Production Engineering (2020) 14:319–328
https://doi.org/10.1007/s11740-020-00967-8

PRODUCTION MANAGEMENT

Reinforcement learning for an intelligent and autonomous production control of complex job-shops under time constraints

Thomas Altenmüller[1] · Tillmann Stüker[1] · Bernd Waschneck[1] · Andreas Kuhnle[2] · Gisela Lanza[2]

Received: 23 March 2020 / Accepted: 28 May 2020 / Published online: 7 June 2020

Abstract
Reinforcement learning (RL) offers promising opportunities to handle the ever-increasing complexity in managing modern production systems. We apply a Q-learning algorithm in combination with a process-based discrete-event simulation in order to train a self-learning, intelligent, and autonomous agent for the decision problem of order dispatching in a complex job shop with strict time constraints. For the first time, we combine RL in production control with strict time constraints. The simulation represents the characteristics of complex job shops typically found in semiconductor manufacturing. A real-world use case from a wafer fab is addressed with a developed and implemented framework. The performance of an RL approach and benchmark heuristics are compared. It is shown that RL can be successfully applied to manage order dispatching in a complex environment including time constraints. An RL-agent with a gain function rewarding the selection of the least critical order with respect to time-constraints beats heuristic rules strictly by picking the most critical lot first. Hence, this work demonstrates that a self-learning agent can successfully manage time constraints with the agent performing better than the traditional benchmark, a time-constraint heuristic combining due date deviations and a classical first-in-first-out approach.

Keywords Complex job shop · Production planning and control · Reinforcement learning · Time constraints

1 Introduction

Manufacturing companies are subjected to constant transformations of their internal processes and their environment [24]. Globalization and competition through emerging industries in developing nations are already well-known [11]. Especially, with the ongoing digitalization accelerating markets are becoming unpredictable, and companies are in need to react quickly and decisively [1]. Optimally exploiting the operational abilities and resources is of utmost importance under these conditions.

In the semiconductor industry, complex job shops are facing similar challenges [23]. Job shops are widely evaluated in other industries, since the requirements for flexible and rapidly changeable production systems are increasing [4]. Complexity, as well as opportunities and the pressure to make use of the latter, are therefore essential endeavours to ensure a competitive position [23].

The emergence of artificial intelligence methods such as deep learning, reinforcement learning (RL) and other machine learning approaches bear the potential to encounter these challenges with new quantitative methods. The methods are backed by significantly decreased computational times, the development of easy to use open source libraries and achievements like beating the human experts in strategic games [19].

The objective of this paper is the development and implementation of an autonomous and self-learning algorithm addressing order dispatching with strict time constraints in complex job shops. This is the first publication which utilizes RL to manage an environment with strict time constraints.

The research is structured as follows: Sect. 2 introduces the fundamentals of time constrained production planning and control, the essential characteristics of job shops in the semiconductor industry, and the basics of RL. Section 3 describes the method and RL-algorithm that has been

✉ Thomas Altenmüller
thomas.altenmueller@infineon.com

Andreas Kuhnle
andreas.kuhnle@kit.edu

[1] Infineon Technologies AG, Am Campeon 1-12, 85579 Neubiberg, Germany

[2] wbk Institute of Production Science, KIT, Kaiserstr. 12, 76131 Karlsruhe, Germany

developed in the present work, leading to the computational result in Sect. 4.

2 Fundamentals and literature review

2.1 Production planning and control

Production planning and control (PPC) focuses on organizing and optimizing the internal processes within a manufacturing system [3]. Production planning predefines the production program, including the production process setup information. Production control takes these predefined inputs to schedule all processes necessary to fulfill the production plan and, thereby, optimally utilize the available production factors given changing circumstances such as shortages of material, machines or workforce [12, 18]. The order release is the interface between production planning and production control.

Order dispatching, which is the focus of this work, is one PPC task and considers the assignment of orders to the next processing machine given a set of available machines and orders. Other tasks are, for instance, order sequencing which defines the sequences in which individual machines process orders [5, 17].

Optimization and decision support methods that are used for PPC are often categorized into three classes [14]: first, there is mathematical optimization, which allows finding the optimal solution but requires high computational effort. Second, heuristics are in wide-spread use to overcome the computational drawbacks in large real-world problems. They achieve acceptable results in much less time but fall short of the optimal solution in most cases. Learning-based techniques, the third class of methods, are considered in the present work.

2.2 Complex job shops in semiconductor manufacturing

Certain features are typical for the wafer fabrication [14]. Re-entrant flows let wafers run through the same machine group more than once. Some machine types are relatively unreliable by nature of the physical manufacturing process and need frequent maintenance measures. Within the same process chain, there are not only serial but also batch processes. As a consequence, massive queues are caused by the succession of parallel batching tools with serial tools. Moreover, dynamic bottlenecks occur. A multifaceted product portfolio and changing product mix ensure constantly varying boundary conditions. For some tools, setup times and multiple processing times are further challenges. Finally, highly competitive due dates prevail.

All in all, wafer fabs belong to the category of complex job shops as a distinct job shop type with the features described above [23].

2.3 Handling of time constraints in PPC

Time constraints, also called time-coupling constraints, describe the maximum time that is allowed to pass between the end of one process step and the start of another one (see [14]). Time constraints are a significant challenge in modern manufacturing systems, but in particular present in the semiconductor industry [12]. Hence, time constraints are an essential part of realistic manufacturing process modelling. Some research focused on the incorporation of time constraints in PPC methods.

Klemmt and Mönch model time constraints and present an approach for solving a scheduling problem with time constraints [6]. Besides customer due dates, time constraints can also exist between two or more consecutive process steps. The authors classify time constraints in five different types, depending on the consecutiveness, adjacency, and overlapping of time constraints. Overlapping time constraints occur in complex job shops, which makes the computation of an optimal solution infeasible for real-world problems.

Sun et al. introduce delay time constraints in the application domain of semiconductor manufacturing [2]. Delay time constraints refer to a period within a consecutive process needs to be started based on a start process step. The authors present objective function and constraint formulations that are required to consider such time constraints in existing manufacturing control systems. The approach is based on mixed-integer programming and constrained programming models. However, experimental results from an application are not presented.

Knopp addresses scheduling problems in complex semiconductor job shops with heuristic and metaheuristic approaches [7]. The focus of his work is on batching processes and time constraints. The time constraints are considered as maximum time lag constraints and implemented in the model as soft constraints with violation costs attached. Reworkable and non-reworkable time lags are distinguished. Reworkable time lags lead to a lot requiring rework to be done in case of violation. This results in increased cycle time for the lot itself and additional machine capacity, which raises its relevance for the overall fab performance. Non-reworkable time lags describe a point in time at which the risk of the wafers to become defective starts to increase for the violation.

2.4 Basics of RL

Machine learning techniques cover a variety of algorithms that can process large amounts of data and identify patterns that can be transferred to new situations [16]. They are suitable to meet the demand for dynamic, real-time production control applications [21].

In RL, being one type of machine learning algorithms, agents constantly adapt and learn strategies in a known or unknown environment through feedback received: the agent perceives the state of the environment $s_t \in S$ from a finite set of possible states at each time t. On this basis, it selects an action $a_t \in A$ from the set of possible actions. The environment responds with the resulting state s_{t+1} and a feedback, the reward $r_t \in \mathbb{R}$, before the next iteration starts [21]. This iteration can be described by a Markov decision process, i.e.s_{t+1} depends only on s_t, not on previous states. The agent's goal is to optimize its strategy to maximize the cumulative reward. The strategy is represented by a probability distribution for choosing a specific action in a given state.

For a detailed overview of the latest RL research in the domain of production planning and control, we refer to the work of [9, 10, 20, 22].

3 Use case description and RL modelling

This section introduces the use case taken from a real-world wafer-fab. The RL modelling with respect to the state and action space, reward function, optimization algorithm, as well as the time constraint consideration are outlined. Overall, the following questions are to be answered:

- Is the proposed single-agent based RL approach able to learn and, therefore, improve over time its ability in optimizing the adherence to a time constrained schedule?
- Can the proposed approach perform better than established competitive heuristics?

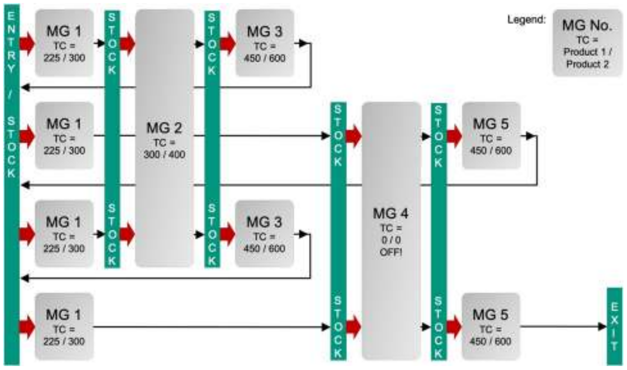

Fig. 1 Job shop layout of the wafer-fab use case

Table 1 Parametrization of the simulation in the wafer-fab use case

System parameters	Wafer-fab use case
Number of machine groups	5
Number of machines per machine group	4-1-2-1-2
Number of product types	2
Probability of product type being generated	50%, 50%
Size of buffers before machine groups	20 order slots
WIP restriction	80
Order release time interval	70
Recipe (machine group sequence)	(1-2-3-1-4-5) x2
Average RPT (per machine)	Between 20 and 80
Machine failures (per machine)	MTBF between 1700 and 4000, MTTR between 15 and 60
Setup times (per machine)	Between 0 and 15
Target flow factor (per product type)	12, 8
Time-coupling constraint (per product type)	Between 225 and 600

3.1 Description of the wafer-fab use case

The considered use case represents a production process with ten specialized machines sub-organized in five machine groups. Every machine group, also called work center, has a buffer stock with a fixed number of twenty buffer slots. Challenges arise from many time-coupled process steps ranging over multiple work centers. The high share of time coupling constraints for around 30% of the operations, increases the complexity of the order process flow and the internal logistics. Moreover, machines also suffer from reduced availability. Furthermore, order flow is re-entrant in earlier visited machine groups. Finally, machines require a product specific set-up. Hence, the use case incorporates the following complexity drivers introduced in Sect. 1: non-linear process flows, re-entrance flows, sequence-dependent setup times, and time constraints. This renders an appropriate scenario for the application of RL to analyse the influence of time constraints within a complex environment.

Here, two product variants are modelled with an equal product mix proportion but different process times and lengths of time constraint intervals (Table 1). The first product variant is faster but has more stringent time constraints than the second. Both product variants can be processed on all machines and have the same basic recipes and, thus, identical process chains. So, on average, orders of both product variants basically have to wait for the same time depending on what orders are waiting in front of the machine. However, the agent's decisions could favor one product variant over the other (Fig. 1).

3.2 State space modelling

The state space depicts the information given to the agent as basis for decision-making on which action to select.

The state space defined in this work is quite rich and has 210 entries consisting of the following elements, which are described in the remainder of this section:

- Current machine where the action is asked for (10 entries, binary, one-hot coding for 10 work groups)
- Loading status of all machines (10 entries, binary, idle or busy)
- Product setup per machine (50 entries, binary, one-hot coding for 5 product variants and ten machines)
- Product variant in the current machine's buffer slots (100 entries, binary, one-hot coding for up to twenty buffers and 5 product variants)
- Order status per buffer slot (20 entries, real-value, 20 buffer slots)
- Full or empty status per buffer slot (20 entries, real-value, 20 buffer slots)

The state values are mostly binary and ranging from 0 to 1. In order to normalize categorical data, one-hot encoding is applied. The state values are determined based on the state of the environment at the point of decision-making.

In this use case, the order status observed by the RL-agent is given by an order's rank of "urgency ratios" (explained in the next section, taking a value between − 1 and 1). The three orders with the highest urgency are given the rank values 3, 2, and 1. All other orders are labelled with a 0. Blank slots in the buffer of a machine group are indicated by a value of − 5. Thus, jobs of different product variants with different time constraints and different processing times are made comparable. In addition to the order's urgency ratio, the benchmark heuristics use a due date deviation based on planned cycle time of a job as second prioritization rule, determined by the raw process time multiplied with a target flow factor. The raw process times are given by the technological process parameters and the product variant. The target flow factors is a predetermined input values, that considers strategic objectives such as customer lead times.

It is important to note that the observation of the orders waiting at a work center is shuffled with respect to buffer slot position before communicating the order status per buffer slot to the agent. This procedure leads to a random allocation of slots for every order and every empty slot. This is a crucial element in preventing the agent from learning a biased policy to pick slot-oriented and not order-specific.

Further, the setup status type of every tool is included, in order to enable the agent to optimize setup sequences of the machines. Combining the machines' setups with respect to the product variants should result in a better setup and load management.

3.3 Action space modelling

Whenever the agent is requested, it chooses one of 21 options:

- Selecting an order from a buffer slot (20 actions)
- Idle and choose no order (1 action)

Each of the twenty slots of a buffer is either occupied or empty. Only selecting an occupied slot is considered a valid action. Invalid actions are not executed (i.e. the simulation state is not changed) but used as feedback for learning: the agent is penalized with a negative reward and requested to select another action.

When experimenting with global rewards (see Sect. 3.5) the agent is given the option to increase or decrease the rate of new order starts to allow control and optimization of the WIP level in the line. This leads to two more options for action:

- Increase loading by reducing current inter arrival times at start buffer
- Decrease loading by increasing current inter arrival times at start buffer

3.4 Modelling extensions for time constraints

Time-coupling constraints are measured for every order individually. In the present work, time constraints are only considered between two consecutive process steps. The length of the time constraint is product variant-specific and depends on the next process step. If a constraint is violated nothing directly happens to the affected order, however, it influences the agent's reward. The number of time constraint violations is a key indicator for the performance of a learned strategy or heuristics.

The state space observed by the agent reflects the state with respect to time constraint-specific urgency ratio *UR*. The urgency ratio is also considered for the time constraint-specific reward function and for the benchmark heuristics. The urgency ratio *UR* indicates how well an order performs according to its time constraint (T_C) and is calculated via:

$$UR := \max\left\{\frac{T_C - (t - t_{\text{finished last step}})}{T_C}, -1\right\} \tag{1}$$

The time t refers to the current simulation time and $t_{\text{finished last step}}$ is the point in time when the last process step was completed. Hence, no distinction is made for orders that are delayed at least 100% of their time constraint. If an order does not have a time constraint for the currently next process step, it is set to $UR = 1$ per default. Therefore, the following cases can occur:

- $UR > 0$: The order is considered to be on time.
- $UR = 0$: If the order is not selected in this time step, it will violate its time constraint.
- $UR < 0$: The order has violated its time constraint.

We use as metrics to evaluate the performance of the system with respect to time constraint violation avoidance the total number of time constraint violation events by one job at any machine stock during the production process. Additionally, a cumulative variant of the urgency ratio best described as cumulated delay $D_{cum}(p)$ is considered:

$$D_{cum}(p) \; := \left\langle \sum_{j \in \text{production steps}} \left(1 - UR_j\right) \right\rangle_{\text{finished lots}(p)} \tag{2}$$

which is the average over all lots of product type p of the cumulated delays measured at the exit buffer.

3.5 Reward function modelling

The reward function r is the key element that directs the agents behavior and determines the optimal policy. As introduced above, invalid actions, i.e. picking an empty slot, are rewarded with $r = -1$ and a reward of $r = 0$ is given for the idle-action, i.e. doing nothing. The exponential time constraint reward function used is modelled as a reward continuously following the UR ratio:

$$r_{\text{local}} = 2 \cdot 5^{(UR+1)/2} \tag{3}$$

local reward r_{local} is different from the global reward r_{global}, which will be introduced afterwards. As the reward is just based on the status at the specific machine for which the action had been selected, it is called “local”. The local reward is bound between 2 and 10. Contrary to the approach used by heuristics, the agent is rewarded for picking orders with a low urgency (i.e. $UR \geq 0$), a reward only sustainable achievable if the agent finds a strategy that removes unfavorable, i.e. high urgency orders, from the system.

In addition to the local reward, two global rewards are investigated. The first directly awards a low number of average time constraint violations measured at the end of the production line:

$$r_{\text{global}} = 10 \cdot \left\langle \exp\left(-\frac{V_p}{3}\right)\right\rangle_p \tag{4}$$

Here, V_p is the average number of time constraint violations over the last 50 orders for product type p . The notation $\langle\,\rangle_p$ refers to the average over all products p in the exit buffer. The exponential function dampens the reward for increasing V_p.

Furthermore, experiments showed a tendency of the agent to reduce the order release and reduce new order starts. This leads to an overall less loaded system where time constraints are easier to handle. Therefore, two additional actions are included to increase and decrease order starts and combine this option with a second global reward component r_{WIP} awarding adherence to a predefined WIP level over all buffers i:

$$r_{\text{WIP}} = \prod_{i \in \text{buffers}} 1 + \max\left\{\beta \cdot (\text{WIP}_i - \text{WIP}_i^{\text{target}}), 0\right\} \tag{5}$$

$\text{WIP}_i^{\text{target}}$ refers to the target WIP level in buffer i and β represents a balancing parameter which controls exceeding the target WIP level. Hence, β controls the trade-off between a production line prioritizing high WIP levels at the cost of cycle time and time constraint violations, on the one hand, and a line favoring cycle time and time constraint violation avoidance over high WIP levels, on the other hand.

The effective total reward transits over the course of the exploration phase from r_{local} to r_{global} to smoothly aim at global learning:

$$r_{\text{total}} = r_{\text{WIP}} \cdot r_{\text{local}} \longrightarrow r_{\text{WIP}} \cdot r_{\text{global}} \tag{6}$$

3.6 RL algorithm and simulation of environment

The DQN-agent [13] provided by the library Keras-RL [15] is the RL-agent used in this research. Hereinafter, a short overview of some of the most important settings and hyperparameters is given (see Table 2), based on a review of other DQN applications (e.g. [22]). A sequential replay memory is used with a limit of one million experiences. The policy deployment uses linear annealing with an ϵ starting at 100%

Table 2 The settings for the DQN-agent used in this work for the wafer-fab use case scenario

Parameter	Default value
Number of decisions by the agent	1.2 million steps
Learning until	600,000 steps
Inner policy	Epsilon greedy Q policy
Policy	Linear annealed policy
Exploration/exploitation	ϵ = 100% → 1% / 1%
Target model update	Every 10,000 steps
Metric	Mean absolute error (MAE)
Warm-up phase	10,000 steps
Size of sequential memory	1,000,000
Discount factor	$\gamma = 0.9$
Optimizer	Adaptive moment estimation (Adam)
Learning rate	$\alpha = 0.00025$

and linearly decreasing to 1% over time. Depending on ϵ the policy chooses either to "explore" by taking a random action with probability ϵ or to "exploit" by going for the best learned action with probability $1-\epsilon$. The agent warms up for the first 10,000 steps, as the simulation first needs to fill up and pass the transient phase. The target network is updated every 10,000 steps. The discounting factor is set to $\gamma = 0.9$ and mean absolute error is used as metric. The adaptive moment estimation optimizer (Adam) is used with a learning rate of $\alpha = 0.00025$.

3.7 Benchmark heuristics

The performance of the RL approach is compared to a benchmark, state-of-the-art time constraint control heuristics. A combined time constraint heuristic (short, *TC*) chooses the order with the lowest urgency ratio min *UR*, if min $UR \leq 0.5$ and, otherwise, the order with the largest due date deviation. The idea behind the combined rule is to model a more realistic approach with time constraints only being considered if an order comes close to violating its time constraint.

Additionally, a first-in-first-out (FIFO) heuristics is included as benchmark heuristic that is well-established in production control application but not directly considering time constraints and due dates.

4 Results

The computational results are based on simulation runs for the above presented scenario. In every experiment, 1.2 million simulation steps are performed, whereby a simulation step is defined as an agent's action selection and execution. The exploration value ϵ, which refers to the share of explorative actions, is initially set to 1 and decreases linearly to 0.01 over the first 600,000 steps. Hence, for the last 600,000 steps the agent just exploits the best learned action so far. However, it is important to note that learning will not stop after the exploitation phase because random machine failures and order sequence will introduce randomness into the learning process.

4.1 RL-agent and heuristic performance analysis—time constraint violations

The benchmark heuristics results are shown in Fig. 2 in the first two charts to the left, the two charts to the right show the results of the RL-agent for local and global rewards. The charts show the frequency of time constraint violations V_p per product type after finishing processing. The entire data set of 1.2 million simulation steps is split into six equal-sized phases with a length of 200,000 simulation steps that are shown separately in order to see the learning process for the learning-based agents.

First, one can see that the values spread significantly, especially for the rule-based heuristics on the left due to the system immanent stochastic processes. As expected, the heuristic performance does not change over the course of the phases, as no learning process applies. More interestingly, the TC benchmark achieves results which are similar to FIFO (see Sect. 3.7). For both products, the two heuristics achieve an average number and standard deviation of time constraint violations of 3.7 ± 1.6 for product 0 and 3.1 ± 1.5 for product 1 (FIFO), respectively 3.7 ± 1.6 and 3.7 ± 1.6

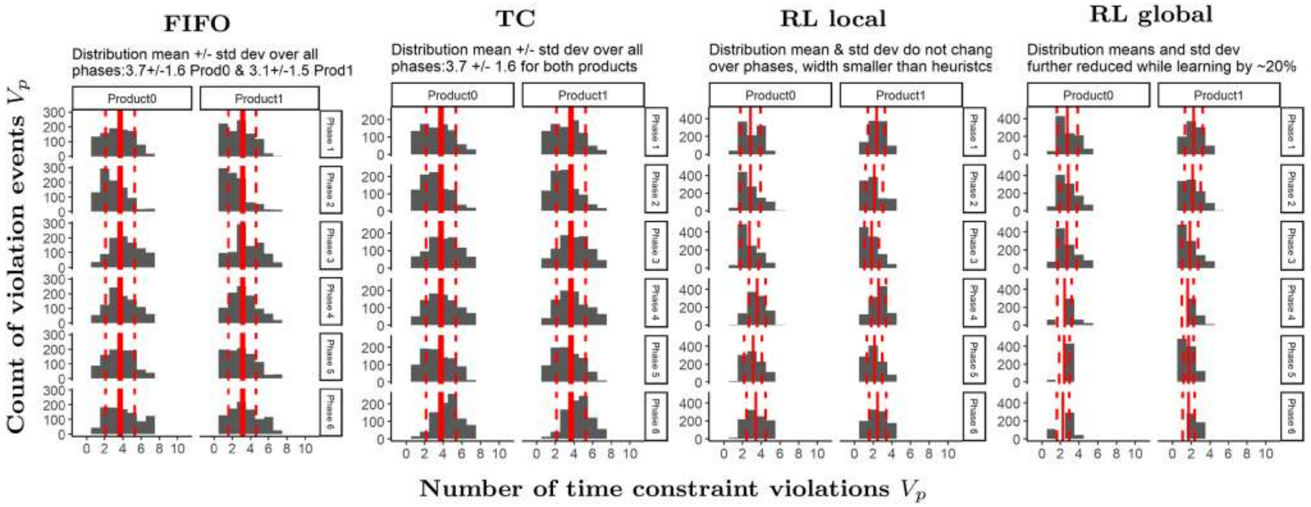

Fig. 2 Count of average number of time constraint violation events for both products measured at the exit buffer. The entire learning process is separated into six phase with equal length. Vertical red lines correspond to the mean (solid) and mean ± one standard deviation (dashed)

(TC). Focusing only on time constraints, one would expect the TC heuristic to perform better in avoiding violations.

On the one hand, the FIFO heuristic does not specifically consider time constraints, it is following waiting times. Hence, it is profiting from the correlation between the waiting time of an order in the current buffer and the probability of an order violating its time constraint. As shown in Table 1, orders of product type 0 have 25% less time to finish than orders of type 1. The heterogeneous product types are not addressed accordingly by FIFO and, hence, the violations are about 20% more frequent for product type 0. On the other hand, TC levels the number of violations for both product types better than FIFO. As it always picks the order with the lowest urgency ratio, neither product type is preferred.

Next, the two charts to the right in Fig. 2 show the result of the RL-agents (DQN-agent). The first represents the agent just with the local reward function r_{local} and the second the reward of the second agent transitioning between Phase 4 and Phase 6 from a purely local reward to an entirely global reward r_{total}. After Phase 3, no-deliberate random actions due to ϵ are chosen and the agent decisions are taken as is.

Overall, one can say that the developed reward functions succeed in minimizing the violations against time constraints and clearly outperform both benchmark heuristics. Moreover, the trend over the six phases reveals that the agent is successfully minimizing the number of violations and the average sum of deviations is reduced. For both products, a local reward trained agent achieves an average number of time constraint violations of 3.1 ± 1.0 and 2.1 ± 0.8, respectively, and a global reward trained agent even 2.4 ± 0.5 and 1.7 ± 0.5, respectively. Note, that in Eq. (4) V_p is explicitly entering the global reward function, while in Eq. (3) the urgency ratio UR is taken into account for the local reward.

Figure 3 shows the same simulation experiments but with respect to the summed delay $D_{cum}(p)$ per product type. This parameter measures the cumulative UR-performance and not only the number of TC constraint violation events. Distribution averages and standard deviations for product types 0/1 are: for FIFO $7.9 \pm 2.6/6.6 \pm 2.3$, TC 8.0 ± 2.6 $/8.0 \pm 2.6$, RL local $6.8 \pm 1.8/5.2 \pm 1.6$, and RL global $5.6 \pm 0.9/4.1 \pm 0.9$, respectively.

The RL-agent learns based on a cumulative reward that confronts it with the consequences of past actions, especially relevant in a re-entrant flow scenario like it is used here. This fact makes the difference between the agent and heuristic approaches, as the latter are limited to a rule-based procedure with local information that is included in the decision rule. In contrast to this, the agent captures during the training phase the relationship between the product type, the order's urgency ratio, and the combined information of product types and urgency ratios of the other orders in the buffer. Moreover, in the case of the local reward, the agent is rewarded for picking orders with a low criticality (i.e., high urgency ratio), which leads to a strategy that ultimately avoids the occurrence of critical orders. In the case of the global reward, the agent is awarded higher if the final count of time constraint violations after processing is as low as feasible. All this results in agents that are not simply picking the order with the lowest urgency ratio, but instead the order most endangered to a negative urgency ratio. In other words, it is able to create a farsighted control strategy which is more successful than the myopic and greedy approach used by heuristics.

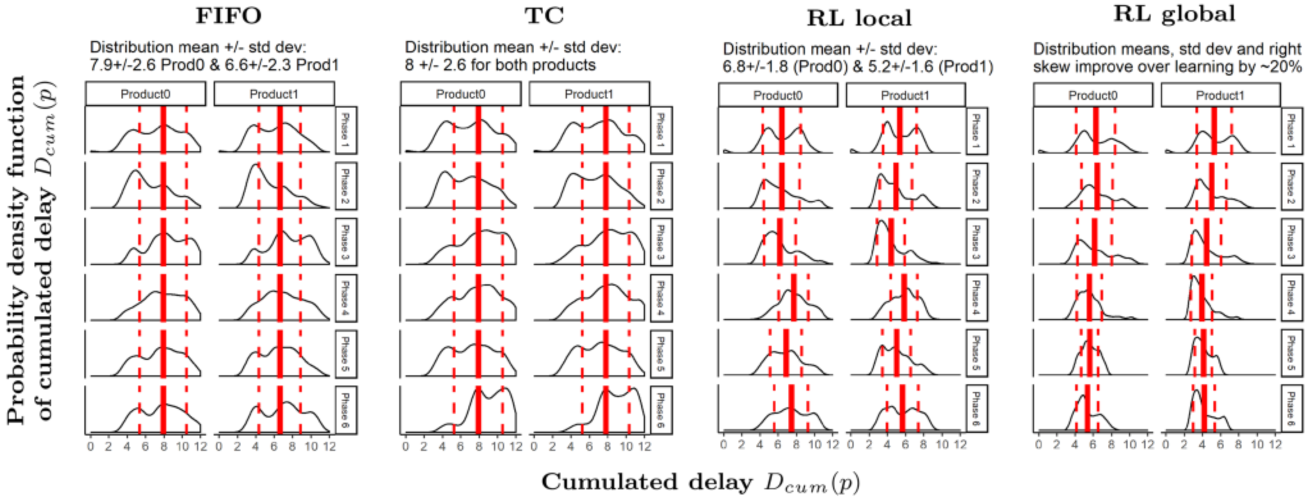

Fig. 3 Density plot of cumulated delay $D_{cum}(p)$ measured at the exit buffer for both products. The entire learning process is separated into six phase with equal length. Vertical red lines correspond to the mean (solid) and mean ± one standard deviation (dashed)

4.2 Investigation of RL-agent's action selection policy

The local reward function awards high positive urgency ratios UR according to Eq. (3).

On the left of Fig. 4 it is shown how the local reward approaches over the course of the exploratory learning phase during the first 600,000 steps an average value of about 7.5 by increasingly eliminating invalid actions and neutral actions as well as selecting positive UR orders, as any $UR > 0$ leads to a positive reward larger than $2 \cdot \sqrt{5}$. The upper and lower bounds of the local reward according Eq. (3), the zero reward for idle actions and the punishment for invalid actions at -1, are also clearly visible.

In contrast to that, the global reward directly awards the desired overall result, i.e. a small number of time constraint violations V_p (see Eq. 4). Moreover, it creates an even more complex behaviour due to two additional actions, which increase or lower the number of order releases at the start of the production line. As discussed above, these actions are balanced by a WIP-sensitive multiplicative reward component as defined in Eq. (5) to prevent an under-loaded system.

The resulting reward is displayed on the right in Fig. 4. Note the difference in the y-axis scaling due to the multiplication with the WIP reward component. It is interesting to observe how fluctuations induced by WIP imbalances are narrowed down as a balance between order starts, time constraint violations, and predefined WIP target is reached over the learning process.

It should also be emphasized how a minor extension of the action space to include order release actions balanced by a corresponding WIP reward allows to broaden the problem scope and thus solution space. This demonstrates the superior versatility of RL approaches in complex real-world manufacturing settings.

Figure 5 compares the agent's action selection with two graphs showing the actual buffer slot selected by the agent. Recall that the actions are shuffled in every step and it is, therefore, prevented that the agent memorizes single slots instead of actually learning a control policy. Recall also, that FIFO strictly picks the first slot, i.e. slot 0, which is also preferred but not strictly adhered to by the TC heuristic. However, the two RL-agent strategies vary this pattern by including also late entries to the buffer, i.e. higher slot numbers. While the local reward function favors orders that are well in time compared to their time constraint, possibly letting a certain amount of orders run entirely out of time, the global reward function focuses on minimization of time constraint violations, which also means taking care of potential time constraint violations more holistically.

Most of the time, human decision makers would choose orders that can be avoided from running out of time. But this is apparently not the case for the locally rewarded agent, which goes with the two "extremes" of reaping high rewards for many orders and no rewards for late orders. Interestingly, the global reward mitigates this drastic local strategy by continuously smoothing the order picking between the first entered lot all the way down to the last lot.

5 Summary and outlook

5.1 Summary

The opportunities and challenges of manufacturing in times of pervasive digitalization lead to a new era of operations management. Intelligent and autonomous control approaches

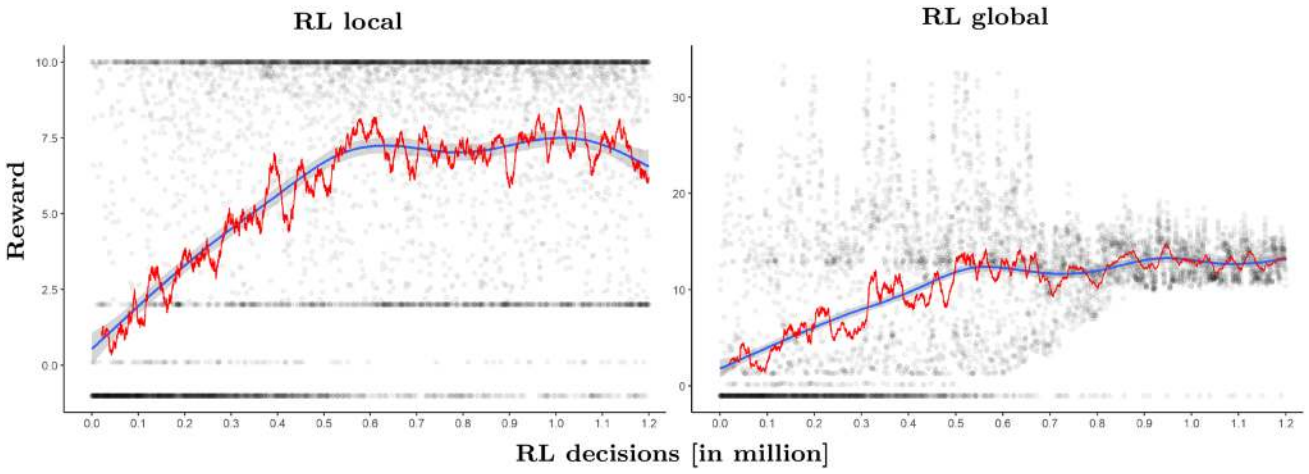

Fig. 4 Development of the RL reward over the entire training process. The moving average over the last 100 values (red) as well as a running median (blue) are highlighted

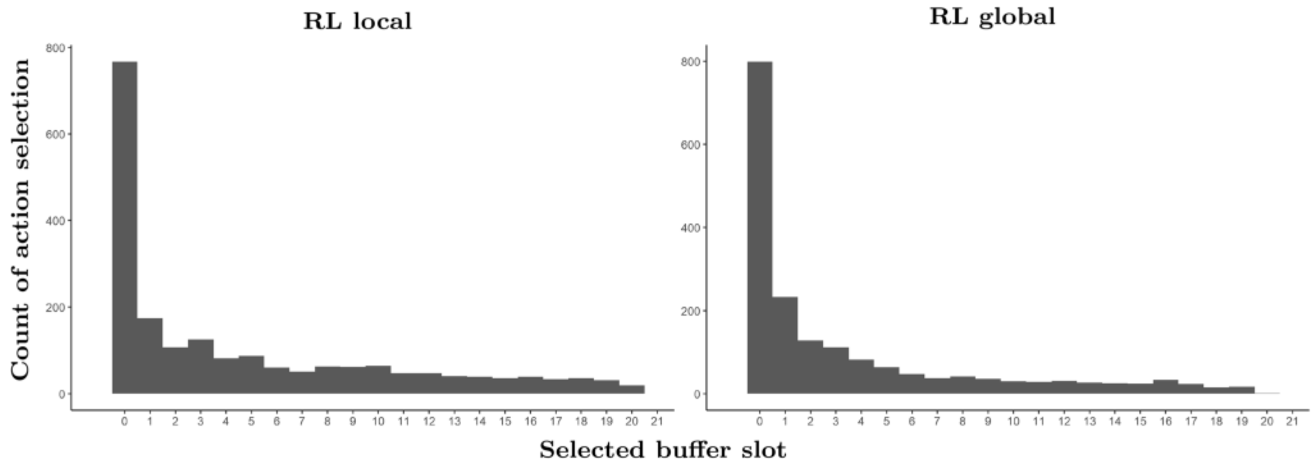

Fig. 5 Frequency of different actions by the RL-agent, i.e., which buffer slot is selected. The numbers are based on the last two of the total six phases

of complex job-shops are enhanced by reinforcement learning. This work contributes to this research by an investigation of order dispatching under time-constraints in complex job-shops. The computational results are based on a discrete-event simulation of a semiconductor wafer fab and reveal the high potential of a model-free temporal-difference Q-learning algorithm (DQN).

The results show that RL-agents can be successfully applied to control order dispatching in a real-world-sized application use case. Moreover, time constraints are managed better than industry-wide established benchmarks, i.e. a heuristic that optimizes time constraints or a FIFO-oriented heuristic.

Furthermore, the experiments show that modelling of rewards is a major and critical part for successful RL applications. This is, in particular, relevant for constraint problems such as order-based time constraints. Thereby, this research contributes to the research question of how to best enforce RL-agents to optimize action validity. Moreover, it is shown how a minor extension of the action space in conjunction with a complementing reward allows to easily increase the scope of action and policy space.

5.2 Outlook

Elaborating and optimizing reward functions as well as the state space seem to offer further potentials for even more improved results. For instance, including the buffer utilization or expected residual machine life time in the state space would allow the agent to optimize the WIP flow and even preventive maintenance schedules [8]. With respect to the deep learning capabilities, inserting additional layers and more sophisticated layer structures should be envisaged. Moreover, the numerous opportunities for hyper-parametrizing RL-agents are so far exploited only to a certain degree. Finally, alternative RL-agents, e.g. advanced policy-based RL algorithms, might achieve good results, too, as they especially prove to be robust given changing problem characteristics.

The long-term vision of self-learning manufacturing systems in real-world is certainly still a long way down the road. For this to happen, improvements need to be made in multiple areas: data availability and quality is still a core issue and a decisive constraint for many companies. Additionally, cyber-security and management of anomalies are major concerns when implementing autonomous systems. Humans need to get convinced to understand and accept autonomous decisions. Therefore, RL, being the most promising way to go, still needs to be understood deeper and developed further in order to be used as single decision-making controller in manufacturing systems.

Acknowledgement Open Access funding provided by Projekt DEAL.

References

1. Bauernhansl T, ten Hompel M, Vogel-Heuser B (2014) Industrie 4.0 in produktion, automatisierung und logistik. Springer Fachmedien Wiesbaden, Wiesbaden
2. Sun D-S, Choung Y-I, Lee Y-J, Jang Y-C (2005) Scheduling and control for time-constrained processes in semiconductor manufacturing. In: ISSM 2005, IEEE international symposium on semiconductor manufacturing, pp 295–298
3. Eversheim W, Wiendahl HP (eds) (2000) Wörterbuch der PPS—Dictionary of PPC: Deutsch–Englisch/Englisch–Deutsch|German–English/English–German. Springer, Berlin
4. Greschke P, Schönemann M, Thiede S, Herrmann C (2014) Matrix structures for high volumes and flexibility in production systems. Proced CIRP 17:160–165
5. Kiener S, Maier-Scheubeck N, Obermaier R, Weiß M (2017) Produktionsmanagement Grundlagen der Produktionsplanung und -steuerung, 11th edn. De Gruyter, Berlin
6. Klemmt A, Mönch, L (2012) Scheduling jobs with time constraints between consecutive process steps in semiconductor manufacturing. In: Proceedings of the 2012 winter simulation conference (WSC), pp 1–10
7. Knopp S (2016) Complex job-shop scheduling with batching in semiconductor manufacturing. Ph.D. thesis, L'Université de Lyon
8. Kuhnle A, Jakubik J, Lanza G (2019) Reinforcement learning for opportunistic maintenance optimization. Prod Eng Res Devel 13:33–41
9. Kuhnle A, Röhrig N, Lanza G (2019) Autonomous order dispatching in the semiconductor industry using reinforcement learning. In: 12th CIRP conference on intelligent computation in manufacturing engineering, Procedia CIRP, vol 79, pp 391–396
10. Kuhnle A, Schäfer L, Stricker N, Lanza G (2019) Design, implementation and evaluation of reinforcement learning for an adaptive order dispatching in job shop manufacturing systems. In: 52nd CIRP conference on manufacturing systems, Procedia CIRP, vol 81, pp 234–239
11. Lanza G, Ferdows K, Kara S, Mourtzis D, Schuh G, Váncza J, Wang L, Wiendahl HP (2019) Global production networks: design and operation. CIRP Ann 68(2):823–841
12. Lödding H (2016) Verfahren der Fertigungssteuerung Grundlagen, Beschreibung, Konfiguration, 3rd edn. Springer, Berlin
13. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
14. Mönch L, Fowler J, Mason S (2013) Production planning and control for semiconductor wafer fabrication facilities: modeling, analysis, and systems. Springer, New York
15. Plappert M (2016) Keras-rl. https://github.com/keras-rl/keras-rl. Accessed 17 Oct 2019
16. Russell SJ, Norvig P (2009) Artificial intelligence: a modern approach, 3rd edn. Pearson Education, Upper Saddle River
17. Schuh G, Stich V (2012) Produktionsplanung und-steuerung 1: Grundlagen der PPS, 4th edn. Springer, Berlin
18. Schuh G, Stich V (2012) Produktionsplanung und-steuerung 2: evolution der PPS, 4th edn. Springer, Berlin
19. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, Chen Y, Lillicrap T, Hui F, Sifre L, Van Den Driessche G, Graepel T, Hassabis D (2017) Mastering the game of Go without human knowledge. Nature 550(7676):354–359
20. Stricker N, Kuhnle A, Sturm R, Friess S (2018) Reinforcement learning for adaptive order dispatching in the semiconductor industry. CIRP Ann 67(1):511–514
21. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction, 2nd edn. MIT Press, Cambridge
22. Waschneck B, Altenmüller T, Bauernhansl T, Knapp A, Kyek A, Reichstaller A, Belzner L (2018) Deep reinforcement learning for semiconductor production scheduling. In: 2018 29th annual SEMI advanced semiconductor manufacturing conference (ASMC), pp 301–306
23. Waschneck B, Altenmüller T, Bauernhansl T, Kyek A (2016) Production scheduling in complex job shops from an Industrie 4 . 0 perspective : a review and challenges in the semiconductor industry. In: Proceedings of of SamI40 workshop at i-KNOW '16
24. Wiendahl HP, ElMaraghy H, Nyhuis P, Zäh M, Wiendahl HH, Duffie N, Brieke M (2007) Changeable manufacturing—classification, design and operation. CIRP Ann 56(2):783–809

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