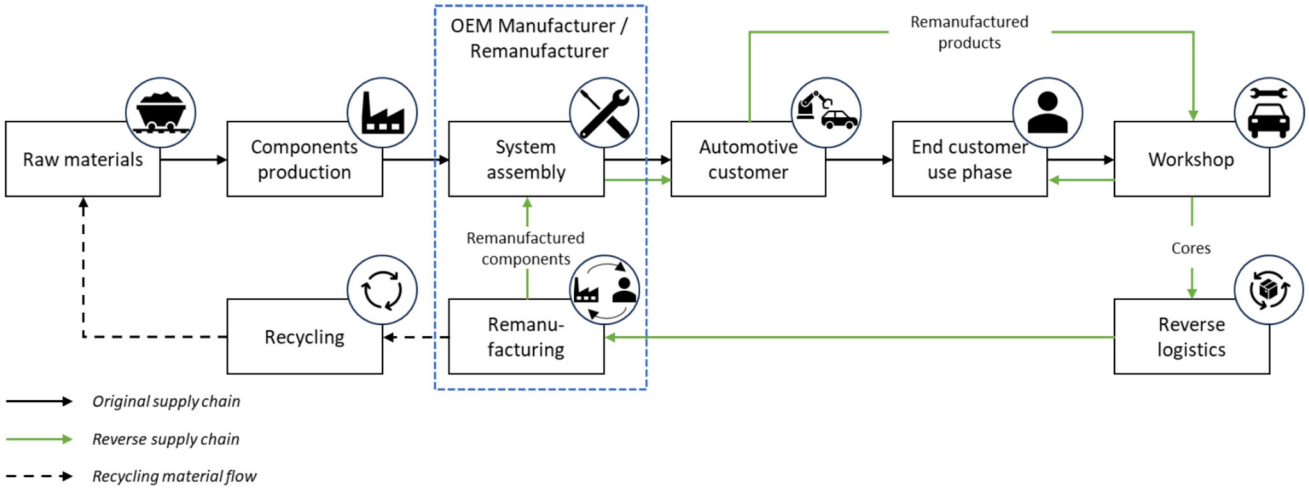

Fig. 8 OEM remanufacturer´s circular supply chain

will require more time to disassemble all the cores, and the state space will be updated without any delay caused by parallel decision-making. The RL agent types were trained in this test environment.

(2) Disassembly test environment with two parallel disassembly stations $D_{1.1}$ and $D_{1.2}$

- In this test environment we will duplicate the single disassembly station $D_{1.1}$, and assign a second worker to disassemble another core in parallel at $D_{1.2}$. The main difference from test environment 1 is that the state space update will be delayed or abrupt because of the parallel decision-making, resulting in an uncertainty.

(3) Disassembly test environment with a disassembly line

- In this test environment, three disassembly stations $D_{1.1}$, $D_{1.2}$ and $D_{1.3}$ are arranged in sequence. Each station is responsible for a specific disassembly operation: OP_1, OP_2 or OP_3. After a station completes its designated disassembly operation, the core is transferred to the next station. However, this transfer is contingent on the decision of the core agent type, which will only initiate the move when it decides to disassemble the next component. This setup will accelerate the disassembly process.

To evaluate the developed decision model, a practical case study was conducted at an OEM remanufacturer for electric power steering (EPS) systems. This application was chosen due to the high complexity of disassembly processes in EPS remanufacturing. EPS systems comprise various mechanical and electronic components, such as the steering control unit (SCU), ball nut assembly, and sensor unit (SU), whose condition significantly influences the disassembly strategy.

The company operates within a closed-loop supply chain, which includes both the original supply chain for manufacturing EPS units for new vehicles and a reverse supply chain for remanufacturing EPS units at the end of their life for the automotive aftermarket. Additionally, the recycling material flow ensures the return of materials to the raw material stream. Despite the profitability of remanufacturing, the OEM remanufacturer prioritises meeting its delivery obligation period, ensuring component availability after series production ends. Within this process, the remanufacturer operates in a collaborative three-party framework, working closely with both the reverse logistics supplier and the automotive customer to facilitate efficient operations.

Figure 8 illustrates the circular supply chain of the OEM remanufacturer.

The data collection process combines real production data with simulated disassembly scenarios. Observations from the actual remanufacturing environment – including processing times, disassembly steps, and quality data of returned EPS systems – were incorporated into the simulation to replicate real-world conditions as accurately as possible. By integrating real process data with simulated learning, the RL approach can be comprehensively evaluated to ensure alignment with practical constraints and industry requirements, forming a robust foundation for assessing its effectiveness in industrial applications.

Through simulation, the RL agent types were exposed to a wide range of variations in the state space across numerous training episodes. This extensive exploration enabled the RL model to adapt to diverse disassembly conditions and optimise its decision-making strategy.

To support the transition from simulation to real production, potential discrepancies between simulated and actual production environments can be mitigated by applying a low learning rate. This adjustment allows the agent to adapt to

evolving process conditions while maintaining stable and reliable decision-making.

Scalability is a crucial aspect of the RL approach, especially when applied to various remanufacturing systems. Remanufacturing environments can vary significantly in terms of complexity, product types, and operational processes. A scalable RL solution ensures that the agent can adapt to these diverse conditions, handling different types of components and quality classes efficiently. As the remanufacturing system expands or changes, the RL agent must be able to scale its operations without compromising performance. We demonstrate this scalability with our different test environments and defined scenarios. In these scenarios, the RL agent types do not need to communicate directly with each other; instead, changes in the state space alone are sufficient to inform decision-making. This independence highlights the RL approach's ability to efficiently manage complex remanufacturing systems by relying solely on state space updates, ensuring integration and operation across various system configurations.

As mentioned, this study explores different combinations of RL agent types that primarily focus on decision-making but differ in their level of decentralization. The action space and the reward function are the same for all combinations and will not be mentioned separately.

The six combinations are as follows:

1st combination: 1 core agent & 1 component agent

In each case, the following states are observed: a single core agent responsible for all core decisions, and a separate component agent accountable for all component decisions. At the third disassembly step, only the component agent determines the fate of the final component, making any extra decision-making by a core agent unnecessary. This is applicable to all combinations. The state space of RL agents is defined as follows:

- ***Core Agent (12 states)*** $\lambda_i(t)$, $SL_{i_j}(t)$, V_i^p, $TL_j(t)$, $CL_j(t)$, $C_{que}^{dis}(t)$ $fori = 1,2,3$; $j = 1,2,\ldots,N$
- ***Component Agent at the decision point preceding*** OP_1 ***(6 states)*** $\lambda_1(t)$, $SL_{1_j}(t)$, V_1^p, $TL_j(t)$, $CL_j(t)$, $C_{que}^{dis}(t)$ $forj = 1,2,\ldots,N$
- ***Component Agent at the decision point preceding*** OP_2 ***(6 states)*** $\lambda_2(t)$, $SL_{2_j}(t)$, V_2^p, $TL_j(t)$, $CL_j(t)$, $C_{que}^{dis}(t)$ $forj = 1,2,\ldots,N$
- ***Component Agent at the decision point preceding*** OP_3 ***(6 states)*** $\lambda_3(t)$, $SL_{3_j}(t)$, V_3^p, $TL_j(t)$, $CL_j(t)$, $C_{que}^{dis}(t)$ $forj = 1,2,\ldots,N$

It should be noted that in the first step, the core agent receives all 12 states, but at the second decision point, the values of component 1 are irrelevant and are therefore set to 0. Depending on the decision point before the individual OP_i, the component agent obtains a total of six states, corresponding to the component currently being disassembled.

2nd combination: 2 core agents & 1 component agent

In this combination, a separate core agent takes action at each of the two core decision points before OP_1 and OP_2. The state space of RL agents is defined as follows:

- ***Core Agent 1 (12 states)*** $\lambda_i(t)$, $SL_{i_j}(t)$, V_i^p, $TL_j(t)$, $CL_j(t)$, $C_{que}^{dis}(t)$ $fori = 1,2,3$; $j = 1,2,\ldots,N$
- ***Core Agent 2 (9 states)*** $\lambda_{2,3}(t)$, $SL_{2,3_j}(t)$, $V_{2,3}^p$, $TL_j(t)$, $CL_j(t)$, $C_{que}^{dis}(t)$ $forj = 1,2,\ldots,N$
- ***Component Agent (6 states)***: The same as in combination 1.

3rd combination: 1 core agent & 3 component agents

The 3rd combination involved implementing a centralised core agent with decentralised component agents. The state spaces only change for the component agents, and each individual component agent observes the state of the component for which they are responsible. The state space of the core agent remains the same as in the first combination. For instance, the state space of the component agents is:

$\lambda_i(t)SL_{i_j}(t)$, V_i^p, $TL_j(t)$, $CL_j(t)$, $C_{que}^{dis}(t)$ $fori = 1,2,3$; $j = 1,2,\ldots,N$

4th combination: 2 core agents & 3 component agents

The 4th combination represents the most decentralised approach. In this method, a separate RL agent is responsible for making decisions for both the core and the component at every point. The state space of the core agent is equivalent to that of combination 2, while that of the component agents corresponds to combination 3.

5th combination: 3 component agent

The 5th combination consists of three component agents and therefore one of the agents makes the decision for its component only. The decisions of the core agent type are disabled for this combination, meaning that the decision for the core is always 'Disassemble' and the action is taken into account in the reward to ensure comparability of results. The state space corresponds to combinations 3 and 4.

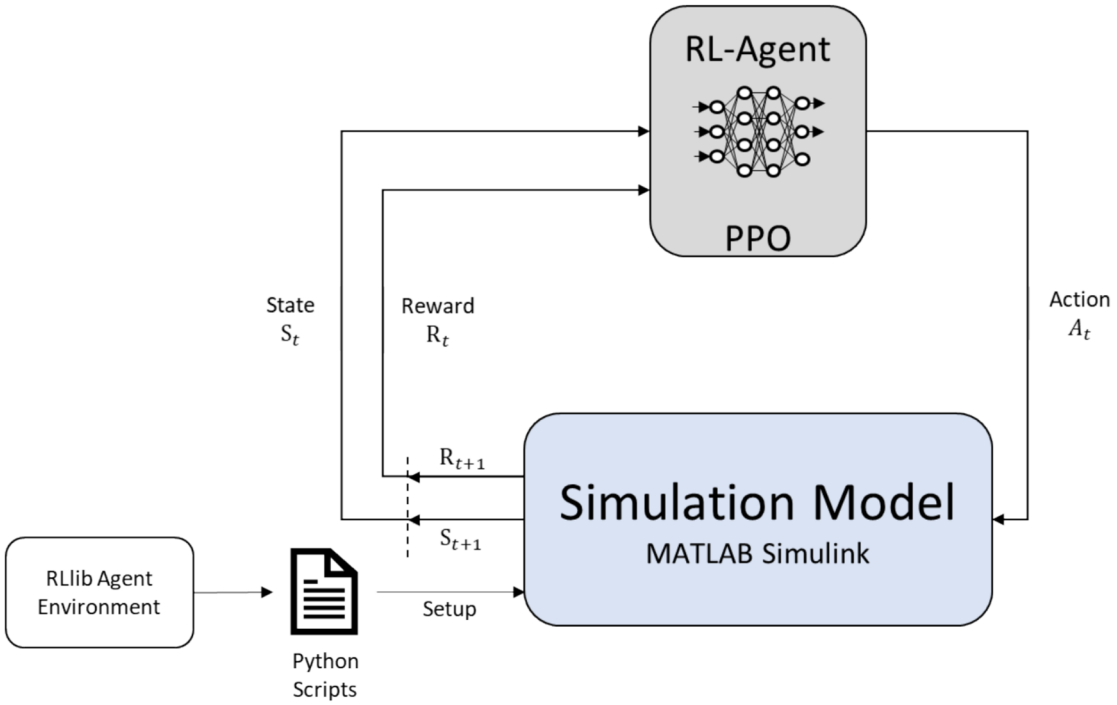

Fig. 9 Software implementation—interactions between different system components

6th combination: 1 component agent

The 6th combination includes a component agent that makes decisions for each individual component decision point. In this case, the decision for the core is also deactivated, similar to the 5th combination. The state space is the same as that of the component agent in combinations 1 and 2.

The 5th and 6th combinations are used to test whether the core agent type is necessary or whether the component agent types are sufficient for decision-making.

The system comprises a simulation model, created using MATLAB (Version 2020b) and Simulink, which provides a virtual representation of the disassembly system. The RL agents are trained and tested using the RLlib library, while an API library in Python is used to establish communication between the simulation and the RL agents. The simulation model requests a decision when necessary and provides rewards immediately after-action execution. As previously mentioned, real production data from the practical use case was incorporated to parameterise the model, ensuring alignment with actual manufacturing conditions.

Figure 9 illustrates the interaction between each component.

The simulation comprises several processes, including an order creation process that simulates the arrival of orders at the source. New orders are generated including the target stock and the IR used. Each core of the generated order is randomly assigned a quality class at the analysing station, simulating the uncertainty in core quality. The probability of failure associated with each quality class was derived from real production experience, ensuring a realistic representation of failure rates. The implementation prioritised clear modularisation and structuring through object-oriented programming. Individual modules can be easily replaced or expanded, to adapt the simulation model to changes in the real production environment. A timeline of events is established through the use of discrete event simulation (DES).

We used a Windows-based devise with an AMD Ryzen 7 5700X 8 core processor with 3.4 GHz and a RAM memory of 32 GB, and an SSD for storage. A powerful computer should be used for implementation in a real disassembly system.

As scalability plays a crucial role in real-world remanufacturing applications, the computational complexity of training the RL agent is a critical consideration for understanding the resources required and the feasibility of the approach. In this study, the RL agent was trained over 1.5 million episodes with changing values per episode: normally distributed target stock $\overline{S_{ij}}$ (mean 50 and standard deviation 20) and a uniformly distributed IR between 1.3 and 1.7. The RL agents were trained in their combinations in test environment 1 (see Fig. 7). The training process, executed on the specified hardware configuration, extended over a duration of 10 days. Throughout the training, RAM memory usage was a significant constraint. The training required a storage space of 4 GB to save progress. The computational complexity analysis evaluates how the decision-making process scales with an

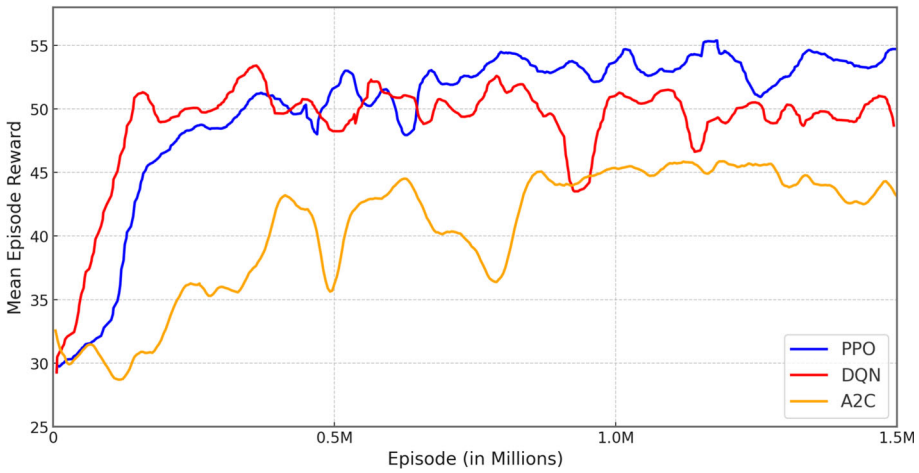

Fig. 10 Training performance comparison between PPO, DQN and A2C

increasing number of components, with the results detailed in Sect. "Results and discussion" (scenario 4).

To provide a basis for comparison, additional methods will be considered. The DQN algorithm was selected for comparison with the PPO algorithm. The DQN agent is based on the 6th combination and is trained in a similar manner to the PPO agents. The next method is the basic heuristic, which involves disassembling the components without considering the state space. The final method is the One Disassembly heuristic, which involves disassembling the components until the stock value of the individual component reaches the target stock level, after which the remaining components are scrapped. This One Disassembly heuristic is designed to achieve the defined service level optimisation goal, whereby the different RL combinations must perform better in terms of on-time delivery. The comparisons are based on the reward function, using the average reward over 10 episodes to ensure a stable assessment and minimize the influence of outliers.

Figure 10 illustrates the training performance comparison between the PPO (6th combination), DQN and A2C algorithms (based on the 6th combination).

Based on the training performance, we decided to take the PPO and the DQN algorithm for our case study and not to further pursue the A2C algorithm. The exclusion of A2C from further testing is due to its inability to show significant performance improvements in our environment.

To evaluate the effectiveness of decision making in the disassembly process, we defined five scenarios. The first two scenarios will be simulated and compared in the various test environments with the defined purposes and are designed to identify the optimal PPO agent combination, which will then be employed in the subsequent three scenarios. Subsequently, scenarios 3 to 5 will be simulated and compared using only test environment 1. The objective of the three last scenarios is to assess the impact of varying control parameters, the stability and variability across a broader range of episodes, and the scalability of the RL approach, as illustrated in scenario 5. In this instance, the optimal PPO combination derived from the two scenarios will be employed, with the DQN and additional heuristics utilised to facilitate a comparative analysis of the outcomes. Each combination of PPO agents, the DQN agent and the other heuristics, are evaluated through the mean final cumulative reward achieved which is calculated by a sample of ten simulation replications per scenario with varying seeds to account for stochasticity. The overall increase in system efficiency is presented in Sect. "Results and discussion".

The first scenario entails a comparative analysis of the various PPO combinations with the DQN agent, the basic heuristic, and the One Disassembly heuristic, across the three specified test environments.

The following variables are defined as basic for the purposes of each simulation, when there is no other definition mentioned:

- $\overline{S_{i_j}}$ mean 50 and standard deviation 20.
- Uniformly distributed IR between 1.3 and 1.7.
- Probability of failure and related quality class ($\overline{\lambda} = 5\%$ ($q = 1$), $\overline{\lambda} = 20\%(q = 2)$, $\overline{\lambda} = 40\%(q = 3)$) and quality class related variable processing times (Table 2).
- 3 components intended for remanufacturing.
- The relevant processing times are presented in Table 2.

In the second scenario, only cores of quality class 3 will be available. The presence of poor-quality cores will result in increased processing times and higher failure rates. This scenario will demonstrate the impact of poor-quality cores on the reward.

In the third scenario, we analyse the influence of changing control parameters and validate the effect on the reward. The following control parameters will be changed:

a. Changing processing times (see Table 5 and Table 6).
b. Changing monetary values.

Table 2 Processing time PT_i^q disassembly station for each component i and based on q

Disassembly step	Processing Time $\boldsymbol{PT_i^q}$ (mean, standard deviation)
Disassembly Component 1	$PT_1^1 = (70s, 2s)$ *per Component*
	$PT_1^2 = (100s, 5s)$ *per Component*
	$PT_1^3 = (140s, 10s)$ *per Component*
Disassembly Component 2	$PT_2^1 = (120s, 2s)$ *per Component*
	$PT_2^2 = (150s, 5s)$ *per Component*
	$PT_2^3 = (190s, 10s)$ *per Component*
Disassembly Component 3	$PT_3^1 = (40s, 2s)$ *per Component*
	$PT_3^2 = (50s, 5s)$ *per Component*
	$PT_3^3 = (90s, 10s)$ *per Component*

c. Increased probability of failure: $\overline{\lambda} = 10\%(q = 1)$, $\overline{\lambda} = 30\%(q = 2)$, $\overline{\lambda} = 60\%(q = 3)$.

The fourth scenario is designed to assess the feasibility of handling a greater number of components in the remanufacturing process. Its objective is to illustrate the adaptability of the proposed approach without requiring retraining, while also evaluating the computational complexity as the number of decision points increases.

The final scenario presents a comparison of the performance of the PPO and DQN agents with the two heuristics under training conditions, with the objective of demonstrating the robustness of the algorithms and the variability.

Table 2 lists the disassembly times per component i and estimated quality class based on $\overline{\lambda}$. The processing times were extracted from real production data and represented as normally distributed values.

If the component agent type decides to not rework the component, then the disassembly process time is reduced by 10 s.

The processing times for the other process steps in the remanufacturing process are as follows:

- Pre-cleaning: mean 100; standard deviation 10
- Analysing: mean 300; standard deviation 20
- Finale Inspection and cleaning: mean 80; standard deviation 10

The monetary values for the individual components were selected as follows: $V_1 = 50$, $V_2 = 150$, and $V_3 = 10$. This is essential for accurately interpreting outcomes and guiding decision-making by RL agents within the given combination. For this particular application, we selected a value of 0.4 for both γ and β, $k = 10$ and $IP = 0.5$, as it yielded the most favourable outcomes in the simulation analyses. For $R_i(t)$ and $R_c(t)$, we use the weighting factor $w_f = 0.75$.

Results and discussion

The different combinations of PPO agent types were compared using a basic heuristic, One Disassembly heuristic and a DQN agent similar to the 6th combination. The highlighted results in blue represents the combination that achieved the best performance in their respective scenarios.

Scenario 1

Scenario 1 was conducted in three different test environments, and the PPO agents' combinations, DQN agent, as well as the Basic and One Disassembly heuristics were tested. The results show different performance profiles in each test environment, measured by the average reward ($\overline{R_f}$) and standard deviation. Table 3 shows the results for scenario 1.

In the test environment 1, the PPO agents with the 6th combination achieved the highest average reward of 55.46, followed by the DQN agent with 52.65 and the One Disassembly heuristic with 54.75. The Basic heuristic demonstrated the lowest performance with 39.46.

The results for the second test environment indicate that the 6th combination (55.01) and the DQN agents (54.97) achieved similarly high average rewards. The One Disassembly heuristic had a lower average of 52.83 in this environment, while the Basic heuristic also performed lower at 40.27. These results suggest that the 6th combination and the DQN agent remain competitive in this environment.

Figure 11 illustrates the distribution of the cumulative reward over the different test environments.

Overall performance in scenario 1 across all test environments:

- PPO agents' combination: The 6th combination demonstrated the best overall performance, with an average of 54.88 and a standard deviation of 1.81. This indicates a high degree of consistency and effectiveness across all environments. The second combination also demonstrated satisfactory performance (average: 47.71, standard deviation: 4.27). The service levels of individual components were as follows on average: 0.98 for Component 1, 0.99 for Component 2, and 0.96 for Component 3. These results demonstrate that the RL agent effectively incorporates the higher monetary value into its decision-making process.
- DQN agent: The DQN agent demonstrated robust performance, with an overall average of 53.31 and a standard deviation of 5.83. However, its performance exhibited greater variability than that of the best PPO combination.

Table 3 Scenario 1: Comparison of mean final reward $\overline{R_f}$ across test environments

Test environment		1	2	3	Overall average $\overline{R_f}$	Standard Deviation
PPO-Agents	1st Combination	43,91	37.75	43.94	**41.87**	**5.63**
	2nd Combination	48.67	49.43	45.02	**47.71**	**4.27**
	3rd Combination	43.41	44.78	44.8	**44.33**	**3.1**
	4th Combination	46.73	49.46	46.42	**47.23**	**4.45**
	5th combination	44.32	46.2	48.86	**46.46**	**3.22**
	6th combination	*55.46*	*55.01*	*54.63*	***54.88***	**1.81**
	DQN Agent	52.65	54.97	52.3	**53.31**	**5.83**
	Basic Heuristic	39.46	40.27	40.92	**40.24**	**2.46**
	One Disassembly Heuristic	54.75	52.83	50.74	**53.11**	**5.93**

The overall average and standard deviation are given in bold
The best result in the individual test environment are given in italics

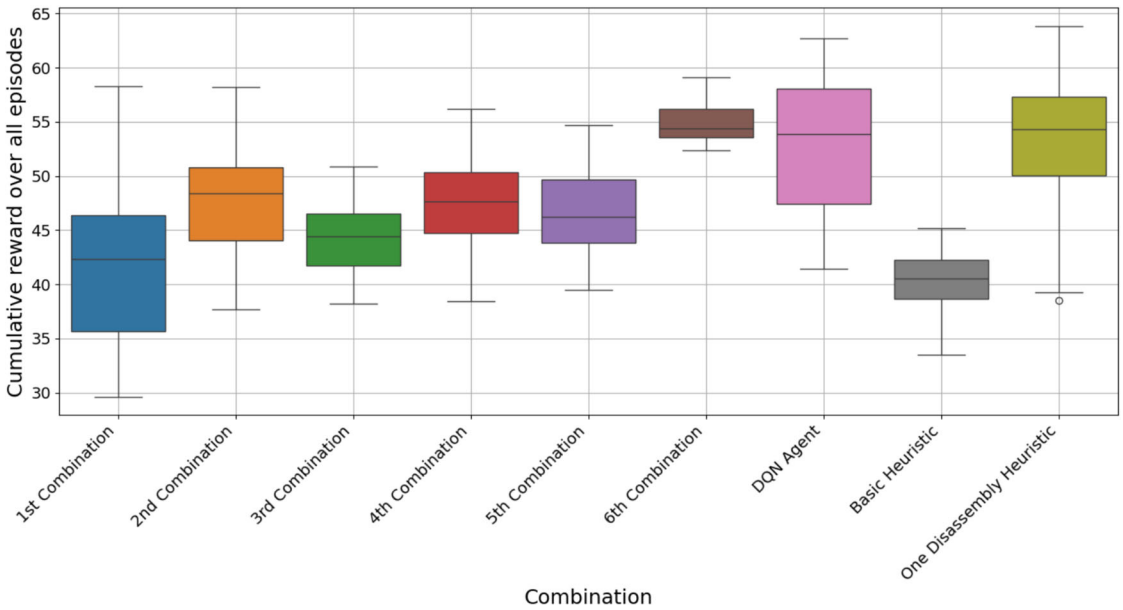

Fig. 11 Distribution of the cumulative reward over different test environments in scenario 1

- Basic and One Disassembly heuristic: The Basic heuristic was observed to be the least effective across all environments, with an overall average of 40.24 and a low standard deviation of 2.46. In contrast, the One Disassembly heuristic achieved an overall average of 53.11, with a higher variation (standard deviation: 5.93), indicating that it performs well in some environments but is less consistent overall.

The One Disassembly heuristic is designed to simulate the optimum service level. The standard deviation demonstrates the impact of the uncertainties in the remanufacturing system. The results demonstrate that the RL agents, specifically the 6th combination and the DQN agent, outperform the heuristic in terms of service level and on-time delivery. The 6th combination proved to be particularly effective and consistent across all environments in scenario 1.

Scenario 2

The second scenario is employed to simulate the situation in which only cores of poor quality are received and must be processed. In this case, the cores have a quality class of 3. The objective is to measure the extent to which the results of

Table 4 Scenario 2: Comparison of mean final reward $\overline{R_f}$ over test environments

Test Environment		1	2	3	Overall average $\overline{R_f}$	Standard Deviation
PPO-Agents	1st Combination	34.06	31.12	32.59	**32.49**	**5.03**
	2nd Combination	29.64	34.35	40.1	**34.7**	**6.05**
	3rd Combination	33.96	41.28	38.12	**37.79**	**5.77**
	4th Combination	32.16	42.78	40.8	**38.58**	**6.17**
	5th combination	34.61	47.2	42.2	**41.35**	**5.29**
	6th combination	*40.7*	45.74	*44.17*	***43.54***	**6.33**
	DQN Agent	24.05	39.09	43.07	**35.39**	**6.1**
	Basic Heuristic	35.62	42.6	41.25	**39.82**	**5.56**
	One Disassembly Heuristic	35.51	41.42	42.88	**39.93**	**5.04**

The overall average and standard deviation are given in bold
The best result in the individual test environment are given in italics

the various approaches are influenced by the poor quality of the cores.

Table 4 shows the results of the scenario 2.

In the first test environment, the 6th combination achieved the highest average reward of 40.7, outperforming the DQN agent, which exhibited a notably lower average of 24.05. The average rewards for the Basic and One Disassembly heuristics were found to be 35.62 and 35.51, respectively. The performance of the PPO agents was superior in this environment, indicating that more advanced PPO configurations can be highly effective. It is noteworthy that the poor quality of the cores has a significant impact on the reward. This can be observed based on the results for the Basic and One Disassembly heuristics, which are typically not that close.

In the second test environment, the 6th combination once again demonstrated robust performance, with an average reward of 45.74. In this environment, the 5th combination outperformed the 6th combination, with an average reward of 47.2. The One Disassembly heuristic and Basic heuristic achieved similar results, with averages of 41.42 and 42.6, respectively. The DQN agent scored 39.09, reflecting competitive but slightly lower performance compared to PPO agents and heuristics.

In the third test environment, the 6th combination demonstrated the highest average reward, with a value of 44.17. The DQN agent demonstrated a notable improvement, with an average of 43.07, while the 3rd combination achieved an average of 38.12 for the PPO agents. The Basic heuristic and One Disassembly heuristic were relatively similar, with averages of 41.25 and 42.88, respectively. The DQN agent showed considerable efficacy, indicating its growing effectiveness in this environment.

Figure 12 illustrates the cumulative reward outcomes across all episodes and test environments for scenario 2, presented as a box plot.

Overall performance in scenario 2 over all test environments:

- PPO agents' combinations: The 6th combination achieved the highest overall average reward of 43.54, with a standard deviation of 6.33, indicating both high performance and variability. The 5th combination also demonstrated a notable level of success, with an average of 41.35.
- DQN agent: The DQN agent had an overall average reward of 35.39 with a standard deviation of 6.1. While competitive, it generally lagged behind the best-performing PPO combinations.
- Basic and One Disassembly heuristic: The Basic heuristic had an overall average of 39.82 with a standard deviation of 5.56. With an overall average reward of 39.93 and a standard deviation of 5.04, the One Disassembly heuristic demonstrated similar performance to the Basic heuristic and showed consistent results across the environments.

Overall, the 6th combination consistently outperformed other methods, demonstrating both high rewards and effectiveness across varying test environments.

This scenario illustrates the influence of low-quality cores on the reward and the Basic heuristic, in comparison to scenario 1, which is closely aligned with the One Disassembly heuristic. The quality of the cores has a considerable impact on the outcome or reward of the remanufacturing system.

In the remaining scenarios, the PPO agents will continue to use the 6th combination, as this achieved the highest overall reward. A comparison is made with the DQN agent, the Basic heuristic and the One Disassembly heuristic.

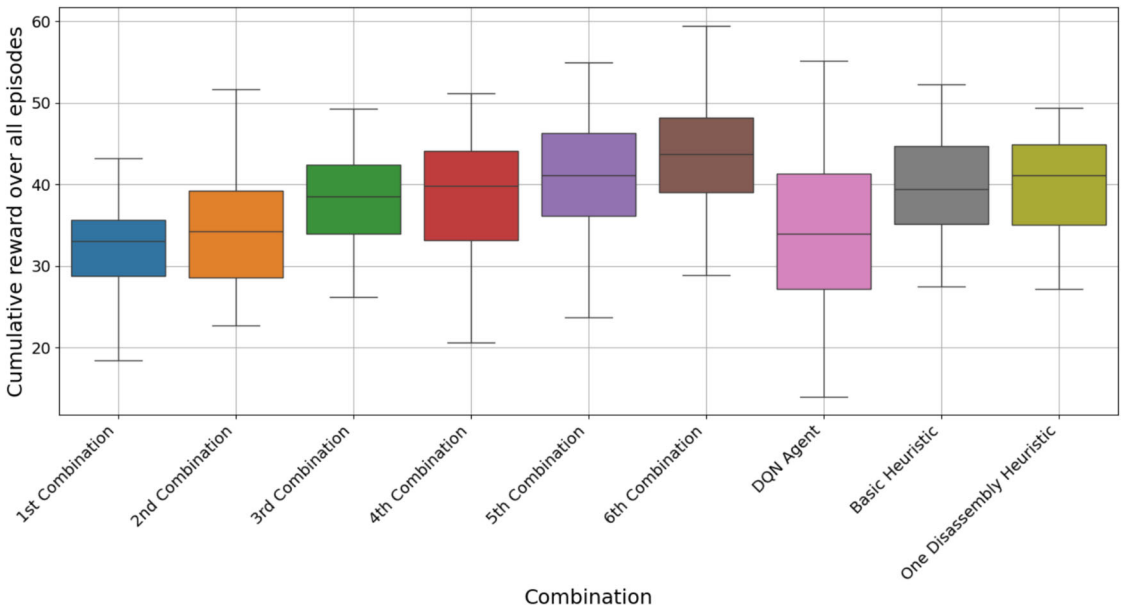

Fig. 12 Distribution of the cumulative reward over different test environments in scenario 2

Scenario 3

The third scenario examines the impact of modifications to the control parameters. In this context, three key parameters exert a significant influence on decision-making processes and, consequently, on the level of reward. The impact of each parameter is subsequently examined. The scenario 3 will be simulated and compared in test environment 1.

Modification of process times

As a consequence of the modified process times, the planned lead time for the order is also subject to alteration. The following section will analyse whether the RL agent achieves a higher reward compared to the One Disassembly heuristic, which determines outcomes independently of adherence to on-time delivery, given that the framework conditions have changed.

The initial step is to alter the process times in a manner that results in the last and, simultaneously, most inexpensive component having the longest process time.

Table 5 shows the changed processing time PT_i^q for the individual component.

The 6th combination achieved the highest overall average reward of 52.25, with a relatively low standard deviation of 3.97. This indicates that the 6th combination not only produced the highest rewards but also demonstrated consistency and stability in performance across different test runs, thereby providing evidence that it is a robust and reliable combination.

Table 5 Processing times PT_i^q variation for evaluation

Disassembly step	Processing time $\boldsymbol{PT_i^q}$ (mean, standard deviation)
Disassembly Component 1	$PT_1^1 = (50s, 2s)$ *per Component*
	$PT_1^2 = (70s, 5s)$ *per Component*
	$PT_1^3 = (110s, 10s)$ *per Component*
Disassembly Component 2	$PT_2^1 = (80s, 2s)$ *per Component*
	$PT_2^2 = (100s, 5s)$ *per Component*
	$PT_2^3 = (140s, 10s)$ *per Component*
Disassembly Component 3	$PT_3^1 = (130s, 2s)$ *per Component*
	$PT_3^2 = (150s, 5s)$ *per Component*
	$PT_3^3 = (190s, 10s)$ *per Component*

The DQN agent achieved an average reward of 51.59, which is comparable to the performance of the 6th combination. However, the DQN agent showed a considerably higher standard deviation of 8.35, indicating greater variability in its performance across different test environments. This variability indicates that while the DQN agent can achieve high rewards, it is less reliable than the PPO agent combination.

The mean reward achieved by the One Disassembly heuristic was 50.34, with a standard deviation of 5.68. Although it demonstrated satisfactory performance, it was outperformed by both the 6th combination and the DQN agent. Both approaches resulted in a better outcome with respect to the multi-objective optimisation issue.

Table 6 Processing time PT_i^q variation with increased time for high value component

Disassembly step	Processing Time $\boldsymbol{PT_i^q}$ (mean, standard deviation)
Disassembly Component 1	$PT_1^1 = (30s, 2s)$ *per Component*
	$PT_1^2 = (60s, 5s)$ *per Component*
	$PT_1^3 = (100s, 10s)$ *per Component*
Disassembly Component 2	$PT_2^1 = (220s, 2s)$ *per Component*
	$PT_2^2 = (250s, 5s)$ *per Component*
	$PT_2^3 = (290s, 10s)$ *per Component*
Disassembly Component 3	$PT_3^1 = (40s, 2s)$ *per Component*
	$PT_3^2 = (50s, 5s)$ *per Component*
	$PT_3^3 = (90s, 10s)$ *per Component*

The Basic heuristic produced the lowest average reward (39.7) of all the methods under consideration. It is significantly less effective than more advanced techniques such as PPO and DQN agents.

In the second step, the process times are modified so that the component with the highest value has the longest process time. The respective process times are presented in Table 6.

In the conducted experiments, the 6th combination achieved the highest average reward of 49.61, with a relatively low standard deviation of 2.48, indicating consistent performance across different test environments. The One Disassembly heuristic closely followed, with an average reward of 48.84 and a standard deviation of 5.36, showing a slightly higher variability. The DQN agent, while effective, obtained a lower average reward of 42.78 and exhibited the highest variability with a standard deviation of 9.24, suggesting inconsistent results. In contrast, the Basic heuristic performed the worst, achieving a significantly lower average reward of 33.36.

Monetary value

As part of the investigation into the influence of changed process times on the decision-making of the RL agent, it is also necessary to analyse the extent to which adjustments are made to other monetary values. This is a relevant area of investigation because it has been demonstrated that the sequence of components linked to monetary values has no influence on the reward. Instead, the RL agent adapts and continues to achieve a high reward. The following variations are taken into account in the study:

1. $V_1 = 10$, $V_2 = 50$, and $V_3 = 150$
2. $V_1 = 150$, $V_2 = 10$, and $V_3 = 50$

In both variations, the 6th combination achieved the highest reward value in comparison to the other methods. The mean cumulative reward is approximately similar in both cases, with a value of 52.99 and 52.32, respectively. The decision-making process, as measured by the service levels of individual components, adjusted accordingly to the modified monetary values. In the first case, the service levels were 0.91 for Component 1, 0.90 for Component 2, and 0.89 for Component 3. In the second case, the service levels were 0.94 for Component 1, 0.87 for Component 2, and 0.89 for Component 3. The DQN agent achieved an average cumulative reward of 50.96 and 49.07. The One Disassembly heuristic is closely aligned with the RL agents, with average cumulative rewards of 50.31 and 52.14. In contrast, the Basic heuristic demonstrated the lowest reward, with average cumulative rewards of 41.22 and 40.84.

Upon analysis of the service level state of the components, it became evident that the RL agents consider the monetary value of the component and adapt their decision-making accordingly.

Increased failure probability

The final modification is related to the probability of failure. As part of this, the impact of an increased probability of failure across all quality classes on the reward is examined. The probability of failure was modified as follows:

- Changed probability of failure $\overline{\lambda}$: $\overline{\lambda} = 10\%$ $(q = 1)$, $\overline{\lambda} = 30\%(q = 2)$, $\overline{\lambda} = 60\%(q = 3)$

In this experimental setup, the failure probability of components was adjusted to increase the likelihood of failures across all quality classes, simulating an overall decline in component quality. Under these conditions, the 6th combination demonstrated the highest average reward of 47.55, with a standard deviation of 7.32, indicating moderate variability in performance. The One Disassembly heuristic followed with an average reward of 45.11 and a standard deviation of 6.07, showing slightly more consistent results. The Basic heuristic achieved a similar reward of 44.12 but with a lower variability, as indicated by its standard deviation of 3.32. The DQN agent, however, yielded the lowest average reward of 36.65 and a standard deviation of 7.54, highlighting both reduced performance and relatively high variability under these altered conditions.

Scenario 4

In the conducted experiments, the proposed approach for component disassembly was tested in various scenarios. Initially, the effectiveness of the methods was investigated in a case involving three components. From these scenarios, the

6th combination achieved the best results. This combination employs a single component agent that makes disassembly decisions of the currently considered component.

A significant benefit of this 6th combination is the configuration of its state space, which enables the extension to multiple components to be disassembled. In this scenario, the RL agent observes only the value of the affected component, thereby enabling adaptation to different scenarios without the necessity for retraining. This configuration enables the demonstration of the scalability of the RL agent.

To analyse computational complexity and to assess the scalability of the RL agent, the disassembly process was simulated with an increasing number of components: four components and ten components. An increase in the number of components introduces greater complexity, enabling a more detailed assessment of the agent's adaptability to varying conditions. As the number of components increases, the number of disassembly decisions per episode grows proportionally. The computation time scales with:

- The number of disassembly steps required per component.
- The increased number of core agent and component agent decisions.
- The time required to process additional reward updates and environmental interactions.

While alternative RL agent combinations (e.g., those with multiple core agents) would increase the state space, our analysis shows that the selected configuration maintains computational efficiency without introducing significant overhead from an RL training perspective.

The performance of the 6th combination was once more evaluated in comparison with other methods, namely the DQN agent, the One Disassembly heuristic, and the Basic heuristic. In terms of cumulative reward, the 6th combination demonstrated the most optimal performance, with a mean cumulative reward of 128.99. In comparison, the DQN agent achieved a mean cumulative reward of 99.66, the One Disassembly heuristic yielded 124.66, and the Basic heuristic obtained 95.51.

The comparative results show that the RL agent in the 6th combination exhibits consistent and scalable performance in both lower and higher complexity scenarios. This highlights the agent's ability to respond to a larger number of components without retraining and still develop an effective disassembly strategy.

The results demonstrate the scalability and robustness of the RL-based approach, particularly in complex scenarios with an increased number of components (e.g., 10 components). The method consistently outperforms traditional approaches, ensuring feasibility for larger-scale implementations. Since no retraining is required, the RL model remains applicable across various disassembly configurations. The computation time scales predictably with the number of components, increasing by approximately 16% when moving from 3 (1 h and 6 min) to 4 (1 h 17 min) components and by 120.5% when scaling to 10 components (2 h 50 min).

Scenario 5

To illustrate the complexity of decision-making, 100 episodes were simulated in scenario 5. This approach was designed to assess the variability, robustness, and scalability of the selected methods. The results confirm that the chosen state values allow for optimal decision-making based on the optimisation objective. To ensure comparability, the simulation environment was consistent with the training phase of the RL agents.

Figure 13 illustrates the comparison of the cumulative reward over 100 episodes in the training test environment.

The results demonstrate that the 6th combination achieved the highest average reward of 53.43 with a standard deviation of 4.07, indicating both a high level of performance and robustness with low variability. The One Disassembly heuristic resulted in a reward of 51.48 and a standard deviation of 6.68, indicating a satisfactory level of performance, albeit with a marginally higher degree of variability. The DQN agent demonstrated a comparable reward of 50.02, however, with a higher standard deviation of 7.98, indicating greater sensitivity to different scenarios. The Basic heuristic performed notably less effectively, with an average reward of 39.76 and the lowest variability, as indicated by its standard deviation of 2.65. These findings reinforce the efficiency of the state selection methodology for optimal decision-making and demonstrate the adaptability of the proposed approach to varying degrees of decision-making complexity.

The proposed RL-based framework demonstrates several key advantages, particularly in the context of dynamic and uncertain disassembly environments:

- **Adaptability to Dynamic Systems** The framework leverages PPO to adapt to changing operational conditions, such as varying disassembly depths and unpredictable core qualities. This adaptability ensures the RL agents can make effective decisions even in environments where quality and component availability are uncertain.
- **Scalability for Complex Systems** As illustrated in Scenario 4, the framework scales effectively with an increasing number of components per product. For instance, simulations involving up to 10 components per product – a significant number in industrial contexts – highlight that the RL agents maintain performance without significant degradation. The results demonstrate the system's ability to handle longer disassembly times and manage increased failure probabilities, showcasing its robustness in complex settings.

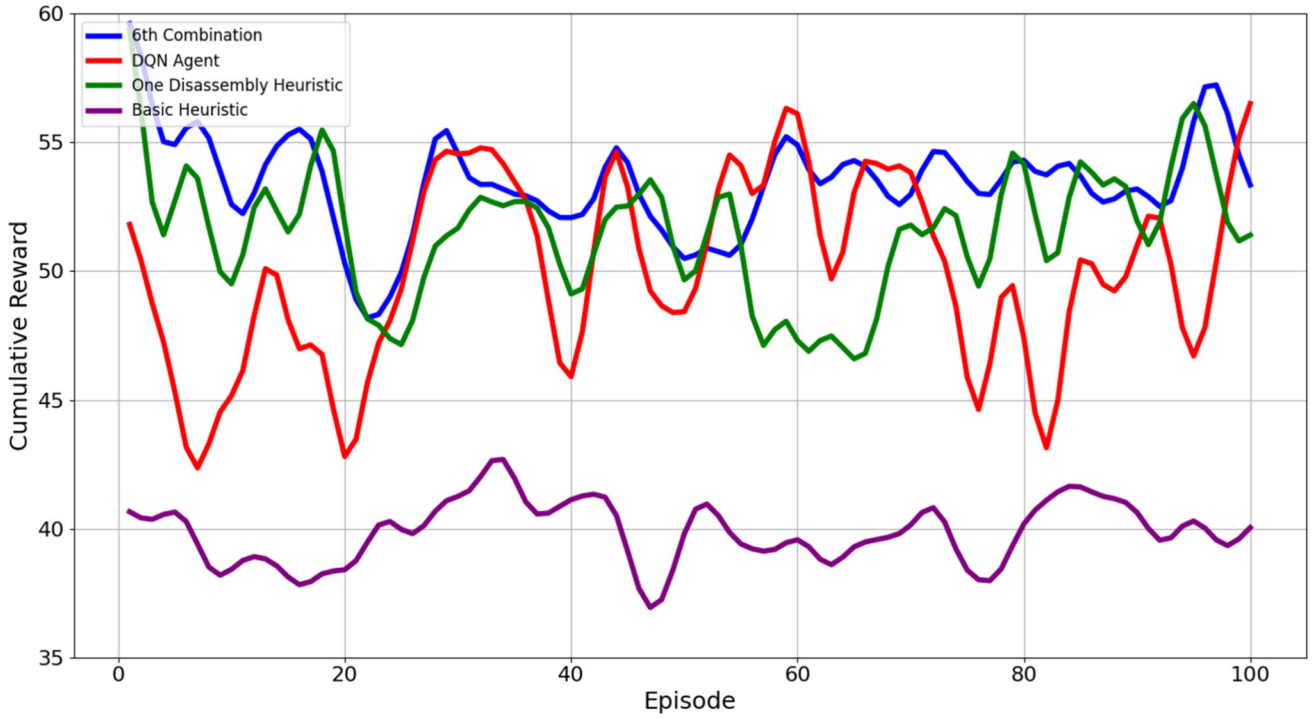

Fig. 13 Comparison of the cumulative reward over 100 episodes in training test environment

- **Optimisation of Key Performance Metrics** The approach prioritises critical metrics such as on-time delivery (OTD) and service level (SL). By dynamically balancing these objectives, the framework ensures that decisions align with operational objectives while minimising delays and maximising overall efficiency.
- **Integration of Data-Driven Decision-Making** The RL agents incorporate real-time data, such as failure rates and service levels, into their decision-making processes. This integration allows the framework to dynamically adjust actions based on the current state, enhancing its responsiveness and accuracy.
- **Potential for Customisation** The weighting system used to train the RL agents offers flexibility to prioritise objectives based on organisational objectives. For example, weights can be adjusted to account for factors such as energy consumption or environmental impact, enabling the framework to support sustainability initiatives while maintaining economic performance.
- **Industrial Relevance and Practicality** The framework has been designed with industrial applicability in mind. It addresses common challenges in remanufacturing, such as uncertainty in core and component quality, as well as variability in processing times, making it suitable for deployment in real-world settings.

Conclusion

The process of decision-making in the context of disassembly presents a significant challenge due to the high degree of unpredictability inherent to used products. The existing literature predominantly addresses deterministic conditions, which are insufficient for fully capturing the complexities of real-world disassembly. To address this limitation, our study presents a RL-based approach, specifically tailored for decision-making in the disassembly of cores. This method takes into account the inherent uncertainties associated with the process and considers the competing optimisation goals of improving service levels and ensuring on-time delivery.

The principal benefit of our RL approach is its capacity to make decisions based on production states without the necessity for a complete and detailed system model. By incorporating real-time production states, the RL algorithm is capable of adapting to changes within the system in real time, thereby optimising decisions for disassembly tasks under uncertain conditions. Furthermore, the technique is capable of efficiently balancing multiple objectives.

The results of our validation demonstrate that the proposed RL approach outperforms traditional heuristic and other decision-making algorithms. In particular, it achieves a 22% higher reward than a basic heuristic method, a 12% improvement over the DQN algorithm, and a 4% higher reward than the One Disassembly heuristic. These findings

illustrate the robustness and efficiency of our approach across a range of disassembly scenarios.

Moreover, our findings indicate that the RL approach is scalable. The system is capable of adapting to a range of configurations without the necessity for retraining, thereby ensuring a high degree of flexibility in different production environments. This scalability is particularly advantageous for industries where system conditions can change frequently, and the capacity for quick adaptation is crucial.

In the future, the RL solution has the potential to be implemented in the real world through the integration of data with existing processes. By establishing a connection with live data streams, the algorithm is able to make real-time decisions in production settings, thereby further enhancing its practical application. Furthermore, this RL-based approach can be extended by integrating with other algorithms that address related challenges in remanufacturing, such as disassembly line balancing or planning and control problems. This would result in a more comprehensive decision-making framework for complex manufacturing systems.

Author contributions Felix Paschko and Steffi Knorn wrote the main manuscript text and prepared all figures and tables. Felix Paschko, Steffi Knorn, Abderrahim Krini and Markus Kemke reviewed the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability All data generated or analysed during this study are included in this published article.

References

Altenmüller, T., Stüker, T., Waschneck, B., Kuhnle, A., & Lanza, G. (2020). Reinforcement learning for an intelligent and autonomous production control of complex job-shops under time constraints. *Production Engineering, 14*, 319–328. https://doi.org/10.1007/s11740-020-00967-8

Bi, Z., Guo, X., Wang, J., Qin, S., Qi, L., & Zhao, J. (2022). A Q-Learning-based Selective Disassembly Sequence Planning Method. In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Prague, Czech Republic, 09.10.2022 - 12.10.2022* (pp. 3216–3221). IEEE. https://doi.org/10.1109/SMC53654.2022.9945073.

Cong, L., Zhao, F., & Sutherland, J. W. (2019). A design method to improve end-of-use product value recovery for circular economy. *Journal of Mechanical Design*. https://doi.org/10.1115/1.4041574

Cunha, B., Madureira, A. M., Fonseca, B., & Coelho, D. (2020). Deep Reinforcement Learning as a Job Shop Scheduling Solver: A Literature Review. In A. M. Madureira, A. Abraham, N. Gandhi, & M. L. Varela (Eds.), *Hybrid Intelligent Systems* (Vol. 923, pp. 350–359, Advances in Intelligent Systems and Computing). Cham: Springer International Publishing.

Feng, Y., Zhou, M., Tian, G., Li, Z., Zhang, Z., Zhang, Q., et al. (2019). Target disassembly sequencing and scheme evaluation for CNC machine tools using improved multiobjective ant colony algorithm and fuzzy integral. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49*, 2438–2451. https://doi.org/10.1109/TSMC.2018.2847448

Fleischmann, M., Bloemhof-Ruwaard, J. M., Dekker, R., van der Laan, E., van Nunen, J. A., & van Wassenhove, L. N. (1997). Quantitative models for reverse logistics: A review. *European Journal of Operational Research, 103*, 1–17. https://doi.org/10.1016/S0377-2217(97)00230-0

Gao, K. Z., He, Z. M., Huang, Y., Duan, P. Y., & Suganthan, P. N. (2020). A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing. *Swarm and Evolutionary Computation, 57*, Article 100719. https://doi.org/10.1016/j.swevo.2020.100719

Güngör, A., & GUPTA, S. M. (1999). Disassembly line balancing. *Proceedings of the 1999 Annual Meeting of the Northeast Decision Sciences Institute*, 193–195.

Guo, X., Bi, Z., Wang, J., Qin, S., Liu, S., & Qi, L. (2023). Reinforcement learning for disassembly system optimization problems: A survey. *International Journal of Network Dynamics and Intelligence*. https://doi.org/10.53941/ijndi0201001

Guo, X., Zhang, Z., Qi, L., Liu, S., Tang, Y., & Zhao, Z. (2022). Stochastic hybrid discrete grey wolf optimizer for multi-objective disassembly sequencing and line balancing planning in disassembling multiple products. *IEEE Transactions on Automation Science and Engineering, 19*, 1744–1756. https://doi.org/10.1109/TASE.2021.3133601

Guo, X., Zhou, M., Abusorrah, A., Alsokhiry, F., & Sedraoui, K. (2021). Disassembly sequence planning: A survey. *IEEE/CAA Journal of Automatica Sinica, 8*, 1308–1324. https://doi.org/10.1109/JAS.2020.1003515

Heuillet, A., Couthouis, F., & Díaz-Rodríguez, N. (2021). Explainability in deep reinforcement learning. *Knowledge-Based Systems, 214*, Article 106685. https://doi.org/10.1016/j.knosys.2020.106685

Hoffmann, M., & Knorn, S. (2024). Optimization of resource allocation in remanufacturing systems: A labor and automation perspective. *IFAC-PapersOnLine, 58*, 150–155. https://doi.org/10.1016/j.ifacol.2024.10.253

Hoffmann, M., Krini, A., Mueller, A., & Knorn, S. (2025). Remanufacturing production planning and control: Conceptual framework for requirement definition. *Journal of Remanufacturing, 15*, 97–126. https://doi.org/10.1007/s13243-025-00149-8

Kaiser, J.-P., Gäbele, J., Koch, D., Schmid, J., Stamer, F., & Lanza, G. (2024). Adaptive acquisition planning for visual inspection in remanufacturing using reinforcement learning. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-024-02478-0

Klügel, R. (2024). *Entwicklung eines neuen Ansatzes zur Bestimmung der Wiederverwendungsgrenzwerte von elektromechanischen Lenksystemen*. Universitäts- und Landesbibliothek Sachsen-Anhalt.

Kuhnle, A., & Lanza, G. (2019). Application of Reinforcement Learning in Production Planning and Control of Cyber Physical Production Systems. In J. Beyerer, C. Kühnert, & O. Niggemann (Eds.),

Machine Learning for Cyber Physical Systems (Vol 9, Technologien für die intelligente Automation). Berlin Heidelberg: Springer Berlin Heidelberg. 123–132

Kuhnle, A., Kaiser, J.-P., Theiß, F., Stricker, N., & Lanza, G. (2021). Designing an adaptive production control system using reinforcement learning. *Journal of Intelligent Manufacturing, 32*, 855–876. https://doi.org/10.1007/s10845-020-01612-y

Liang, J., Guo, S., Du, B., Liu, W., & Zhang, Y. (2022). Restart genetic flatworm algorithm for two-sided disassembly line balancing problem considering negative impact of destructive disassembly. *Journal of Cleaner Production, 355*, Article 131708. https://doi.org/10.1016/j.jclepro.2022.131708

Liu, H., & Zhang, L. (2021). Optimizing a disassembly sequence planning with success rates of disassembly operations via a variable neighborhood search Algorithm. *IEEE Access, 9*, 157540–157549. https://doi.org/10.1109/ACCESS.2021.3101221

Lund, R. T. (1984). *Remanufacturing: The experience of the United States and implications for developing countries* (1st ed., Integrated resource recovery, Vol. 2). Washington, DC: The World Bank.

Mao, H., Liu, Z., & Qiu, C. (2023). Adaptive disassembly sequence planning for VR maintenance training via deep reinforcement learning. *The International Journal of Advanced Manufacturing Technology, 124*, 3039–3048. https://doi.org/10.1007/s00170-021-08290-x

Mei, K., & Fang, Y. (2021). Multi-Robotic Disassembly Line Balancing Using Deep Reinforcement Learning. In *ASME 2021 16th International Manufacturing Science and Engineering Conference, Virtual, Online, 21.06.2021 - 25.06.2021*. American Society of Mechanical Engineers. https://doi.org/10.1115/MSEC2021-63522.

Meng, K., Xu, G., Peng, X., Youcef-Toumi, K., & Li, J. (2022). Intelligent disassembly of electric-vehicle batteries: A forward-looking overview. *Resources, Conservation and Recycling, 182*, Article 106207. https://doi.org/10.1016/j.resconrec.2022.106207

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., et al. (2016). Asynchronous methods for deep reinforcement. *Learning*. https://doi.org/10.48550/arXiv.1602.01783

Ong, S. K., Chang, M. M. L., & Nee, A. Y. C. (2021). Product disassembly sequence planning: State-of-the-art, challenges, opportunities and future directions. *International Journal of Production Research, 59*, 3493–3508. https://doi.org/10.1080/00207543.2020.1868598

Paprocka, I., & Skołud, B. (2022). A predictive approach for disassembly line balancing problems. *Sensors*. https://doi.org/10.3390/s22103920

Paschko, F., Knorn, S., Krini, A., & Kemke, M. (2023). Material flow control in Remanufacturing Systems with random failures and variable processing times. *Journal of Remanufacturing, 13*, 161–185. https://doi.org/10.1007/s13243-023-00126-z

Ren, Y., Meng, L., Zhang, C., Zhao, F., Saif, U., Huang, A., et al. (2020). An efficient metaheuristics for a sequence-dependent disassembly planning. *Journal of Cleaner Production, 245*, Article 118644. https://doi.org/10.1016/j.jclepro.2019.118644

Reveliotis, S. A. (2007). Uncertainty management in optimal disassembly planning through learning-based strategies. *IIE Transactions, 39*, 645–658. https://doi.org/10.1080/07408170600897536

Riggs, R. J., Battaïa, O., & Hu, S. J. (2015). Disassembly line balancing under high variety of end of life states using a joint precedence graph approach. *Journal of Manufacturing Systems, 37*, 638–648. https://doi.org/10.1016/j.jmsy.2014.11.002

Rizova, M. I., Wong, T. C., & Ijomah, W. (2020). A systematic review of decision-making in remanufacturing. *Computers & Industrial Engineering, 147*, Article 106681. https://doi.org/10.1016/j.cie.2020.106681

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms* .

Stecke, K. E., & Solberg, J. J. (1981). Loading and control policies for a flexible manufacturing system. *International Journal of Production Research, 19*, 481–490. https://doi.org/10.1080/00207548108956679

Sutton, R. S., & Barto, A. (2020). *Reinforcement learning: An introduction (Adaptive computation and machine learning)*. The MIT Press.

Toffel, M. W. (2004). Strategic management of product recovery. *California Management Review, 46*, 120–141. https://doi.org/10.2307/41166214

Tuncel, E., Zeid, A., & Kamarthi, S. (2014). Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. *Journal of Intelligent Manufacturing, 25*, 647–659. https://doi.org/10.1007/s10845-012-0711-0

Vongbunyong, S. (2015). *Disassembly Automation: Automated Systems with Cognitive Abilities (SpringerLink Bücher)*. Springer International Publishing.

Wang, Di., Zhao, J., Han, M., & Li, L. (2024). Deep reinforcement learning-based energy-aware disassembly planning for end-of-life products with stimuli-activated self-disassembly. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-024-02527-8

Wang, K., Li, X., Gao, L., & Li, P. (2020). Energy consumption and profit-oriented disassembly line balancing for waste electrical and electronic equipment. *Journal of Cleaner Production, 265*, Article 121829. https://doi.org/10.1016/j.jclepro.2020.121829

Wenzel, S., & Peter, T. (2017). Simulationsunterstützte Entwicklung von Methoden zur reaktiven Steuerung von Demontagelinien. *Simulation in Produktion und Logistik, 2017*, 325.

Wurster, M., Michel, M., May, M. C., Kuhnle, A., Stricker, N., & Lanza, G. (2022). Modelling and condition-based control of a flexible and hybrid disassembly system with manual and autonomous workstations using reinforcement learning. *Journal of Intelligent Manufacturing, 33*, 575–591. https://doi.org/10.1007/s10845-021-01863-3

Xanthopoulos, A. S., Koulouriotis, D. E., Gasteratos, A., & Ioannidis, S. (2016). Efficient priority rules for dynamic sequencing with sequence-dependent setups. *International Journal of Industrial Engineering Computations*. https://doi.org/10.5267/j.ijiec.2016.2.002

Xu, W., Tang, Q., Liu, J., Liu, Z., Zhou, Z., & Pham, D. T. (2020). Disassembly sequence planning using discrete Bees algorithm for human-robot collaboration in remanufacturing. *Robotics and Computer-Integrated Manufacturing, 62*, Article 101860. https://doi.org/10.1016/j.rcim.2019.101860

Zhang, W., Zheng, Y., & Ahmad, R. (2023). The integrated process planning and scheduling of flexible job-shop-type remanufacturing systems using improved artificial bee colony algorithm. *Journal of Intelligent Manufacturing, 34*, 2963–2988. https://doi.org/10.1007/s10845-022-01969-2

Zhao, M., Guo, X., Zhang, X., Fang, Y., & Ou, Y. (2019). ASPW-DRL: Assembly sequence planning for workpieces via a deep reinforcement learning approach. *Assembly Automation, 40*, 65–75. https://doi.org/10.1108/AA-11-2018-0211