Predicting performance indicators with ANNs for AI-based online scheduling in dynamically interconnected assembly systems

Amon Göppert¹ · Leon Mohring¹ · Robert H. Schmitt²

Received: 3 March 2021 / Accepted: 27 April 2021 / Published online: 8 May 2021
© The Author(s) 2021

Abstract

Mass customization demands shorter manufacturing system response times due to frequent product changes. This increase in system dynamics imposes additional flexibility requirements especially on assembly processes, as complexity accumulates in this last step of value creation. Flexible and dynamically interconnected assembly systems can meet the increased requirements as opposed to traditional dedicated assembly line approaches. The high complexity and dynamical environment in these kinds of systems lead to the demand for real-time online control and scheduling solutions. Within the decision-making of online scheduling, the capability of predicting the consequences of available actions is crucial. In real-time environments, running extensive discrete-event simulations to evaluate how actions unfold requires too much computing time. Artificial neural networks (ANN) are a viable alternative to quickly evaluate the potential future performance value of a production state for online production planning and control. They can predict performance indicators such as the expected makespan given the current production status. Leveraging recent advances in artificial intelligence (AI) game algorithms, an assembly control system based on Google DeepMind's AlphaZero was created. Specifically, an ANN is incorporated into the approach that suggests favorable job routing decisions and predicts the value of actions. The results show that the trained network can predict favorable actions with an accuracy of over 95% and estimate the makespan with an error smaller than 3%.

Keywords Neural networks · Manufacturing · Scheduling · Assembly · Control systems

1 Introduction

The market demand for highly customized products is rising and product life-cycles kept shortening over the last decades [1]. In manufacturing this trend is termed mass customization. Among others, one upcoming additional requirement is more flexibility in terms of shorter manufacturing system

response times due to frequent product changes [2]. The resulting increase in complexity and system dynamics poses a challenge for production systems; flexible and fast adaptation to constantly changing market requirements is a key competitive factor [3]. Adaption of production scenarios has a high efficiency impact especially on the assembly systems, due to the increase and accumulation of complexity along the value chain and the number of sub-processes that are required to be conducted [4]. Hence, high variety product portfolios can become economically feasible in the final assembly by using adaptable systems configurations [5, 6].

In addition to the market side, on the manufacturing side, the organization of assembly systems advances as a reaction to the described additional market requirements. Especially the layout and the material flow are affected by these new organizational forms, described in the following, as they are derived from the concept of job-shop manufacturing [7]. Restrictions of a linear transfer and a super-ordinate cycle time are omitted to create higher degrees of freedom in the job route, which is the sequence of operations performed on the product at a sequence of

✉ Amon Göppert
a.goepfert@wzl.rwth-aachen.de

Leon Mohring
l.mohring@wzl.rwth-aachen.de

Robert H. Schmitt
r.schmitt@wzl.rwth-aachen.de

¹ Laboratory for Machine Tools and Production Engineering WZL, RWTH Aachen University, Campus-Boulevard 30, 52074 Aachen, Germany

² Laboratory for Machine Tools and Production Engineering WZL, RWTH Aachen University Fraunhofer Institute for Production Technology IPT Aachen, Campus-Boulevard 30, 52074 Aachen, Germany

resources [8]. This concept is coined under the terms of matrix assembly [9], modular assembly systems [10] or dynamically interconnected assembly systems (DIAS) [11]. DIAS is taken as a reference system in the following, since it suits most to the dynamical challenges that today's production systems are facing, as illustrated above. Since the job route has increased degrees of freedom, the production planning control is especially affected due to higher solution spaces and computational complexity by the upcoming organizational forms of assembly systems.

From the computer science domain, digital technologies are pushing beyond the existing level of manufacturing execution [12] and enable IT systems to plan and control mass customization and the production of highly customized products [13]. Production planning and control generally comprise the determination of product quantities to be produced in accordance with an optimization of the enterprise's objective such as productivity, profitability, and adherence to due dates [14]. Also, it includes steering of the production processes, the synchronization of stations, and the allocation of stations to tasks over given time periods [15, 16]. The latter is attributed to production scheduling, which is especially influenced by production turbulences, which results in the need of time-sensitive decision-making to quickly react to the turbulences and disruptions? A resulting applied method is re-scheduling with focus on time-sensitive decision-making [17]. Concerning scheduling of complex production systems such as job shops, several review surveys show that machine learning techniques such as supervised learning are increasingly utilized to cope with the challenge of optimizing production schedules [18–21].

Another influence is the more complex job route in DIAS, which increases the solution space for an optimal schedule. The dynamical characteristics of DIAS require timely scheduling and rapid decision-making. The speed at which the control system in DIAS needs to react is bound to real-time conditions, limiting the time-span for a response to a certain threshold [22]. Consequently, online scheduling algorithms, which are applied during the manufacturing execution phase, are suitable for production control in DIAS [23].

An outstanding example of innovations in computer science is the application of machine learning for an artificial intelligence (AI) player in strategic board games, which showed groundbreaking performance results. In 2016 Silver et al. published an AI algorithm, called AlphaGo, that achieved super-human performance in the strategy board game Go and defeated the world champion [24].

The basic components of AlphaGo are a Monte Carlo tree search (MCTS) algorithm to model the decision-making within the game and to find optimal moves. A deep artificial neural network (ANN) is trained to select moves and another ANN is trained for a rapid evaluation of board states.

In 2017 Silver et al. generalized the AlphaGo Zero approach [25], which was a follow-up version of AlphaGo, into the single algorithm AlphaZero and achieved a super-human level in the games of go, chess, and shogi (Japanese chess). AlphaZero starts from random play without any specific expert domain knowledge except the game rules [26]. The universal algorithm architecture and the more general applicability of AlphaZero shows a high potential for use in other domains as also stated by Silver et al. [26].

DIAS require timely decision-making by computing and evaluating large solution spaces for the tasks of online scheduling. The requirements in strategic board game AI development are comparable. For instance, in a board game the reaction times are limited to a range of sub-minutes, which also applies for a broad variety of assembly scenarios, where the processes or takt times are bound to sub-minute ranges. Therefore, an application of the algorithmic architecture from AlphaZero in online scheduling is encouraging.

A critical part of the decision-making in online scheduling is the evaluation of possible decision outcomes [27]. A common approach is using discrete-event simulation (DES) to predict the outcome of a decision as multiple publications show [28–31]. For complex systems a high computational time for the setup and execution of the simulation models is required. This is not following the needs of time-sensitive decision-making [32]. The knowledge approximation from board game AI with ANNs has its strength on time-sensitive decision-making, which is comparable to the needs of online scheduling in DIAS, and showed an extraordinary performance and is, therefore, a promising approach. The ANN offline learning of expert domain knowledge supported by simulation is the main driving factor for the success of AlphaGo and AlphaZero. For online scheduling problems, this would be equivalent to offline learning with DES. As a result, the AlphaZero ANN architecture can easily be transferred and adopted, while the expert domain knowledge in the DIAS online scheduling case is provided by established simulation models. The advances of AlphaGo Zero and AlphaZero consist of the full replacement of expert domain knowledge with simulations predicting the performance impact of game action decisions [26]. As a deduction, for online scheduling a time-sensitive method for evaluating the performance impact of possible scheduling decisions needs to be developed. Recent publications in the field of online scheduling and performance indicator prediction focus on problem types that are not directly comparable to DIAS (e.g. flow shop, sheet-metal production [21, 33, 34]) or do not take the AlphaZero architecture for the ANN input and output layer design into account (e.g. [35–37]). Therefore, the main goal of this work is to decrease the time-span of predicting the makespan of a given system state by training and implementing an ANN that follows the architecture of the promising AlphaZero approach. This ANN is a component

of a potential AlphaZero online scheduling and control algorithm that utilizes its time-sensitive prediction capabilities.

This paper is structured as follows: In Sect. 2, the boundary conditions of the production system and the online scheduling problem as well as the foundations of the AI algorithm AlphaZero will be outlined. In Sect. 3 various relevant publications are summarized and evaluated based on the requirements. Section 4 describes the implementation of a DIAS in a discrete-event simulation. This section also includes the AlphaZero-based scheduling agent, a greedy scheduling agent for comparison, and how they interact with the simulation. In Sect. 5 the data set generation and ANN training procedure are explained. In Sect. 6 the outcome of the supervised learning and hyperparameter optimization are shown and discussed. Finally, in Sect. 7 the work is briefly summarized and reflected, and further research opportunities are presented.

2 Foundations

The foundations create the knowledge base to understand the subsequent chapters, e.g. the state of the art, the presented developments and the results. This chapter comprises explanations on the assembly organisation form of dynamically interconnected assembly systems (DIAS), the flexible job shop problem as a widespread scheduling problem type, online scheduling that is specifically fitting to DIAS and, finally, the relevant basics of AlphaZero as an board game AI.

2.1 Dynamically interconnected assembly systems

Two central aspects of dynamically interconnected assembly systems are routing and resource flexibility. Routing flexibility is the ability to select new assembly stations and assembly steps for each product individually based on the availability of resources [38]. In DIAS resources are stations that offer operations to complete the jobs. Although the station themselves are immovable, there are no spatial restrictions regarding the transport of jobs between any pair of stations. Resource flexibility describes the availability of redundant operations at the assembly stations. Each station should either provide several different operations for one product or one operation for different products to enable a high utilization. High route and resource flexibility enable DIAS to assemble large numbers of variants efficiently. Since the assembly stations work independently in both spatial and temporal dimensions, the assembly system can easily scale up by adding the required stations to the system, while running [8].

The assembly procedure is defined by job routes, which are sequences of stations selected for the assembly of a product. Figure 1 shows examples of job routes for two product types (product A and product B). The figure shows two different job routes for product A. Products with multiple routes can be leveraged to achieve higher utilization of assembly resources throughout the system by choosing less frequented routes. New stations, added to meet an increase in demand, are part of new possible job routes [8].

Fig. 1 The control system is the central software component responsible for scheduling jobs within the DIAS. In contrast to linear assembly systems, the control system assigns the next assembly station and

next operation to each job individually and dynamically, i.e. upon completion of the previous operation [8]. Based on [11]

2.2 Flexible job shop problem

The DIAS scheduling problem is a special case of the *flexible job shop problem* (FJSP). In *flexible job shops* (FJS) an operation can be processed on any machine from a set of alternative machines [19].

A FJSP is characterized by the following data:

- (i) a set of n jobs J_i ($i = 1, \dots, n$)
- (ii) a set of k machines (stations) M_l ($l = 1, \dots, M_k$)
- (iii) each job J_i has a sequence of n_i operations
- (iv) the j^{th} operation ($j = 1, \dots, n_i$) of job J_i is denoted by o_{ij}
- (v) executing operation (o_{ij} on machine M_l takes a processing time of $p_{ij,l}$
- (vi) Each operation o_{ij} has to be performed to complete the job J_i

FJSP is a theoretical model that matches the scheduling problem in dynamically interconnected assembly systems. However, there are some additional complications. First, in FJS operations of a job must be completed in an a priori determined order. DIAS are more flexible and only impose precedence constraints between operations, so there is a degree of flexibility concerning the order of operations of a job. Second, assembly stations are subject to a certain breakdown behavior leading to downtimes during which the machine is not available to execute operations. Finally, each station has a first in, first out queue with limited capacity for jobs. The additional complexity introduced by limiting the number of jobs in the system can have a high impact, e.g. the occurrence of deadlocks. Jobs that just completed an operation may not be able to exit the machine since the downstream resources have no remaining queue capacity.

2.3 Online scheduling

An *offline scheduling* algorithm allocates complete routes, including the machines and starting times, to jobs before any process starts. In contrast, online scheduling takes real-time scheduling decisions based on the current system status. The online approach is especially useful in dynamic environments that require swift adjustments to unforeseen events, e.g. station breakdowns, while offline scheduling is advantageous under static and deterministic conditions. As a consequence the DIAS control system is based on an *online scheduling* approach, i.e. it assigns new routes to jobs dynamically and in real-time during production.

Ouelhadj et al. (2009) differentiate between *predictive-reactive* and *completely reactive scheduling* [39]. Predictive-reactive scheduling consists of two steps. The first one is to

generate a predictive production schedule offline. In a second step, the schedule is updated each time an unexpected disruption occurs to minimize the negative impact on the system performance [40]. Alternatively, in the completely reactive scheduling approach, decisions are made locally and in real-time without a pre-computed schedule. Completely reactive approaches require a small latency between a request and a response to reduce the time a job waits for an assignment while the system is running. Due to the dynamic environment, the DIAS control system is based on a completely reactive online scheduling approach.

2.4 AI in board games: AlphaZero

In recent years, the performance of AI-based algorithms improved considerably. One prominent example is AlphaZero, which achieved breakthrough results in various board games, especially in the game of Go [26]. The holistic approach and the use of the ANN as a source of expertise in the decision process make it a natural fit for the complex DIAS online scheduling problem.

The ANN provides the algorithm with the capability to predict the outcome of a game (*value*) and determine the most favorable actions (*policy*) given the current game state. Each of these two outputs, value and policy, has its own head, which is connected to the main body of the ANN, the residual tower (Fig. 4 shows the ANN structure of our own implementation). The ANN f_θ is characterized by its parameters θ . These parameters are predominantly weights used in the convolutional and dense layers. One of the two outputs of the ANN is the value v of a game state s , which is the predicted outcome of the game. The second output is a policy vector $\mathbf{p}$ that assigns a probability to each potential action available in s [25].

$$(v, \mathbf{p}) = f_\theta(s) \quad (1)$$

A game state s contains the eight most recent positions of the black and white stones on the Go board, including the current positions. Also, it indicates the player who is to play next. The game state is processed by a residual tower consisting of one convolutional block and 19 residual blocks. The convolutional block is composed of a convolutional layer, batch normalization, and a rectifier non-linearity. The characteristic property of the 19 residual blocks is the additional skip connection that offers an alternative route to two convolutional layers. The residual tower processes the received game state and extracts the essential features for the value and the policy head [25].

To obtain the value v of the game state s as well as the policy vector $\mathbf{p}$, the ANN has two heads. The value head

ends with a one-neuron dense layer combined with a *tanh* activation function and estimates the game outcome from the perspective of the color to play. The value v is a scalar between -1 and 1 . An estimated value v that is close to the upper bound indicates a high victory probability. Accordingly, estimated values close to the lower bound indicate a high defeat probability. The last layer of the policy head is a dense layer with the number of neurons corresponding to the number of actions in the game. A softmax activation function generates a probability distribution over all legal actions, this is the policy vector $\mathbf{p}$. In Go, an action is placing a stone on one of the board's 361 fields or to pass and let the opponent play next, so the size of the vector is 362. The policy vector $\mathbf{p}$ provides AlphaZero with the likelihood of each field being selected as the next action and the likelihood of passing. Each likelihood correlates with the success probability of the corresponding action [25].

The AlphaZero decision is based on a Monte Carlo tree search (MCTS) version that integrates the described ANN. The nodes of the search tree represent states s , while the edges represent actions a . During the search tree construction, the algorithm allocates the value v of a state to the corresponding node and the prior action probability p_a of a specific action to the corresponding edge. The MCTS procedure focuses on paths with high-value v states, and edges that represent actions with high likelihoods.

$$l = (z - v)^2 - \pi^T \log(\mathbf{p}) + c \|\theta\|^2 \quad (2)$$

The ANN improves its parameters θ by training on data generated through self-play. As the loss function l (see Eq. 2) shows that the *reinforcement learning* process aims at simultaneously solving a regression task (with the value head) and a classification (with the policy head). The mean squared error of the predicted outcome v and the actual outcome z of a game represents the regression part. The policy head is trained to minimize the cross-entropy of the policy vector $\mathbf{p}$ and the MCTS probabilities π . The decision quality of MCTS is superior to the raw ANN since it can incorporate prediction. Hence, it is possible to train the classification capability of the policy head based on the labels provided by MCTS. The last term of the loss function serves the purpose of regularization [25].

Each decision in every game of Go is a data point that the ANN can train on. The data points are generated entirely through *self-play*, i.e. two versions AlphaZero playing against each other. The ANN is retrained every 30 games on 4,096 randomly drawn decision samples. The final version of AlphaZero for the game of Go has been retrained 700,000 times over the course of 21,000,000 training games [26].

3 State of the art: use of ANNs in online scheduling algorithms

Various publications leverage the capabilities of ANNs in different completely comparable production planning and control scenarios.

Cadavid et al. (2020) present a review on machine learning applied in production planning and control [21] on various use-cases. The most addressed use-cases are in smart planning and scheduling. Also, Cadavid et al. show that the use of ANNs for machine learning applications increased significantly in the last years. An overview of ANN applications and other supervised learning techniques is given in the publication.

A concrete approach to combine supervised learning and DES is the multi-model based prediction of throughput time by Pfeiffer et al. [33]. The described use-case is a flow shop. A DES maps a control system to generate data for the supervised learning technique. A multivariate regression that is closely related to ANNs is trained, validated, and tested on the simulation data. By inserting the statistical prediction models into a simulation-based decision support system, a control system anticipating deviations was created. The simulation was used for reactive control of disturbances as well as for training the prediction models.

Murphy et al. [35] address the problem of order due date assignment in production scheduling. For this purpose, machine learning technologies estimate job makespan in a dynamical job shop with nine stations. A DES framework was utilized to model the dynamical job shop and to provide 10,000 training data records for the machine learning algorithms. As a result, the performance of the proposed machine learning algorithms is better than the conventional due date assignment methods. Within the different machine learning methods that were added for comparison, a random forest approach performs better at high complexity and an ANN produces a higher accuracy on low complexity.

Jong et al. [36] train a multilayer perceptron ANN in a job shop and flow shop environment to quickly predict makespan for a wide range of system configurations. A simulation model serves as the data source. The input parameters are layout (job shop or flow shop), the statistical distribution of job types, processing time on a machine, and number and speed of automated transport vehicles serves as the data source. With a full factorial experiment setup and 100-fold repetition of all experiments to minimize stochastic influences, 230,400 training data sets are generated. In addition to an ANN, the three further machine learning algorithms mixture of experts, extreme learning machine, and support vector machine are trained

separately for the two layout options as a reference. The ANN has the least error, but the training time is longer than for the other algorithms. The higher training time is put into perspective by the fact that an ANN is trained once with a large data set and can then be trained and improved incrementally. Therefore, multilayer perceptron ANN algorithms prove to be superior estimators compared to the other three machine learning algorithms.

Motivated by the significant progress in the field of deep learning and deep Q-learning neural networks, Waschneck et al. (2018) applied these methods to flexible job shop scheduling in the context of semiconductor production. They implement and train one deep Q-network (DQN) agent per station, which is composed of multiple machines. Each agent has the task of selecting a job from the station's queue and allocating it to a machine. The ANN is trained to approximate a policy that selects actions given the state of the system such that the Q value as a sum of rewards is maximized. Waschneck et al. designed a reward function that incentivizes the minimization of the deviation of completion times from due dates. While the cycle times increase slightly, the DQN approach achieves a reduction of delayed jobs by up to 90% [37].

Rinciog et al. implemented a scheduling agent based on AlphaGo Zero [25] to reduce tardiness and material waste in sheet-metal production equivalent to a job shop problem. The scheduling agent allocates the different operations of each part to the machines in the production system. The agent interacts with a discrete event simulation as a surrogate for the real-world production system and receives a scheduling request whenever a machine becomes idle. Rinciog et al. adjusted AlphaGo Zero in terms of state and action space, ANN architecture, and the loss function to fit the sheet-metal production scheduling problem. The input to the ANN is a representation of the production state at the moment of the scheduling decision. The ANN consists of two fully-connected layers with 1024 and 512 neurons, respectively. The ANN has a policy head that allocates prior probabilities to all jobs, and a value head estimating the quality of the current state. Rinciog et al. defined a custom state value function that rewards reducing waste in the cutting process, punishes tardiness, and prioritizes valuable jobs. Rinciog et al. train the network in two stages. First, the network undergoes a pre-training routine, learning from the earliest due date dispatching rule. In the subsequent self-play phase the network learns from the data generated by the agent itself. To validate the AlphaGo Zero agent, the authors compared it to the earliest due date heuristic. After pre-training AlphaGo Zero and earliest due date go head to head. After reinforcement learning, AlphaGo Zero achieved the best score, with lower tardiness and less waste, in 64% of all trials, outperforming earliest due date [34].

The publications presented in this section showed how machine learning models, especially ANNs, sustain decision-making processes. They improve the scheduling agents' performances by providing an a priori estimation of the quality of a production state or a possible action. Most of the research is focused on applications of machine learning agents to job shop and flow shop problems. An ANN especially for dynamically interconnected assembly systems was developed, which differs from most publications, where their problem type definition differs from DIAS. More specifically, Rinciog et al. [34] focus on sheet-metal production, which is not comparable to assembly systems and Rinciog et al. utilize the ANN architecture of AlphaGo Zero, whereas in this work the architecture of the newer AlphaZero is used. AlphaZero is more promising for a wider range of applications outside of the board game domain. Therefore, the approach in this work embeds the predictive qualities of ANNs, according to the AlphaZero architecture.

4 DIAS simulation and ANN implementation

This section provides further details on the implementation of DIAS in a discrete-event simulation, the AlphaZero-based scheduling agent, and how they interact.

4.1 Simulation model architecture

Using simulation enables quickly generating the required amount of data to train the ANN as well as validating its performance. Figure 2 illustrates the simulation of a dynamically interconnected assembly system. The JobGenerator creates new jobs, corresponding to products until the maximum work in progress in the system is reached. The job type allocation object assigns the job type to the newly created job or in other words it specifies the product variant and the operations required to assemble the variant. The job loops through the system using the transport mechanism to visit different stations and implement the required operations. Upon completion of the last operation, the job takes the path to the sink.

Each station's queue holds the jobs awaiting processing. The queues are based on the *first in, first out* principle. The station objects are at the heart of the DIAS simulation model and perform two tasks. They conduct the requested assembly operations and then request a new scheduling assignment for the next loop upon completion of the operation. The station sends the request for the current job to an external scheduling server, which encapsulates the decision-making algorithm. The server returns an assignment, consisting of the next station and the operation (see Fig. 2). If all possible next stations' queues are filled to maximum capacity the job has no other option but

Fig. 2 Simulation of a dynamically interconnected assembly system

to remain in the station and block it. The scheduling server attempts to resolve blocked states whenever queue capacity frees up. The stations are subject to stochastic breakdown behavior. The duration between two subsequent failures is drawn from a Gaussian distribution, whereas the duration of the downtime is based on a uniform distribution.

To embed the simulation logic described above, a new discrete event simulation tool in python named *DIAS-Sim* was developed. It has several advantages over alternative open-source DES modules. *DIAS-Sim* does not cover a broad variety of simulation use-cases but is a light and fast DES tool custom-built for dynamically interconnected assembly systems. The entire simulation logic is condensed into one class. An object of the class can be stored at any decision point. This functionality makes it possible to store a simulation state and relaunch the simulation from there later. Different actions can be tested at any decision point. This is a crucial feature for the construction of the search tree used in the AlphaZero scheduling agent. As mentioned before, scheduling decisions are taken by an external entity, the scheduling server. The server itself is contained in a single class and can quickly load one of the various decision-making algorithms, enabling plug-and-play testing of new scheduling agents. The server can run as a local web application to connect to other simulation programs and real-world applications.

DIAS-Sim has a set of simulation-defining parameters, such as the number of stations, the number of job types, and the maximum work in progress. Upon creation of an instance, *DIAS-Sim* requires a set of values for the parameters. The values specify the layout and characteristics of the assembly system and program to be simulated. The set of values defines the *DIAS* scenario.

4.2 AlphaZero agent: ANN design

An AlphaZero scheduling agent for *DIAS* was developed based on the explanations given in Sect. 2.4. The AlphaZero agent assigns a station-operation tuple to a job based on ANN guided Monte Carlo tree search and relies on *DIAS-Sim* to make simulation states available in all nodes of the search tree. This subsection presents the ANN of the modified AlphaZero.

The ANN provides an estimate of the value of a given state and an a priori indication of how favorable the available actions are. In the case of the *DIAS* application of AlphaZero, the ANN's input is a one-dimensional feature vector and a representation of the legal actions. The feature vector represents the state of the assembly system and consists of different components shown in Fig. 3. It can be extracted from *DIAS-Sim* at any decision point. The left side of Fig. 3 shows some general metrics included in the feature vector, like the simulation time or the processing times of the operations. The italic font under the name of the components indicates the size of the component. Since they are among the main objects of the simulation there are more detailed statistics for the stations in the feature vector. For each station, the vector contains statistics like the number of jobs in transport headed to the station or the queue length.

The legal actions are all feasible station-operation tuples expressed in a binary vector. Table 1 shows the legal actions in a matrix format.

In the given example the only available scheduling options are operation 1 at station 5 or operation 3 at station 2. Flattened, this matrix corresponds to the legal actions vector.

Fig. 3 Overview of the components of a feature vector representing a DIAS simulation state

Table 1 Representation of legal actions

		Operation							
		1	2	3	4	5	6	7	8
Station	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0
	5	1	0	0	0	0	0	0	0

Fig. 4 Residual tower, value head, and policy head of the AlphaZero scheduling agent ANN

The feature vector and the legal actions combined are processed by the ANN shown in Fig. 4. It starts with a block consisting of batch normalization, a dense layer with $n_{Neurons}$ neurons, a ReLU activation function and dropout (see Fig. 4). This block repeats n_{Blocks} times. The number of neurons, the number of blocks, the bias term, and the dropout rate are subject to hyperparameter optimization.

The value head consists of another dense layer with a ReLU activation function. The output is the state value v . The value is greater than zero and smaller than one, enforced by the sigmoid activation function. It is an estimate of the simulation progress, which is defined as the ratio of the simulation time to the makespan. Of course, the simulation time is a known variable in the equation and the value network could estimate the makespan directly. However, the selected approach of predicting the simulation progress has two advantages. First, an absolute state value less or equal than one is best practice in the majority of MCTS applications and facilitates comparability. Second, the performance of the ANN might decrease in case of a large discrepancy between the output magnitudes of the value and the policy head.

As shown in Fig. 4 the policy head starts with a dense layer and a *ReLU* activation function, identical to the value

head. Before applying a softmax function, the logits of the final dense layer are multiplied with the binary legal actions vector. The multiply layer masks illegal actions, which are infeasible station-operation combinations. Analogous to the approach in the game of Go the policy vector p assigns a likelihood to the legal actions that correlate to their success probabilities.

Both the sizes of the input and the output of the ANN are dependent on the number of stations and operations in the assembly system. Therefore, one ANN is dedicated to a fixed number of stations and operations. Varying one of these parameters requires training a new adequately scaled ANN.

4.3 Greedy agent

Another agent to schedule jobs in the DIAS is the greedy agent. It is required to convert the pretraining scenarios in training data for the supervised learning phase of the AlphaZero agent's ANN. It iterates through all scheduling options (station-operation tuples) and selects one of them according to algorithm 1. The greedy approach aims at finding a local optimum and minimizing the time until the job can start the next operation.

Table 2 All factors and levels used for the purpose of generating a large set of scenarios for training and validation

Factor	Level I	Level II	Level III	#Levels
Number of stations	5			1
Number of jobs	25	40	55	3
Number of job types	1	5	9	3
Number of operations	8			1
Min. operations per job type	1	4	7	3
Max. operations per job type	3	8		2
Number of additional operations	2	10	18	3
Transport speed	1	4		2
Max queue length	3	8	13	3
Distance	28			1
Min. processing time	30			1
Max. processing time	100	600		2
Operational flexibility	2	6	10	3
Random seed (replications)	422			1
WIP	4	17	30	3
MTBF (mean, std)	(250, 30)	(1500, 200)		2
MTTR (min, max)	(10, 60)			1

Result: Next Station-next operation tuple

Initialization:

1. Generate a list of feasible scheduling options, i.e. station-operation combinations
2. Initialize $t_{min} = \infty$

```

for (station,operation) in list do
  p = processing time of the operation
  r = remaining processing time of the station
  q = combined processing time of the jobs in the station's queue
  m = combined processing time and remaining transport time of the jobs headed
  to the station
   $t = p + r + q + m$ 
  if  $t < t_{min}$  then
     $t_{min} \leftarrow t$ 
     $next\_station \leftarrow station$ 
     $next\_operation \leftarrow operation$ 
  end
end

```

Algorithm 1: The greedy scheduling decision process.

5 Validation and experiment setup

The AlphaZero agent and its ANN require a large data set for training and validation. The data set must be based on a wide range of different DIAS scenarios to prevent overfitting. The left-hand column of Table 2 lists the parameters that can be varied to generate different DIAS scenarios. A full-factorial DOE approach to vary these factors and considered the levels given in Table 2 was used.

Each level combination of the listed factors corresponds to one scenario. Multiplying the number of levels per factor, listed in the right-hand column of the table, yields the theoretical number of scenarios. In this case, it is 34,992. However, unrealistic combinations that would cause errors in the simulation, e.g. a scenario where stations offer fewer operations than required for the jobs, were filtered. The number of remaining scenarios is 10,944. The scenario analysis generates a simulation data file for each of these scenarios. Throughout the training procedure, it showed that the average number of scheduling decisions per scenario is about 180. Each scheduling decision is a data point consisting of a feature vector, legal actions-vector, and labels for the value and policy head. The total number of data points is roughly $1.97 * 10^6$.

The number of stations and the number of operations are constant (see Table 2). These two parameters directly affect the architecture of the ANN. Altering them would require a complete rerun of the training procedure from the start. However, the agent can train its scheduling capabilities for new DIAS setups by interacting with the simulation leaving the real-world system unaffected. Once the agent is fully trained it can be deployed in the control system.

Analogous to Rinciog et al. [34] and Waschneck et al. [37] the training phase is split into two phases, pretraining (supervised learning) and self-play (reinforcement learning).

Within the scope of this paper, the focus is to find a suitable ANN structure for the task at hand using the pretraining data set. The ratio of scenarios used for pretraining to those used for self-play is 2 to 1. The greedy agent presented in 4.3 converts the pretraining scenarios in training data for the supervised learning phase of the AlphaZero agent's ANN. Each greedy scheduling decision consists of the labels for the value head and the policy head, z and π , as well as the state representation s comprising the feature vector and the legal actions-vector. This corresponds to one data point. The algorithm is quite fast and requires only about 30 min to simulate the scenarios in the pretraining data set. That is a share of 40% of all generated scenarios and amounts to about $0.78 * 10^6$ data points. The greedy algorithm is designed to minimize the period of time before a job starts its next operation. This approach also reduces the overall makespan and is a solid basis for pretraining the AlphaZero ANN.

First, a grid-search based optimization of the hyperparameters *number of neurons*, *number of blocks*, *bias*, and *dropout rate* was conducted. Then, the most promising ANN version to completion were trained. The aim is to find the ANN with the highest capability to accurately evaluate the state of the simulated DIAS and select favorable actions.

6 Results

6.1 Hyperparameter optimization

The hyperparameter optimization uses 85% of the pretraining data set to train different ANNs. Training aborts after 10 epochs. An epoch is one training cycle through the entire data set. Given the complex structure of the ANN and the size of the pretraining data set, it takes considerable computational time to conduct the hyperparameter optimization.

Fig. 5 Influence of the ANN parameters on the network’s performance

Fig. 6 Scatter plot of the ANN hyperparameter optimization results

Even with the number of epochs restricted to 10, it takes about 20 h to train the different ANN versions. In a final step, the other 15% of the data set that the ANN did not train on, are used to validate the networks’ performances.

The validation statistics are the basis for selecting one of the ANN configurations.

The prototype ANN has four hyperparameters that can be optimized (see Fig. 4). The parameters and the tested values are:

- n_{Blocks} : 3, 4, 5
- $n_{Neurons}$: 128, 256, 512
- $dropout_rate$: 0, 0.2
- $bias$: False, True

Figures 5 and 6 illustrate the results of the hyperparameter optimization. Figure 5 illustrates the impact of the four hyperparameters on the performance of the ANN. Each of the subplots shows the policy head accuracy on the y-axis (see Eq. 4) and the value head mean absolute error on the x-axis (see Eq. 5) of each value combination from the list above.

$$Accuracy = \frac{\sum_{j=1}^N f(p_j, \pi_j)}{N} \quad (3)$$

Table 3 Characteristics of the network configurations with the smallest losses

	Neurons	Bias	Dropout	Blocks
1	512	True	0.0	5
2	512	False	0.0	4
3	512	False	0.0	5

$$f(\mathbf{p}, \boldsymbol{\pi}) = \begin{cases} 1, & \text{if } \mathbf{p} = \boldsymbol{\pi} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

$$MAE = \frac{\sum_{j=1}^N |v_j - z_j|}{N} \quad (5)$$

The accuracy reflects the probability of the policy head correctly predicting the action selected by the greedy agent across all data points j in a data set of size N . That is the case, when the predicted policy vector $\mathbf{p}$ corresponds to the policy head label $\boldsymbol{\pi}$. The mean absolute error (MAE) can be interpreted as the average discrepancy between the progress predicted by the value head v and the actual progress z . Since a high accuracy and a low MAE are desired the network configurations in the top left corner of the scatter plots are of interest.

The top left subplot shows that ANNs with more neurons in the dense layer generally score better than those with fewer neurons. The points in the top left corner all have 512-neuron dense layers. There is no obvious indication of whether the bias term has a positive or negative influence on the

network's performance (see top right subplot). However, the bottom left plot clearly shows that dropout harms the accuracy and the MAE. Possibly, the positive effect of dropout starts to settle in later epochs. The bottom right subplot indicates that networks with fewer blocks achieve a lower MAE, while networks with more blocks have higher accuracy.

Figure 6 plots the policy head accuracy and the value head mean absolute error of each hyperparameter combination. Additionally, labels of the three data points with the smallest loss are included in the plot. The accuracy of the policy head and the MAE of the value head of all three ANN configurations are promising.

The final decision is to move forward with an ANN consisting of four blocks, with 512-neuron dense layers, no dropout, and no bias term, as indicated with 1 in Table 3. The other two candidates, which consist of five blocks, would require longer training times in the following steps, but do not demonstrate significantly better performance. The selected ANN achieves an MAE of 3.4% and an accuracy of 93.8%.

6.2 Supervised learning

During supervised learning, the ANN configuration selected in the hyperparameter optimization step is trained until overfitting settles in. The ANN trains on the entire pretraining data set. Figure 7 shows the training and validation statistics regarding loss, accuracy, and MAE throughout pretraining.

Identical to the hyperparameter optimization, the validation split is 15%. The base learning rate is 0.0001. The

Fig. 7 Loss, value head MAE, and policy head accuracy throughout the supervised-learning phase. Final validation loss: 0.065|final validation MAE: 2.99%|final validation accuracy: 95.14%

learning rate controls the pace of the learning process. The selected value facilitates a quick yet stable learning process. With the help of callback functions, the learning rate can be lowered throughout the training procedure when learning slows down. A learning rate reduction by a factor of 10 on a plateau in epoch 26 causes a sudden increase in the accuracy and a corresponding decrease in loss. The training stops when the validation loss does not decrease in six consecutive epochs. This is the case in epoch 38. As the left subplot in Fig. 7 shows, overfitting starts to settle in at this point as the validation loss starts to overtake the training loss. From this point on, the ANN may improve its performance on the training data set but the generalizability of the learned patterns fades. The final validation loss is 0.065, the final validation accuracy is 95.14%, and the final MAE is 2.99%. In other words, the ANN can correctly predict the greedy decision in over 95% of all cases and can estimate the simulation progress, and therefore the makespan, with a margin of error of approximately 3%.

7 Conclusion and outlook

In this work, a performance prediction method with ANNs for online scheduling problems within dynamically interconnected assembly systems (DIAS) was proposed. As the dynamical environment of DIAS requires fast online decision-making, discrete-event simulation is not a possible method to be applied, due to the high needed computational times. The recent breakthroughs in AI algorithms for strategic board games present a promising solution for fast decision-making in a setup comparable to online scheduling problems. The online scheduling support task that was chosen for this work was the prediction of a performance indicator (makespan) to quickly evaluate a given system state. For this task, the advances from the AI game algorithm AlphaZero were leveraged. Specifically, an ANN that can suggest favorable job routing decisions and predicts the value of actions was created. The estimations rest upon a custom-designed feature vector and a legal actions matrix. Optimization of the ANN hyperparameters showed that a network with four layers, no dropout, no bias term, and 512 neurons per dense layer performs best. The fully trained network can predict favorable actions with an accuracy of over 95% and estimate the makespan with an error smaller than 3%.

There are several options to further develop this project and enhance the AlphaZero agent's performance. The data set used for training can be extended. Broader value ranges of the scenario parameters increase the generalizability of the learned strategies. Switching to fractional factorial DOE scenario creation or random scenario selection can help to reduce the computational burden of training data

generation while maintaining a wide diversity of scenarios. Even though it requires training a new ANN each time, the number of stations and operations should be varied beyond the values five and eight as used in this work. The increased complexity of larger DIAS layouts might benefit holistic scheduling approaches like AlphaZero.

The capabilities of the ANN could be further enhanced by including the history of the preceding states in the input. The most recent DIAS feature vectors could be stacked and serve as input to a convolutional neural network. Neural network configurations can be compared using enhanced hyperparameter tuning methods, e.g. Bayesian optimization

Throughout this work, the focus was the design of the ANN and the supervised learning stage. The parameters of the underlying MCTS were set by conducting a small-scale DOE. Further research should investigate a broader range of MCTS parameter constellations to optimize the performance of AlphaZero in flexible assembly systems. Also, future research on the subject should consider the reinforcement learning phase. The ANN requires training on data generated through self-play to surpass the heuristic it learned from in the supervised learning phase. Another open task is thoroughly benchmarking the AlphaZero scheduling agent. The algorithm must be compared to the greedy heuristic and schedules generated by an integer-linear program solving the DIAS scheduling problem optimally.

Acknowledgements This work is part of the research project “AIM-FREE” that is funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) within the “directive on a joint funding initiative to fund research and development in the field of electro mobility” (Funding number: 01MV19002A) and supported by the project management agency German Aerospace Center (DLR-PT). The author is responsible for the content of this publication. Simulations were performed with computing resources granted by RWTH Aachen University under project thes0843.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

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

References

- Groß E, Siegert J, Bauernhansl T (2017) Changing requirements of competence building due to an increase of personalized products. In: *Procedia Manufacturing*, vol. 9, pp. 291–298, 2017. 7th Conference on Learning Factories, CLF
- Trattner A, Hvam L, Forza C, Herbert-Hansen Z (2019) Product complexity and operational performance: a systematic literature review. *CIRP J Manuf Sci Technol* 25:69–83
- ElMaraghy H, Schuh G, Elmaraghy W, Piller F, Schönsleben P, Tseng M, Bernard A (2013) Product variety management. *CIRP Ann Manuf Technol* 62(2):629–652
- Wiendahl H-P, Gerst D, Keunecke L (2004) *Variantenbeherrschung in der Montage*. Springer, Berlin, Heidelberg
- Hu S, Ko J, Weyand L, ElMaraghy H, Lien T, Koren Y, Bley H, Chrysosouris G, Nasr N, Shpitalni M (2011) Assembly system design and operations for product variety. *CIRP Ann* 60(2):715–733
- Chinnathai MK, Alkan B, Harrison R (2017) Convertibility evaluation of automated assembly system designs for high variety production. *Procedia CIRP* 60:74–79
- Khayankit Y, Liangrokapart J (2018) Lean application in a jop shop: a case study of tyre mold plant. *Kasem Bundit Eng Jou* 8:386–399
- Schmitt R, Göppert A, Hüttemann G, Lettmann P, Schönstein D, Schreiber A, Serbest E, Steffens M (2017) *Frei verkettete wandlungsfähige Montage*. In: *Internet of Production für agile Unternehmen*. AWK Aachener Werkzeugmaschinen-Kolloquium 2017
- Greschke P (2016) *Matrix-Produktion: Konzept einer taktunabhängigen Fließfertigung*. PhD thesis, Dissertation, Technische Universität Braunschweig
- Kern W, Rusitschka F, Kopytynski W, Keckl S, Bauernhansl T (2015) Alternatives to assembly line production in the automotive industry. In: *Proceedings of the 23rd International Conference on Production Research (IFPR)*
- Hüttemann G, Göppert A, Lettmann P, Robert H (2017) Dynamically interconnected assembly systems—concept definition. *Requir Appl Anal*
- Park B, Jeong J (2020) A cps-based iiot architecture using level diagnostics model for smart factory. In: Gervasi O, Murgante B, Misra S, Garau C, Blečić I, Taniar D, Apduhan BO, Rocha AMA, Tarantino E, Torre CM, Karaca Y (eds) *Computational Science and Its Applications— ICCSA 2020*. Springer International Publishing, Cham, pp. 577–587
- Kim M, Jeong J, Bae S (2019) Demand forecasting based on machine learning for mass customization in smart manufacturing. In: *Proceedings of the 2019 International Conference on Data Mining and Machine Learning, ICDML 2019*, (New York, NY, USA), p. 6–11, Association for Computing Machinery
- Hu Y, Guan Y, Han J, Wen J (2017) Joint optimization of production planning and capacity adjustment for assembly system. In: *Procedia CIRP*, vol. 62, pp. 193–198. 10th CIRP Conference on Intelligent Computation in Manufacturing Engineering - CIRP ICME '16. [Edited by: Roberto Teti, Manager Editor: Doriana M. D'Addona]
- Chapman S, Gatewood AK, Arnold TK, Clive L (2016) *Introduction to materials management*. Pearson Higher Ed
- Moeuf A, Pellerin R, Lamouri S, Tamayo-Giraldo S, Barbaray R (2018) The industrial management of smes in the era of industry 4.0. *Int J Prod Res* 56(3):1118–1136
- Pinedo ML (2014) *Scheduling*. Springer, 5th ed. 2016 ed
- Mohan J, Lanka K, Rao A (2019) A review of dynamic job shop scheduling techniques. 30:34–39
- Chaudhry IA, Khan AA (2016) A research survey: review of flexible job shop scheduling techniques. *Int Trans Oper Res* 23(3):551–591
- Xie J, Gao L, Peng K, Li X, Li H (2019) Review on flexible job shop scheduling. *IET Collab Intell Manuf* 1(3):67–77
- Cadavid JPU, Lamouri S, Grabot B, Pellerin R, Fortin A (2020) Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. *J Intell Manuf*: 1–28
- Harmonosky C, Robohn SF (1991) Real-time scheduling in computer integrated manufacturing: a review of recent research. *Int J Comput Integrat Manuf* 4(6):331–340
- Suwa H, Sandoh H (2013) Online Scheduling in Manufacturing
- Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap T, Leach M, Kavukcuoglu K, Graepel T, Hassabis D (2016) Mastering the game of Go with deep neural networks and tree search. *Nature* 529(7587):484–489
- Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, Chen Y, Lillicrap T, Hui F, Sifre L, Van Den Driessche G, Graepel T, Hassabis D (2017) Mastering the game of Go without human knowledge. *Nature* 550(7676):354–359
- Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T, Lillicrap T, Simonyan K, Hassabis D (2017) Mastering chess and shogi by self-play with a general reinforcement learning algorithm. [arXiv: 1712.01875](https://arxiv.org/abs/1712.01875)
- Frantzén M, Ng AH, Moore P (2011) A simulation-based scheduling system for real-time optimization and decision making support. In: *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 4, pp. 696–705. Conference papers of Flexible Automation and Intelligent Manufacturing
- Hollenstein L, Lötscher A, Luccarini F (2018) Simlack: simulation-based optimization and scheduling of generic powder coating lines. In: 24. Symposium Simulationstechnik (ASIM 2018), Hamburg, Deutschland, 4.-5. Oktober 2018, pp. 45–50, Argesim
- Frazzon E, Albrecht A, Pires M, Israel E, Kück M, Freitag M (2018) Hybrid approach for the integrated scheduling of production and transport processes along supply chains. *Int J Prod Res* 56(5):2019–2035
- Vieira G, Kück M, Frazzon E, Freitag M (2017) Evaluating the robustness of production schedules using discrete-event simulation. *IFAC-PapersOnLine* 50(1):7953–7958
- Dallasega P, Rojas R, Bruno G, Rauch E (2019) An agile scheduling and control approach in eto construction supply chains. *Comput Ind* 112
- Cashmore M, Coles A, Cserna B, Karpas E, Magazzeni D, Ruml W (2018) Temporal planning while the clock ticks. 2018:39–46
- Pfeiffer A, Gyulai D, Kádár B, Monostori L (2016) Manufacturing lead time estimation with the combination of simulation and statistical learning methods. In: *Procedia CIRP*, vol. 41, pp. 75–80. Research and innovation in manufacturing: key enabling technologies for the factories of the future—Proceedings of the 48th CIRP Conference on Manufacturing Systems
- Rinciog A, Mieth C, Scheikl PM, Meyer A (2020) Sheet-metal production scheduling using AlphaGo Zero. In: *Conference On Production Systems and Logistics*, pp. 1–10
- Murphy R, Newell A, Hargaden V, Papakostas N (2019) Machine learning technologies for order flowtime estimation in manufacturing systems. *Procedia CIRP* 81:701–706
- de Jong AW, Rubrico JJ, Adachi M, Nakamura T, Ota J (2017) Big data in automation: towards generalized makespan estimation in shop scheduling problems. In: 2017 13th IEEE Conference on Automation Science and Engineering (CASE), pp. 1516–1521, IEEE

37. Waschneck B, Reichstaller A, Belzner L, Altenmuller T, Bauernhansl T, Knapp A, Kyek A (2018) Deep reinforcement learning for semiconductor production scheduling. In: 2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference. ASMC 2018:301–306
38. Browne J, Dubois D, Rathmill K, Sethi SP, Stecke KE et al (1984) Classification of flexible manufacturing systems. *FMS Mag* 2(2):114–117
39. Ouelhadj D, Petrovic S, Scheduling A (2009) Survey of dynamic scheduling. *J Schedul* 12:27
40. Vieira GE, Herrmann JW, Lin E (2003) Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *J Schedul* 6(1):39–62

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