Simulation in Produktion
und Logistik 2021
Jörg Franke & Peter Schuderer (Hrsg.)
Cuvillier Verlag, Göttingen 2021

Towards Production-Ready Reinforcement Learning Scheduling Agents: A Hybrid Two-Step Training Approach Based on Discrete-Event Simulations

Beitrag zu einem produktionsreifen Einsatz von Reinforcement Learning-Planungsagenten: Ein hybrider zweistufiger Trainingsansatz basierend auf ereignisdiskreten Simulationen

Marco Kemmerling, Vladimir Samsonov, Daniel Lütticke, IMA, RWTH Aachen University, Aachen (Germany), marco.kemmerling@ima.rwth-aachen.de, vladimir.samsonov@ima.rwth-aachen.de, daniel.luetticke@ima.rwth-aachen.de

Günther Schuh, Andreas Gützlaff, Matthias Schmidhuber, Tim Janke, Werkzeugmaschinenlabor WZL der RWTH Aachen, Aachen (Germany), g.schuh@wzl.rwth-aachen.de, a.guetzlaff@wzl.rwth-aachen.de, m.schmidhuber@wzl.rwth-aachen.de, t.janke@wzl.rwth-aachen.de

Tobias Meisen, TMDT, University of Wuppertal, Wuppertal (Germany), meisen@uni-wuppertal.de

Abstract: Reinforcement Learning (RL) applications for the tasks of Production Planning and Control (PPC) are gaining the broad attention of the research community. Most of the current research results concentrate on PPC problems such as order release, sequence building, or transportation planning and rely on custom-made production simulations for RL. While this is well suited for conceptual research, it is currently inconceivable in practice due to the complexity of real production environments and the trend towards standardized software applications. Since interfacing with standardized simulations to train RL agents limits the speed at which training can occur, we propose a hybrid RL approach involving pre-training on a fast and flexible python-native production simulation with subsequent training and evaluation on the industrial-grade commercial solution. To enable pre-training in similar conditions as encountered in practice, we further demonstrate how to generate realistic production scenarios using real production data.

1 Introduction

Manufacturing companies are reacting to the increasing demand for customized products by designing their production to be more flexible and by moving towards job shop production structures with individual workstations and undirected material flows (Schuh et al., 2019; Zijm and Regattieri, 2019). This flexibility comes at the cost of significantly increased complexity in Production Planning and Control (PPC) (Schlegel et al., 2019). In order to fulfil the general production targets and keep logistical costs low, advanced order release and sequencing methods are required. Multiple exact methods, meta-heuristics and dispatching rules already exist to address these problems (Xie et al., 2019). However, finding a complete scheduling solution upfront, which may then become obsolete shortly afterwards, unnecessarily consumes computational time. For this reason, continuous online scheduling, where only parts of a solution are computed as they are required based on information of the system's current state, might be preferable in practice (Lang et al., 2019). However, there is no universal method suitable for online scheduling capable of finding close-to-optimal solutions in a limited time.

Multiple recent studies investigate the applicability of Reinforcement Learning (RL) for order release control as well as sequence building in production (Kuhnle et al., 2020; Lang et al., 2020; Waschneck et al., 2018) and demonstrate that the application of RL can yield competitive solutions in this domain. Especially job shop production systems – further gaining in importance for customized industrial goods – are difficult to optimize with present methods. Training of RL agents requires trial-and-error and is thus not advisable within a real production system, because an untrained agent will perform many deficient decisions. To avoid unnecessary costs such agents are typically trained on simulation environments.

Once trained, an agent can be transferred from the simulation to the real counterpart. These simulations need to closely depict the complexity of a real production system while being performant, scalable and offering the ability to interface with RL agents at the same time. Industrial-grade commercial solutions such as Automode, AnyLogic or Plant Simulation fulfill the complexity requirements, but lack defined communication interfaces with RL agents and possibilities in parallelization and thus limit the feasibility of fast training. Currently, as far as the literature research has shown, no research is available on how to combine the flexibility and parallelization of custom-made simulation solutions with the error-proofed realism of commercial solutions. In this study, we investigate the aspect of combining an RL agent, fast python-native discrete-event simulations and further detailed commercial simulation models (using Plant Simulation) for order release tasks in a job shop production, which offers a novel strategy to mitigate current shortcomings and brings RL-based solutions one step closer to deployment.

2 State-of-the-Art

Considerable uncertainty levels and constantly changing conditions in production environments require adaptive online planning. Simulation methods offer high flexibility in dealing with uncertainty and finding planning solutions that satisfy the production goals at hand. However, the number of parameters to consider in complex production setups makes the search space too large for an exhaustive search. It is common to apply hybrid methods combining production simulation with metaheuristic, Mixed-Integer Programming (MIP) or Constraint Programming (CP) solvers. Nevertheless, such hybrid solutions result in high computational times for the considered planning period and leave little space for online scheduling applications. (Xie and Allen, 2015)

One promising research direction is the application of learning-based methods such as Reinforcement Learning for PPC activities. The main advantage of learning-based methods is the possibility to anticipate good solution strategies for given production conditions and goals with little computation time involved. The option to continuously learn and refine solution strategies without human supervision gives the learning-based methods a considerable advantage over common heuristic approaches (Bello et al., 2016).

Multiple studies formulate PPC as a Markov Decision Problem (MDP) (Howard, 1960) with subsequent use of RL. Building on the MDP process formulation, Qu et al. (2015) adopt a single Q-learning RL agent to control a multi-stage production system with three machines and the goal of minimizing the overall waiting time. Evaluation against a First-In-First-Out (FIFO) priority rule, a custom-made greedy heuristic and multi-agent Q-learning demonstrates better performance of the proposed scheduling RL agent. Training and evaluation are conducted in a custom-made simulation environment implemented in python.

The study of Waschneck et al. (2018) demonstrates one of the first deep RL applications on scheduling tasks for the example of semiconductor wafer processing. A custom-made production model implemented in MATLAB is used for training and evaluation. Three Deep Q-Network (DQN) agents independently control three machines and minimize the cycle time spread for three product groups. An alternative scenario used for comparative evaluation relies on a production setup with each machine being controlled according to an individual heuristic. The authors demonstrate a lower variation of production cycle times by using RL agents.

Thomas et al. (2018) adopt a single-agent DQN RL setup for job shop scheduling. The overall goal is the maximization of the production system throughput. Additionally, a neural network is introduced for continuous bottleneck detection. Recognized bottlenecks are mitigated via a temporal increase of throughput for involved machines. The production model is implemented as a discrete event simulation in the AnyLogic software.

Schneckenreither and Haeussler (2019) propose a DQN-based order-release mechanism superior to the backward infinite loading heuristic. The considered application covers a two-stage flow shop with three machines. An RL agent monitors arriving orders and determines when a batch of accumulated orders should be released in the production system. Implementation details of the production simulation are not provided.

Gannouni et al. (2020) address the problem of sequence-dependent setup waste minimization for a single-stage production process. A single RL agent builds production sequences for three machines and delivers good solution results in a fraction of the time required by Google OR-Tools and Gurobi solvers. The adopted neural architecture is a modification of the REINFORCE RL with Pointer Network (Nazari et al., 2018). The material flow and production processes simulation is built in python and uses Machine Learning (ML) models trained on real production data.

Scheduling of flexible job shop production is addressed in the work of Lang et al. (2020). Two independent DQN agents with Q-values approximated by recurrent neural networks are responsible for the selection of orders and machines. Schedules generated by RL agents for production layouts with up to eight machines result in lower total makespan and tardiness compared to the randomized adaptive search (GRASP) metaheuristic, which is selected as a baseline. The production simulation is implemented using the open-source Salabim library in python.

Samsonov et al. (2021) propose a DQN-based RL solution for job shop scheduling problems for makespan minimization. The developed approach features an action space agnostic solution design and an exponential reward shape facilitating learning of near-optimal solutions. Evaluation is conducted on a production layout with six machines in a custom-made simulation implemented in python.

A problem of dynamic route guidance for material handling systems in automated production sites is investigated in the work of Hwang and Jang (2020). The authors propose a new RL algorithm based on the Q(λ) learning approach with low computation overhead. The production and transport system simulation is conducted in the commercial software AutoMode. The achieved average delivery time and the total amount of deliveries suggest a good performance with little computation time compared to several common static and dynamic routing algorithms.

The study of Kuhnle et al. (2020) investigates several action spaces and reward designs for a learning-based production control system. The work focuses on the control of order dispatching and transportation between machines using a multi-agent RL setup and a production simulation implemented in python with the open-source library SimPy. The considered job shop production consists of eight machines unified in three machine groups. An evaluation against rule-based heuristics such as nearest job first (NJF) and FIFO demonstrates that the proposed RL setup can be successfully used for control for various production scenarios.

As Figure 1 shows, most of the RL-based solutions for PPC tasks which provide competitive results for online scheduling tasks rely on custom discrete-event simulations. Tailored solutions offer almost unlimited possibilities for customization and fast distributed training making it a well-suited choice for conceptual studies.

	Task	Order release	Order scheduling	Material routing	**Process flow**	Job shop	Flow shop	Batch process	Continous process	**Simulation**	Python (custom)	MATLAB (custom)	AnyLogic	AutoMode	**Real data**	yes	no
Qu et al. (2015)		○	●	○		○	○	●	○		●	○	○	○		○	●
Waschneck et al. (2018)		●	●	○		●	○	○	○		○	●	○	○		○	●
Thomas et al. (2018)		○	●	○		●	○	○	○		○	○	●	○		○	●
Schneckenreither and Haeussler (2019)		●	○	○		○	●	○	○		○	○	○	○		○	●
Gannouni et al. (2020)		○	●	○		○	○	○	●		●	○	○	○		●	○
Lang et al. (2020).		○	●	○		●	○	○	○		●	○	○	○		○	●
Samsonov et al. (2021)		●	○	○		●	○	○	○		●	○	○	○		○	●
Hwang and Jang (2020).		○	○	●		●	○	○	○		○	○	○	●		●	○
Kuhnle et al. (2020)		●	○	●		●	○	○	○		●	○	○	○		○	●

Figure 1: *Overview of the considered approaches*

However, for widespread use of such methods in manufacturing companies, the applications must be robustly deployable and based on simulation tools already in use to be an alternative to current PPC software. The call for implementation in real production applications will gradually turn the spotlight on the questions of governance and certifications of production simulations used for training and production deployment of RL agents. Current approaches mostly focus on a job shop production showing a promising alternative for heuristics used in the PPC, because of the high complexity of today's manufacturing systems, which can't be controlled down to the smallest detail. In addition to the lack of practical relevance of some approaches in terms of the simulation tools used, the basis of purely synthetic data is also a shortcoming of the approaches used to date and calls for a more practical solution.

3 Approach and Methodology

Our experimental setup aims to demonstrate the general workflow for training and evaluation of RL agents for the task of order release under conditions close to practical applications. We use historical production data and a job shop based production structure implemented in Siemens Plant Simulation as a standard simulation software in the production domain. We design a suitable RL solution and devise an approach for deployment using a combination of two simulations as described in the following.

3.1 Job Shop Scheduling with Reinforcement Learning

Before an RL agent can be successfully applied to a given problem, it needs to go through a training period in which it learns how to solve the problem at hand. Once trained, an agent can make decisions very quickly, but the training period is often lengthy and requires many interactions with the given environment. Minimizing the time required for each individual interaction with the simulation environment is thus of high importance. In practice, such an environment is usually represented by a simulation of the process at hand. Rather than working with a static dataset collected upfront, a simulation allows for the possibility of exploration, i.e. the agent can influence the data it receives by performing actions and receives realistic subsequent state data.

In the production domain, simulations are often implemented with the help of commercial software such as Plant Simulation. If an RL agent is to be of any practical use, it needs to be interoperable with simulation tools already used as a standard by the domain experts. Enabling this interoperability, however, requires overcoming a set of challenges, because the existing commercial simulations are generally not explicitly designed to be used in the RL context. The central problem is that the state-of-the-art RL software packages are not supported by standard simulation software used in the production domain. Therefore, a communication layer must be created between the RL agent and Plant Simulation. While such an interface is well suited for the use with an already trained agent, it is not the ideal solution for training an agent due to the additional time delay introduced by the communication layer.

Hence, we propose a hybrid approach consisting of three stages as depicted in Figure 2: (1) Training of the agent in a custom-made simulation in SimPy closely imitating the desired target simulation and designed to minimize the training time. (2) Transfer learning, i.e. further training of the already trained agent in Plant Simulation, to adapt to minor differences between the two simulations. The number of training steps in this stage is thus kept to a minimum. If the difference between the two simulations is sufficiently small, the second step may be skipped entirely. (3) Application of the trained agent and running simulation in Plant Simulation for dynamic order release in a real production environment.

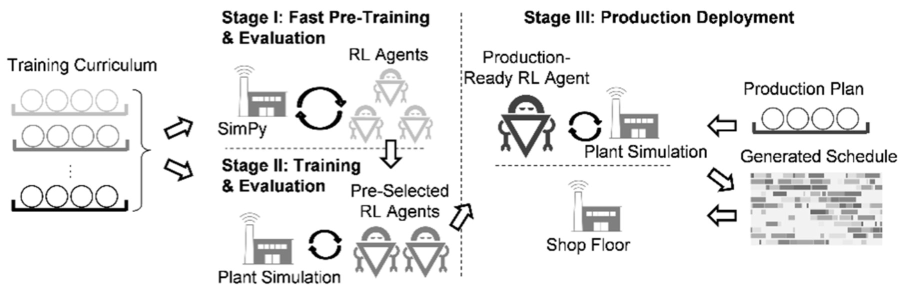

Figure 2: *Hybrid training and evaluation scheme*

To demonstrate the general feasibility of the methodology described above, we use the agent design, state space, and reward function introduced by Samsonov et al. (2021) and apply it to the more limited problem of order release, in which our agent does not schedule every operation, but only the first operation of every order. The remaining operations of each order follow the production plan and are scheduled on each machine according to the FIFO principle.

Due to the more limited scope of the problem, the state space of the agent contains only the backlog and remaining processing time on each machine, the duration of the first operation for each order, the total processing time required for each order and the type of machine required for the first order as well as the current simulation time. The agent is rewarded based on the comparison of the makespan it achieves for the given problem instance and the optimal makespan for this instance, as described in Samsonov et al. (2021).

3.2 Production Data and Simulation in Plant Simulation

The focus on order release in a real production environment necessitates the use of simulation software commonly used in real production plants. Moreover, validation against real historical production data and typical heuristics for order release is suggested, e.g. constant work in progress (CONWIP) or workload control. Therefore, a representative job shop production setup with originally 200 machines and a total of 15,000 depicted production orders is used for this purpose instead of creating an imaginary simulation model well suited for the used methodology but with no further regard to the industry's reality.

For the final setup of dynamic order release at the appropriate time and rapid adaptability of the experiments, we use Siemens Plant Simulation as it is widely used for discrete-event simulation of real production plants, closely modelling a wide range of production characteristics such as layout specific transition times.

Further research on the state space representation is needed to enable the RL methods used in this study to work with production scenarios consisting of hundreds of machines and thousands of orders. To test the hybrid training and evaluation concept proposed in this work, we hence downscale the model and the data set of the real production in two steps, such that the original model and the downscaled model deliver similar results. For this purpose, a simulation of the entire model is carried out and only those orders are considered that occupy the machines during the defined period. Then, the machines to be included are reduced by considering only those that have the highest utilization in the time interval under consideration. In this process, the processing times on missing work centers are represented by additional transition times. This results in a new data set with 160 orders produced on 10 machines in three days, which is an appropriate size for the methods examined here and will serve as the basis for the present study. To ensure comparability, the results of both the original and reduced simulations are evaluated and show a maximum deviation of 3% for the workstations' utilization and 10% for the throughput times.

3.3 Synthetic Training Data and Simulation in SimPy

To enable the first training stage of our approach, we implement a discrete-event production simulation for order release using the SimPy open-source library in python following the OpenAI Gym API, which is specifically designed to be a fast and flexible tool for RL training.

The production structure and program can change considerably over time. Such major changes might require retraining of the order release RL agent while having a limited amount of suitable historical data available. Starting with a limited amount of data consisting of the 20 production scenarios with different available machines and orders to produce (see Section 3.2), we generate a larger amount of synthetic data by mimicking the characteristics of this limited initial data set. Investigation of the operation time histogram and the Q-Q plot (see Fig. 3) of the log-transformed data demonstrates that the production data follows a lognormal distribution. Further investigations of transportation times and operation numbers per order demonstrate similarities to lognormal distributions.

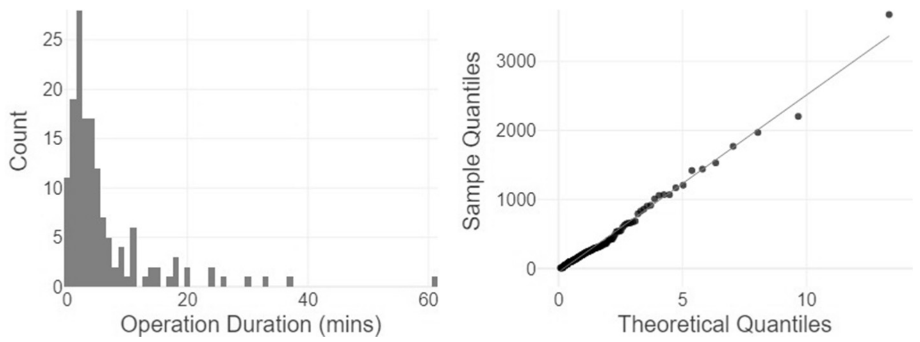

Figure 3: *Distribution of operation durations & Q-Q plot of log transformed operation durations*

Using the 20 available production scenarios, we estimate the probability density of the lognormal distributions for operation and transportation times, as well as the operation numbers per order, in terms of the mean and standard deviation. Those parameters are utilised to generate realistic synthetic production data and ensure the exposure of the RL agent to a wide range of production situations during training.

To evaluate the proposed hybrid training and evaluation scheme, several validation runs demonstrate identical behaviour with the production model implemented in Plant Simulation, yet it potentially lacks the level of implementation reliability provided by Plant Simulation. The python-based simulation is used primarily for fast training at the first stage of the proposed hybrid approach.

3.4 Connecting Plant Simulation and RL Agent

To enable communication between the RL agent and Plant Simulation, an interface to exchange messages between the two components is implemented via network sockets. The proposed methodology requires a bidirectional information flow, with the simulation sending the current state to the agent and the agent in return deciding which orders to release based on this state. This decision is then transferred back to the simulation, which continues to run until another selection is pending. In the resulting message exchange, each component alternatingly sends a message and subsequently waits for a response from the other component.

While messages from the agent to the simulation merely contain the order numbers to be released, the structure of messages from the simulation to the agent is more complex and contains several sections described in Table 1.

Table 1: *Sections of a state message as sent from the simulation to the agent*

Section	Content
Header	Starting positions of the remaining sections in the message
Available Orders	List of orders currently available for release
KPIs	Number of orders that are late, simulation time to date in seconds, total slack of all orders that are not late, delay (negative slack) of late orders
Machine Information	List containing a triplet (Machine name, remaining processing time on machine, machining and setup time in machine queue) for each machine
Finished orders	Orders which have been finished between the previous message and the current one

The state information exchanged through the interface contains only information that may change from step to step. Additional, static information required by the agent is loaded upfront from a file describing the current scheduling instance in order to avoid unnecessarily long messages, which negatively affect the speed of communication. This static information includes the processing times for individual operations, and on which machine each operation needs to be processed.

Knowing the length of a message in advance enables each component to ascertain whether a particular message has been received fully and thus ensures a more robust communication between the two components. Since the state message as communicated by the simulation varies in size from step to step, the simulation will communicate the length of the incoming message before sending its contents.

4 Experimental Results

To validate the approach presented here, we first evaluate the performance of our agent by investigating whether the agent's performance improves during training and how the trained agent fares in comparison with commonly used order release rules in Plant Simulation. As Figure 4 shows, the agent's performance, as measured by the achieved makespan in a problem instance, improves during training, indicating that the agent is learning how to solve the problem.

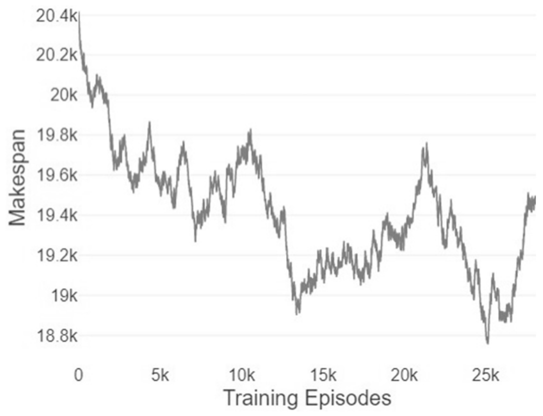

Figure 4: *RL agent performance during the training process (total makespan)*

To investigate the performance of the trained agent, we apply it to each of the instances described in Section 3.2. As a comparison, we further apply the CONWIP rule (Lödding, 2016) with 15, 30 and 50 concurrent orders in the production simulation. Each of these four approaches is then evaluated on two metrics, the utilization rate and the average waiting time, which are deliberately chosen instead of the makespan. The makespan is well suited for training because each training instance consists of a limited number of orders to be released and the agent can then learn to minimise the time it takes for the system to process all orders. In practice, however, there typically is no limit on the number of orders, but rather a continuous influx of new orders, such that the task is never "done". We replicate this behaviour and simply cut the simulation off at a predefined point in time to arrive at an evaluation period. Since the overall makespan is now no longer a useful metric anymore, we turn to the two commonly used metrics described above.

The results are illustrated in Figure 5, where each data point represents the solution of one problem instance by one particular method. Each different method forms a clear cluster with different trade-offs between the utilization rate and the average waiting time. The RL agent, however, achieves both comparatively high utilization rates and low average waiting times.

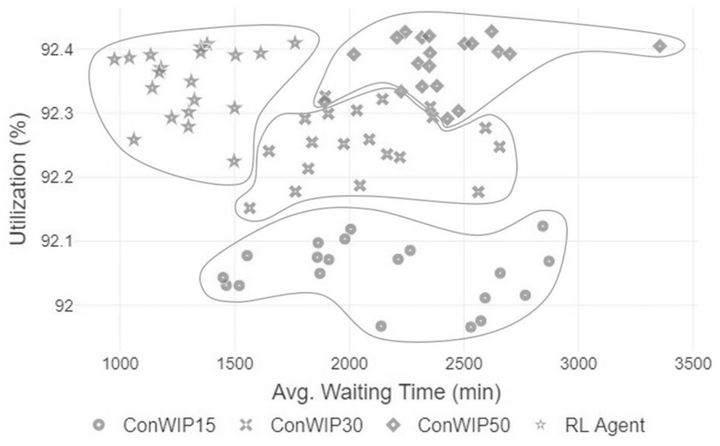

Figure 5: *Quality of the solutions obtained by the agent and CONWIP*

To quantify and emphasize the necessity of the hybrid simulation approach presented here, we investigate the time individual simulation steps require in an RL setting using both simulations. As Figure 6 elucidates, there is a marked difference between the two simulations, with the average step in Plant Simulation requiring 37.6 ms while the average step in the native python simulation only requires 1.4 ms. Although both values may seem fairly small at first glance, the sheer number of steps required in RL, especially in more complex problem settings can lead to hours or days of training time. A simulation that is slower by a factor of ~27, as in this case, can make an already lengthy training process impractically long. This difference in speed is not necessarily because Plant Simulation is inherently slower, but because communication through the socket interface incurs extra time costs.

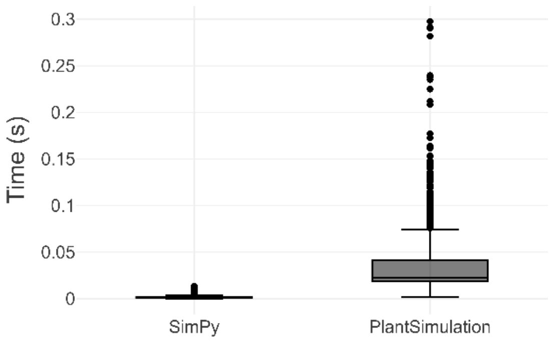

Figure 6: *Time required for one simulation step*

5 Conclusion

In this work, we argue that while RL is evolving into a promising approach to solve common PPC tasks, integrating it into the existing simulation ecosystem used in the production domain remains non-trivial. To enable the usage of RL in conjunction with commercial production simulations, we propose a hybrid methodology and demonstrate its feasibility on the example of order release tasks. Since communication between the commercial production simulation and the RL agent is costly in terms of time, the training becomes impractical. Instead, we propose to perform the bulk of the training in a native python simulation, which closely resembles the original model, and then transferring the trained agent to the commercial simulation. If necessary, the agent can be retrained for a much more limited duration to compensate for any potential differences between the two simulations.

We show that an RL agent trained in the native python simulation on synthetic data designed to resemble the characteristics of real production data can be successfully transferred to a Plant Simulation production environment. Not only does the communication protocol allow the RL agent to control this simulation, but the initially learned dispatching strategies lead to satisfactory outcomes even without additional retraining of the RL agent. We measure these outcomes by the average waiting times and utilization rates across 20 problem instances. As the results in Section 4 show, the RL agent achieves higher utilization rates on average and lower average waiting times compared to the commonly used CONWIP heuristic.

While we demonstrate the general validity of our approach, extending our work to more complex PPC tasks such as sequence building is the main direction for future work. Although further training of the agent in the commercial simulation was not necessary in this example, increasing both the complexity of the tasks performed and the simulation scenarios may lead to greater differences between the two simulations in use. The second training step should therefore remain a part of the presented approach and its necessity should be evaluated on an individual basis.

Acknowledgement

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany‘s Excellence Strategy – EXC-2023 Internet of Production – 390621612

Bibliography

Bello, I.; Pham, H.; Le, Q.V.; Norouzi, M.; Bengio, S.: Neural combinatorial optimization with reinforcement learning. arXiv:1611.09940 (2016).

Gannouni, A.; Samsonov, V.; Behery, M.; Meisen, T.; Lakemeyer, G.: Neural Combinatorial Optimization for Production Scheduling with Sequence-Dependent Setup Waste. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2020, pp. 2640–2647.

Howard, R.A.: Dynamic programming and markov processes (1960).

Hwang, I.; Jang, Y.J.: Q(λ) learning-based dynamic route guidance algorithm for overhead hoist transport systems in semiconductor fabs. International Journal of Production Research 58 (2020) 4, pp. 1199–1221.

Kuhnle, A.; Kaiser, J.-P.; Theiß, F.; Stricker, N.; Lanza, G.: Designing an adaptive production control system using reinforcement learning. Journal of Intelligent Manufacturing 54 (2020) 4, pp. 1196.

Lang, S.; Behrendt, F.; Lanzerath, N.; Reggelin, T.; Müller, M.: Integration of Deep Reinforcement Learning and Discrete-Event Simulation for Real-Time Scheduling of a Flexible Job Shop Production. In: Bae, K.-H.; Feng, B.; Kim, S.; Lazarova-Molnar, S.; Zheng, Z.; Roeder, T.; Thiesing, R. (Hrsg.): Proceedings of the 2020 Winter Simulation Conference. Orlando, USA 2020, pp. 3057–3068.

Lang, S.; Schenk, M.; Reggelin, T.: Towards Learning- and Knowledge-Based Methods of Artificial Intelligence for Short-Term Operative Planning Tasks in Production and Logistics: Research Idea and Framework. IFAC-PapersOnLine 52 (2019) 13, pp. 2716–2721.

Lödding, H.: Verfahren der Fertigungssteuerung: Grundlagen, Beschreibung, Konfiguration. Berlin, Heidelberg: Springer Vieweg 2016.

Nazari, M.; Oroojlooy, A.; Snyder, L.V.; Takáč, M.: Reinforcement Learning for Solving the Vehicle Routing Problem, 2018,

Qu, S.; Chu, T.; Wang, J.; Leckie, J.; Jian, W.: A centralized reinforcement learning approach for proactive scheduling in manufacturing. In: Proceeding of the 20th Conference on Emerging Technologies & Factory Automation 2015, pp. 1–8.

Samsonov, V.; Kemmerling, M.; Paegert, M.; Lütticke, D.; Sauermann, F.; Gützlaff, A.; Schuh, G.; Meisen, T.: Manufacturing Control in Job Shop Environments with Reinforcement Learning. In: International Conference on Agents and Artificial Intelligence (ICAART), 2021, pp. 589–597.

Schlegel, T.; Siegert, J.; Bauernhansl, T.: Metrological Production Control for Ultra-flexible Factories. Procedia CIRP 81 (2019) 3, pp. 1313–1318.

Schneckenreither, M.; Haeussler, S.: Reinforcement Learning Methods for Operations Research Applications: The Order Release Problem. In: Nicosia, G.; Pardalos, P.; Giuffrida, G.; Umeton, R.; Sciacca, V. (Hrsg.): Machine Learning, Optimization, and Data Science. Cham: Springer 2019, pp. 545–559.

Schuh, G.; Prote, J.-P.; Sauermann, F.; Franzkoch, B.: Databased prediction of order-specific transition times. CIRP Annals 68 (2019) 1, pp. 467–470.

Thomas, T.E.; Koo, J.; Chaterji, S.; Bagchi, S.: Minerva: A reinforcement learning-based technique for optimal scheduling and bottleneck detection in distributed factory operations. In: 2018 10th International Conference on Communication Systems & Networks (COMSNETS), Bengaluru, 1/3 - 1/7, 2018, pp. 129–136.

Waschneck, B.; Reichstaller, A.; Belzner, L.; Altenmuller, T.; Bauernhansl, T.; Knapp, A.; Kyek, A.: Deep reinforcement learning for semiconductor production scheduling. In: Proceedings of the 29th Annual 2018, pp. 301–306.

Xie, C.; Allen, T.T.: Simulation and experimental design methods for job shop scheduling with material handling: a survey. The International Journal of Advanced Manufacturing Technology 80 (2015) 1-4, pp. 233–243.

Xie, J.; Gao, L.; Peng, K.; Li, X.; Li, H.: Review on flexible job shop scheduling. IET Collaborative Intelligent Manufacturing 1 (2019) 3, pp. 67–77.

Zijm, H.; Regattieri, A.: Manufacturing Planning and Control Systems. In: Zijm, H.; Klumpp, M.; Regattieri, A.; Heragu, S. (Hrsg.): Operations, Logistics and Supply Chain Management. Cham: Springer International Publishing 2019, pp. 251–271.