Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Luisa Alina Kinat

Assembly Line Sequencing with Collaborative Robots using Reinforcement Learning

Master's thesis in Global Manufacturing Management
Supervisor: Mirco Peron
June 2023

NTNU
Norwegian University of
Science and Technology

Luisa Alina Kinat

Assembly Line Sequencing with Collaborative Robots using Reinforcement Learning

Master's thesis in Global Manufacturing Management
Supervisor: Mirco Peron
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

The introduction of collaborative robots in assembly lines offers great potential to enhance productivity and reduce ergonomic risks for human operators. However, it leads to an increasingly complex environment. Therefore, in order to exploit the full potential, advanced solutions for task allocation and sequence planning are necessary. The use of reinforcement learning offers a promising approach to solving the optimization problem and enable real-time decisions. So far, the research in this area is mainly focused on economic factors such as a short completion time. The goal of this thesis is, firstly, to identify the state of the art of addressing the assembly line sequencing problem within this context by conducting a systematic literature review, and secondly, to develop an optimization model using reinforcement learning and taking economic as well as ergonomic factors into account. The objectives are to identify relevant literature, to evaluate their strengths and limitations, to derive areas where future research is needed, to develop an optimization model based on the previous findings, and finally, to train and test the model on an exemplary assembly task.

In the systematic literature review, only a limited number of documents was found with minimized completion time as single objective for most models. One multi-objective model was identified taking completion time and task difficulty into account. The pure focus on ergonomics was not a subject of research so far. The within this thesis developed model aims to optimize the completion time as well as the ergonomic stress level of the human using multi-agent reinforcement learning. The results show that multiple objectives can be considered and weighted by assigning rewards to multiple conditions.

Limitations of the review lay in the low number of documents, the use of only one database in which solely English literature was considered and that a degree of subjectivity remains during the selection of the documents. Limitations of the developed model are the support of only cooperation between robot and human in one workstation, its limited capabilities to adapt to unforeseen behavior and its stochastic elements which might not always lead to the choice of the optimal solution.

Keywords: assembly line sequencing, collaborative robots, reinforcement learning, ergonomics

Sammendrag

Introduksjonen av samarbeidsroboter i samlebånd gir et stort potensial for å øke produktiviteten og redusere ergonomiske risikoer for menneskelige operatører. Men det fører også til et stadig mer komplekst miljø. For å utnytte det fulle potensialet er det derfor nødvendig med avanserte løsninger for oppgavefordeling og sekvensplanlegging. Bruken av reinforcement learning tilbyr en lovende tilnærming til å løse optimaliseringsproblemet og muliggjøre sanntidsbeslutninger. Så langt har forskningen på dette området først og fremst fokusert på økonomiske faktorer som kort gjennomføringstid. Målet med denne oppgaven er for det første å identifisere den nyeste teknologien innen håndtering av samlebåndssekvenseringsproblemet i denne sammenheng ved å gjennomføre en systematisk litteraturstudie, og for det andre å utvikle en optimaliseringsmodell som bruker reinforcement learning og tar hensyn til økonomiske samt ergonomiske faktorer. Målene er å identifisere relevant litteratur, vurdere deres styrker og begrensninger, utlede områder der fremtidig forskning er nødvendig, utvikle en optimaliseringsmodell basert på tidligere resultater, og til slutt trene og teste modellen på en eksemplarisk monteringsoppgave.

I det systematiske litteraturstudiet ble det kun funnet et begrenset antall dokumenter med minimert gjennomføringstid som ett mål for de fleste modellene. En multi-objective modell ble identifisert som tok hensyn til gjennomføringstid og oppgavevanskelighet. Det rene fokuset på ergonomi har så langt ikke vært gjenstand for forskning. Modellen utviklet i denne oppgaven har som mål å optimalisere gjennomføringstiden samt det menneskelige ergonomiske stressnivået ved å bruke multi-agent reinforcement learning. Resultatene viser at flere mål kan vurderes og vektes ved å tildele belønninger til flere forhold.

Begrensningene i litteraturstudien ligger i det lave antallet dokumenter, bruken av kun én database hvor kun engelsk litteratur ble vurdert, og at det fortsatt er en viss grad av subjektivitet under utvelgelsen av dokumentene. Begrensningene til den utviklede modellen er at den kun støtter samarbeid mellom robot og menneske på én arbeidsstasjon, dens begrensede evne til å tilpasse seg uforutsett atferd og dens stokastiske elementer, som kanskje ikke alltid fører til en optimal løsning.

Table of contents

List of figures

List of tables

1 Introduction

In this chapter, introductory remarks to this thesis are provided. To start, section 1.1 shows the background and motivation, followed by the detailed problem description in section 1.2. The research questions and the thesis scope are outlined in section 1.3. Lastly, section 1.4 explains the report structure. This thesis is based on the specialization project preceding this work, in which a systematic literature review on assembly line balancing and sequencing in the context of Industry 4.0 technologies was conducted (Kinat 2022).

1.1 Background and motivation

The world of manufacturing is revolutionized by the introduction of Industry 4.0 technologies such as collaborative robots. In assembly systems, collaborative robots lead to enhanced productivity while reducing ergonomic risks for human workers. Beyond that, the employment of human workers regardless of their abilities or their strength might be enabled. Humans can not only be supported, but substituted in dangerous or heavy tasks (Bragança et al. 2019). This is feasible in most of the common manufacturing environments. The use of human-robot collaboration can be highly beneficial. In addition, the utilization of collaborative robots offers a low-threshold option for implementing partial automation. A growing number of manufacturing companies aim to make use of this technology.

However, the great potential of the technology is often not fully exploited. Its introduction leads to an increasingly complex environment with many parameters. Advanced approaches for system control and decision-making are necessary. The allocation of tasks to robot and human worker as well as the definition of the best task sequence come with extensive effort. Industrial planners might face issues when trying to solve these tasks with analytical methods such as mixed-integer linear programming. The implementation of machine learning such as reinforcement learning offers a solution for finding an optimized task allocation and sequence in this context. Until recently, reinforcement learning methods have been limited to low-dimension problems. Nevertheless, thanks to advancements in deep learning, recent reinforcement learning algorithms can cope with highly complex problems (Yu at al. 2020).

Assembly line sequencing has been studied for decades. Many mathematical models to solve the optimization problem with various objectives exist. Nevertheless, only a limited number of models were developed considering collaborative robots. The use of reinforcement learning to solve the assembly line sequencing problem in this context has become an emerging topic. So far, the research in this area is focused on economic factors such as a short cycle time. To also put the focus on ergonomic factors forms the basis for the motivation of this thesis.

1.2 Problem description

This research aims to develop an optimization model to solve the assembly line sequencing problem in the context of collaborative robots using reinforcement learning and taking economic as well as ergonomic factors into account. The focus will be on workstations where one human worker and one collaborative robot complete tasks to reach a shared assembly goal. Starting point will be to study the state of the art in dealing with assembly line sequencing with collaborative robots using reinforcement learning. Assembly line sequencing refers to defining the order of tasks to be done taking objectives and constraints into account. In addition, a decision must be made about task assignment when a human worker and a robot can perform tasks. Collaborative robots are able to correct insignificant mistakes of the human during work (Alessio et al. 2022). However, by their implementation, additional complexity is introduced to the assembly process. Their utilization should be optimized. The following research objectives are derived from this demand:

- Identification of relevant literature
- Evaluation of strengths and limitations of the different approaches and identification of areas for further research
- Development of an optimization model for assembly line sequencing with collaborative robots using reinforcement learning taking ergonomic factors into account
- Training and testing of the model using an exemplary assembly task

1.3 Research questions and thesis scope

The formulation of the research questions is based on the previously defined objectives of this project. As a result, the following research questions are pursued:

1) What is the state of the art of addressing the assembly line sequencing problem with reinforcement learning when using collaborative robots?
2) How can a reinforcement learning based optimization model be formulated to solve the assembly line sequencing problem with collaborative robots under consideration of ergonomic factors?

The first research question deals with the current state of knowledge regarding assembly line sequencing in the specified context. A literature review is carried out regarding this topic. The review focuses only on publications that specifically addresses sequencing with collaborative robots. The scope of the review is further narrowed to research using reinforcement learning to solve the optimization problem as indicated in the research question. Only studies providing models to solve the sequencing problem in sufficient detail should be relevant. This criterion

is met when an optimization model is introduced. In addition, only literature regarding assembly workstations is relevant. To answer the first research question, a systematic literature review should be used as methodology. The defined scope serves as criterion for determining the relevance of the publications included in the review.

As formulated in the second research question, an optimization model using reinforcement learning for solving the sequencing problem will be developed within this thesis. The model should be applicable for workstations with one human worker and one collaborative robot. The task allocation to human and robot should be included in the model. To sum up, the model should find the sequence of tasks as well as the allocation of the tasks to reach the assembly goal.

1.4 Report structure

The remainder of this thesis report is structured as follows. Next, in chapter 2, relevant theoretical background to the topic of this thesis is explained. Chapter 3 describes and justifies the used methodologies to answer the research questions in detail. This includes definitions of the assembly line sequencing problem and collaborative robots as well as a description of reinforcement learning. In chapter 4, a systematic literature review to answer the first research question is carried out: It includes the study selection, the descriptive analysis of the selected documents as well as the content analysis according to defined categories. Based on the findings of the literature review, the development of an optimization model follows in chapter 5. The model is tested on an example assembly task and discussed in chapter 6. Finally, the conclusion of this thesis including a summary of the findings, limitations as well as an outlook on future research is given in chapter 7.

2 Theoretical background

In this chapter, the relevant theoretical background on the main topics addressed in this thesis is outlined. It creates the basis for the literature review as well as the development of the optimization model which are both used to answer the research questions. Firstly, definitions for assembly line sequencing as well as collaborative robots are provided, as both terms are central to the research questions. Subsequently, reinforcement learning is explained in detail.

2.1 Assembly line sequencing

Assembly line sequencing is an active area of research faced in operations management with the goal of optimization. It impacts the productivity and thereby the cost-effectiveness of an assembly line. Sequencing refers to identifying an optimal task order to finish an assembly task. Defined objectives and constraints must be considered when finding a sequence. Often, a short cycle time achieved through, for example, a minimized idle time is the objective for optimization. Constraints can be given by, for example, dependencies of tasks, availability of resources, or product variations. There are various approaches to solving the assembly line sequencing problem. The choice of model is dependent on the specific characteristics of the assembly line as well as on the company's objectives. When developing a model, various factors should be considered: The assembly plan, as e.g. mixed, batched, or single, the physical line layout, as e.g. straight, parallel or U-shaped lines, the work transportation method, as e.g. conveyor or pallet-based, as well as variations in processing workstations, as e.g. manual, robotic or hybrid stations. (Kamal et al. 2011)

2.2 Collaborative robots

Collaborative robots, also called cobots, are robots developed to operate alongside human workers in a shared workplace. When applied in an assembly line, they can support the human workers in physical as well as in cognitive tasks. The implementation of collaborative robots in an assembly line has several advantages. As they are capable to work alongside humans without physical barriers, they can work together with the human worker to complete an assembly task. For example, they could be used to adapt tools according to the movements of the human worker. Additionally, they can support or even substitute humans in dangerous or heavy tasks, as the handling of heavy loads. Human can be supported in cognitive tasks, for example, by the visualization of alternative decisions. Collaborative robots can lead to an enhanced productivity and efficiency, reduce operation costs, improve the product quality, as well as the ergonomic working conditions for the human workers. However, there are also

potential disadvantages to consider. The implementation of collaborative robots can come with high cost as it may require significant investments in training as well in as infrastructure. In addition, due to possible limitations in handling certain types of materials or working in certain environments, collaborative robots might not be able to perform all tasks that human workers can. These technological limitations need to be compensated by the human workers. They need to adapt to the new working environment by acquiring and improving certain skills, for example IT skills. (Bragança et al. 2019)

2.3 Reinforcement learning

Reinforcement learning is an active area of research and commonly used to train artificial intelligence systems to perform tasks such as playing games or controlling robots. The solution of complex optimization problems that are difficult to solve with traditional algorithms is possible. Reinforcement learning is one of three broad areas of machine learning alongside with unsupervised and supervised learning. Unlike the two other frameworks, the goal of reinforcement learning is not to cluster or label data, but to train an agent to take actions to maximize a reward. There is no information provided about which action to take, instead the agent must discover which actions yield the highest reward in what situation by trying them out. The taken actions define the reward as well as the situation change. The basic principle of reinforcement learning is illustrated in figure 2-1. In the following, the main basic elements of a reinforcement learning system are described. The information in this section provides an introduction to reinforcement learning and can be looked up in detail in Sutton and Barto (2018).

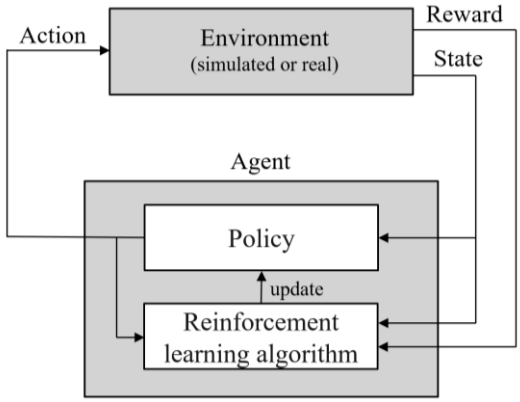


Figure 2-1: Basic reinforcement learning system.

Environment

The environment is the system being controlled by the agent. It might be a real physical environment the agent interacts with, or a simulated model of the environment. Often, for

training the agent, simulated environments are used. The environment provides observations and reward resulting from an action of the agent. At each time step, the agent receives an observation and chooses an action. The environment responds by transitioning to a new state and providing a scalar reward signal. For the definition of the state, the framework of Markov decision processes (MDPs) is typically used. Sequential decision-making can be formulated with MDPs, in which actions have an impact not only on the immediate reward but also on following states, and consequently on the overall long-term rewards.

Model

It is possible to provide a model of the environment to the agent, independent from if a real or simulated environment is used. This allows the agent to make inferences about the behavior of the environment. Agents can be categorized in model-based and model-free agents. Model-free agents learn by pure trial-and-error, while model-based agents already have knowledge of parts of the environment.

Policy

The policy defines the behavior of the agent at a point in time. During training, the agent updates its policy based on the reward it received for an action in a specific state of the environment. In the next time step, the new policy is then used to determine the action considering the recent environment state. By that, the agent should learn a policy that maximizes the cumulative reward over time. Policies can be represented by simple functions or lookup tables, however, for increasingly complex problems, neural networks are used. A neural network is generally a universal function approximator and consists of a group of connected nodes. The higher the number of nodes and connections, the more complex functions can be approximated, but the longer the identification of parameters takes. Policies are, simply said, a function from the environmental state to actions. They are the core of reinforcement learning agents as they determine the behavior. Usually, policies are stochastic and specify a probability for each action. The training of the agent has the goal to find the best policy parameters.

Reward signal

The environment provides the reward to the agent as a single number on each time step based on the state transition of the environment. The state transition is a result of the action that the agent took before. The only objective of the agent is the maximization of the total reward over time. Usually, reward signals are stochastic functions of the environment state and the taken action.

Value function

The value of a state of the environment is determined by the total accumulated reward that the agent can expect in the future. While the reward signal defines the immediate desirability of the current environmental state, the value indicates the long-term desirability considering the states and corresponding rewards to follow. For example, it is possible that the current state of the environment provides a low reward, however, its value might be high when it is regularly followed by states with high rewards. The value of a certain state is estimated and re-estimated by the agent from the observations it made throughout the previous training. As the agent seeks to maximize the provided reward over time, or with other words, to achieve states with a high value, the efficient estimation of the value of a certain state is crucial for most of the reinforcement learning algorithms.

Reinforcement learning algorithms

The learning process of an agent is guided by a reinforcement learning algorithm. There are various reinforcement learning algorithms which are often already implemented in commercial software such as *Matlab*. Simplified, they can be categorized in three groups: policy-based, value-based, and actor-critic approaches. Policy-based algorithms train a neural network with the state as input and the action as output. This neural network is called actor as it determines the action to take. One of the challenges when using policy-based algorithms is that they can converge on a local maximum reward rather than the global maximum reward.

Value-based algorithms train a neural network with the state and one possible action as input and the value of the state as output. This network is called critic as it criticizes the action of the agent. As the network has no action as output, which is needed to represent a whole policy, another step is necessary. The value of every possible action is checked and then the action with the highest value is finally chosen. However, this makes value-based algorithms not suitable for continuous action spaces as it is not possible to calculate the value of an infinite set of actions.

The combination of policy-based and value-based algorithms is called actor-critic. Then, the critic only defines the value of the action that the actor chose. Hence, continuous action spaces are possible. Afterwards, the critic determines the accuracy of its value prediction based on the reward provided by the environment for the chosen action and updates its network. The actor updates its network based on the corrected value of the chosen action provided by the critic. In other words, it uses the value provided by the critic instead of the reward to update its policy.

3 Methodology

In the following, the methodology used in this thesis is described in detail. Section 3.1 explains how to conduct a systematic literature review. This methodology is used to answer the first research question. In section 3.2, the methods modeling and simulation are outlined. They are used to answer the second research question

3.1 Systematic literature review

A systematic literature review is performed to investigate the state of the art of addressing the assembly line sequencing problem within the defined scope. This methodology was chosen to answer the first research questions as it offers a replicable, scientific and transparent process with minimized bias (Tranfield et al. 2003). By the identification of the available knowledge, the evaluation of strengths and weaknesses and the determination of literature gaps, the first research question can be fully answered. The recommendations of Tranfield et al. (2003) to perform the systematic literature review in three steps are followed in this thesis. The three steps are described in the next sections, starting with *Planning the review*. *Conducting the review* is the second step explained subsequently. The last step, *Reporting and dissemination* of the results, is outlined lastly. The information provided in these sections originates from Tranfield et al. (2003), Snyder (2019) as well as Seuring et al. (2020) and can be found there in further detail.

Planning the review

The first step to plan a systematic literature review is to form a review panel. It has the task of directing the process as well as resolving disputes about the inclusion or exclusion of documents. The members of the review panel should be experts in the study area and the methodology itself.

After forming the review panel, the review protocol is specified. It is composed of the research questions, the chosen database or databases as well as the keywords for the literature search, and the inclusion and exclusion criteria. The review protocol ensures objectivity by documenting all decisions on these items. The clear definition of the research questions is crucial as it guides the remainder of the review. The number of used databases as well as the selection of keywords depends on the review scope. The keywords are defined based on the research questions. Their selection is fundamental as it determines the review quality. For example, too broad terms lead to the finding of numerous publications beyond the review scope. On the other hand, the selection of too specific terms results in the non-identification of

potential relevant documents. The existing knowledge of the research topic can then not be assessed completely. This may lead to inaccurate conclusions, as for example regarding literature gaps. The terms are searched, for example, in the title, abstract as well as the keywords of a study. Often, various groups of terms are specified and then combined in a search string. Then, one term of each defined group must be found in order to include the corresponding publication in the review. The identified documents are then filtered according to the inclusion and exclusion criteria. Only documents meeting the inclusion criteria and not meeting any of the exclusions criteria are relevant for the review. Often, the subject area, the year of publication, the language or the type of document such as, for example, journal article or conference paper, are used as criteria.

Conducting the review

After the planning of the review, it is conducted starting with a literature search according to the defined review protocol. With help of the search string, a list of publications is obtained from the chosen database or databases. This list is then filtered regarding the defined inclusion and exclusion criteria. There are various approaches to scan the remaining documents. In case a low number of publications is retrieved, reviewers might choose to read the full text of each document. On the one hand, this leads to accurate decisions regarding the relevance for the review. On the other hand, it is time-consuming. Therefore, in particular when the number of retrieved documents is high, reviewers might choose to proceed in stages. Firstly, the title and the abstract of the publications are read and articles being clearly out of the scope of the conducted review can be excluded. Finally, in the second stage, the full text is read and the ultimate decision on the relevance of the article is taken. The documentation of the selection process including the number of publications excluded at each stage as well as the reason for exclusion is important for full transparency. However, the decision on the relevance of an article remains relatively subjective. For this reason, it is beneficial if more than one reviewer selects the documents. In case of disagreements between the reviewers, the review panel can be consulted.

Reporting and dissemination

The identified relevant publications are analyzed in the third step of a systematic literature review. The analysis can be divided into two parts starting with the descriptive analysis. It includes, for example, the display of the distribution of publishing years, journals, the type of study, and the authors. This is followed by the content analysis or also called thematic analysis. Per category, the content of the chosen publications is outlined, the level of shared consensus among researchers is assessed, and emerging themes are identified. To do so, the documents are categorized first. There are two main approaches to determine the categories. When the deductive approach is chosen, the categories are specified from existing theories before the

analysis. This should be done when there are already many theories of the research topic existing. Otherwise, or if there are only a few theories, the inductive approach can be selected. Then, the categories are defined during the analysis of the relevant documents under constant review and comparison in an iterative process. As the categories are directly derived from the publications, they are constantly extended and adapted.

The output of descriptive and content analysis can differ depending on the scope of the review. For example, it might be a timeline for predicting where a research area is headed, a comparison of different concepts that could serve as a basis for the development of a theory, the identification of new theories, research agenda, or research propositions, or the disclosure of literature gaps. The contribution of the literature review can provide a foundation upon which the area of study can be further advanced.

3.2 Modeling and simulation

To answer the second research question of this thesis, a model to solve the assembly line sequencing problem will be designed. A reinforcement learning algorithm will be used to solve the optimization problem and is further described in section 2.3. In order to apply the algorithm, a model of the environment is required which can be used for simulating the assembly process. This is necessary in order to evaluate the outcome of an assembly process. The information is used by the reinforcement learning algorithm to find the best sequence of tasks.

Modeling and simulation are quantitative methods. They are used to predict how a system will behave under certain conditions. Robinsons (2008a) defines conceptual modeling in his research. A model is abstracted from a real or proposed system and corresponds to a simplification of the system. Assumptions about the real world might be taken regarding uncertainties. Modeling and simulation have the advantages that large and complex situations can be analyzed in a short time and with low cost in comparison to real world experiments. However, the simulation results are highly dependent on the extent to which model and reality deviate from each other. A certain degree of inaccuracy is always present as simplifications and assumptions were made.

Robinsons (2008b) proposes a framework for conceptional modeling consisting of five key activities. The first step is to understand the problem situation. From this, the aim of the model and the general project objectives are derived. In the next steps, the model outputs and the model inputs are identified. The model outputs correspond to the response of the system and the model inputs to the experimental factors. Finally, the model content in terms of its scope and level of detail is specified. The identification of simplifications and assumptions is done during the process.

From the conceptional model, the computer simulated model is derived. Within this thesis, the model code is developed in the programming environment *Simulink* based on the programming language *Matlab*. The developed model of the environment is described in detail in section 5.2.

4 Systematic literature review on assembly line sequencing with collaborative robots using reinforcement learning

In order to answer the first research question and to build up a foundation to develop an optimization model which is used to answer the second research question, a systematic literature review is carried out. To identify already developed models using reinforcement learning in the context of collaborative robots as well as potential literature gaps, the existing research in this context needs to be reviewed. The methodology of a systematic literature review, which is carried out in this chapter, is described and justified in detail in section 3.1. In the following, section 4.1 documents the selection of studies with the final list of relevant documents as output. The descriptive analysis of these documents follows in section 4.2 and the content analysis per category in section 4.3.

4.1 Studies selection

This section documents the process of study selection according to the first two steps of the selected methodology, *Planning the review* and *Conducting the review*, which are described on a general level in section 3.1. The first step is to form a review panel. As the scope of the review is limited, it consists of only one member, namely the supervisor of this thesis, Mirco Peron. In the next step, the review protocol is defined. It can be found in appendix A. It contains, among other, the first research question of this thesis which should be answered by this review, see also section 1.3.

> What is the state of the art of addressing the assembly line sequencing problem with reinforcement learning when using collaborative robots?

As database for the literature review, *Scopus* is chosen. It offers an interdisciplinary compilation of peer-reviewed literature and is one of the main sources for citations (Mongeon and Paul-Hus 2016). The keywords, which are used for the literature search on the database, are listed in table 4-1.

Table 4-1: Search keywords.

Group 1	Group 2	Group 3
“assembly”	“sequenc*”	“cobot*”
	“ALSP”	“collaborative robot*”
		“co-bot*”
		“human-robot collaboration”
		“HRC”

The search keywords are based on the research question. They contain synonyms and abbreviations of the main terms. If only the base of a term is searched for, a star is added in the end of the word base. Quotation marks indicate loose phrases. There are three groups of keywords. In the first one, only *"assembly"* is defined as the review is limited to this context. The second group contains the word base and abbreviation of sequencing as this is the main topic of this review. In the third group, terms to describe collaborative robots are stated. In total, the chosen keywords cover the entire scope and lead to a specific output of documents from the database without being too broad. The search string for the database is stated in the review protocol in appendix A. It is generated by combining the keywords using Boolean operators based on the defined groups. In that way, one match with a keyword of each group is necessary.

As last point, the review protocol outlines the inclusion and exclusion criteria which are used to filter the obtained list of publications from the database. Only articles published in English are taken into consideration, while the subject areas are restricted to engineering, computer science, decision sciences, business, management and accounting, as well as economics, econometrics and finance. To ensure the high-quality of publications, the document type is also considered as a criterion, with journal articles being the preferred choice. However, due to the anticipated limited number of relevant studies, conference papers will also be included in the review.

The *Scopus* database search identified 89 documents. They were filtered according to the inclusion and exclusion criteria. Within this process, 17 articles were excluded. The title and abstract of the remaining 72 publications were read to assess whether they are within the review scope leading to 62 exclusions. Finally, the full text of the remaining 10 documents was read and 4 final articles were identified to be included in this review. Documents were excluded based on the scope defined in section 1.3, for example, because of insufficient study details, such as the absence of a developed model to address the optimization problem, or when a study falls outside the scope of this thesis, such as not focusing on sequencing or not using reinforcement learning. An overview of the study selection process with indication of the reasons for exclusion is illustrated in figure 4-1 in the form of a PRISMA diagram based on Moher et al. (2009). Given that the document filtering process was conducted by a single reviewer, it retains a certain degree of subjectivity. However, any doubts regarding inclusion or exclusion were discussed with the review panel. This approach ensures quality and a validation in alignment with the objectives of this thesis. To achieve reliability, the study selection process was carefully documented which allows for transparency.

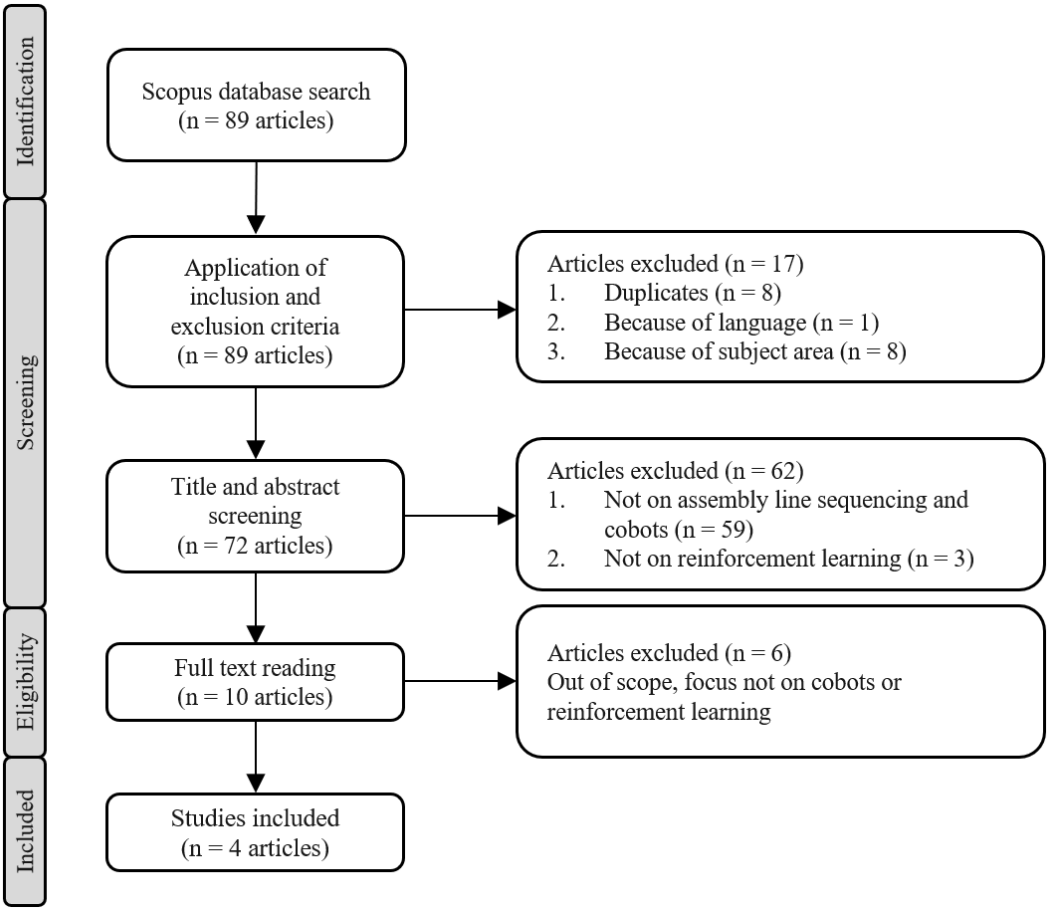


Figure 4-1: PRISMA diagram of the study selection process based on Moher et al. (2009).

In table 4-2, the output of the studies selection is shown in chronological order. Based on this list, the descriptive and the content analysis of the selected documents is carried out according to the third step of the methodology as explained in detail in section 3.1. The analyses follow in the next sections.

Table 4-2: List of final documents.

Title	Authors	Publication year
A reinforcement learning method for human-robot collaboration in assembly tasks	Zhang et al.	2022
Robust Adversarial Reinforcement Learning for Optimal Assembly Sequence Definition in a Cobot Workcell	Alessio et al.	2022
Robust Assembly Sequence Generation in a Human-Robot Collaborative Workcell by Reinforcement Learning	Antonelli et al.	2021
Mastering the working sequence in human-robot collaborative assembly based on reinforcement learning	Yu et al.	2020

4.2 Descriptive analysis

In the following, the descriptive analysis results of the systematic literature review are presented and discussed. In particular, the distribution of the studies, starting with the temporal and geographical distribution, then the distribution among journals and the most frequent used keywords are shown.

The publication years of the selected articles are shown in figure 4-2. As there was no starting date within the inclusion criteria of this review, the eraliest research on task sequencing with collaborative robots using reinforcement learning was published in 2020. Subsequently, the analysis period of this review is from the year 2020 until the year 2022. As visible in the figure, the number of publications was stable in 2020 and 2021 and increased in 2022.

The results imply that the scope of this review is an emerging topic as no documents were published before 2020 and the number of documents within the scope of this review is rather low. This means that only a limited amount of research was conducted in this area so far. Even though assembly line sequencing was studied for years, the embedding of collaborative robots and the usage of reinforcement learning are new approaches which offer great potential. Reason for this is, on the one hand, the relatively novel approach to use machine learning for optimization problems, and, on the other hand, the nascent stage of implementation of collaborative robots resulting in lack of wide-scale uptake in assembly lines. As a result, researchers may have perceived the topic as having a low practical value, which might have led them to pursue other areas of research instead. However, the rising number of publications in 2022 emphasizes the increasing relevance and shows that the topic becomes more and more in focus of research. Therefore, the number of publications can be expected to further increase in the next years. The selected documents only include journal articles. As a low number of documents was expected, conference papers were included in this review.

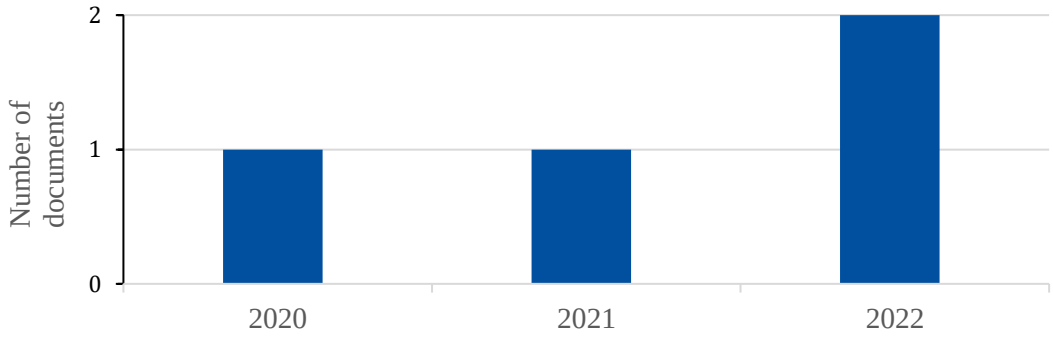


Figure 4-2: Year-wise distribution of publications (n = 4).

Figure 4-3 shows the geographical distribution of the selected publications, which displays the number of documents originating from each country. As visible, the majority of the research

was conducted in Italy (2). The remaining articles were written by authors from China (1) and the USA (1).

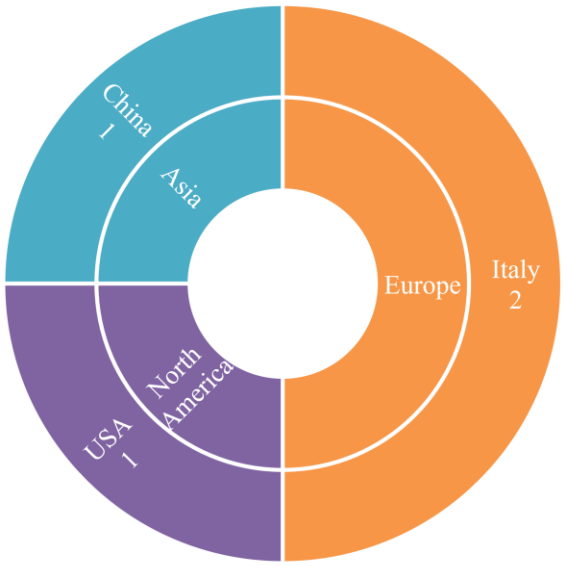


Figure 4-3: Geographical distribution of publications (n = 4).

The results show that researchers affiliated with institutions in Italy, China and the USA are interested in the topic of research. One reason that these countries are especially interested in the automation of assembly lines might be that China (28.7%), USA (16,8%) and Italy (2.1%) together have a share of 47,6% of the global manufacturing output (Richter 2021)[1]. Hence, these countries might focus on funding programs for research in this area and the practical interest of researchers might be higher in comparison to other countries.

The publishing journals of the documents can be seen in figure 4-4. All articles were published in different journals. This indicates that the interest in the topic of this review is not fouces on one specific academic sector.

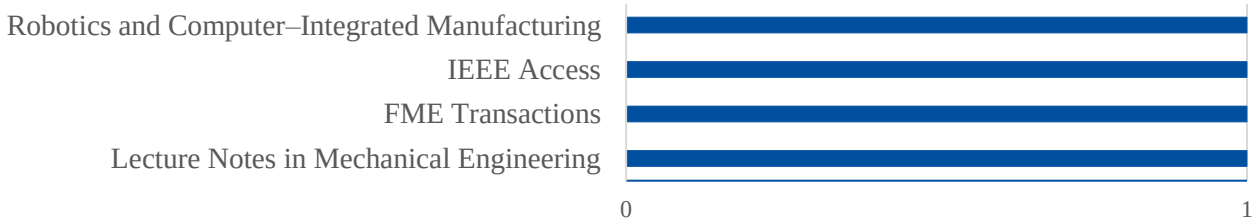


Figure 4-4: Journal distribution of publications (n = 4).

[1] Numbers are from 2019 and measured on a value-added basis in U.S. dollars

The most frequent used keywords are shown in figure 4-5. In total, the selected articles contain 12 distinct keywords. All documents list the keyword *Human-robot collaboration*.

Figure 4-5: Most frequent keywords.

4.3 Content analysis

The content analysis follows the third step of the methodology described in section 3.1. It is carried out by category. In this section, the findings are structured in the categories and discussed. The determination of the categories was done by following the inductive approach for category selection as the topic of this review was not dealt with in detail yet. The approach is described in detail in section 3.1. The following categories were defined: Firstly, the objectives for optimization for each of the models are compared. This provides an overview of the assembly system characteristics that have been the primary focus of optimization in prior research. In the next section, the environment with which the reinforcement learning algorithm interacts is outlined for each publication, followed by the reinforcement learning algorithm itself. The last category is the application area as well as validation of the studies. Per category, similarities, disagreements and potential literature gaps are identified. The theoretical background of reinforcement learning is described in section 2.3.

4.3.1 Objective for optimization

As only models using a reinforcement learning algorithm were in the scope of this review, the objective to be optimized is determined by the reward provided within the models. This may involve a single or multiple objectives. The models of Yu et al. (2020), Antonelli et al. (2021) and Alessio et al. (2022) pursue a single objective. Yu et al. (2020) aims to minimize the completion time of an assembly task for maximizing the working efficiency. For this, the best sequence of tasks and the best operator for a specific task are chosen so that the entire assembly process is finished in the shortest time. The same as in various of the other studies, operator refers to both human workers and robots. The goal of Antonelli et al. (2021) and Alessio et al. (2022) is also to find the task sequence with the lowest completion time. For example, in the model of Alessio et al. (2022), this is achieved by providing a negative a reward for idle time. Constraints, such as a movement out of the used grid environment, are met in the same way,

by providing a negative reward for these actions. In contrast to these studies, the research of Zhang et al. (2022) follows a multi-objective approach. Besides to the minimization of the assembly completion time, a constant task difficulty should be achieved. To do so, the reward function considers the time spent for each action as well as the difficulty of executing the operation, individually defined for both human and robot.

The results show that the minimization of the completion time is in focus of researchers. While the majority follows a single-objective approach, only Zhang et al. (2022) consider multiple objectives by taking the difficulty to solve a task into account. Ergonomic factors among others are up to a certain amount included in the difficulty of a task. However, a pure focus on ergonomics was not a subject of research so far.

4.3.2 Environment

The environment defines the state transition following an action and provides the reward to the agent. In all the selected documents, the environment is composed of a Markov Decision Process (MDP). The MDP represents the set of all feasible tasks at a certain point in time. Yu et al. (2020) uses a chessboard game to formulate the assembly process. The rules in the game reflect the required constraints. Alessio et al. (2022) uses a *Matlab* built-in grid world which shows the sequence of possible assembly steps. The agent explores the grid world and searches for terminal states which indicate the completion of the assembly process. The observation provided to the agent is thereby composed of four channels: one for obstacles which define the structure of the grid world, one for the path covered by the robot, one for the path covered by the human, and one for information on whether a terminal state was reached. However, the use of a grid world as environment is also a limitation of the model. While it provides a clear visualization of the results, it reflects a real scenario only with restrictions as the behavior of one agent cannot change the state of both the agents. In the model, individual states are assigned to the agents, and they cannot be in the same state at the same time. Zhang et al. (2022) express the assembly task by vectorizing the bill of material of the product. By this, sequence constraints are established.

All researchers use a MDP environment which is usual when applying a reinforcement learning algorithm. Also, they all implement the sequence constraints. However, different approaches are identified in the literature, as for example, the use of a grid world.

4.3.3 Agent

The models of Yu et al. (2020) and Antonelli et al. (2021) both use single-agent reinforcement learning. The collaborative robot is thereby represented by the agent. In contrast to that, Alessio et al. (2022) and Zhang et al. (2022) both use multi-agent reinforcement learning which is an

extension of the single-agent approach. Both robot and human worker are then represented by an agent. The reinforcement learning algorithm used to update the policy of Yu et al. (2020) is inspired by Google Deep Mind's Alphago Zero which is an algorithm based on Monte Carlo Tree Search. Antonelli et al. (2021) uses Q-learning which adjusts an approximator for the best action-value function. Alessio et al. (2022) apply proximal policy optimization (PPO) which is a policy gradient reinforcement learning method where the weights of the policy are estimated by the gradient ascent algorithm. As they focus on the behavior of the robot, the robot agent is trained to use the best policy while the human agent follows a random objective. In this way, the robot has to correct the mistakes of the human. Zhang et al. (2022) uses a deep deterministic policy gradient (DDPG) method as well as Q-value based deterministic strategy. Noise is added to the actor network of the agent representing the human to increase the robustness of the system when influenced by uncertain factors.

The results show that the researchers were unanimous about the choice of reinforcement learning method. Using multi-agent reinforcement learning comes with several advantages. Different rewards can be provided to the agents and thereby a different focus can be set. In addition, it enhances the robustness of the system as small mistakes, which the human might do during a real assembly task, can be simulated. As explained in section 2.3, reinforcement learning agents can be categorized as model-free and model-based. All researchers use a model-free algorithm which means no model of the environment is provided to the agent. In addition, reinforcement learning algorithms can be categorized as policy-based, value-based, and actor-critic approaches. The majority of the selected studies use an actor-critic approach (Yu, Alessio, Zhang), while Antonelli et al. (2021) uses with Q-learning a value-based approach.

4.3.4 Application area and validation

All researchers developed a model to find the sequence of tasks to assemble a product by a human worker and a collaborative robot with shared workspace at one workstation (Yu, Huang, and Chang 2020; Antonelli et al. 2021; Alessio, Aliev, and Antonelli 2022; Zhang et al. 2022). However, the models support different levels of interaction between human worker and collaborative robot as well as their abilities to react to foreseen behavior.

For the level of interaction between human and robot, three approaches can be found in the models. When cooperating, the robot and human are working independently on different tasks to complete a joint assembly goal. When working in collaboration, they complete a shared task together. The support of one of the two or both at the same time is possible. The research of Yu et al. (2020) and Alessio et al. (2022) only consider cooperating while the model of Antonelli et al. (2021) only supports collaboration, meaning that all operations to complete an assembly task are conducted by human and robot together. Both cooperation and collaboration are considered by Zhang et al. (2022).

The developed models are validated in different ways. Yu et al. (2020) performed extensive numerical studies with 100 different randomly generated assembly chessboards as well as a simulated case study to assemble a desk. Furthermore, the developed model was compared with a conventional mathematical optimization method regarding the assembly completion time and the time needed by the model to take a single decision. Their model outperformed the conventional optimization method in these points. The authors see a further improvement of their model in the consideration of random robot failures and human errors, as their model does not consider possible mistakes of the human or random failure of the robot.

Antonelli et al. (2021) validated their model by testing it in a real-life experimental setup. The case study included the mounting of two flanges on a base. Their model allows disassembly actions when assembled wrong due to an error and is therefore able to cope with changes. However, if the deviation from the proposed sequence is too high, they found that their model might not be able to adapt. For a further improved model, the authors suggest as next step using an adversarial reinforcement learning algorithm designed for games. By that, the authors expect a better handling of unforeseen human interferences in assembly tasks. In their opinion, when using an adversarial algorithm, the reinforcement learning agent will aim to complete the assembly job despite any interference as it considers the human operator as an opponent. In the authors' opinion, this leads to an enhanced robustness. The suggested approach is followed with some deviations by Alessio et al. (2022). However, they disagree partly with the claims of their fellow researchers. They argue that a traditional adversarial approach, in which both agents compete against each other as in a game, is not fully appropriate to complete an assembly task as both agents must work toward a common goal. A robust design requires that the robot adjusts its next actions when the human makes a mistake. In their opinion, the use of an entirely adversarial approach could, in some cases, make the completion of the assembly process impossible. This issue is solved by the subtraction of 10% of the human's total reward from the total reward of the robot. Then, a mistake of the human leads to a decreased reward of the robot which prompts the robot to adjust its actions based on the best strategy to complete the assembly process. However, their algorithm was only tested in a virtual environment for a limited set of tasks. According to the authors, a modification of the training environment as well as the conduction of a case study with a real assembly process will improve their model.

Zhang et al. (2022) validated their models by testing it in a real-life experiment. The behavior of the agent representing the human is shown on the display to the operator in real-time who can then perform the tasks in the proposed sequence. Also, tools which might be required for certain tasks as well as the time to operate them are considered. Their model is able to adapt to unforeseen changes during the process and is capable of finding an alternative sequence to achieve the assembly target. To improve their model, the authors want to implement the

computation of operating time and resource consumption and apply their model to more complex assembly tasks.

The results show a progression in the development of the models. The model of Yu et al. (2020) does not consider human mistakes, while Antonelli et al. (2021) takes human failures into account, however, with limited robustness. Alessio et al. (2022) enhances robustness with their adapted version of adversarial multi-agent reinforcement learning. Zhang et al. (2022) developed the first model pursuing multiple objectives taking the difficulty of a task into account. Also, the considered scenarios have different levels of complexity.

5 Development of an optimization model

In order to answer the second research question, an optimization model using multi-agent reinforcement learning for solving the sequencing problem is developed. The model is created based on the findings of the systematic literature review in chapter 4. As the focus on ergonomic factors has not been in focus of research so far, the model follows two objectives. Next to the optimization of completion time, the ergonomic stress level of the human worker should be minimized. The goal of the model is to find a task sequence in connection with the task allocation to robot and human worker to complete an assembly process considering the two defined objectives.

In section 5.1, the model structure is explained followed by a detailed description of the environment in section 5.2 as well as a description of the agents in section 5.4. The training and simulation of the model is presented in section 5.5. The application of the model on an exemplary task and its discussion is outlined in chapter 6.

5.1 Model structure

In reinforcement learning, as described in section 2.3, an agent interacts with an environment. The state of the environment is changed, and a reward is provided based on the actions of the agent. Based on the state transition and the reward, the agent updates its policy. Reinforcement learning has been extensively studied at the trajectory level of robots. As shown in the literature review in chapter 4, it is a relatively new approach to use it for solving the assembly line sequencing problem. In multi-agent reinforcement learning, two or more agents interact with an environment. Within this thesis, two agents are used. One represents the collaborative robot and the other one the human worker. Therefore, the model is applicable for workstations with one human worker and one collaborative robot and considers ergonomic factors. To do so, multiple objectives are pursued, namely a short assembly completion time and a low ergonomic stress level for the human. The agents can be classified as model-free, meaning that no model of the environment is provided to the agents, also see section 2.3. The reinforcement learning model supports only cooperating of human and robot, meaning that both operators work independently to complete a joint assembly process. The model is programmed using *Matlab* and *Simulink*. Functions of the *Reinforcement Learning Toolbox™* (2023) are used. The core reinforcement model structure can be seen in figure 5-1, which is a screenshot of the *Simulink* programming environment. The environment and both agents representing robot and human as well as the data transfers between them are visible. Each time step, the operators provide their action to the environment resulting in its state transition. Then, an observation, which describes the state transition, as well as the respective rewards are supplied to both operators. In addition,

Boolean information indicating the completion of the assembly process is given. The development of the environment, based on the methods described in section 3.2, and of the agents are described in the next sections in detail.

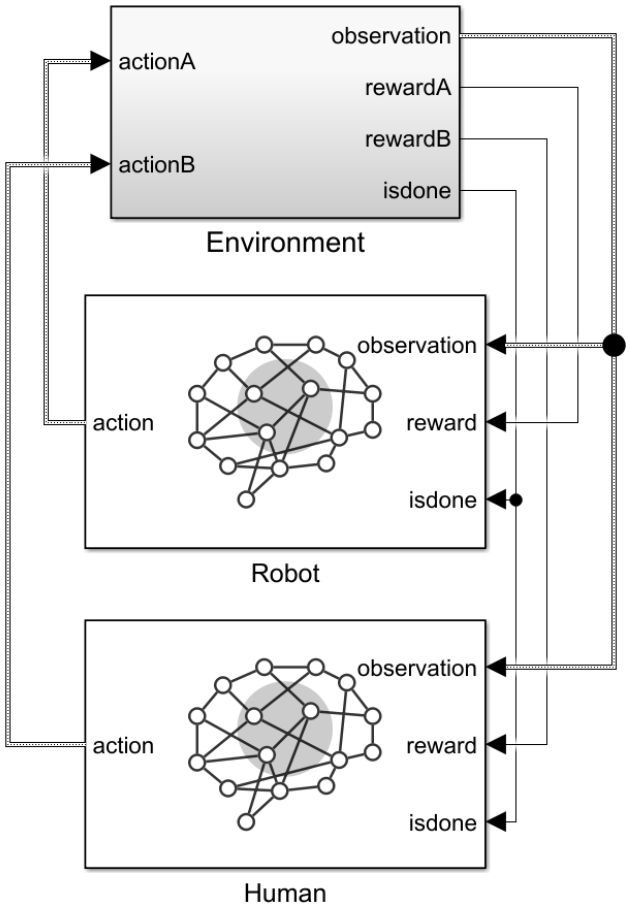


Figure 5-1: Model structure.

A *Matlab* script is used to initiate the model in the figure above. It is provided in appendix B. In the script, firstly, parameters are defined, and the action space as well as the observation space are established in *Bus* format which are later used to set up the agents. Then, the *Simulink* model is created as well as the function *resetEnvironment* is set as reset function for the environment. Afterwards, the two agents are generated, and training parameters are determined. Finally, functions to validate the environment, to start the training of the agents or to load pre-trained agents as well as to simulate an assembly task with trained agents are given.

The function *resetEnvironment* is attached in appendix C. It resets the *Simulink* model when a new simulation is started by defining the initial state of the environment.

5.2 Environment

The environment is modelled using the framework for conceptional modelling of Robinsons (2008b) which is described in section 3.2. The conceptional model of the environment is then translated into programmed model code. The model should represent a physical assembly process at one workstation and should be used to train two reinforcement learning agents. Assembly constraints should be represented in the model. The two actions a_R and a_H chosen by the agents should serve as model inputs. The model outputs need to provide sufficient information about its state for the agents so that they are able to adapt their policy to find the best task sequence. They are the response to the inputs of the agents. As the agents do not share information among themselves, the model should provide the number of task d_R and d_R which both are currently doing. The time which is needed by robot and human to finish each task, is provided within the vectors t_R and t_H. The ergonomic stress level to do a task for the human is provided within the vector e_H and the dependencies of each assembly task within the matrix C. As a requirement to use reinforcement learning, the rewards r_R and r_H must be provided to each agent dependent on the chosen action. Finally, information is needed about whether or not the assembly process was finished. This is given within the binary value f. The notations of all model inputs and outputs are listed in table 5-1.

Table 5-1: Inputs and outputs of the environment model.

Type	Notation	Format	Explanation
Input	a_R	Scalar	Action chosen by the agent representing the robot
Input	a_H	Scalar	Action chosen by the agent representing the human
Output	d_R	Scalar	Task which the agent representing the robot is currently doing
Output	d_H	Scalar	Task which the agent representing the human is currently doing
Output	t_R	Vector	Remaining time for each task for the robot
Output	t_H	Vector	Remaining time for each task for the human
Output	e_H	Vector	Ergonomic stress level for each task for the human
Output	C	Matrix	Dependencies of each assembly task
Output	r_R	Scalar	Reward for the agent representing the robot
Output	r_H	Scalar	Reward for by the agent representing the human
Output	f	Boolean	Binary value indicating completion of assembly task

The model should have a sufficient level of detail to simulate the assembly process to find the best task sequence as well as the best allocation of tasks. In addition, it should offer the possibility to exclude robot or human to do a certain task, meaning that specific tasks must be performed by the robot or by the human. This can be indicated by setting the corresponding

task time to zero. The following simplifications and assumptions are taken when formulating the conceptional model. The model represents a MDP as all information about the state of the environment required for decision-making are provided. This includes that only the current state of the environment is considered, and the history of past states and actions is discarded. The possible actions for the agents are finite. Each time step, they can choose a task number or choose to wait and doing nothing. The operation in discrete time steps is another simplification. Actions are only possible and observations are only provided after a certain time interval. Also, the process is assumed to be stationary and other possible influences on the assembly process are not considered, as for example the availability of components and the usage of tools. As the state of the environment is only represented by a limited number of variables, the model freedom and dependencies might be omitted. The model does not support collaboration between robot and human to work on the same task together. Besides, once started a task, the agent must finish it in the predefined time before choosing the next task. Difficulties which might appear when performing the task and lead to a longer task time are not taken into account, similarly to when a task is finished earlier as expected. Therefore, accurate task time estimations for both robot and human are necessary in order to provide an accurate model of the assembly process.

After the outline of the conceptional model, the programmed model code is explained in the following. The environment subsystem of the *Simulink* model can be seen in figure 5-1 which represents the whole customized programmed environment. The opened subsystem is shown in appendix D. For more clarity, data operations are performed on this level. This includes handling of the *Bus* datatype as well as ensuring the data transfer between different rates and tasks. The opened next subsystem is visible in figure 5-2. It shows all inputs and outputs to a *Matlab Function block* in which the state transition of the customized environment is programmed.

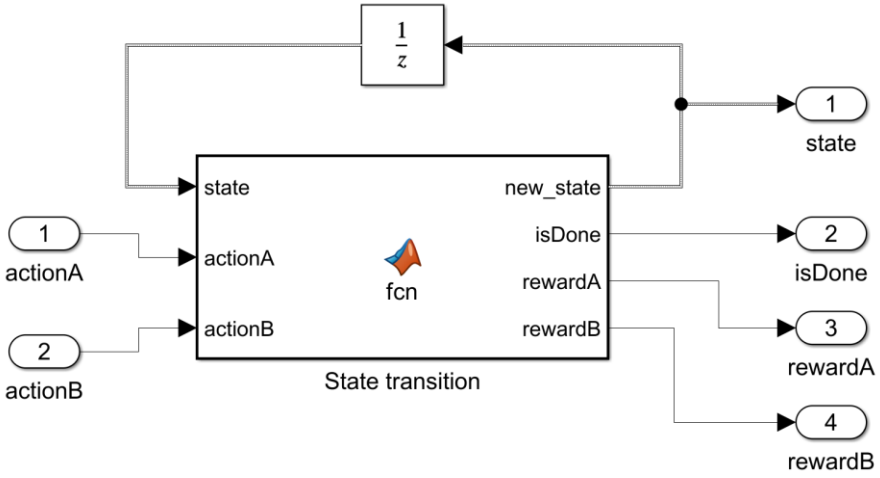


Figure 5-2: Subsystem with *Matlab Function block*.

The variable *state* contains the variables d_R, d_H, t_R, t_H, e_H and C explained in table 5-1 in *struct* format. The state after transitioning serves as starting point for the next state transition. To enable the use of an output as an input, a delay block was added in between the two. The program code of the *Matlab Function block* is attached in appendix E.

First, the rewards for both agents for several cases are defined, for example if the assembly process is finished or if the agent chooses an unavailable task. Positive and negative rewards are possible. The choice of rewards is further justified in section 5.3. Then, further variables are initialized. Within a for-loop, the chosen action is validated and, dependent on the case, the corresponding reward is provided. This is performed for both agents one after the other. It is checked if the action leads to idle time. This would be the case when no action is chosen even though the agent is free. In this case, $a_{R,H} = 0$ and $d_{R,H} = 0$. Then, several checks are performed to determine if the chosen action is illegal. It is examined if an action was chosen even though the agent is busy. This is the case when $a_{R,H} \neq 0$ and $d_{R,H} \neq 0$. The next check is the sequence dependency of the chosen task. If the task is dependent on the completion of another task before, the corresponding column lists at least one 1, so the equation

$$C(i,j) = 0 \text{ for } i = a_{R,H} \text{ and } j = 1 \dots m \text{ with } m = \text{total number of tasks}$$

must be true. Finally, it is assessed if the chosen task is currently being done by the other agent, already finished or must be done by the other agent. In all those three cases, the corresponding task time $t_{R,H}(a_{R,H}) = 0$. If an illegal action was selected, it is set to $a_{R,H} = 0$ and is thereby ignored. In case the agent is currently performing a task and did not choose an illegal action, the time corresponding to one time step is deducted from the task time $t_{R,H}(a_{R,H})$. If this leads to $t_{R,H}(a_{R,H}) = 0$, then the variable for the task which the agent is currently doing is set to $d_{R,H} = 0$ and the dependencies of other tasks on the assembly task are set to

$$C(i,j) = 0 \text{ for } i = 1 \dots m \text{ and } j = a_{R,H} \text{ with } m = \text{total number of tasks.}$$

In addition, a reward for finishing the task can be provided. Due to the design of the model, as the agents do not communicate directly with each other, the chance exist that they might choose the same valid task. In this case, the action of the robot is set to $a_R = 0$ and no reward is provided. Next, if the agent is free and a new not illegal action was chosen, the new task is assigned to the agent. This is the case when $a_{R,H} \neq 0$ and $d_{R,H} = 0$. This leads to $d_{R,H} = a_{R,H}$. Besides, the task time for the other agent is set to $t_{H,R}(a_{R,H}) = 0$. By this, the other agent will not be able anymore to choose the action. Also, a positive or negative reward for ergonomics can be assigned here.

Finally, the variables d_R, d_H, t_R, t_H, e_H and C are saved in *struct* format and are provided as the environment state to the agents. The Boolean value f, which indicates if the assembly process is finished, is changed to *true* when

$$t_{R,H}(i) = 0 \text{ for } i = 1 \dots m \text{ with } m = \text{total number of tasks.}$$

Then, the simulation is stopped.

5.3 Rewards and model objectives

By using multi-agent reinforcement learning, different rewards can be provided to the two agents. This allows for the promotion of different behaviors and ultimately leads to different policies. A local as well as a global reward is provided to the agents. The global reward r_g is the same for both agents, while the local reward $r_{lR,H}$ is assigned individually. The rewards, which are specified within the programmed environment, are crucial for the training success and the realization of the determined model objectives.

First of all, the agents should learn which actions are allowed in a certain environment state. For this reason, a negative reward is added to the local reward of an agent when choosing an illegal action. The ultimate goal is to finish the assembly process. Therefore, when finished, a relatively high global reward is assigned to the reward function. Applying these rewards alone, however, could lead to sparse rewards during the assembly, especially in the beginning of the training when the agents might not always finish the process or it could take a long time. To prevent this, reward shaping is used by providing a local small intermediate reward for completing one assembly task to guide the agents better.

In order to achieve a minimized completion time of the assembly process, a global negative reward is appointed for every time step. In addition, a local negative reward for idle time is assigned to the agents. To prevent the convergence of the policy to only choosing waiting as action instead of exploring the action space, this reward is slightly higher than the one for an illegal action.

The second objective is a low ergonomic stress level for the human worker. To achieve this, when choosing a specific task, the ergonomic stress level of this task times a factor is added to the local reward of the agent representing the robot or subtracted to the local reward of the agent representing the human. By this, the robot-agent gets promoted to take over tasks which would result in a high ergonomic stress level of the human while the human-agent is trained to perform preferably tasks with a low ergonomic stress level. The weighing between the two objectives *completion time* and *ergonomic stress level* can be adjusted by the amount of the factor.

An overview of all rewards per action for each agent can be seen in table 5-2. The total reward provided to the two agents is calculated by $r_R = r_g + r_{lR}$ for the agent representing the robot and $r_H = r_g + r_{lH}$ for the agent representing the human. The values of r_g and $r_{lR,H}$ are assigned

within the programmed environment model based on the current environment state and the chosen action of the agents.

Table 5-2: Overview of provided rewards per action for each agent.

Type	Explanation	Robot reward	Human reward
Global	Reward for finishing the whole assembly process	50	50
Global	Negative reward for every time step	-1	-1
Local	Reward for completing one assembly task	1	1
Local	Negative reward for illegal action	-10	-10
Local	Negative reward for idle time	-11	-11
Local	Reward for choosing a task i with a certain ergonomic stress level	$e_H(i) * 1$	$-e_H(i) * 1$

5.4 Agents

The agents interact with the environment according to the model structure in figure 5-1. One agent represents the collaborative robot and the other one the human worker. As reinforcement learning algorithm, for both agents, proximal policy optimization (PPO) is chosen. Thereby, built-in functions of the *Reinforcement Learning Toolbox™* (2023) are used.

According to section 2.3, reinforcement learning algorithms can be categorized as policy-based, value-based and as actor-critic. The chosen algorithm is an actor-critic approach. The critic network then evaluates the decision made by the actor network. The PPO agents use a value function critic and a stochastic policy actor. The literature review in section 4.3.3 showed that this approach is used by most of the examined studies. It has the advantage that both discrete and continuous action and observation spaces are supported. During training, PPO uses a policy gradient method which means that it adjusts the policy of the agent based on the feedback it gets. This is done by collecting data through interaction with the environment and then optimizing the policy using a technique called stochastic gradient descent. To ensure stability during training, the amount by which the policy is changed at each time step is limited. This prevents drastic changes and leads to more efficient learning. Each time step, the probabilities for taking each action of the action space is estimated. The final action is chosen by randomly selecting an action based on those probabilities. The agents are model-free. This means that no information on the simulated environment is provided to the agents, and they learn by trial-and-error.

Several options can be customized when creating PPO agents in *Matlab*. The choice of parameters influences the behavior of the agents and is highly dependent on the reinforcement learning problem. For example, the entropy loss weight can be adjusted. It is a scalar value

between 0 and 1 which is added as factor of the entropy loss term to the actor loss function. The higher the entropy value, the more uncertain the agent is about which action to take next. Therefore, a high entropy loss weight increases the uncertainty of the agent and thereby promotes exploration. By a high exploration, the agent gets less easily trapped on a local reward maximum. Another relevant parameter is the discount factor. During training, it is applied to future rewards and thereby determines the weighing of immediate rewards over future rewards. Also, the number of hidden fully connected layers of the neural networks of the agent can be specified. The higher the number of layers, the more complex function can be approximated. However, finding the best policy becomes more and more computationally expensive and the training time increases.

5.5 Training and simulation

During training, the assembly process is simulated with the help of the environment model. The two agents interact with the environment and update their policies. By simulating the process, the system behavior can be predicted.

Several training options can be adjusted. As learning strategy, a decentralized training is chosen. This means that the agents do not share their experiences and update their policy individually. The maximum number of episodes determines how often the assembly process should be simulated before the training is ended. One episode refers to the simulation of one assembly process. Also, the maximum number of time steps per episode is defined. Especially in the beginning of the training, the agents might not complete all assembly tasks within the maximum time steps. When starting the training, a window opens where the training progress can be monitored. The current number of episode as well as one graph per agent showing the episode reward for all previous episodes, a curve of the averaged reward with a window size of 30 episodes, and the value estimation of the critic network. Usually, the rewards in the beginning of the training are low and then slowly converging to the ideally highest reward.

It is possible to fine-tune the agents by conducting the training in two stages. In the first stage, a high exploration rate can be chosen to promote exploring the action space. However, the policy of the agents will then hardly converge to a maximum value as the agents receives negative reward when they are too certain about a certain policy. This promotes a policy change each time the policy converges. Therefore, a second training stage with a low exploration rate can be beneficial. Then, the policy can finally converge to the overall maximum.

When the training of the agents is finished, they can be saved from the workspace to file. By this, they can be loaded into the workspace again at a later point in time or be made available on other devices. The assembly process can be simulated with the trained agents. The result is the optimal sequence and task allocation according to the two objectives regarding completion time and ergonomic stress level.

6 Testing and discussion of the model

In this section, the developed optimization model is tested and discussed. To validate the model, a computational experiment using an exemplary system without practical reference is carried out. It is composed of seven tasks which the collaborative robot and the human worker finish in cooperation to reach the assembly goal taking several constraints into account. The sequence constraints can be seen in figure 6-1.

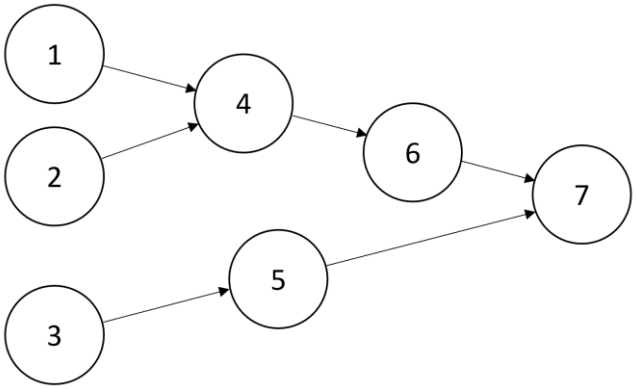


Figure 6-1: Sequence constraints of the example task.

The sequence constraints result in the dependency matrix

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

Each row indicates which tasks have to be finished before the respective task can be started. Thereby, the availability and processing time of each task are different for the two operators. Robots and humans have their own distinct strengths and skills which might be beneficial for certain types of tasks depending on, for example, material characteristics and tolerance limits. Some tasks might not be suitable for robot or human or the processing time can differ between both. To cover the application possibilities of the model, this is reflected in the exemplary system by setting the task times for the robot to

$$t_R = [\,9 \quad 7 \quad 0 \quad 2 \quad 4 \quad 10 \quad 2\,]$$

and for the human to

$$t_H = [\,10 \quad 8 \quad 10 \quad 0 \quad 2 \quad 3 \quad 4\,].$$

The third task cannot be processed by the robot and the fourth task cannot be processed by the human. In addition, the ergonomic stress level

$$e_H = [\,10 \quad 11 \quad 2 \quad 0 \quad 4 \quad 1 \quad 5\,]$$

for the human is assigned to the tasks. As the exemplary system is not derived from a real assembly task, the values are randomly selected. Since the human worker is not able to process the fourth task, a value of 0 is given, meaning that no additional reward will be provided to the robot when choosing this task. The ergonomic stress level of an assembly task is, among others, dependent on body motion and postures, physical effort as well as environmental stressors (Tropschuh et al. 2021). As the examination of the ergonomic stress level is not within the scope of this work, a detailed explanation is omitted at this point and reference should be made to the literature.

To test the developed model and assess the influence of the provided reward regarding the ergonomic stress level, the model is applied multiple times on the example system. The first time, no reward regarding ergonomics is assigned. By that, only the goal of a minimized completion time should be followed by the agents. In the second run, a reward of $e_H(i) * 1$ and $-e_H(i) * 1$, respectively, is given when taking over a task. Finally, the factor is increased and a reward of $e_H(i) * 5$ and $-e_H(i) * 5$, respectively, is provided. The proposed optimal assembly sequences and task allocations for these three cases are compared with each other in the following.

Some options for the agents as well as for the training are highly dependent on the complexity of the assembly task. As explained in section 5.5, the training is conducted in stages. For the selected example, an entropy loss weight of 0.7 is chosen to promote exploration in the first stage. Also, a negative reward for idle time is provided to prevent the policies to converge towards a solution where no action is chosen at all to not risk taking illegal actions. In the second stage, after the agents learned how to prevent illegal actions, the reward for idle time is set to 0, as the solution with the least idle time does not necessarily have the shortest completion time due to the task time differences for the operators. In the last stage, the entropy loss weight is reduced to 0.01 to promote convergence. The number of hidden layers of the neural networks of the agents is set to 256. Regarding the training options, a maximum number of episodes 800 is selected for the first stage, 500 for the second stage and 800 for the last stage. The maximum number of time steps per episode is 1400. During training, it is visible that these numbers are high enough in order to enable a learning progress of the agents without being too computationally expensive.

The training progress of the first and second run is visualized in figure 6-2, above for the agent representing the robot and below for the agent representing the human. The reward for each episode is illustrated in light blue and the running average reward value over the last 30 episodes in dark blue. Since the agents are composed of an actor and a critic, the critic estimates the value of a certain state of the agent. The estimated value for the initial state to the beginning of each episode, meaning the cumulative discounted long-term reward for the episode, is shown in orange. In the progress of the training, the critic learns to estimate the value better and better. The high exploration of the agents is clearly visible in the first 800 episodes. Afterwards, when no negative reward for idle time is provided anymore, the policies adapt to this. In the following, the curves are slowly converging.

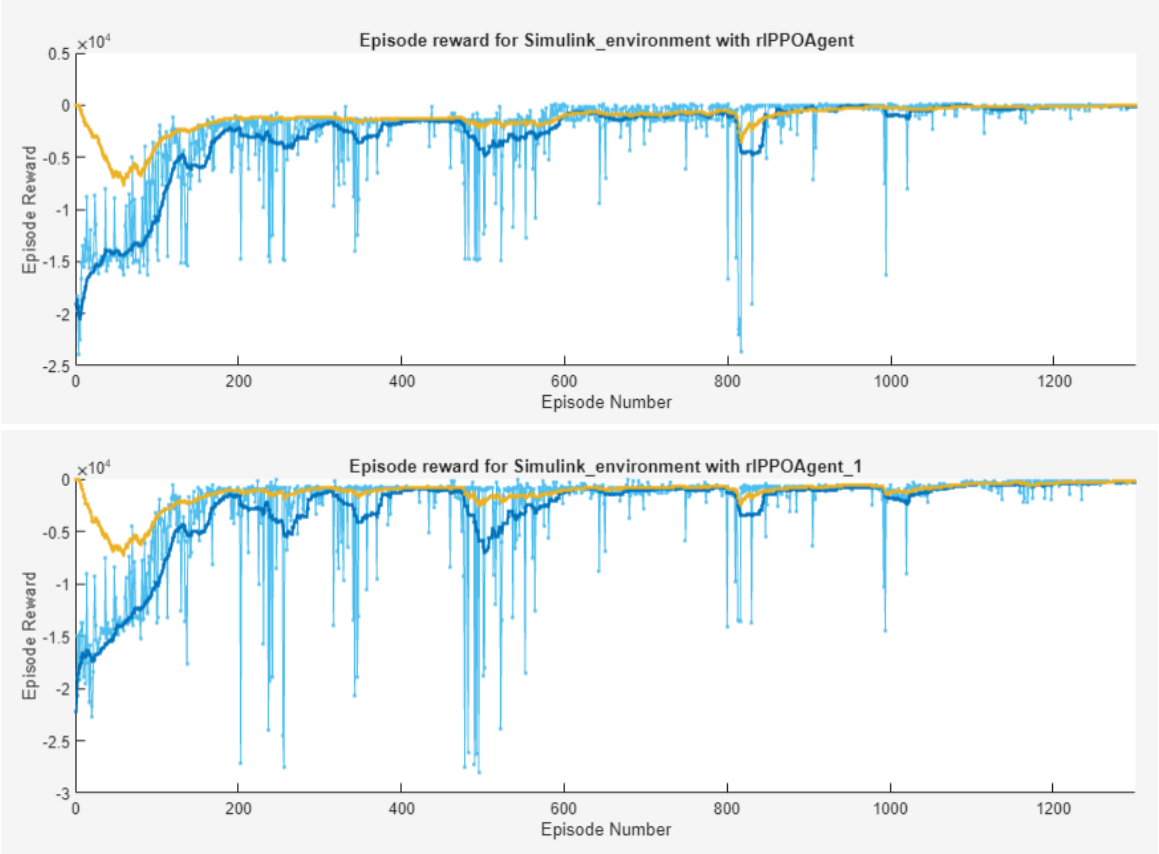


Figure 6-2: Training progress.

The solution for the task sequence and allocation found by the model in the first run is shown in figure 6-3 for the robot and figure 6-4 for the human. The course of the two variables d_R and d_H over time without any reward for choosing a task with a certain ergonomic stress level is visible.

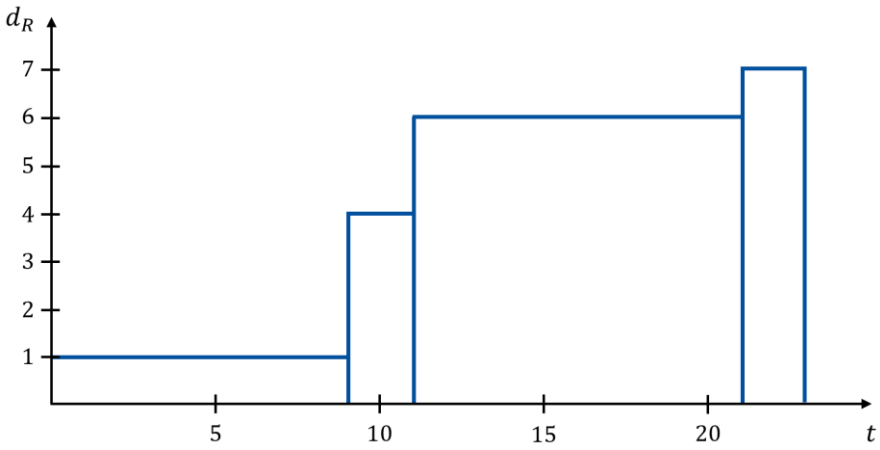


Figure 6-3: Task sequence for the robot over time for only optimized completion time.

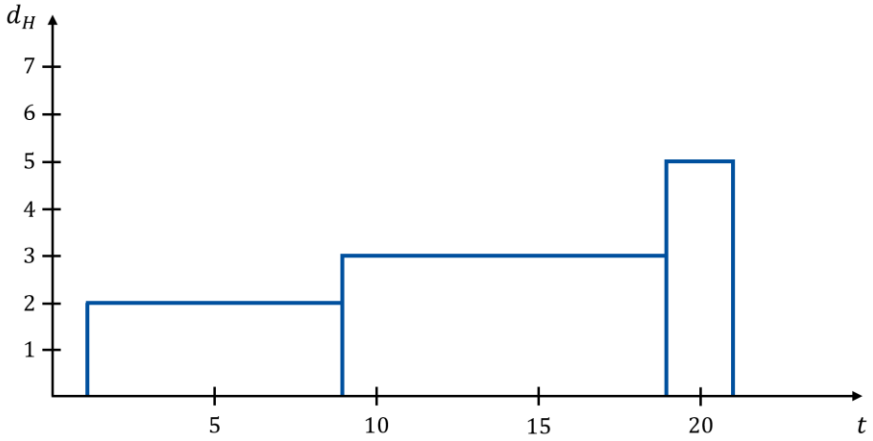


Figure 6-4: Task sequence for the human over time for only optimized completion time.

The tasks with the number 1, 4, 6 and 7 are assigned to the robot, while the human does the tasks with the number 2, 3 and 5. All constraints regarding the sequence, the task allocation and the task time are fulfilled. The total time for completing the assembly process amounts to 23 seconds. Minimizing this time was the only objective as the ergonomic stress level was not taken into account by the agents. The ergonomic stress level according to e_H amounts to 17.

Next, a reward for choosing a task with a certain ergonomic stress level is provided with the factor 1. The model output for d_R and d_H follow in figure 6-5 and figure 6-6.

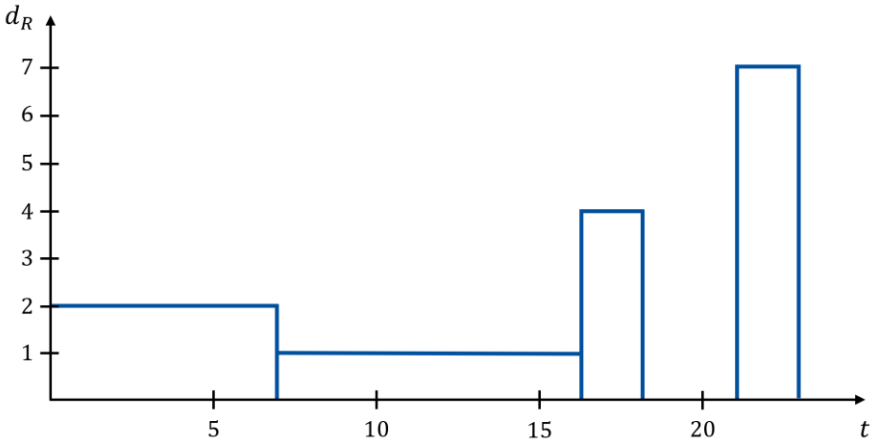


Figure 6-5: Task sequence for the robot over time considering ergonomics with factor 1.

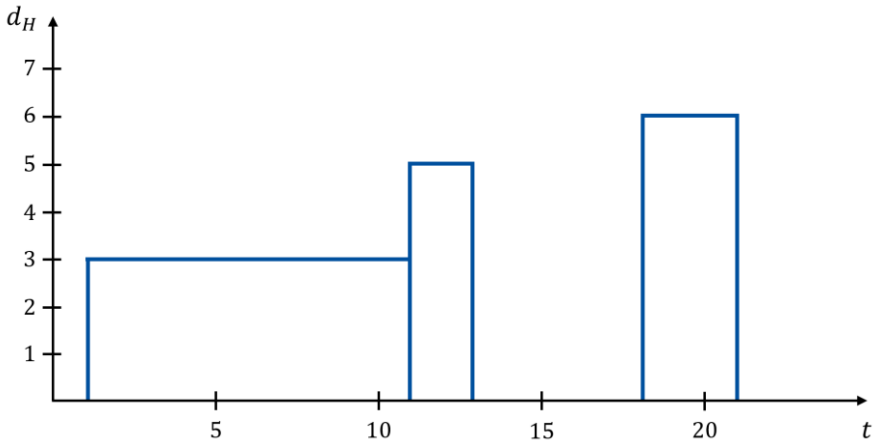


Figure 6-6: Task sequence for the human over time considering ergonomics with factor 1.

The completion time stays at 23s, however, another sequence and task allocation are chosen resulting in a total ergonomic stress level of 7. Again, all given constraints are thereby fulfilled.

In the last run, the factor for calculating the reward for choosing a task with a certain ergonomic stress level is increased to 5. By this, the weighing between the two objectives is changed in favor of ergonomics.

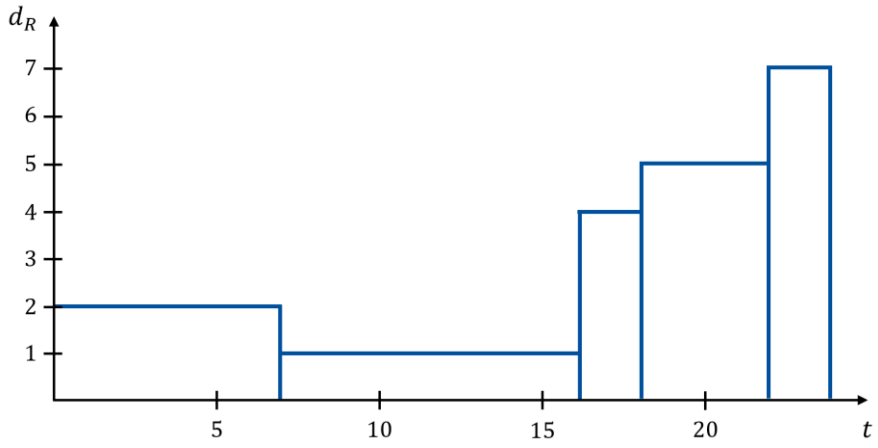


Figure 6-7: Task sequence for the robot over time considering ergonomics with factor 5.

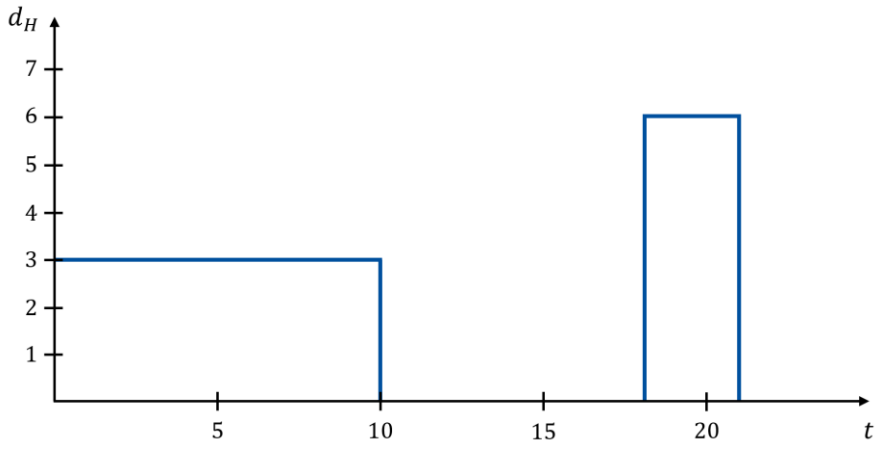


Figure 6-8: Task sequence for the human over time considering ergonomics with factor 5.

The completion time changed to 24s seconds while the ergonomic stress level is reduced to 3. A summary of the output for the three model runs is given in table 6-1. As expected, the ergonomic stress level overall decreases while the completion time increases.

Table 6-1: Model outputs respectively to provided reward regarding ergonomics.

Run	Scenario	Robot reward (ergonomic)	Human reward (ergonomic)	Completion time	Ergonomic stress level for the human
1	Only focus on optimizing total completion time	0	0	23s	17
2	Ergonomic stress level of the human is considered	$e_H(i) * 1$	$-e_H(i) * 1$	23s	7
3	Higher weighing of the ergonomic stress level	$e_H(i) * 5$	$-e_H(i) * 5$	24s	3

The validation of the model shows that it is possible to consider multiple objectives when using reinforcement learning for optimization problems by assigning rewards to multiple conditions. The model outputs for the three different runs imply that the weighing between the objectives can be adjusted by changing the provided rewards. Also, illegal actions can successfully be prevented by assigning a negative reward to them. By that, given constraints can be met. It should be noted that, when an illegal action is taken by one agent, the action should not only be ignored by the environment, but no further state transition should be performed. For example, if the agent is currently doing a task and already chooses the next task while it is occupied, no time should be deducted from the task time. That way, the agent learns that it can only finish a task by not performing illegal actions. It is necessary to remark that the exemplary assembly process used to test the model is a rather simple system with a limited number of tasks which might be solved with conventional optimization methods with smaller computational effort. However, the goal of this thesis is to develop a model taking multiple objectives into account and to make use of the advantages of reinforcement learning over conventional optimization methods in this context.

Another implication of the findings is the importance of selecting parameters which are appropriate for the specific assembly process. This includes fine-tuning of the rewards, options for the agents as well as training options. The results clearly demonstrate that the choice of parameters significantly impacts the training progress and the proposed solution. When testing the system, for example, no negative reward for idle time led to a policy in favor of taking no action at all as the agents tried to avoid negative reward for illegal actions. That hindered the exploration of the action space. The provision of a negative reward which is slightly higher than for illegal actions led to the preference of an illegal action over idle time in the beginning of the training. By taking also illegal actions, the agents explored the action space and finally adjusted their policies in favor of taking legal over illegal actions. In addition, the agents learned to choose idle time rather than an illegal action, as no state transitions are performed when an illegal action is chosen as described above.

The findings show that the learning curve of the agent, which represents the robot, is in general lower than the one of the agent representing the human. This is influenced by the fact that both agents might choose the same valid task at the same time. As explained in section 5.2, when this happens, the action of the robot-agent is set to 0 and the action of the human-agent is progressed. The reason is that the robot should be able to react to the human and choose another task. However, it slows down the learning progress of the robot as a valid action was chosen but no state transition is processed based on it.

In the developed model, the robot is able to adapt to the human and choose alternative assembly sequences based on the human choices. This is beneficial, as a certain degree of uncertainty is present in an assembly process involving human workers. When using conventional methods,

an adaption would not be possible which results in a halt of the process. However, the proposed model does not support disassembly actions as in the model of Antonelli et al. (2021). The support would allow a greater flexibility and represents therefore one possible improvement of the proposed model.

The training of the reinforcement learning agents to solve the exemplary system was conducted in three stages. First with a high exploration rate and later with a low exploration rate. The reason is to balance exploration and exploitation of the agents. The trade-off between exploration and exploitation is a general challenge in reinforcement learning (Sutton and Barto 2018). The agents should exploit the paths they already experienced but also discover new paths. By using a high exploration rate in the first two stage of the training, the agents have the opportunity to exploit the action space. Afterwards, in the third stage, the policies are able to converge towards the optimal solution.

One might assume that a minimization of idle time would lead automatically to a minimization of completion time. However, this is not the case. As the task times for robot and human can be different, it might be beneficial if the operator with the lowest time conducts the task even though if it might lead to more idle time for the other operator.

A weakness of using a reinforcement learning model is that the training process will not be exactly the same when repeated even though when the exact same parameters are used. The reason is that stochastic choices are made by the agents which influence the further development of the policies. For example, if the agents randomly discover choices which lead to a high reward in the beginning of the training, they will quicker come to other choices with high rewards. However, agents might also explore long until they find choices with high rewards. It cannot be guaranteed that the model always finds the same and optimal solution, especially when the number of episodes during training is not high. When the training is finished and an assembly process with identical parameters is simulated multiple time, the model might not always perform the same and best sequence, even though when the underlaying policies converged towards the optimal solution. The reason is the stochastic nature of the model which might lead to a different outcome in the same state of the environment.

7 Conclusion

The implementation of collaborative robots to support the human workers in assembly lines offers a great potential to enhance productivity and reduce ergonomic risks. However, it leads to an increasingly complex planning environment and requires optimization models for sequence planning and task allocation. Reinforcement learning can be used to develop optimization models. So far, the research in this context is mainly focused on economic factors such as a short completion time. This thesis aimed to investigate the following two research questions:

- What is the state of the art of addressing the assembly line sequencing problem with reinforcement learning when using collaborative robots?
- How can a reinforcement learning based optimization model be formulated to solve the assembly line sequencing problem with collaborative robots under consideration of ergonomic factors?

The objectives of this thesis included the identification of relevant literature, the evaluation of their strengths and limitations, the derivation of areas for future research, the development of an optimization model, and finally, the training and testing of the model on an exemplary assembly task.

To answer the first research question, a systematic literature review on assembly line sequencing with collaborative robots using reinforcement learning was carried out. In total, four documents were selected. As the first document was published in 2020 and the number of relevant documents is low, the research area is an emerging topic. Single- and multi-objective approaches were identified. The minimization of completion time is an objective in all models. In addition, only Zhang et al. (2022) takes a constant task difficulty into account and thereby follows multiple objectives. The pure focus on ergonomics was not a subject of research so far. All models use a MDP environment as common when using reinforcement learning. The researchers used different approaches, such as for example a grid world. Single-and multi-agent reinforcement learning is used in the studies. Thereby, different kinds of agents are applied, such as Q-learning, PPO and DDPG. The majority of the models use an actor-critic approach, and all models apply model-free agents. Only the model of Zhang et al. (2022) supports cooperation and collaboration between robot and human. A progression in the development of the models is visible. The complexity of the considered scenarios, adaption to human failures, model robustness, and consideration of multiple objectives are the main points of development.

To answer the second research question, an optimization model using multi-agent reinforcement learning was developed. The model follows two objectives, namely the minimization of the completion time and a minimization of the ergonomic stress level of the

human worker. It is composed of two model-free PPO agents which represent robot and human. The agents interact with a customized environment programmed using *Matlab* and *Simulink* taking constraints into account. The environment was programmed based on a conceptional model of an assembly process and represents a MDP. Each time step, the agents choose an action, which triggers a state transition in the environment, and receive a reward. An optimized assembly sequence and task allocation is found for a workstation where one robot and one human work in cooperation to complete a joint assembly. Depending on the assembly task, the rewards and parameters for the agents and the training can be adjusted. The training can be conducted in multiple stages, for example, with different exploration rates. The developed model was tested on an exemplary assembly process. Thereby, the proposed solution without and with consideration of the economic stress level of the human worker were compared. The test showed that the model successfully considered the second objective as it proposed a solution with a slightly higher completion time, but significantly lower ergonomic stress level. Given constraints were met by providing a negative reward for illegal actions. It can be concluded that reinforcement learning can be used to solve the assembly line sequencing problem considering multiple objectives such as a short cycle time and low ergonomic stress. The adaption to small mistakes and uncertainties during the assembly process is possible. It should be noted that the results are highly dependent on the choice of parameters.

The contributions of this thesis are as follows. The findings of the literature review can be used by researchers to quickly gain an understanding of the state of the art of assembly line sequencing with reinforcement learning when using collaborative robots. The review guides future research and can be used by practitioners to classify their current optimization model based on the existing knowledge as well as to better understand how to optimize the usage of collaborative robots through the application of reinforcement learning. On a general level, the review demonstrates how a systematic literature review can be performed for an emerging research area. The development of the optimization model highlights important factors that should be considered when using reinforcement learning for assembly line sequencing. In addition, it emphasizes the suitability of reinforcement learning for decision-making processes in robotics.

The objectives of this thesis were achieved. Limitations of the review are the use of one single database in which solely English literature was considered, and the fact that a degree of subjectivity remains in the selection of documents. In addition, as the review topic is specific with a narrowed scope, only a small number of relevant studies were found. A source of error could be the selection of search keywords, as it is possible that relevant studies might have been excluded. In further research, the scope of the review could be broadened by including other optimization problems. Limitations of the developed optimization model are as follows. Regarding the level of interaction between robot and human, only cooperation is supported. The application of the model is restricted to one workstation with one robot and one human

worker. Adjustments to unforeseen behavior are possible only to a limited extent. Also, the robustness is limited. As the used agents have stochastic elements, they might take different actions in the same state of the environment. Therefore, the best solution might not always be chosen, even though it has the highest reward. In the next step, to improve the robustness of the policy, the training could be conducted with randomized elements of the environment as well as with the addition of noise. The usage of non-stochastic agents is possible. In addition, a case study to test the developed model on a real-life assembly task could be carried out. As this increases the number of environment states, an adequate exploration of the action space might be a challenge and would require adjustments of parameters. Further improvement of the model is possible through adding the support of disassembly processes in case of mistakes by the human.

References

Alessio, Alessandro, Khurshid Aliev, and Dario Antonelli. 2022. "Robust Adversarial Reinforcement Learning for Optimal Assembly Sequence Definition in a Cobot Workcell." *Lecture Notes in Mechanical Engineering*: 25–34.

Antonelli, Dario, Qingfei Zeng, Khurshid Aliev, and Xuemei Liu. 2021. "Robust Assembly Sequence Generation in a Human-Robot Collaborative Workcell by Reinforcement Learning." *FME Transactions* 49(4): 851–58.

Bragança, Sara, Eric Costa, Ignacio Castellucci, and Pedro M. Arezes. 2019. "A Brief Overview of the Use of Collaborative Robots in Industry 4.0: Human Role and Safety." In *Occupational and Environmental Safety and Health*, Cham: Springer International Publishing, 641–50.

Kamal, Mohammad, and Jose Luis Martinez Lastra. 2011. "Assembly Line Balancing and Sequencing." In *Assembly Line - Theory and Practice*, InTech.

Kinat, Luisa A. 2022. "Assembly Line Balancing and Sequencing in the Industry 4.0 Era: A Systematic Literature Review." Specialisation Project. Norwegian University of Science and Technology.

Moher, David, Alessandro Liberati, Jennifer Tetzlaff, and Douglas Altman. 2009. "Preferred Reporting Items for Systematic Reviews and Meta-Analyses: The PRISMA Statement." *Open medicine* 3(2): 123–30.

Mongeon, Philippe, and Adèle Paul-Hus. 2016. "The Journal Coverage of Web of Science and Scopus: A Comparative Analysis." *Scientometrics* 106(1): 213–28.

Reinforcement Learning Toolbox Release R2023b. 2023. Natick: The MathWorks, Inc.

Richter, Felix. 2021. "China Is the World's Manufacturing Superpower." *Statista*. https://www.statista.com/chart/20858/top-10-countries-by-share-of-global-manufacturing-output/ (April 28, 2023).

Robinson, Stewart. 2008a. "Conceptual Modelling for Simulation Part I: Definition and Requirements." *Journal of the Operational Research Society* 59(3): 278–90.

Robinson, Stewart. 2008b. "Conceptual Modelling for Simulation Part II: A Framework for Conceptual Modelling." *Journal of the Operational Research Society* 59(3): 291–304.

Seuring, Stefan, Yawar, Sadaat Ali, Land, Anna, Khalid, Raja Usman, and Sauer, Philipp C. 2020. "The Application of Theory in Literature Reviews – Illustrated with Examples from Supply Chain Management." *International Journal of Operations & Production Management* 41(1): 1–20.

Snyder, Hannah. 2019. “Literature Review as a Research Methodology: An Overview and Guidelines.” *Journal of Business Research* 104: 333–39.

Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. Second edition. Cambridge, Massachusetts: The MIT Press.

Tranfield, David, David Denyer, and Palminder Smart. 2003. “Towards a Methodology for Developing Evidence-Informed Management Knowledge by Means of Systematic Review.” *British Journal of Management* 14(3): 207–22.

Tropschuh, Barbara, Sina Niehues, and Gunther Reinhart. 2021. “Measuring Physical and Mental Strain during Manual Assembly Tasks.” *Procedia CIRP* 104: 968–74.

Yu, Tian, Jing Huang, and Qing Chang. 2020. “Mastering the Working Sequence in Human-Robot Collaborative Assembly Based on Reinforcement Learning.” *IEEE Access* 8: 163868–77.

Zhang, Rong et al. 2022. “A Reinforcement Learning Method for Human-Robot Collaboration in Assembly Tasks.” *Robotics and Computer-Integrated Manufacturing* 73: 102227.

Appendix

A Review protocol

Review protocol

Research question

What is the state of the art of addressing the assembly line sequencing problem with reinforcement learning when using collaborative robots?

Database

Scopus

Search keywords

Group 1	Group 2	Group 3
“assembly”	“sequenc*”	“cobot*”
	“ALSP”	“collaborative robot*”
		“co-bot*”
		“human-robot collaboration”
		“HRC”

Search string

(TITLE-ABS-KEY ("sequenc*") OR TITLE-ABS-KEY ("ALSP") AND TITLE-ABS-KEY ("assembly") AND TITLE-ABS-KEY ("cobot") OR TITLE-ABS-KEY ("collaborative robot*") OR TITLE-ABS-KEY ("human-robot collaboration") OR TITLE-ABS-KEY ("HRC"))

Inclusion and exclusion criteria

- Published in English language
- Subject areas: “Engineering”, “Computer Science”, “Decision Sciences”, “Business, Management, and Accounting”, “Economics, Econometrics, and Finance”
- Document type: Journal articles or conference papers

B *Matlab* function *Main.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script to find the optimal assembly sequence and task allocation for ...
    a workstation
% with one collaborative robot and one human worker considering completion
% time and ergonomic stress level using multi-agent reinforcement learning

clear
clc
close all

% When doTraining = false, already trained agents can be used for ...
    simulation
doTraining = false;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Define variables
num_tasks = 7; % Total number of tasks
max_time = 60*60; % Maximum time per task
max_erg = 100; % Maximum ergonomic stress level per task

% Create a vector containing all possible actions
possible_actions = zeros(1,num_tasks+1);
for i = 0:num_tasks
    possible_actions(1,i+1) = i;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Define the action space (later used to create the RL agents)
% Number of task which the agent chooses to do
act(1) = Simulink.BusElement;
act(1).Name = "action";
act(1).Dimensions = 1;

% Create bus object
actBus = Simulink.Bus();
actBus.Elements = act;
actInfo = ...
    bus2RLSpec("actBus","DiscreteElements",{"action",possible_actions});

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Define the observation space (later used to create the RL agents)
% Number of task which the robot-agent is currently doing (discrete)
obs(1) = Simulink.BusElement;
obs(1).Name = "isDoingA";

% Number of task which the human-agent is currently doing (discrete)
obs(2) = Simulink.BusElement;
obs(2).Name = "isDoingB";

% Vector with the time for each task for the robot between 0 and max_time
obs(3) = Simulink.BusElement;
obs(3).Name = "timeA";
```

```
obs(3).Dimensions = [1 num_tasks];
obs(3).Min = 0;
obs(3).Max = max_time;

% Vector with the time for each task for the human between 0 and max_time
obs(4) = Simulink.BusElement;
obs(4).Name = "timeB";
obs(4).Dimensions = [1 num_tasks];
obs(4).Min = 0;
obs(4).Max = max_time;

% Vector with the ergonomic value for each task between 0 and max_erg
obs(5) = Simulink.BusElement;
obs(5).Name = "erg";
obs(5).Dimensions = [1 num_tasks];
obs(5).Min = 0;
obs(5).Max = max_erg;

% Matrix with the dependencies of the tasks to meet assembly constraints
obs(6) = Simulink.BusElement;
obs(6).Name = "dependencies";
obs(6).Dimensions = [num_tasks num_tasks];
obs(6).Min = 0;
obs(6).Max = 1;

% Create bus object
obsBus = Simulink.Bus();
obsBus.Elements = obs;
obsInfo = bus2RLSpec("obsBus","DiscreteElements",{"isDoingA",...
    possible_actions,"isDoingB",possible_actions});

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Create the environment
mdl = "Simulink_environment";
open_system(mdl);
blks = [mdl+"/Robot", mdl+"/Human"];
obsInfos = {obsInfo,obsInfo};
actInfos = {actInfo,actInfo};
env = rlSimulinkEnv(mdl,blks,obsInfos,actInfos);
env.ResetFcn = @(in) resetEnvironmnet(in);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Create the agents representing robot and human
agentOptions = rlPPOAgentOptions(...
    ExperienceHorizon = 512,...
    ClipFactor = 0.2,...
    EntropyLossWeight = 0.7,...
    MiniBatchSize = 128,...
    NumEpoch = 4,...
    AdvantageEstimateMethod = "gae",...
    GAEFactor = 0.95,...
    SampleTime = 0.1,...
    DiscountFactor = 1);
agentOptions.ActorOptimizerOptions.LearnRate  = 1e-4;
agentOptions.CriticOptimizerOptions.LearnRate = 1e-4;

```

```
robot = rlPPOAgent(obsInfo,actInfo, ...
    rlAgentInitializationOptions(NumHiddenUnit= 256),agentOptions);
human = rlPPOAgent(obsInfo,actInfo, ...
    rlAgentInitializationOptions(NumHiddenUnit= 256),agentOptions);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Validate the environment (optional)
% validateEnvironment(env)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Training of the agents
if doTraining
    % Log the training data in the end of each episode in the folder "logs"
    lgr = rlDataLogger();
    lgr.EpisodeFinishedFcn = @logExperienceData;
    lgr.LoggingOptions.LoggingDirectory = "logs";

    % Define training options
    trainOpts = rlMultiAgentTrainingOptions(...
        AgentGroups = "auto",...
        LearningStrategy = "decentralized",...
        MaxEpisodes = 800,...
        MaxStepsPerEpisode = 1400,...
        ScoreAveragingWindowLength = 30,...
        StopTrainingCriteria = "EpisodeReward",...
        StopTrainingValue = [1000 1000]);

    % Start the training
    training_results = train([robot,human],env,trainOpts,logger=lgr);

    % Change of parameters for the second stage of training (optional)
    % robot.AgentOptions.EntropyLossWeight = 0.01;
    % human.AgentOptions.EntropyLossWeight = 0.01;
    % training_results(1, 1).TrainingOptions.MaxEpisodes = 1300;
    % training_results(1, 2).TrainingOptions.MaxEpisodes = 1300;
    % training_results = ...
       train([robot,human],env,training_results,logger=lgr);

    % Save trained agents and training results
    save("Robot","robot")
    save("Human","human")
    save("Training-results","training_results")
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Load trained agents and simulate assembly process
if ¬doTraining
    % Robot-agent and human-agent of the same case should be selected
    file1 = uigetfile("Trained_agents\");
    file2 = uigetfile("Trained_agents\");
    load("Trained_agents\"+file1)
    load("Trained_agents\"+file2)
    out = {};
    reward=[];
    for i = 1:1
```

```
160         out{i} = sim([robot,human],env); % Simulation of the assembly ...
                process
161         time(i) = out{1, i}(1).IsDone.Time(end) + 0.1;
162     end
163     time_min = find(time == min(time));
164     ind = time_min(1);
165
166     % The task time -1 is shown, every task actually starts 1 second before
167     figure
168     tasksRobot = out{1,ind}(1).Observation.isDoingA.plot;
169     ylim([0 num_tasks+1]);
170     title("Robot"); xlabel("Time"); ylabel("Task");
171     figure
172     tasksHuman = out{1,ind}(1).Observation.isDoingB.plot;
173     ylim([0 num_tasks+1]);
174     title("Human"); xlabel("Time"); ylabel("Task");
175 end
```

```
1 function dataToLog = logExperienceData(data)
2 dataToLog.Experience = data.Experience;
3 end
```

C *Matlab* function *resetEnvironment.m*

```
function in = resetEnvironmnet(in)
% Reset of the Simulink environment and definition of initial values

state0 = struct();

state0.isDoingA = 0; % Task assigned to the robot-agent (normally 0)
state0.isDoingB = 0; % Task assigned to the human-agent (normally 0)
state0.timeA = [9,7,0,2,4,10,2]; % Time for each task for the robot
state0.timeB = [10,8,10,0,2,3,4]; % Time for each task for the human
state0.erg = [10,11,2,0,4,1,5]; % Ergonomic stress level for each task ...
    for the human

% Dependencies of each the tasks to meet sequence constraints (0 or 1)
state0.dependencies(1,:) = [0,0,0,0,0,0,0];
state0.dependencies(2,:) = [0,0,0,0,0,0,0];
state0.dependencies(3,:) = [0,0,0,0,0,0,0];
state0.dependencies(4,:) = [1,1,0,0,0,0,0];
state0.dependencies(5,:) = [0,0,1,0,0,0,0];
state0.dependencies(6,:) = [1,1,0,1,0,0,0];
state0.dependencies(7,:) = [1,1,1,1,1,1,0];

for i = 1:length(state0.timeA)
    if state0.timeA(i) ~= 0
        state0.timeA(i) = state0.timeA(i)-1;
    end
    if state0.timeB(i) ~= 0
        state0.timeB(i) = state0.timeB(i)-1;
    end
end

in = setVariable(in,'state0',state0);

end
```

D Environment subsystem in *Simulink* model

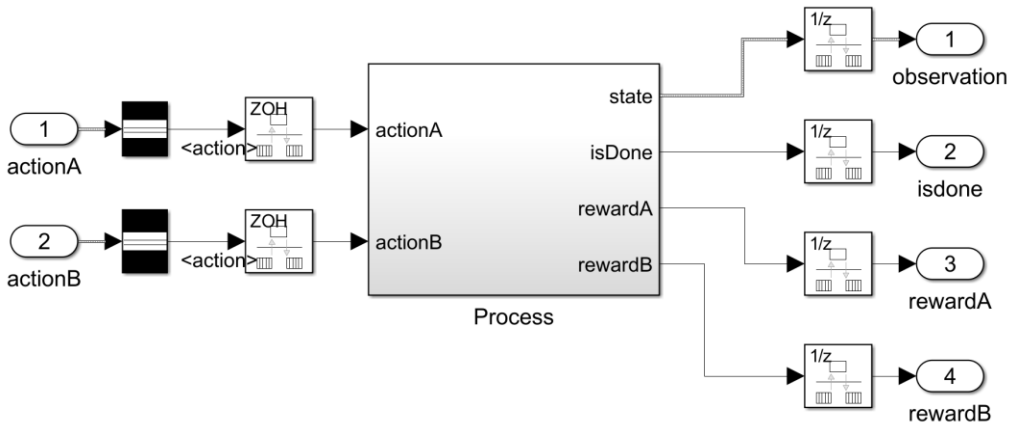


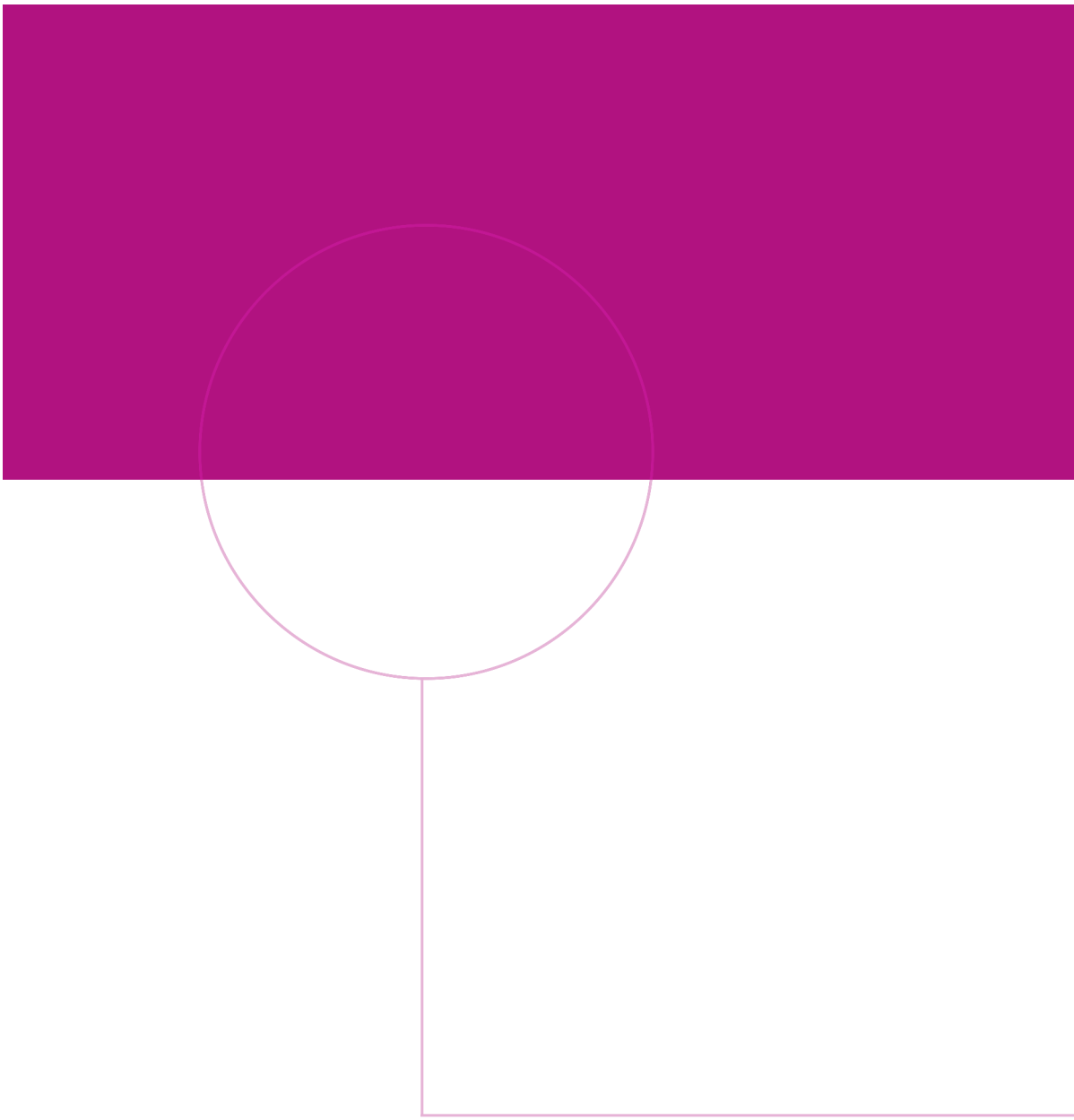
Figure D-1: Subsystem with data operations.

E State transition in *Simulink* model

```
function [new_state,isDone,rewardA,rewardB] = fcn(state,actionA,actionB)
% Script to transform the state of the environment based on the action of
% the agents with the new state and the reward for both agents as output
%
% Variables:
% state.isDoingA - Number of task which the robot-agent is currently doing
% state.isDoingB - Number of task which the human-agent is currently doing
% state.timeA - Vector with the time for each task for the robot
% state.timeB - Vector with the time for each task for the human
% state.erg - Vector with the ergonomic stress level for each task
% state.dependencies - Matrix with the dependencies of the tasks (0 or 1)
% isDone - Boolean value if all tasks are finished
% rewardA - Reward for the robot-agent
% rewardB - Reward for the human-agent

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Define rewards
reward_info.step = -10;
reward_info.ergonomic = 0; % Case 1: 0, case 2: 1, case 3: 5
reward_info.action_while_busy = -10;
reward_info.action_on_dependent_task = -10;
reward_info.action_on_unavailable_task = -10;
reward_info.idle_time = 0; % -10 for the first stage of training
reward_info.done = 50;
reward_info.one_task_done = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Define further variables
Δ_t = 1;
actions = [actionA actionB];
isDoing = [state.isDoingA state.isDoingB];
time = [state.timeA;state.timeB];
is_illegal_move = boolean([0 0]);
reward_local = [0 0];
reward_global = reward_info.step; % Negative reward for every step

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Validate the actions
for i = 1:length(actions)
    idx_isDoing = isDoing(i); % Number of current task
    idx_action = actions(i); % Number of chosen task

    % Give negative reward for idle time (not choosing an action even ...
       though
    % the agent is free)
    if actions(i) == 0 && isDoing(i) == 0
        reward_local(i) = reward_local(i) + reward_info.idle_time;
    end

    if idx_action ≠ 0
        % Check if an action was chosen even though the agent is busy
        if isDoing(i) ≠ 0
            is_illegal_move(i) = true;
            % Give negative reward
            reward_local(i) = reward_local(i) + ...
                reward_info.action_while_busy;
```

```
55          end
56
57          % Check if the chosen task is dependent on another unfinished task
58          if sum(state.dependencies(idx_action,:)) ≠ 0
59              is_illegal_move(i) = true;
60              % Give negative reward
61              reward_local(i) = reward_local(i) + ...
                    reward_info.action_on_dependent_task;
62          end
63
64          % Check if task is already done or task must be done by the ...
                other agent
65          if time(i, idx_action) == 0
66              is_illegal_move(i) = true;
67              % Give negative reward
68              reward_local(i) = reward_local(i) + ...
                    reward_info.action_on_unavailable_task;
69          end
70
71          % If an illegal action was chosen, ignore the action
72          if is_illegal_move(i) == true
73              actions(i) = 0;
74          end
75      end
76  end
77
78  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
79  %% Perform state transition
80  for i = 1:length(actions)
81      idx_isDoing = isDoing(i); % Number of current task
82      if is_illegal_move(i) == false
83          % Decrement time for the task the agent is currently doing
84          if isDoing(i) ≠ 0
85              time(i,idx_isDoing) = time(i,idx_isDoing) - ∆_t;
86              % If the task is finished, reset the status of the agent ...
                    and remove
87              % dependencies
88              if time(i,idx_isDoing) == 0
89                  isDoing(i) = 0;
90                  state.dependencies(:,idx_isDoing) = ...
                        zeros(size(state.dependencies,1),1);
91                  reward_local(i) = reward_local(i) + ...
                        reward_info.one_task_done;
92              end
93          end
94      end
95  end
96
97  % If robot and human choose the same (valid) task, the task is assigned ...
        to the
98  % human and the robot will wait one timestep
99  if actions(1) == actions(2)
100     actions(1) = 0;
101 end
102
103 % Assign status of the agents and set time for the other agent to zero
```

```
104 if actions(1) ≠0 && isDoing(1) == 0
105     idx_action = actions(1);
106     isDoing(1) = actions(1);
107     time(2,idx_action) = 0;
108     reward_local(1) = reward_local(1) + state.erg(idx_action) * ...
            reward_info.ergonomic; % Global or local reward here? (Was ...
            global before)
109 end
110 if actions(2) ≠0 && isDoing(2) == 0
111     idx_action = actions(2);
112     isDoing(2) = actions(2);
113     time(1,idx_action) = 0;
114     reward_local(2) = reward_local(2) - state.erg(idx_action) * ...
            reward_info.ergonomic;
115 end
116
117 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
118 %% Assign the new state
119 state.isDoingA = isDoing(1);
120 state.isDoingB = isDoing(2);
121 state.timeA = time(1,:);
122 state.timeB = time(2,:);
123 new_state = state;
124
125 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
126 %% Stop when all tasks are finished
127 isDone = false;
128 if sum(time(:)) == 0
129     isDone = true;
130     % Give reward
131     reward_global = reward_global + reward_info.done;
132 end
133
134 % Calculate final reward
135 rewardA = reward_local(1) + reward_global;
136 rewardB = reward_local(2) + reward_global;
137
138 % Print state transition (optional)
139 isDoing
140 time
141 state.dependencies
142 isDone
143 end
```



NTNU

Norwegian University of
Science and Technology