Application of Reinforcement Learning in Production Planning and Control of Cyber Physical Production Systems

Andreas Kuhnle¹, Gisela Lanza¹

¹ wbk Institute of Production Science, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
andreas.kuhnle@kit.edu

Abstract. Cyber Physical Production Systems (CPPS) provide a huge amount and variety of process and production data. Simultaneously, operational decisions are getting ever more complex due to smaller batch sizes (down to batch size one), a larger product variety and complex processes in production systems. Production engineers struggle to utilize the recorded data to optimize production processes effectively.

In contrast, CPPS promote decentralized decision-making, so-called intelligent agents that are able to gather data (via sensors), process these data, possibly in combination with other information via a connection to and exchange with others, and finally take decisions into action (via actors). Modular and decentralized decision-making systems are thereby able to handle far more complex systems than rigid and static architectures.

This paper discusses possible applications of Machine Learning (ML) algorithms, in particular Reinforcement Learning (RL), and the potentials towards an production planning and control aiming for operational excellence.

Keywords: Production planning and control; Order dispatching; Maintenance management; Artificial intelligence; Reinforcement Learning.

1 Introduction

The productivity of manufacturing systems and thus their economic efficiency depends on the performance of production control mechanisms. Because of an increasing global competition and high customer demands, the optimal use of existing resources is ever more important. Optimizing production control is hence a central issue in the manufacturing industry.

Companies are additionally facing complex manufacturing processes due to high product diversity, lot size reduction and high quality requirements. In the herein considered real-world example of the semiconductor industry, complexity arises through a high number of manufacturing processes and their precision on a nanometer level [1]. Planning and coordinating processes is a challenging task and requires appropriate control methods and decision support systems.

Moreover, production control has to deal with a dynamic and non-deterministic system inside a volatile environment and thus has to handle uncertainty and unexpected incidents [2]. Currently, production planning and control systems such as mathematical programming, heuristics and rule-based approaches are highly centralized and monolithic and not able to meet these needs [3]. Therefore, the dynamic characteristics of production systems are poorly met.

Through the integration of manufacturing components, enhanced process monitoring and data collection, Cyber Physical Production Systems (CPPS) provide real-time data such as order tracking, machine breaks and inventory levels. This makes it possible to apply data-driven techniques, such as Machine Learning (ML) algorithms. Additionally, these are able to adjust to the current system state by analyzing the available data in real-time. This paper shows the successful implementation of a decentral production control system that is based on ML algorithms. The system focuses on the following two use cases: order dispatching and maintenance management. As performance benchmark an existing rule-based heuristic is considered. The real-world use case is taken from a semiconductor manufacturing company that is regarded as a highly suitable example of a cyber physical and digitized production system.

2 Fundamentals and literature review

2.1 Requirements within the semiconductor industry

The semiconductor manufacturing is classically divided into two parts: the front-end, before splitting the wafers, and the subsequent back-end. The front-end comprises all processing steps before cutting the silicon wafer. It consists of several thousand individual processes and lasts between 11 and 20 weeks. Generally, semiconductor manufacturing is considered as one of the most complex manufacturing processes in discrete manufacturing [4]. Between the actual manufacturing processes, control and cleaning processes are required repeatedly. Many of these processes are also performed several times on a wafer so that in general the entire process is not linear. Certain processes are recurrent to build up layers in and on the silicon wafer. Moreover, there are time restrictions between process steps as wafers contaminate quickly when not processed further [1].

2.2 Order dispatching and maintenance management

The assignment of orders to machines for processing is addressed in the so-called order dispatching. Dispatching is an optimization problem that aims to assign orders to resources and hence determines the sequence and schedule of orders. It directly influences the objectives utilization, throughput time (TPT) and work-in-process (WIP).

Next to an optimal order assignment, the robustness of each resource of the system to failures is crucial and has a high influence on these objectives. Therefore, the goal of maintenance management is to maintain availability at minimal cost. Reactive maintenance, i.e. repairs, is balanced with inspection and preventive maintenance measures with the goal to achieve the highest possible uptime of the resources.

Given the challenges of wafer fabrication, order dispatching and maintenance management becomes crucial. Based on real-time process and product data the dispatching and maintenance decisions can be enhanced by ML algorithms in order to optimally match the current manufacturing situation and objectives.

2.3 ML in production planning and control

ML refers to a subsection of artificial intelligence. Many other disciplines of artificial intelligence, such as the processing of natural language or robotics, whose intelligent behavior presupposes a broad knowledge base, are based on this.

There are various industrial applications where ML algorithms are applied with promising results [5]: In [6] an ML algorithm is implemented to control the process parameter power in a laser welding process. The experimental results for a particular setup show that the algorithm generates stable solutions and is suitable for a real-time and dynamic control mechanism. In the context of production control, other authors investigated the usage of ML for order scheduling. The scheduling approaches differ in their overall architecture. The system proposed in [2] and [3], for example, focuses on a highly distributed form, where each resource and each order are considered as intelligent agents. In this kind of architecture resources bid for the allocation of an order depending on the estimated processing cost when being selected. To reduce computational complexity a ML-based solution is presented to estimate the benefit of allocating a job to a specific resource. The implemented ML algorithm uses a table representation in a single objective problem. The work of [7] applies Q-learning to a single-machine scheduling problem and a layout with a few process steps. The order scheduling at each machine and the order release are performed by ML-based agents.

These examples demonstrate the wide range and successful application of ML algorithms in the domain of production engineering. Based on this research the broader application of ML in production planning and control is considered in this paper.

3 Application of reinforcement learning in CPPS

Reinforcement Learning (RL) as one subcategory of ML algorithms addresses the question of how an autonomous, intelligent program (from hereon also named agent) observes and acts in its environment, learning to choose optimal actions in order to achieve a certain goal defined in the beginning. For this, every action of the agent in the environment is rewarded or punished via a scalar number that indicates the desirability of the action, with respect to the overall objectives. The goal of the agent is to maximize this positive feedback [8]. Thereby, the agent explores its environment and learns the optimal connection between the input signal, i.e. the current state of the system, and the action without having to rely on any previous training [9].

3.1 Agent definition

Agents are an essential concept of not only RL but intelligent computing and distributed system design in general [5]. On a functional level, an agent is a computational system that (i) interacts with a dynamic environment, (ii) is able to perform autonomous actions and (iii) acts with regard to a specific objective [5]. To achieve this behavior an agent architecture that has three key components is proposed [10]: For the interaction with its environment the agent needs sensors to perceive relevant aspects of its surrounding and actuators to execute actions. To generate objective-driven actions, a third component, the so-called agent function is required. These characteristics are in line with the general characteristics of CPPS.

In this model, the agent function is the key component for defining the agent's behavior. It determines how the perceived information is processed to decide on actions that lead to a "good" performance with regard to the overall objectives. At the same time, it needs to compromise the agent's experiences. This is crucial to learn the consequences of the agent's decisions. Eventually, the agent function represents a learned model of the environment. The system can consist of several agents with overlapping environments. In that case it is called a multi-agent system [3].

3.2 Reinforcement Learning algorithm

RL applies the ideas of a learning agent-based approach to optimization problems. Because the learning capability is based on repeated interaction with the environment it is often referred to as "trial and error" learning [11]. Despite the existence of many different RL algorithms that vary in the concrete realization of the learning functionality, they follow the same steps in the agent-environment interaction shown in Fig. 1.

Fig. 1. Agent-environment interaction, derived from [11]

The agent perceives the actual state of the environment as a vector S_t . In order to decide on an action A_t the information is processed in the agent function that stores the current policy $\pi_t(a|s) = \mathbb{P}(A_t = a | S_t = s)$. After the action is performed in the environment the agent perceives the new state S_{t+1} and a reward signal R_{t+1} . Note that the environmental transformation is closely linked to the concepts of Markov Decision Processes (MDP). According to the received feedback, the agent adapts its policy. [11]

These steps are repeated in an iterative procedure. As a result, the agent optimizes its behavior in a way to find a policy π maximizing the long-term reward – and therefore a policy that corresponds best to the agent's objectives. [11]

Finding an optimal policy is a iterative process. In each iteration, the current policy π_t is adapted depending on the latest experiences. There are two main techniques to

determine the new policy: (i) value-based and (ii) policy-based approaches. The main difference between both approaches is that value approximation learns the action-value function during the interaction instead of directly learning a policy π . The value function $q_\pi(s,a)$ defines the expected long-term return when choosing an action a in state s following policy π . The policy is then derived from the estimated value of all possible actions in each state. Policy approximation, on the other hand, directly updates the policy function $\pi_t = \pi_t(a|s)$.

Most real-world problems deal with continuous action and state spaces. Storing and updating the policy or value function in a table is therefore computationally inefficient and requires lots of memory space. One possibility is to store the original policy or value function approximatively. Artificial neural networks are widely used for that purpose, as they are capable of approximating complex functional relationships via multiple weights connecting the neurons within the network and allow the adaption of those weights dynamically during the learning process [11]. As a result, neural networks reduce the computational effort by updating a set of weight parameters instead of the values for each state-action pair in each iteration. A dense fully connected feed-forward network is considered in this paper.

Depending on the dimension and the characteristics of the problem, different learning approaches lead to good results. In recent years, new kinds of RL algorithms such as PPO [12], TRPO [13] and DQN [14] were developed to deal with complex problems in different domains. They can be regarded as advanced policy or value approximation algorithms that are optimized with regard to an efficient and stable learning process. The results of this paper are based on these RL algorithms.

4 Case study and experiment results

4.1 Case study setup and description

The considered production system is the production area for wafer implantation. The layout of the production area is illustrated in Fig 2. It consists of three sections with in total eight machines and one entrance and exit lift per section. Regardless of the sections, the machines are grouped according to the principle of job shop production, which can perform the same processing steps. Processing begins with incoming orders at the lifts and the distribution to the respective, pre-defined machines and ends after the order has been processed on the machine and is transported back to the lift. When unloading orders from the lift, access to the first element is always possible. One worker does the transportation between the resources manually. The worker receives the information which order to transport from a central control system. Intermediate storage does not exist, however the machines have a limited buffer in which order batches can be stored before and after processing. The unprocessed batches in the input buffer are automatically fed by the machine according to the FIFO principle and, after complete processing, automatically put into the output buffer.

For this real-world system, a virtual simulation model has been implemented to derive the computational results and evaluate the performance of the RL algorithm. Both,

the simulation model and the RL algorithm, are implemented in Python to be able to implement the bidirectional interaction of the RL agent with the production system.

Fig. 2. Layout of production area.

4.2 Intelligent order dispatching

Due to multiple stochastic influences, such as volatile processing times, changing product variants, dysfunctional manufacturing resources and the limited number of transportation resources (just one worker), the system demands a highly flexible order dispatching system. So, the RL-based agent, that decides which order to dispatch next, needs to consider the state of the CPPS in real-time, e.g. the location of all unprocessed and processed batches, tool state information and remaining processing time.

However, it considers just the information that is relevant for the optimal behavior. Just the following state information is taken into account: First, the location of the worker. Second, for each machine one variable for the machine's current availability and the buffer filling state to indicate whether an action ending at a specific machine is possible or not. A second variable based on the existence of a processed order in the machine buffer indicating whether an action starting at a specific machine is possible or not. Two variables for the sum of processing times of unprocessed orders and waiting times of processed orders at each machine. Third, for each entered order one variable for the longest waiting order. A second variable indicates on which machine the longest waiting order has to be processed.

There are three types of possible actions for the agent. Standing at a certain location (machine or lift), the agent can either dispatch an unprocessed order to one out of the eight machines, bring a processed order back to a lift or change its location by moving empty-handed. Additionally, there is the possibility to wait in case there is no order to be dispatched. Moreover, it might be beneficial to wait voluntarily knowing that a batch is available at this location soon.

Objective-driven actions require a feedback from the environment to the agent. This feedback has to be a numeric signal that is transferred to the agent after each action. In this use case a reward of zero is given when the agent decides on an action that cannot be executed by the worker, for example due to machine failure or a buffer overflow. A low value indicates that the agent should avoid such kind of actions, whereas a high value makes the agent behave similarly in the future.

It can be shown that the RL algorithm improves its performance over time, proving that it can be applied as flexible order dispatching control that continuously learns the optimal behavior. Fig. 3 shows the development of the reward signal starting from the initial state where the agent's behavior is completely random. The agent successfully learns a high performance behavior, however not losing the desired flexible behavior. The reward fluctuation points out that the agent is adaptive enough to react to changing conditions of the production system (e.g. disturbances, demand fluctuations). The benchmark FIFO-heuristic approach is based on a set of if-then-rules, e.g. "take the longest waiting batch next" and "first dispatch all batches in one area and move to another area afterwards" (to minimize time consuming area changes). According to Fig. 3 the RL-based algorithm yields a superior performance behavior. After the first iterations the utilization drops to a bottom value. In the end, an overall machine utilization of above 90% is achieved, comparing to a utilization of far below 90% for the heuristic. The same applies for the TPT. Moreover, the heuristic results show an almost stable performance that is not able to adapt to changing conditions. [15]

Fig. 3. Reward signal (left), utilization (middle) and throughput time (right); moving average values for 1000 iterations

4.3 Intelligent maintenance management

The aim of the maintenance approach presented in this paper is to predict machine failures and based on this prediction perform the most appropriate maintenance action at the optimal time, which is characterized by a low load of incoming orders, i.e. when the opportunity cost of maintenance are low.

The above presented use case is abstracted and considered as a system that consists of a set of parallel machines, each with a buffer, which receives the orders according to the dispatching. A machine then processes the available orders. The state of each machine is monitored and the state directly affects the performance of the machine, e.g. the operating speed is linked to the achieved output and in case of a failure the machine might only run at a low speed. Initially, the machines operate in a normal mode, where the performance is on the highest level. Each machine fails stochastically. If a critical, failure-initiating value is exceeded, a malfunction begins that ends with the failure after

a certain period. If a machine breaks, a maintenance engineer who is responsible for all machines repairs it and afterwards the machine is back in the desired mode.

In this use case the intelligent maintenance agent is responsible for the decision when and which maintenance action to take. The goal is to reduce the opportunistic maintenance cost, i.e. the optimal action considering the current system load of incoming orders, the cost of the action and the cost of a machine breakdown.

Fig. 4 illustrates the remaining time to failure of a critical state machine at the time the agent performs the action over the learning phase iterations. The agent learns to follow a strategy that brings the action closer to the failure. Additionally, the results proves that the algorithm is able to implicitly learn the prediction and, based on this, perform a suitable preventive action.

Fig. 4 also proves that conducting maintenance as late as possible is able to increase the overall output of the system and comes at lower total cost, since fewer maintenance actions are carried out. The results are compared to two benchmarks: a reactive and a time-based maintenance strategy. The numbers do not take into account the further exploited wear rate of the machine components at the latest possible maintenance time, which is why the actual value tends to be underestimated.

Fig. 4. Remaining time to failure (left, moving average values for 1000 iterations) and cost comparison with benchmark maintenance strategies (right, average values of 40 runs)

5 Conclusion, discussion and outlook

This research has shown that CPPS providing real-time data pave the way for the application of data-driven algorithms to enhance the operational efficiency of production planning and control. RL algorithms are successfully implemented for order dispatching and maintenance management outperforming existing rule-based approaches.

However, ML algorithms are not favorable for all industrial applications. The following properties are advantageous: (i) applications with a limited scope in terms of the number of states and actions (the learning period is dependent on these dimensions), (ii) responsive real-time decision systems (computing the output of a ML algorithm requires just linear operations), (iii) “cheap” training data (the trial-and-error approach is intensively data-driven) and (iv) complex environments that can hardly be described in detail (ability to generalize) [15].

This work brings the application of ML algorithms and the transition towards autonomous production systems one step closer to reality. However, the limitations of ML algorithms and RL in particular still prevail, e.g. in terms of solution robustness. Further research in the area of designing RL algorithms is needed to achieve a broad application also in other areas of production control such as employee allocation and capacity control. Furthermore, research on multi-agent systems is required to broaden the scope of applications.

Acknowledgments

We extend our sincere thanks to the German Federal Ministry of Education and Research (BMBF) for supporting this research project 02P14B161 “Empowerment and Implementation Strategies for Industry 4.0”.

References

1. Mönch L, Fowler JW, Mason SJ (2013) Production planning and control for semiconductor wafer fabrication facilities. Springer, New York
2. Monostori L, Csáji BC, Kádár B (2004) Adaptation and Learning in Distributed Production Control. *CIRP Annals* 53:349-352
3. Csáji BC, Monostori L, Kádár B (2006) Reinforcement learning in a distributed market-based production control system. *Advanced Engineering Informatics* 20:279-288
4. Sturm R (2006) Modellbasiertes Verfahren zur Online-Leistungsbewertung von automatisierten Transportsystemen in der Halbleiterfertigung. Jost-Jetter Verlag, Heimsheim
5. Monostori L, Váncza J, Kumara SRT (2006) Agent-Based Systems for Manufacturing. *CIRP Annals* 55:697-720
6. Günther J, Pilarski PM, Helfrich G, Shen H, Diepold K (2016) Intelligent laser welding through representation, prediction, and control learning. *Mechatronics* 34:1-11
7. Stegherr F (2000) Reinforcement Learning zur dispositiven Auftragssteuerung in der Varianten-Reihenproduktion. Herbert Utz Verlag, München
8. Thomas MM (1997) Machine Learning. McGraw-Hill, Inc., New York
9. Ertel W. (2011) Introduction to Artificial Intelligence. Springer, London
10. Russel S, Norvig P (2016) Artificial intelligence. Pearson Education Limited, Malaysia
11. Sutton RS, Barto AG (1998) Reinforcement Learning: An Introduction. MIT press, Cambridge
12. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal Policy Optimization Algorithms. *arXiv preprint:1707.06347*
13. Schulman J, Levine S, Moritz P, Jordan MI, Abbeel P (2015) Trust Region Policy Optimization. *International Conference on Machine Learning* 2015:1889-1897
14. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing Atari with Deep Reinforcement Learning. *arXiv preprint:1312.5602*
15. Stricker N, Kuhnle A, Sturm R, Friess S (2018) Reinforcement learning for adaptive order dispatching in the semiconductor industry. *CIRP Annals* 67:511-514

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.