Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 88 (2020) 25–30

www.elsevier.com/locate/procedia

13th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME '19

Autonomous production control for matrix production based on deep Q-learning

Constantin Hofmann[a,*], Carmen Krahe[a], Nicole Stricker[a], Gisela Lanza[a]

[a]Karlsruhe Insitute of Technology, Kaiserstrasse 12, 76131 Karlsruhe, Germany

* Corresponding author. Tel.: +49 152 39502583; fax: +49 721 608 - 45005. *E-mail address:* constantin.hofmann@kit.edu

Abstract

Matrix production refers to a highly flexible production system based on independent production cells that are linked by a flexible transportation system. The production control system decides on the sequence of the production steps of each order and their allocation to specific time slots on the available machines. This paper presents an approach based on deep Q-learning that is able to cope with the dynamic events of the system. The performance of the machine learning-based production control is compared to a static rule-based approach. Additionally, the effects of coordination between the independent agents on throughput time is shown.

© 2020 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer review under the responsibility of the scientific committee of the 13th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 17-19 July 2019, Gulf of Naples, Italy.

Keywords: Flexible manufacturing system (FMS); Scheduling; Adaptive control; Machine Learning; Neural network

1. Introduction

Producing companies operate in a market environment constantly demanding state-of-the-art individualized or regionally adapted products [1]. The results are an increased number of variants in production and takt time losses due to varying cycle times. In the course, matrix production as takt time independent concept has received much attention both in practice as well as in research. Matrix production relies on a free material flow between work stations to exploit operation and routing flexibility [2]. To benefit from these degrees of freedom, an appropriate production control system is required. To satisfy the demand for short calculation times, autonomous multi-agent approaches have proven suitable [3]. Previous simulation studies on the performance of a rule-based multi-agent production control in scenarios of various degrees of operational and routing flexibility have shown that a rule-based approach cannot effectively exploit the given degrees of freedom [4]. Thus, two working hypothesis can be derived:

H1: A production control system based on machine learning adapts better to dynamic system states then the corresponding rule-based approach.

H2: An increase in coordination between the agents reduces the number of blocking states and reduces the throughput time as a result.

To safely evaluate hypothesis H1 and H2, the discrete event simulation model of a matrix production already introduced in [4] is used as substitute for a real production system. Intelligent product agents decide which process should be performed on which machine based on a machine learning approach. It is assumed that all necessary production resources, namely personnel, materials and tools, are constantly available, that machine break-downs only occur in-between production steps and that cycle times are deterministic.

2212-8271 © 2020 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer review under the responsibility of the scientific committee of the 13th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 17-19 July 2019, Gulf of Naples, Italy.
10.1016/j.procir.2020.05.005

2. Related Work

The research conducted in the collaborative research center (SFB) 637 Autonomous Cooperation Logistic Processes explores the performance of various autonomous production control methods for a simplified matrix production layout limited to forward-directed material flow and a fixed process sequence. The simulation studies on both static decision rules [5] and bio-inspired control mechanisms [6] revealed that autonomous decision mechanism show good dynamic behavior especially concerning unforeseen events. However, it could be concluded that no single decision mechanism is always dominant, instead the performance depends on the underlying system dynamics. Building on the findings of SFB 637, Echsler Minguillon and Lanza contributed a simulation study focusing on the choice of the best-performing priority rule depending on the current system state using reinforcement learning [7]. Stricker et al. examine a reinforcement learning approach using Q-learning for adaptive order dispatching in wafer fabrication. In contrast to the previous works of [7], this approach does not rely on priority rules [8]. Similar to their work, Stegherr considers reinforcement learning for order dispatching in a multi-agent setting [9]. The learning agents act simultaneously and implement a coordination mechanism. The research on the application of machine learning approaches, notably reinforcement learning for scheduling, shows that these production control mechanisms are well suited to cope with dynamic events. Mnih et al. introduced a Q-learning algorithm for Atari games capable of mastering advanced policies [10]. Their variant of the Q-learning algorithm relies on a neural network as non-linear function approximation procedure. In contrast to SFB 637, e.g. [6] or [11], in this paper a fully connected matrix and variable process sequences are considered. Therefore the production control system needs to decide on the next process step as well as the machine where the production step should be performed.

3. Methodology

The scheduling problem can be modeled as finite Markov Decision Problem (MDP). Each product agent attempts to maximize its reward *R* by finding the optimal action *a* depending on the state *s* of the environment. The environment responds to the agent's action by changing its state. A typical objective in the context of scheduling is to minimize throughput time. However, throughput time, as a result of a sequence of actions, can only be observed after the last process step. Therefore, a solution approach capable of handling this temporal difference is required. To test hypothesis H1, the dynamics of the system are increased by increasing the demand volatility. Demand is modelled using sinus-shaped curves, comparable to [5].

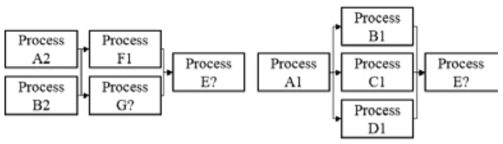

Fig. 1. Precedence graph of both product families

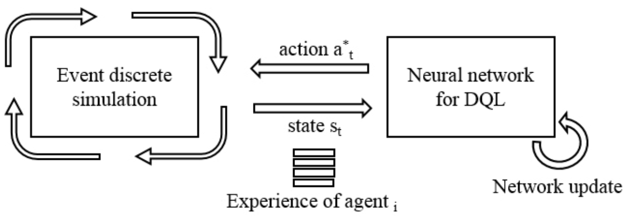

Fig. 2. Interaction of the simulation and the neural network for Q-learning

3.1. Simulation model

The simulation model serves as environment to generate training data for the neural network approximating the Q function and to evaluate the performance of the developed production control, see Fig.2.

The production program consists of two product families with three variants each. Each product variant requires five process steps, see Fig. 1. Orders arrive with an normally distributed inter-arrival time of *N(70,5)*.

The production system is defined by ten workstations of which some are capable of performing multiple processes. The proposed system represents a fully connected matrix production where all workstations are interlinked. Break-downs occur only during idle times, for the mean time between failures (MTBF) and mean time to repair (MTTR) values see Table 1. For each scenario, the results of ten simulation runs, representing a production period of six months, have been averaged.

Table 1. Simulation model

Parameter	Value
Simulation period	6 months
Simulation runs	10
Cycle time [min.]	100
Inter-arrival time [min.]	N(70,5)
MTTF [min.]	5000
MTTR [min.]	N(5,1)
WIP level	10
Set-up same product same process [min.]	1
Set-up different product or process [min.]	5
Set-up different product and process [min.]	10

3.2. Markov decision problem (MDP)

The discussed scheduling problem can be reduced to a MDP represented by state, action and reward tuples.

The state vector incorporates all information regarding the environment. To enable coordination between the independently acting agents, the state vector contains information about other already dispatched product agents as well. At time *t*, the state vector s_t consists of the expected end times for each requested process, the current throughput time and the remaining open processes. To improve coordination, the vector additionally includes entries for all open processes for each dispatched agent. Among these are the expected end times of the current process steps and the expected total throughput times of the dispatched agent. The knowledge about the current throughput times in combination with the remaining process steps allows the agents to estimate whether their own actions potentially have an impact on other agents.

Each agent can choose between a set of actions composed of machine and process combinations. Additionally, an agent can decide to wait. This action is particularly useful if there is no machine offer (e.g. due to machine failure) or to give priority to another agent. The resulting action space is encoded using real numbers. Once an agent has chosen an action, it is executed and thus affects the transition to the subsequent state s_{t+1}. The subsequent state of a transition represents the start state for the next transition.

Since the agents are conditioned to find an optimal behavior regarding throughput time, the rewards for their actions is calculated based on the achieved throughput time. The reward function is composed of two parts: an immediate reward for the selected action and a delayed reward for the achieved throughput time that is determined at the end of production. The objective of the immediate reward is to accelerate the learning process by rewarding possible actions and punishing blocking actions. A blocking state occurs when an agent having several options takes a decision that leaves another agent without options. These decisions are still part of the action space but lead to lower rewards as the reward is based on the overall throughput time of all agents.

Since the reward should be mapped to [-1,1], the throughput times need to be transformed. Taking into account the small rewards for feasible actions of 0.01 and a total of ten decisions, the upper bound of the remaining interval is 0.9. The lower bound of the reward function *R(s,a)* needs to be limited to 0, otherwise the option waiting would represent a favorable behavior to the agent.

Equation 1 shows the calculation for the throughput time (TPT) depended part of the reward function *g(TPT).*

$$g(TPT) = \frac{0.9\,(TPT - TPT_{max})^2}{(TPT_{min} - TPT_{max})^2} \tag{1}$$

To test hypothesis H2, the level of coordination is increased. To enforce coordination between the agents, actions resulting in blocking state for other agents are punished and the reward is given for the sum of all throughput times.

3.3. Q-Learning

A major difficulty when dealing with throughput time as objective is that the effect of a single action cannot be evaluated immediately. Therefore temporal difference (TD) Learning has to be used, e.g. Q-learning. *Bootstrapping* refers to the property of these methods to rely on estimates instead of the actual final outcomes [12]. The behavior strategy of an agent in a MDP is fully defined by a policy π that only depends on the current state s_t. The action-value function (q-function) $q_\pi(s,a)$ describes the expected return starting from state s, taking action a, assuming a certain policy π for all future actions.

$$q_\pi(s,a) = \sum_\pi [\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, A_t = a] \tag{2}$$

Thereby γ describes the discount factor for future rewards R_{t+k+1} with $\gamma \rightarrow 1$ for farsighted and $\gamma \rightarrow 0$ for myopic evaluation. In the evaluated scenario, it is set to 0.99. k represents the number of considered future actions.

One possibility to solve an MDP is to find the optimal q-function q_*, i.e. the q-function of an optimal policy q_{π^*}. An optimal policy satisfies the Bellman optimality equation:

$$\max_\pi q_\pi(s,a) = \sum_\pi \left[R_t + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a\right] \tag{3}$$

Hence, the optimal q-value q_* of a state-action pair describes the highest expected return for a given state s, taking action a and following an optimal policy π_* from there on. To solve the Bellman equation and to find the optimal q-function, iterative methods such as Q-learning can be used. Q-learning is model-free. Therefore the q-value for each state-action-pair is estimated from experience that agents gain by interacting with the environment. No model of the environment has to be known [12]. In the following, the estimated q-function is noted by Q. Since Q-learning is a temporal difference method, the value of a state is approximated by the immediate reward R_t and the estimated future return from state S_{t+1} on. The Q-learning update rule minimizes the difference of the actual and estimated Q-value, called temporal difference error.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha * \left[R_t + \gamma * \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\right] \tag{4}$$

Q-learning learns off-policy. That means that the policy to choose the next action (behavior policy) is different to the policy for which the Q-values are learned for (target policy). To ensure sufficient exploration during the learning phase, the behavior policy is explorative ε-greedy.

In case of a large number of possible state-action pairs, the q-function cannot be expressed explicitly and thus has to be approximated [12]. Neural networks with weights θ are common non-linear approximators for this task. In this paper, Q-learning is used in combination with a fully connected feed forward neural network. Even though multiple product agents act simultaneously in the production system, there is only one neural network with weights θ that is trained to approximate the q-function for all different product types. The input of the network is given by the respective state vector s_t. This information is processed by the neural network and mapped to a corresponding, estimated Q-value of all actions in the given state.

The initial weights θ of the neural network are updated with a variant of stochastic gradient descent (Adam-optimizer). Hereby a sequence of loss functions $L_{\theta i}$ is minimized in each iteration i see equation 5.

$$L_i(\theta_i) = \sum_{s,a\sim\rho(\cdot)} = [(y_i - Q(s,a,\theta_i))^2] \tag{5}$$

$$y_i = \sum_{s'\sim\varepsilon}\left[r + \gamma \max_{a'} Q(s', a', \theta_{i-1}) | s, a\right] \quad (6)$$

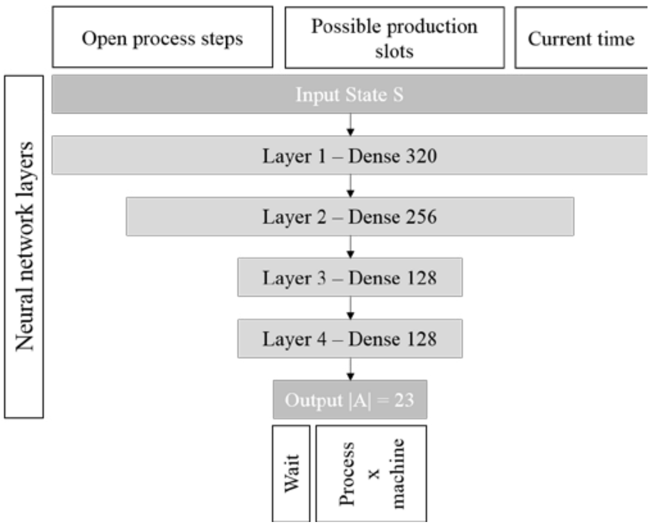

Fig. 3. Neural Network topology

Equation 6 describes the target for iteration *i, ρ(s,a)* the probability distribution over sequences *s* and actions *a* and therefore the behavior of an agent in the environment E.

To soften problems with correlated data and non-stationary distributions an experience replay mechanism is used similar to [10] which randomly samples previous transitions. Moreover, to improve the capacity of generalization, batch learning is applied using mini-batches of data for updates.

3.4. Learning Framework

Fig. 2 illustrates the interaction of the simulation model with the Q-learning based production control system. The simulation model generates the data for the learning process and serves as environment for evaluation. The product agents engage in a sequential decision process until all process steps have been performed.

To learn, the neural network is fed with data sets containing information about the state, action, reward and successor state. Since batch learning with experience replay has been implemented, a randomly chosen batch of 32 past data sets is used to update the neural network, see Fig.2.

To determine a well-suited network topology, the effects of the number of nodes per layer, the drop-out ratio as well as linear, broken-rational and parabolic reward functions have been tested. Fig. 4. illustrates the topology of the fully-connected neural network that has been retained. The best results were obtained without dropout and with a parabolic reward function.

4. Results

In this section, the learning performance is presented followed by a comparison of the rule-based and the deep Q-learning approach in terms of throughput time. Additionally, the results concerning hypothesis H1 and H2 are shown.

4.1. Learning Performance

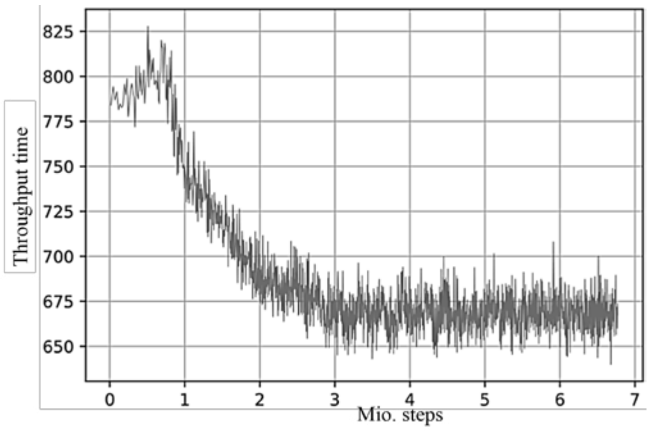

Fig. 4. Throughput evolution during training

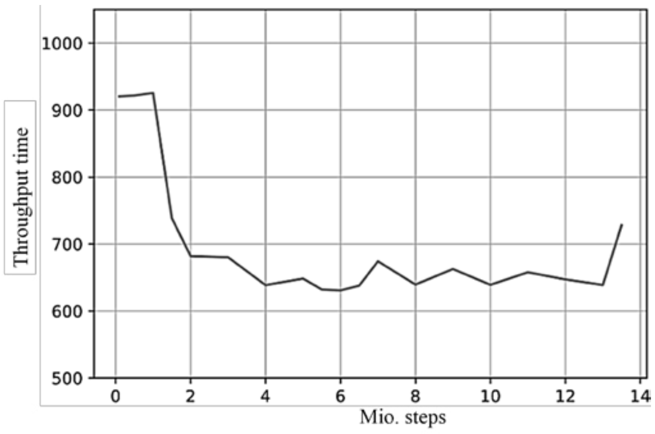

Fig. 5. Throughput time evolution and overfitting

Fig.4. and Fig. 6. visualize the learning curve of the Q-learning framework.

During the first million iterations the agent's behavior is dominated by trial and error leading to poor rewards and high throughput times. During the course of the next two million iterations, the learning process accelerates, the agents take better decisions and therefore improve the throughput time. During the subsequent phase, the performance stabilizes.

The effect of overfitting can be seen in Fig. 6. After 13 million iterations the model has difficulties to generalize and the throughput time increases again.

4.2. Performance compared to rule-based approach

In the following, the performance of the Q-learning production control is compared to a rule-based approach also minimizing throughput time, discussed in [4]. The applied priority rule is similar to the queue length estimator introduced by [13].

Table 2 compares the results in terms of throughput time,

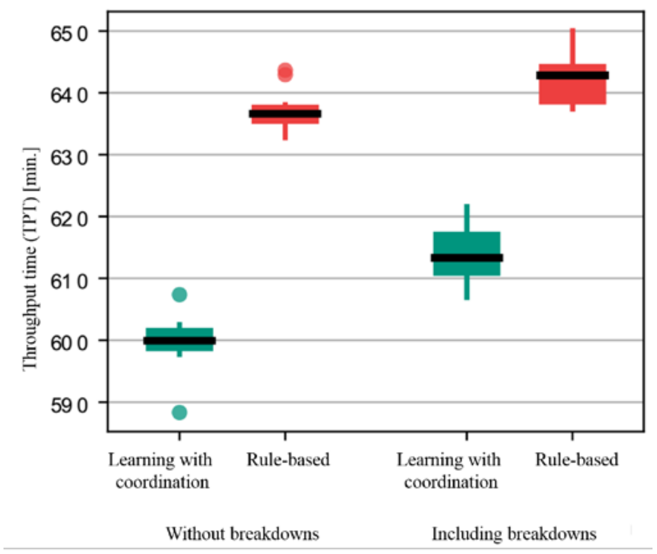

Fig. 6. Boxplot of throughput time for rule-based and deep Q-learning control

output, utilization and tardiness of the two production control approaches. The Q-learning approach outperforms the rule-based approach regarding throughput time by 4.4%. The Q-learning approach prevents actions that lead to blocking states for other orders. These actions only occur to 0.06% while the rule-based approach takes these decisions to 8.4%. Overall, the Q-learning production control takes the same decisions as the rule-based approach in 66%. Utilization is also slightly improved due reduced setup times.

Table 2. Comparison of Q learning to rule-based production control for the scenario with break-downs (value Q-learning, value rule-based)

	Throughput time [min.]	Output [pieces]	Utilization [%]	Relative tardiness [%]
μ	(614.1, 642.6)	(2951.6, 2950.1)	(74.54, 75.38)	(-77.95, -76.27)
σ	(4.76, 4.4)	(6.42, 3.41)	(0.12, 0.08)	(0.34, 0.54)
max	(621.6, 650.1)	(2966, 2956)	(74.8, 75.5)	(-77.5, -75.5)
min	(606.9, 637.4)	(2944, 2946)	(74.4, 75.3)	(-78.5, -76.9)

Fig. 7. shows the throughput time achieved by the rule-based approach and the deep Q-learning control. In the scenario with break-downs the variance of the throughput times increases. It could also be shown that in this case, the Q-learning approach is more sensitive to break-downs, even though the overall performance is still significantly better.

4.3. Hypothesis H1 - machine learning based approach copes better with an increase in dynamic

To model volatile customer demand, the inter-arrival time of the orders has been manipulated. The inter-arrival time of *N(70,5)* is substituted by a sinus-shaped inter-arrival time of 95 minutes with an amplitude of 30 minutes and a wave length of one month.

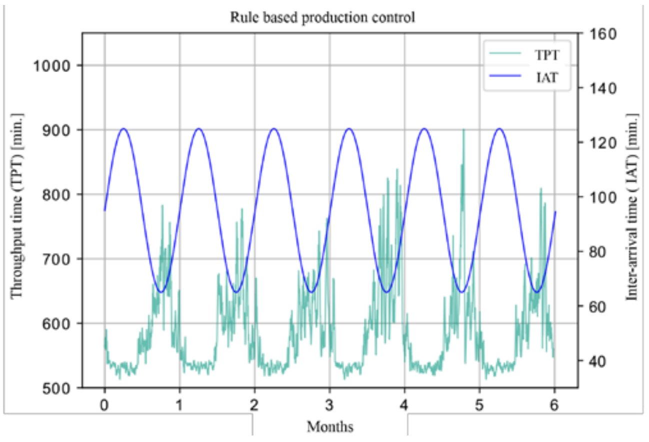

Fig.7. Evolution of inter-arrival time (IAT) and throughput time (TPT) for rule-based control

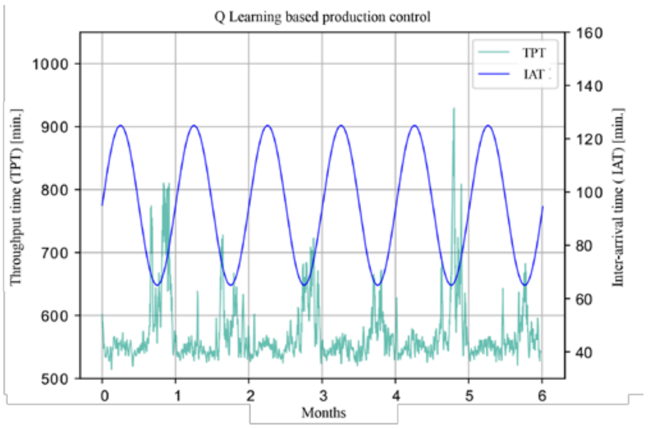

Fig. 8. Evolution of inter-arrival time (IAT) and throughput time (TPT) for deep Q-learning control

In phases of high customer demand, modelled by a short inter-arrival time, the level of work in progress rises in the production. Alternative product routings and redundant machines can be used to soften the effects of these customer demand peaks. The deep Q-learning approach has been trained using the inter-arrival time pattern of *N(70,5)* and is applied in the scenario of increased dynamics.

The rule-based production control shows proportional peaks in the throughput time. In contrast, the machine learning based production control approach is able to soften the impact of the varying inter-arrival times better than the rule-based approach, see Fig.7. and Fig. 9. The average throughput time is also about 3% lower. Due to the longer throughput time of the rule-based production control, the valleys between the peaks are narrower.

4.4. Hypothesis H2 - Increased coordination improves throughput time

To test whether coordination improves the throughput time, hypothesis H2, two neural networks have been designed and

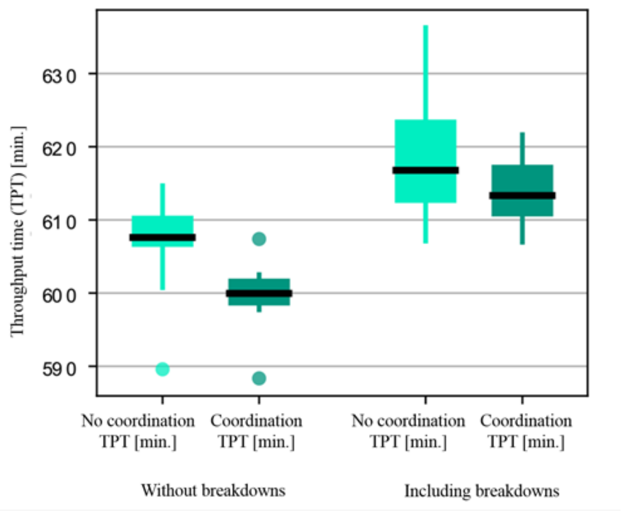

Fig. 9. Boxplot for throughput time with and without coordination

trained. The state vector for the approach without coordination does not contain information about other dispatched agents, as

described in section 3.2. The reward is calculated solely based on the own throughput time. Fig.9. summarizes the results achieved with and without break-downs.

In both scenarios, it can be seen that coordination lowers the throughput time and improves the variance. In the scenario with breakdowns, the coordination between the agents also lowers the occurrence of blocking states that lead to long throughput time losses for the blocked agent.

In case the throughput time of the deciding agent is already much higher than the average overall throughput time, the choice of the self-centered action causing a blocking state for another agent can be beneficial.

5. Conclusion

In this paper, two production control approaches minimizing throughput time have been compared using a simulation model of a fully-connected, forward directed matrix production with machine breakdowns. The comparison of a rule-based approach using a priority rule similar to the queue length estimator and the deep Q-learning approach revealed that the deep Q-learning approach outperforms the rule-based control. It could be shown that the choices of the deep Q-learning control mechanism differed in about 34% of the cases from the priority rule. Moreover, it could be demonstrated that the performance increases further if information about other dispatched orders is fed to the neural network. The resulting coordination reduces decisions leading to blocking states for other orders and improves the overall throughput time. An increase in dynamic in terms of volatile customer demand volume has been modeled using sinus-shaped demand patterns. The production system relying on the proposed deep Q-learning model was able to soften the effect of the changing production volume better preventing long waiting times.

References

[1] Hochdörffer J, Berndt CV, Lanza G. Resource-based Reconfiguration of Manufacturing Networks Using a Product-to-plant Allocation Methodology, in: Proc. 6th Int. Conf. Compet. Manuf, 2016: pp. 511–516.
[2] Schönemann M, Herrmann C, Greschke P, Thiede S. Simulation of matrix-structured manufacturing systems, J. Manuf. Syst. 37 (2015) 104–112.
[3] Scholz-Reiter B, Höhns H. Selbststeuerung logistischer Prozesse mit Agentensystemen, in: Produktionsplan. Und-Steuerung, Springer, 2006: pp. 745–780.
[4] Hofmann C, Brakemeier N, Krahe C, Stricker N, Lanza G. The Impact of Routing and Operation Flexibility on the Performance of Matrix Production Compared to a Production Line, in: R. Schmitt, G. Schuh (Eds.), Adv. Prod. Res., Springer International Publishing, Cham, 2019: pp. 155–165.
[5] Scholz-Reiter B, Freitag M, de Beer C, Jagalski T. Modelling Dynamics of Autonomous Logistic Processes: Discrete-event versus Continuous Approaches, Ann. CIRP. 55 (2005) 413–417.
[6] Scholz-Reiter B, Beer CD, Freitag M, Jagalski T. Bio-inspired and pheromone-based shop-floor control, Int. J. Comput. Integr. Manuf. 21 (2008) 201–205. doi:10.1080/09511920701607840.
[7] Minguillon FE, Lanza G. Maschinelles Lernen in der PPS, Wt Werkstatttechnik Online. 107 (2017) 630–634.
[8] Stricker N, Kuhnle A, Sturm R, Friess S. Reinforcement learning for adaptive order dispatching in the semiconductor industry, CIRP Ann. (2018).
[9] Stegherr F. Reinforcement-Learning zur dispositiven Auftragssteuerung in der Variantenreihenproduktion, Utz, Wiss., 2000. https://books.google.de/books?id=quNCJqWyPeUC.
[10] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M. Playing Atari with Deep Reinforcement Learning, (2013). http://arxiv.org/abs/1312.5602.
[11] Scholz-Reiter B, Rekersbrink H, Görges M. Dynamic Flexible Flow Shop Problems - Scheduling Heuristics vs. Autonomous Control, CIRP Ann. - Manuf. Technol. 59 (2010) 465–468.
[12] Sutton RS, Barto AG. Reinforcement Learning: An Introduction, MIT Press, 1998. http://www.cs.ualberta.ca/~sutton/book/the-book.html.
[13] Scholz-Reiter B, Freitag M, De Beer C, Jagalski T. Modelling and Analysis of Autonomous Shop Floor Control, in: 2005.