Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

DYNAMIC SCHEDULING OF GANTRY ROBOTS USING SIMULATION AND REINFORCEMENT LEARNING

Horst Zisgen
Robert Miltenberger

University of Applied Sciences Darmstadt
Schöfferstr. 3
64295 Darmstadt, GERMANY

Markus Hochhaus
Niklas Stöhr

SimPlan AG
Sophie-Scholl-Platz 6
63452 Hanau, GERMANY

ABSTRACT

Industry 4.0 induces an increasing demand of autonomous interaction between the units of production facilities, like work centers and transportation equipment. This has an impact on the requirements for production scheduling and control algorithms. These must be capable to adapt autonomously to changes on the shop floor. This paper presents a combination of Reinforcement Learning and discrete event simulation for controlling a flexible flow shop using a gantry robot system as transportation unit. In a gantry robot system, parts are transported by carriages fitted with grippers that travel along rails from machine to machine. The presented RL-agent learns autonomously the right control policy to move the carriages. It is shown that in cases the optimal policy can be determined the Reinforcement Learning based policy is optimal and in other cases the achieved throughput does slightly exceed the throughput gained by a heuristic priority rule for controlling the gantry robot.

1 INTRODUCTION

Industry 4.0 induces an increasing demand of autonomous interaction between the units of production facilities, like work centers or transportation equipment. This has an impact on the requirements for production scheduling and control. Corresponding algorithms must be capable to adapt dynamically and autonomously to changing conditions on the shop floor. Typically, dynamic scheduling approaches do apply heuristic dispatching rules. The efficiency of such rules may deteriorate gradually in a stochastic shop floor environment. But a manual development and adaption of these rules is time and resource consuming and relies on the experience of those creating the rules. Therefore, an automated adaption of rules is becoming essential (Renke et al. 2021).

In the course of this paper the dynamic scheduling of flexible flow shop systems with gantry robots based material handling is considered. Typically, high volume manufacturing lines, like those producing components used in automotive or engines, are organized as flow jobs with an automated parts handling system. A flexible approach for such a material handling system are gantry robots, which support alternative routes per part while still be more cost-effective than e.g. Automatic Guided Vehicles (AGVs) or even Autonomous Mobile Robots (AMRs). In a gantry robot system workpieces are transported by carriages fitted with grippers that travel along rails from machine to machine.

An effective gantry robot scheduling and control is crucial for avoiding blocking or starving situations at the machines and for achieving a high throughput rate of the manufacturing line. Given the dynamic and stochastic character of the problem it seems obvious to treat this problem as a Markovian Decision Process (MDP). But even for small systems with just a few machines the size of the state and action space gets intractable. Furthermore, the MDP can not be described completely with regard to the state transition probabilities. Thus a combination of simulation and Reinforcement Learning (RL) is considered

979-8-3503-6966-3/23/$31.00 ©2023 IEEE

as solution approach. The advancements made in the area of RL, in particular from neural fitted Q Iteration towards Deep Q-Networks (DQN), open up new opportunities for the application of RL in the context of manufacturing and Industry 4.0.

This paper provides a RL-agent for controlling a flexible flow shop using a gantry robot system as transportation unit. The presented RL-agent gets trained by using iteratively a discrete event simulation model of the corresponding manufacturing system and a DQN.

The paper is organized as follows. In section 2 we provide a brief review of the related literature. Subsequently gantry robot systems are described in section 3. In section 4 a concise overview of the concepts of RL is given while in the following section 5 the concrete simulation and RL based solution approach is explained. Section 6 presents the evaluation of the RL-agent based on comparison either with available optimal policies or with benchmarks based on a heuristic priority rule. Finally, in section 7 a critical summary and an outlook for future research needs are given.

2 RELATED RESEARCH

There are several RL approaches for production scheduling described in the literature, mainly for job shop scheduling problems. Some recent examples are presented in Waschneck et al. (2018), Xie et al. (2019), Lang et al. (2020), and Zhang et al. (2017). These approaches do not consider constraints due to the limited availability of transportation resources required to load or unload machines. Some recent work which does take Automatic Guided Vehicles (AGVs) as limiting transportation resource into account is presented in Popper et al. (2021), Xue et al. (2018), and Feldkamp et al. (2020). While AGVs are just allowed to move on defined routes, Autonomous Mobile Robots (AMR) can navigate very flexible on the shop floor. Malus et al. (2020) provide a RL based approach for dispatching such AMRs. In Xu et al. (2022) a RL approach for a deterministic job shop scheduling problem with cranes as transportation resources is presented. The problem covered therein assumes that the set of jobs to be scheduled is known upfront while the flow job problem covered by this paper is a dynamic scheduling problem. Furthermore, the approach in Xu et al. (2022) disregards any stochastic interference within the production process.

While there is a lot of research covering RL based job shop scheduling, RL based dispatching for gantry robots has not been studied intensively so far. To the best knowledge of the authors there is just one paper Ou et al. (2018) which provides a tabular Q-learning based dispatching approach for gantry robots. Ou et al. (2019), Ou et al. (2020) are modifications of this approach which are based on tabular Q-learning too. In contrast to these papers this work presents a DQN based approach. Using deep neural networks as function approximators for the state-value function allows to cope with more complex gantry robot systems which are intractable for a tabular Q-learning approach.

A good review of the recent work concerning the application of RL in the context of production scheduling and control can be found in Panzer and Bender (2021).

3 GANTRY ROBOT SCHEDULING

A gantry robot system couples m machines $M_1, \ldots, M_m$ typically arranged in series. The machines are grouped in work centers, whereby a work center could be an individual machine or a set of identical machines performing the same operation. The workpieces to be processed are transported by carriages fitted with grippers that travel along rails from machine to machine. Common gripper types are single or double-gripper. Machines are either busy or idle. Typically, there are no buffers between machines. Workpieces are supplied by an input conveyor adjacent to M_1 where arriving parts are queued and the gantry robot can pick up waiting parts. Analogously, finished parts are deposited by the gantry robot onto an output conveyor - next to M_m - which moves the parts to a stock or another production stage. The gantry robot moves between the machines and picks up or drops off parts. In case of a single gripper the robot can only carry one part at a time. The double-gripper can pick up or drop off parts sequentially and is capable to transport up to two parts a time. In our model fixed pickup and drop-off times are supposed. The travel

times between machines depend on the distance of the machine the robot starts at and the machine the robot is heading for. Thereby the acceleration of the carriages is taken into account.

In our analysis we assume that the parts supply at the input conveyor is stochastically distributed and consequently there might be not always parts available at the input conveyor. Furthermore, processing times are deterministic and machine break downs take place randomly. It is obvious that the key performance indicators, like throughput or lead time, of the production system depend on the control of the gantry robot. Ideally, neither machines are blocked because the robot does not unload a processed part accurately timed nor machines get idle since parts are not supplied on time. A decision has to be made sequentially where to move the robot next. Such a sequential decision making process can be modeled as MDP. Iteratively at each state change of the system the RL-agent has to decide what the best next move of the gantry robot is. Due to the high dimension of adequate state and action spaces and to the lack of transition probabilities the MDP becomes analytically intractable for real size problems and a combination of simulation and RL as approximation needs to be considered. For further details on MDP refer to Puterman (2005).

4 DEEP REINFORCEMENT LEARNING APPROACH

RL is a machine learning approach for solving MDPs. The idea of RL is to learn the optimal decision making policy by training iteratively an RL-agent based on rewards it receives after taking an action. The gantry robot is considered as RL-agent, the action to take is the next movement of the robot and the environment is the production line, including machine states and robot positions, represented by a discrete event simulation model.

Let S_t be the state of the environment at timestep t and A_t the action taken by the RL-agent at timestep t. The reward received in S_t upon action A_t is noted by R_{t+1} and S_{t+1} is the resultant state of the environment after taking action A_t. Eventually, the rewards can be considered as a short term feedback on actions the RL-agent has chosen. In turn, the final policy results from a maximization of the expected accumulated and discounted rewards earned from starting in state S_t, which can be considered as the long term feedback of the environment to the RL-agent for choosing this policy when starting in state S_t. Thus, the objective of the RL-agent is to choose a policy π that makes decisions yielding to a maximization of the so called state-value function

$$v_\pi(s) = E_\pi\left[\sum_{k=t+1}^{\infty} \gamma^{k-t-1} R_k | S_t = s\right] = E_\pi[G_t | S_t = s]$$

for each state s or a maximization of the so called state-action function (or Q-function)

$$q_\pi(s,a) = E_\pi[G_t | S_t = s, A_t = a]$$

for each state-action pair (s,a) respectively, whereby $\gamma \in (0,1)$ denotes the discount factor. In order to avoid a time consuming simulation to evaluate the value of G_t the so called Q-Learning approach was introduced in (Watkins 1989) to approximate the Q-values iteratively by updating the current Q-value by the convex combination

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha \cdot \left[R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t)\right], \alpha \in (0,1)\ ,$$

whereby $R_{t+1} + \gamma \max_a Q(S_{t+1},a)$ is considered as the target and its difference to $Q(S_t,A_t)$ as the error of the update step. In order to have approximations for all state-action pairs it is obviously required that at each state s all possible actions a are chosen at least some times. But choosing an action just based on the so far trained $Q(s,a)$-values, i.e. choosing the action providing the highest $Q(s,a)$-value based on the experiences made so far, does not guarantee that all actions are "tried out". In order to trade off the need for an exhaustive exploration of the state-action space and for a sound exploitation of the current knowledge a so called ε-greedy approach is applied. ε-greedy means that only in $(1-\varepsilon)\cdot 100$ percent of the cases the action is chosen greedily according to $\max_a Q(S_{t+1},a)$ and in the remaining cases randomly,

i.e. presumably nongreedy. The greedy choices are for exploitation and the nongreedy ones for exploration purposes. The percentage of the nongreedy actions gets reduced over time assuming that after a certain period of time the state-action space has been explored sufficiently and the main focus of the training is on the exploitation of the experience made so far.

The Q-learning approach requires that the $Q(s,a)$-values for all state-action pairs (s,a) are approximated and kept stored in a table, called Q-table. Being in state s the corresponding policy can be derived by looking up the action a which yields to the highest $Q(s,a)$-value.

A disadvantage of Q-Learning is that approximating all Q-values requires a sufficiently high number of training updates per state-action-pair which is impossible for problems with a large state and action space. To overcome this drawback DQN uses a neural network as a function approximation method to represent the Q-function generally instead of approximating the Q-values for all possible state-action pairs, as tabular Q-learning does. DQNs were first introduced in Mnih (2015). Each node of the output layer of the neural network is associated to an action a and their activation levels are the estimated $Q(s,a)$-values of corresponding state-action pairs (s,a) while the state s is represented by the network's input. Training the network by backpropagation requires training data. Therefore Lin (1992) introduced a so called replay memory. The replay memory stores at each time step the experience of the RL-agent. Hence, after the RL-agent has chosen at time step t action A_t in state S_t, the subsequent state S_{t+1} and the received reward R_{t+1} are added to the replay memory as tuple $(S_t, A_t, R_{t+1}, S_{t+1})$. In the case of the maximum size of the replay memory is reached this new entry replaces the oldest one in the replay memory. At each time step a mini batch is sampled from the replay memory to perform a weight update accordingly. The DQN approach utilizes two neural networks as parametrized function approximators, the prediction and the target network. These two networks are introduced to counter the known phenomenon that off-policy algorithms diverge when using function approximators, shown by Mnih (2015). The prediction and the target network have an identical network structure and are initialized with equal parameters (weights) θ and θ^-, respectively. The prediction network's θ is updated per time step whereby θ^- is used to provide the target value for the update step. On a regular basis the parameter set θ^- gets replaced by θ. This yields to the update procedure

$$\theta_t \leftarrow \theta_t - \alpha \nabla L_t, \text{ with } L_t = (Q(S_t, A_t; \theta) - (R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta^-)))^2 \quad (1)$$

Further details about RL can be found in Sutton and Barto (2018).

5 SOLUTION APPROACH

5.1 Representation of the State and Action Space

In the presented approach we are using the following state vector describing the gantry robot environment. The robot related state items are given in Table 1.

Table 1: Robot related representation of the state space.

Loader position	current position of the loader
Gripper object nextStep	next process step of the part currently loaded
Gripper2 object nextStep	next process step of the second part currently loaded (double gripper case)

The availability of parts at the input conveyor is given by Table 2 and for each machine M_i, $i = 1, \ldots, m$, the status is given by Table 3.

Table 2: Representation of parts availability at the input conveyor.

Conv_In.fin	indicates whether there is a part available at the input conveyor

Feasible actions are *move to conveyor in/out, move to machine* M_i, $i = 1, \ldots, m$, *gripper 1 picks up a part, gripper 2 picks up a part, gripper 1 drops off a part, gripper 2 drops off a part, wait*.

Table 3: Machine status representation.

Station_i.occ	indicates whether M_i is currently occupied
Station_i.fin	indicates whether there is a finished job at M_i
Station_i.fail	indicates whether M_i is failed or not (only if stations can fail)

Obviously useless actions are marked as infeasible and get blocked. Thus, the RL-agent can not choose to let gripper 1 drop off a part anywhere if gripper 1 has no part nor to choose to pick up a part at a machine which is not loaded. However, in general the maxim was to disable as less state-action pairs as possible and let the RL-agent learn itself what a reasonable action in a certain state is.

5.2 Reward Function Design

Beside the state representation and the definition of the feasible action set, the design of the reward function is crucial for the learning success of the RL-agent. While the definition of the state space and the feasible action set is to some degree predetermined by the production environment, the definition of the reward function is not inevitably induced by the problem. It is rather up to the modeler how to design the rewards.

Learning based on positive rewards receiving for finished goods delivered at the output conveyor turned out to fall short of providing enough feedback to learn a meaningful policy for controlling the robot. Therefore, a more dense reward function has been designed based on providing positive feedback for achieving subgoals, like picking up parts which have completed their service at an operation successfully but are still not finally finished. Initially, rewards have been assigned according to the following rules at time step t:

$$R_{t+1} = \begin{cases} +5, & \text{finished good dropped off at the output conveyor} \\ +2, & \text{part correctly delivered to machine } M_i,\ i = 1,\ldots,m \\ 0, & \text{otherwise} \end{cases}. \tag{2}$$

Using this reward function design enables the RL-agent to learn a policy that maximizes the production throughout and thus meets the objective of the associated MDP. However, there are further requirements to be met by the RL-agent that are not covered by the MDP with the reward function given in (2) and applying the maximum average reward criterion. There is no feedback given to the RL-agent that useless robot moves with empty grippers which do not impact the throughput are not desired. The reward function needs to get enhanced by some non-Markovian components to prevent wasted moves which do not impact throughput. Thus, a negative reward of -1 is given, if two consecutive moves are made, i.e. two moves of the robot without picking up or dropping off a part in between. This allows the RL-agent to earn a higher average reward by choosing a wait operation instead of the useless move operation. On the other side there might be cases where it is more beneficial to move to the machine which completes a part next and to wait there until its completion. Therefore, in addition to the -1 reward a weighting factor $\eta^{t-t'}, 0 < \eta < 1$, for future rewards is introduced and used instead of the discounting factor γ^{k-t-1} in (1). This factor acts as counterbalance to the penalty for waits. If wait operations are unavoidable, the RL-agent gets thereby the feedback to position the gripper as early as possible at the location it is required next and to wait there instead of waiting anywhere and moving later to the next destination. Additionally, the RL-agent receives an reward amounting to $\tau_t/100$ for any load or unload operation, whereby τ_t represents the accumulated throughput of finished goods up to time step t. This makes loads or unloads more profitable than movements and yields to a better training performance.

5.3 Training

The RL-agent is trained over a period of 20,000 episodes. In each episode the RL-agent controls a gantry robot system for a simulated period of 20 minutes. The production gantry robot system is represented by

a simulation model which sends the request for each robot movement to the RL-agent and gets back the RL-agents decision. After 20 minutes of simulated time the system gets reset and initialized randomly to start the training again. The prediction network is trained with probability of $1/3$ after a decision is made within an episode and additionally at the end of each episode. The trained prediction and target network at the end of an episode is used as the initial network for the next episode. To persist successful policies found during training the target network gets stored as policy network if 10 times in a row the throughput achieved during the training episode is higher than the throughput achieved by the previously stored policy. The neural networks consist each of three hidden layers which get activated by a ReLu function. The size of the replay memory is set to 100,000 entries and per training a mini batch of size 64 is sampled out of the replay memory. In order to ensure a sufficiently long exploration period, the decay is set to $\varepsilon^{decay} = 0.99975$.

6 RESULTS

In this section the evaluation of the RL-agent described above is presented. For evaluation purposes the RL-agent has been trained for various settings of a gantry robot system. The general production line setup is as described in section 3. Detailed results are presented for a gantry robot system consisting of three work centers and a single gripper. The work centers have two machines without a buffer each. At each work center a single operation is performed and each part has to go through each operation in a fixed order.

In order to take into account the stochastic nature of the problem, the RL-agent's performance gets evaluated based on 100 validation runs. The simulation model for each run is initialized with a different seed value, and thus the RL-agent sees in each run a different trajectory of the production process it has to deal with. The throughput gained per validation run is compared with either a determined optimal policy or with benchmarks calculated on the heuristic scheduling rule which makes the gripper to take the nearest job first (NJF).

In a first setup the processing times are deterministic and machines do not break down while there is a high probability that a part is available at the input conveyor. In a second setup machines break down randomly. The throughput for both scenarios is presented in Figure 1.

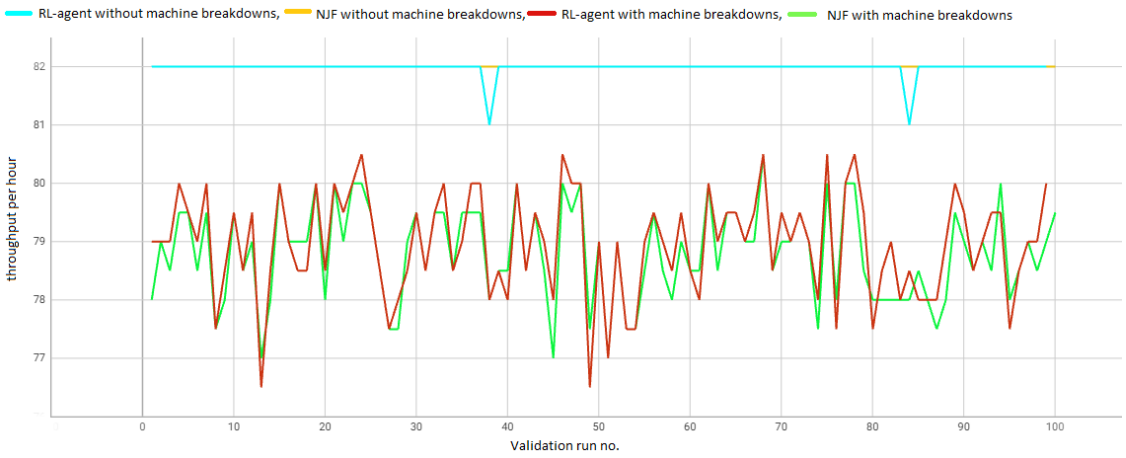

Figure 1: Throughput of an RL-agent controlled gantry robot versus a NJF controlled system serving 3 work centers with and without machine outages.

The upper two lines show the reached throughput in the case of deterministic process times. Therein, the purple one is based on the NJF heuristic and the yellow one is achieved by the RL-agent. Both approaches reach an hourly throughput of 82 which is optimal for this setup. The interarrival times and their variance

is such that they do not really impact the throughput as the constant throughput lines demonstrate. Just in 2 validation scenarios the RL-agent misses the optimal throughput by 1 part per hour. In those two cases the variance of the input stream causes a delayed part arrival at the input conveyor. Since this is a rare event, the RL-agent had no chance to learn an adequate policy for this situation. A training with higher variances of the interarrival times enables the RL-agent to train appropriate strategies, as the results in Figure 2 illustrate. The lower two curves in Figure 1 are showing the throughput achieved in a setup with machine outages. Due to the breakdowns the hourly throughput is obviously less than in the case without outages. Furthermore, it can be seen that in the scenarios with outages the control policy learned by the RL-agent yields to very similar results compared to the NJF heuristic.

To analyze the capability of the RL-agent to cope with a higher variability of the interarrival times at the input conveyor Figure 2 presents results for systems with outages of the input conveyor with 2 different interarrival times having different variance.

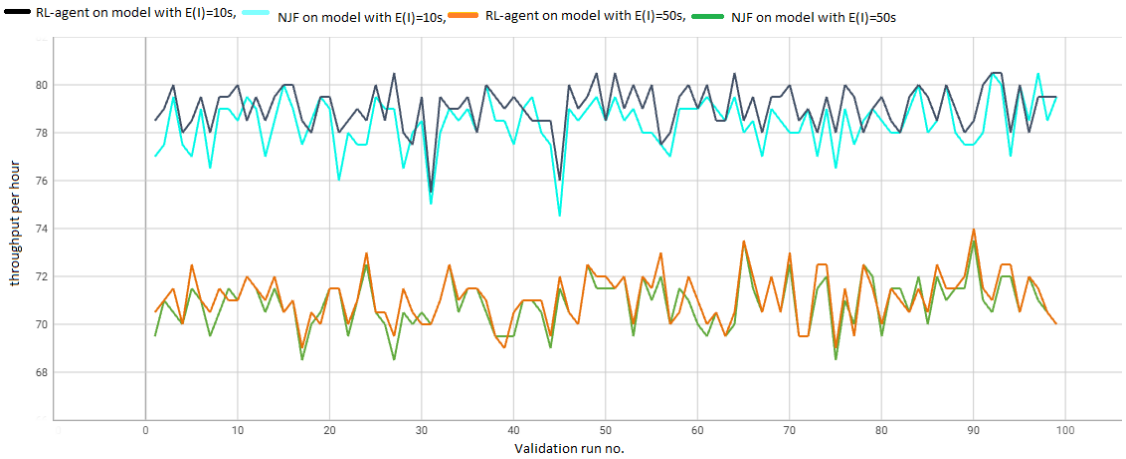

Figure 2: Throughput of an RL-agent controlled gantry robot versus a NJF controlled system serving 3 work centers with input conveyor outages in 2 different settings.

In the setting with higher variance the learned policy of the RL-agent is slightly better than the NJF heuristic. Based on the 100 validation runs the RL-agent achieves a mean throughput of 79.06 parts per hour (the 90% confidence interval is [78.91, 79.21]) while the NJF heuristic reaches 78.40 (90% confidence interval [78.22, 78.58]). There is no overlap in the intervals which supports that the advantage of the RL-agent is small but statistically valid.

Considering a system with both machine outages and input conveyor outages provides similar results (refer to Figure 3). In this case the RL-agent achieves a mean throughput of 73.73 parts per hour with a 90% confidence interval of $[73.52, 73.96]$.

The NJF heuristic achieves in average 73.1 parts per hour with a 90% confidence interval of [72.91, 73.41]. Again the RL-agent reaches a slightly higher mean throughput over 100 validation runs while the confidence intervals are close but do not overlap.

There have been made further evaluation runs with different settings, e.g. an higher amount of work centers, a varying number of machines per work center or using a double gripper. In all theses cases the RL based policy performs comparably well. This supports the properness of the presented RL approach even in more complex settings.

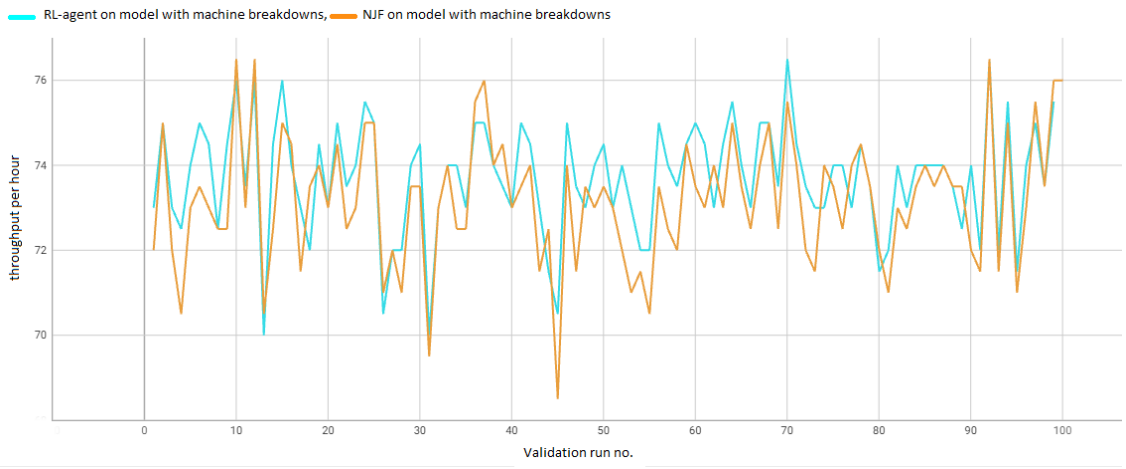

Figure 3: Throughput of an RL-agent controlled gantry robot versus a NJF controlled system serving 3 work centers with input conveyor outages and machine outages.

7 CONCLUSION AND OUTLOOK

In this paper a successful approach for dynamic scheduling of a flow shop with a gantry robot as limiting transportation resource based on simulation and deep reinforcement learning is introduced. In cases the optimal policy can be determined the RL-agent acts optimal concerning the objective of throughput maximization. If the optimal policy can not be identified, the RL approach performs better than the NJF heuristic. In cases with low variability the RL-agent acts slightly better. But the higher the variance of the stochastic interference in the system is the better does the RL-agent compared to the NJF heuristic. Although the throughput advantages of the RL-agent in systems with low complexity are small, they are statistically valid. An additional benefit of the RL-agent is its capability to adjust the policy according to changes on the shop floor autonomously, which is an essential requirement in an Industry 4.0 setting. Eventually, the presented RL based scheduling approach provides several benefits compared to a heuristic policy, like NJF.

Nevertheless, some further research has to be done concerning gantry robot systems with more complex layouts, in particular systems with multiple carriages using single and double grippers or more flexible flow shops producing multiple products with different process flows.

ACKNOWLEDGEMENT

This work (HA project no. 1286/21-187) was financed with funds of LOEWE – Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz, Förderlinie 3: KMU-Verbundvorhaben (State Offensive for the Development of Scientific and Economic Excellence).

REFERENCES

Feldkamp, N., S. Bergmann, and S. Strassburger. 2020. "Simulation-based Deep Reinforcement Learning for Modular Production Systems". In *2018 IEEE International Conference on Industrial Technology (ICIT)*, edited by K.-H. Bea, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, 1596–1607. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Lang, S., N. Lanzerath, F. Behrendt, T. Reggelin, and M. Müller. 2020. "Integration of Deep Reinforcement Learning and Discrete-Event Simulation for Real-Time Scheduling of a Flexible Job Shop Production". In *Proceedings of the Winter Simulation Conference*, edited by K.-H. Bea, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, 3057–3068. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Lin, L.-J. 1992. "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching". *Machine Learning* 8:293–321.

Malus, A., D. Kozjek, and R. Vrabič. 2020. "Real-time Order Dispatching for a Fleet of Autonomous Mobile Robots using Multi-Agent Reinforcement Learning". *CIRP Annals* 69:397–400.

Mnih, V. 2015. "Human-Level Contol Through Deep Reinforcement Learning". *Nature* 518:529–533.

Ou, X., Q. Chang, and N. Chakraborty. 2019. "Simulation Study on Reward Function of Reinforcement Learning in Gantry Work Cell Scheduling". *Journal of Manufacturing Systems* 50:1–8.

Ou, X., Q. Chang, and N. Chakraborty. 2020. "A Method Integrating Q-Learning with Approximate Dynamic Programming for Gantry Work Cell Scheduling". *IEEE Transactions on Automation Science and Engineering* 18:85–93.

Ou, X., Q. Changa, J. Arinez, and J. Zou. 2018. "Gantry Work Cell Scheduling through Reinforcement Learning with Knowledge-guided Reward Setting". *IEEE Access* 6:14699–14709.

Panzer, M., and B. Bender. 2021. "Deep Reinforcement Learning in Production Systems: A Systematic Literature Review". *International Journal of Production Research* 60:1–26.

Popper, J., V. Yfantis, and M. Ruskowski. 2021. "Simultaneous Production and AGV Scheduling using Multi-Agent Deep Reinforcement Learning". *Procedia Cirp* 104:1523–1528.

Puterman, M. L. 2005. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 2nd ed. Hoboken, New Jersey: John Wiley & Sons, Inc.

Renke, L., R. Piplani, and C. Toro. 2021. "A Review of Dynamic Scheduling: Context, Techniques and Prospects". In *Implementing Industry 4.0*, edited by C. Toro, W. Wang, and H. Akhtar. New York: Springer Verlag.

Sutton, R. S., and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*. 2nd ed. London: The MIT Press.

Waschneck, B., A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp, and A. Kyek. 2018. "Optimization of Global Production Scheduling with Deep Reinforcement Learning". *Procedia Cirp* 72:1264–1269.

Watkins, C. J. C. H. 1989. *Learning from Delayed Rewards*. Ph.D. thesis, University of Cambridge, Cambridge, England. https://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf.

Xie, S., T. Zhang, and O. Rose. 2019. "Online Single Machine Scheduling Based on Simulation and Reinforcement Learning". *Simulation in Produktion und Logistik* 1:59–68.

Xu, Z., D. Chang, T. Luo, and Y. Gao. 2022. "Intelligent Scheduling of Double-Deck Traversable Cranes based on Deep Reinforcement Learning". *Engineering Optimization* 1:1–17.

Xue, T., P. Zeng, and H. Yu. 2018. "A Reinforcement Learning Method for Multi-AGV Scheduling in Manufacturing". In *2018 IEEE International Conference on Industrial Technology (ICIT)*, 1557–1561. Lyon, France: Institute of Electrical and Electronics Engineers, Inc.

Zhang, T., S. Xie, and O. Rose. 2017. "Real-Time Job Shop Scheduling based on Simulation and Markov Decision Processes". In *Proceedings of the Winter Simulation Conference*, edited by W. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 3899–3907. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

HORST ZISGEN is a full Professor in the Department of Mathematics and Natural Sciences at the University of Applied Sciences Darmstadt, Germany. His research interests include Monte Carlo methods, queueing theory, stochastic processes and reinforcement learning. He received a M.S. degree and a Ph.D. in mathematics from the Clausthal Technical University. Before joining University of Applied Sciences Darmstadt he worked several years within IBM's research and development organization, where he held several technical leadership positions. His email address is horst.zisgen@h-da.de and his website is https://fbmn.h-da.de/zisgen-horst.

ROBERT MILTENBERGER works as researcher at the Department of Mathematics and Natural Sciences at the University of Applied Sciences Darmstadt, Germany. He is a PhD candidate at the Technical University of Clausthal. His research interests include queueing theory, stochastic processes, biostatistics, and reinforcement learning. He received a M.S. degree in applied mathematics and a M.S. degree in data science at the University of Applied Sciences Darmstadt. His email address is Robert.miltenberger@h-da.de.

MARKUS HOCHHAUS works as a consultant and material flow simulation specialist at SimPlan AG. He received a M.Sc. degree in mathematics in finance and industries at Braunschweig Technical University and joined the SimPlan AG directly after his studies. His email address is markus.Hochhaus@simplan.de.

NIKLAS STÖHR works as a software developer at SimPlan AG, focusing on software development for the interaction, extension, and augmentation of simulation tools. He received an M.Sc. degree in business informatics from the Technical University of Darmstadt and joined SimPlan AG after completing his studies. His email address is niklas.stoehr@simplan.de.