# Implementing an Online Scheduling Approach for Production with Multi Agent Proximal Policy Optimization (MAPPO)

Oliver Lohse, Noah Pütz, Korbinian Hörmann

# Implementing an Online Scheduling Approach for Production with Multi Agent Proximal Policy Optimization (MAPPO)

Oliver Lohse, Noah Pütz, and Korbinian Hörmann

Digital Manufacturing Technologies
Siemens AG, Munich,
Germany
{Oliver.Lohse,Noah.Puetz,Korbinian.Hoermann}@siemens.com

**Abstract.** The manufacturing process relies on a well-coordinated schedule that optimally incorporates all available resources to achieve maximum profit. In the case of machine breakdowns, the created schedule does not contain information on how to proceed further. Manual adjustments to the process order do not guarantee optimal utilization of the available resources, as many interconnections of the manufacturing process are not evident to a human. A reliable method is needed that can react to changing conditions on the shop floor and form well-founded decisions to mitigate negative effects. This paper presents an approach to implement Multi Agent Reinforcement Learning for online scheduling a cell-based manufacturing environment with unpredictable machine breakdowns. The developed "Multi Agent Proximal Policy Optimization"-Algorithm (MAPPO) combines already existing approaches in a novel way, by using the centralized learning and dezentralized execution together with an objective function developed for OpenAIs Proximal Policy Optimization algorithm.

**Keywords:** JSSP · Online Scheduling · Multi Agent Reinforcement Learning.

## 1 Introduction

The ongoing development of reinforcement learning methods in combination with data of digitized plants ("Industrie 4.0") yields high potential for the application of data-driven methods in manufacturing processes. Availability, quality and performance, in other words, the overall equipment effectiveness (OEE), can be increased by the analysis of process data [5]. Scheduling based on Reinforcement Learning (RL) is one of the promising examples to increase the performance of a plant. The so-called online-scheduling, also referred to as dynamic scheduling, can decrease the negative effects of sudden machine breakdowns, keeping the performance of the plant at a profitable level [2].

Advanced planning and scheduling systems (APS) rely on heuristics and meta-heuristics to determine the best possible sequence of manufacturing jobs.

Solving this job shop scheduling problem (JSSP) is time consuming and computationally expensive. When unforeseen disruptions occur, rescheduling cannot occur immediately, resulting in impaired production throughput. This effect is amplified when a production line is converted to matrix production. Here, the production cells are no longer rigidly connected and offer the product many opportunities to pass through production. Due to the high complexity of this optimization task, it is not possible for humans to reroute the products in the best possible way in case of a disruption.

Recent research results in the field of reinforcement learning show that RL is able to incorporate many dependencies, uncertainties and probable future decisions into the decision-making process [11]. Furthermore, it has been shown that multi agent systems are capable of working cooperatively to achieve a common goal [7]. These developments suggest that multi agent reinforcement learning can be used to solve the JSSP. In the paper, the JSSP is formulated as a Markov Decision Process. Building on OpenAI's Proximal Policy Optimization (PPO) [10], a multi agent is developed that is capable of solving the JSSP. The results of the MAPPO are then evaluated using a simple heuristic.

## 2 State of the Art

### 2.1 Planning and Scheduling

The main purpose of scheduling is to ensure that the capacities of the manufacturing plant, including machines and human resources, are utilized to the fullest potential. Sudden machine breakdowns, such as broken drill-bits or hydraulic issues, are the main factor of influence on the calculated finishing time of a product. The previously calculated static schedules don't include disruptions. These incidents are commonly known but not well predictable in the manufacturing process. Foresighted planning of production steps that considers these events is impossible. Such failed equipment has a negative impact on time constraints for the following products. [9]

Optimal scheduling of all production orders is a classic job shop scheduling problem (JSSP). Advanced planning and scheduling IT-systems resort to heuristics and meta-heuristics to solve the JSSP [13][3]. Thus, an approximated solution is found for the JSSP. However, due to the high complexity of the JSSP, these methods are very time consuming and computationally expensive [13]. They are therefore not suitable for reacting immediately to the changed situation on the shop floor in the event of a disruption.

Approaches to solve the JSSP using RL have already been published. There are approaches where the dispatching rules are determined by a decentralized MARL algorithm [8][12][4]. An agent controls the dispatching rules of a machine or work center. The products have no intelligence in these approaches. Other approaches present a centralized MARL approach in which the products decide independently on which machine they want to be processed [1]. For this purpose, a Deep Q-network is used for each product. The paper shows that a centralized MARL approach can in principle solve the JSSP.

### 2.2 Proximal Policy Optimization

The "Proximal Policy Optimization" is a policy gradient method for a single agent approach [10]. Policy gradient methods parameterize the policy directly and alter the weights of the parameterization to produce a policy that maximizes the expected return. These methods are fundamental in recent achievements in using reinforcement learning for mastering video games or the game of Go [11]. However, policy gradient algorithms are sensitive to the adjustment of the step size. The step size determines the extent to which the parameters of the neural net are changed during training; if it is too small, the training is unnecessarily protracted; if it is too large, the changes are too strong and do not lead to the optimum policy.

By using the clipped surrogate objective $L^{Clip}$, the PPO controls the change of the policy during the training. The PPO ensures the necessary sensitive adjustment of the step size for each iteration and additionally prevents excessive fluctuations of the neural net parameters. [10]

## 3 Concept of the MAPPO-Algorithm

The MAPPO algorithm presented is based on the single agent PPO. The PPO, with its agent, describes a single product-agent. In contrast to the single agent PPO, in the MAPPO all product-agents share a centralized critic. The centralized critic evaluates the actions of each individual product-agent in the context of all product-agents. For this purpose, it receives the global state and all actions of time step $t$ as input. The $L^{Clip}$ function of the PPO calculates the parameter $\Theta$, which indicates how strongly the parameters of the neural network of the product-agent must be adjusted, see Fig.1. For the adjustment of the neural network of the critic, in addition to the global state, the assigned rewards are also needed. Thus, the product-agents maximize the overall reward with respect to their own reward. Comparable to the centralized critic of the MADDPG [6], the critic itself is only needed for training. During execution, each product-agent takes an action based on the available state, which implicitly considers the actions of the other product-agents.

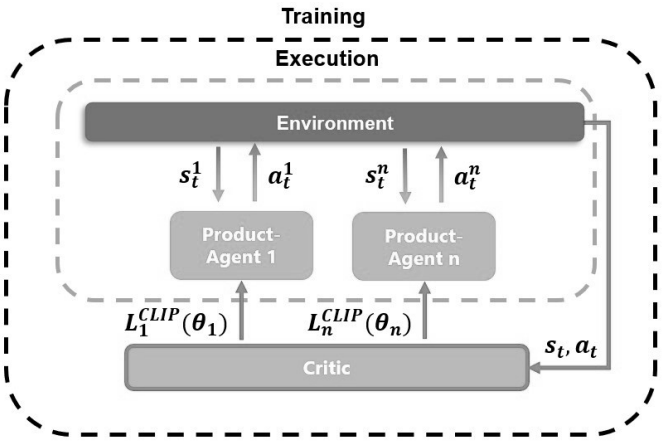


**Fig. 1.** Overview of the MAPPO-Algorithm approach. During the training all product-agents share a centralized critic. In execution the critic is no longer used and each product-agent makes decisions independently.

## 4 Implementation of the MAPPO in a Production Environment

### 4.1 Implementing Product Agents and Environment

At least the following three entities are required to implement MAPPO: Production-Environment, Machines, Products. The entity "Machines" contains the basic logic of all machines. This class is called to create a machine object. Such a machine object corresponds to the virtual representation of a machine in production. Table 1 shows the parameters which are describing this class. If a machine has a queue of more than one product, the dispatching rule always selects the product with the highest priority for processing.

**Table 1.** Parameter to describe the machine entity.

| Parameter | Explanation |
|---|---|
| ID | Identification number of the machine object |
| Processing time | Matrix with all processing times |
| Queue | Length of the queue in front of the machine object |
| Status | Boolean: 0 = no disruption and 1 = disruption |

The product entity contains the logic with which the products can move through the production. The class is used to initialize the products, each of which is assigned to a product-agent. Each product-agent has a set of values that are used to identify and trace the product through production. Parameter which are describing this class are shown in table 2.

**Table 2.** Parameter to describe the product entity.

| Parameter | Explanation |
|---|---|
| ID | Identification number of the product object |
| Status | Current state the product is in, e.g. processing, moving, waiting |
| Position | Indicates on which machine the product is currently located |
| Priority | Priority of the product |
| Process history | Keeps track of which machines the product has already been on. |
| Complete | Boolean that is set to "True" once the product has completed all processing steps. |

In the production environment, the machines and the product entity are called and the respective objects are initialized. The production environment sends the observation and the reward to the product-agents, which then send back their respective actions (cf. Fig 1). The product-agents interact with the production until they are either all processed or the maximum time has been reached. This period of time from the first generation of observations to the last sending of actions is called a training episode. After each training episode, the manufacturing model resets and sends the initial observations, which re-trigger the entire training episode cycle. In the new training episode, the product-agents now have the opportunity to make new decisions and gain new experience. Such training episodes can occur any number of times within a training session.

An observation of a product-agent is a list of values with information about the production and about the product itself. It serves as input for the neural network within the product-agent for selecting an action. All input-values of the neural network are scaled to a value between 0 and 1 before input into the neural network, as this improves the processing quality in the neural network.

### 4.2 Designing the Reward Function

MAPPO learning is analogous to other reinforcement algorithms. At the start of the learning phase, the actions of the products are purely random. Via the reward function, each product-agent receives a reward according to its action. The goal of the product-agents is to maximize this reward. For this purpose, actions that have led to a high reward are executed more frequently. Other actions, where the reward is very low, are not repeated. After many training episodes, the product-agents have learned a behavior that leads to the highest possible reward and thus to the desired behavior in production.

Decisive for the learned behavior is the reward function. For the MAPPO in the production environment, a negative reward of -1 is given to each product at each time step. With this reward, the product-agents learn a behavior that minimizes the negative reward - the product-agents move along the fastest possible route through production. In addition, the product-agents have different priorities that affect their reward per time step. The higher a product is prioritized, the greater the negative reward the product-agent receives per time step. The negative reward per time step is multiplied by the priority. Thus, products with a higher priority receive a higher negative reward per time step. The reward is mathematically defined as follows:

The episode reward $R_a$, defined in Formula 1, for a product-agent is the sum of all rewards achieved per time step t of an episode. The reward per time step is calculated as -1 multiplied by the priority of the product-agent $P_a$.

$$R_a = -1 * \sum_{t=1}^{T} P_a \tag{1}$$

The episode reward of all product-agents $R_{tot}$ is the sum of the episode rewards of all product-agents, see Formula 2.

$$R_{tot} = \sum_{a=1}^{A} R_a \tag{2}$$

## 5 Validation of the MAPPO

To validate the MAPPO algorithm presented in the previous chapter, it is compared with a heuristic. The heuristic distributes all products evenly to the available machines at the beginning of an episode and routs the products to the nearest machine after the production step is completed. This simple heuristic is intended to replicate the behavior of an employee on the shop floor who always moves products to the next possible machine with the lowest transport time. According to the routings, all products must pass through all machines. The sequence in which the products pass through the machines is not specified. In addition, all products have the same priority. Disruptions are not considered with the test runs. The MAPPO and the heuristics are compared in six different sized environments. On the X-axis Fig.2 shows the number of machines times the number of products is given. The Y-axis describes the time needed for the products to pass through all machines. The Travel-Time-Matrix is the same for all runs and is extended when another machine is added. The environments differ only in the number of machines and products. In the following, the environments are therefore given as the number of machines times the number of products. The minimum throughput time for the runs in the 2x3, 3x3, 2x5, and 3x5 environments is determined using a path search algorithm. For the 4x5 and the 4x10 environment, the computational cost of the path search algorithm is too

great. The MAPPO leads to the same throughput time as the path search in the first four environments and has thus reached the global optimum. The heuristic is very close to MAPPO for the first three environments. The distance between MAPPO and the heuristic is about 5% in the 3x5 and 4x5 environment. In the 4x10 environment, the difference between MAPPO and heuristic is the largest. The MAPPO is 10% faster in this case. It can be assumed that as the complexity of the environment increases, the difference between MAPPO and heuristic becomes larger. This is particularly interesting for the implementation of the MAPPO in a real production with significantly more machines and products. Advanced heuristics will perform better in these cases than the heuristic used in this example. But the computation of an optimal solution in case of a disruption is time consuming. Due to the way neural networks work, the learned MAPPO can react immediately to disruptions and reroute products even in complex production environments - it thus complements an APS. It can be assumed that the performance of the MAPPO in a static environment can also be transferred to environments with disruptions. For this purpose, disruptions must already be simulated in the environment during the training of the product-agents. The disruptions in the training should be similar to the failure probabilities of the real machines. The training of the product-agents becomes longer due to the larger number of possible states, but can be limited to realistic training scenarios if real failure probabilities are used.

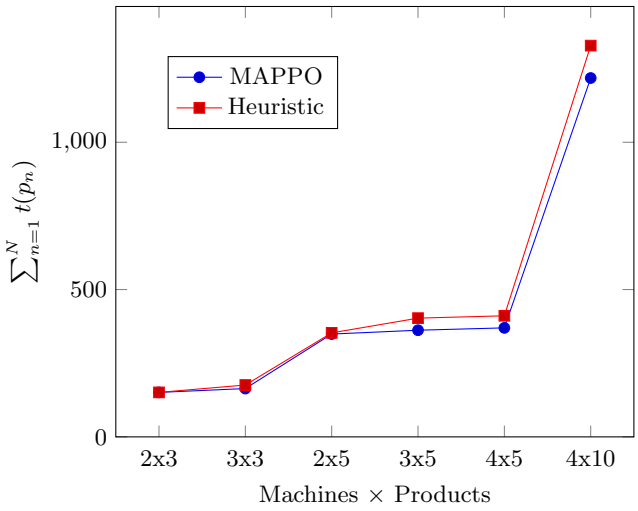


**Fig. 2.** Throughput time comparison in different static environments

The MAPPO can reach the global maximum if the products are homogeneous. This means that the products have the same routings, processing times and the same priority. In this case, the global optimum also corresponds to the local optimum of all products. If the products have different routings or priorities, the global optimum no longer corresponds to the local optimum. In this case, products with a low priority, for example, would have to let other products pass - thus they would worsen their reward in order to optimize the global reward. However, the experiment shown in Figure 3 illustrates that MAPPO and other MARL algorithms that rely on a centralized critc are not able to achieve a global optimum with the given reward function.

Figure 3a shows the initial situation. Three product-agents have to decide to which production cell they will go. In this case, each product has to go through only one machine. The processing times of the machines are the same for every product. Product 1 and 2 have a low priority (LP). Product 3 has a high priority and therefore gets a higher penalty for each time step in production, see 4.2. The experiment is set up so that at least one product has to wait. A global maximum is reached when one LP product waits and the HP product passes through production without waiting. Even in the trained state, the MAPPO keeps product 3 waiting even though it has the highest priority. The product-agents have learned to always bring about their local maximum in different states. Thus, the overall yield improves over all learning episodes until the overall yield converges to a value that represents a local but not a global maximum.

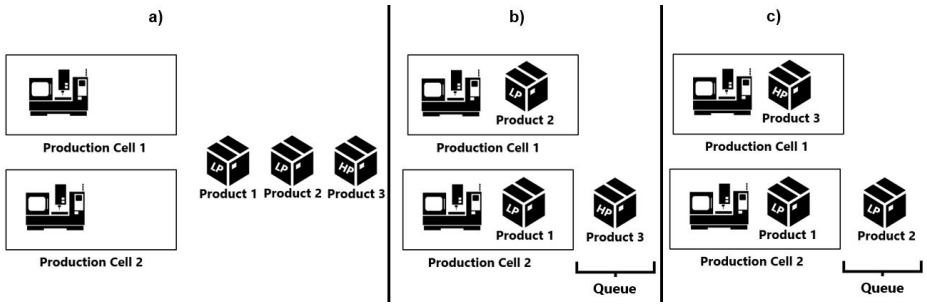


**Fig. 3.** Product prioritization using the MAPPO. Figure 3a depicts the initial situation. The local optimum of the MAPPO can be seen in Figure 3b. Figure 3c shows the global optimum for this problem.

## 6 Conclusion and Outlook

This paper presents an approach for controlling products in a production using multi agent reinforcement learning. The developed MAPPO algorithm is presented in this paper and validated in a production scenario. The MAPPO

is compared with a simple heuristic in a static environment. Several test runs are carried out in environments of different sizes and the throughput time of the MAPPO is compared with that of the heuristic. The MAPPO achieves the minimum throughput time in smaller production environments with homogeneous products. The MAPPO also performs better than the heuristic in larger environments. It can be assumed that the performance of the MAPPO in static environments can also be transferred to environments with disruptions. In this case, the corresponding disruptions must occur in the training runs. However, if the products are no longer homogeneous, the MAPPO cannot generate a global production maximum - in this case, a local optimum is generated, in which each product-agent optimizes its own reward. Future work will investigate whether the MAPPO can also be used to induce a global production maximum for inhomogeneous products. In addition, the MAPPO must also be applied to larger production environments with disruptions. The results of these investigations could then be compared with state-of-the-art heuristics from APS. It can be assumed that the MAPPO holds great optimization potential for production. The MAPPO can react to disruptions without additional computation. Especially for complex cell-based productions with many possibilities for the product to pass through the production online scheduling systems will be necessary. Further studies on the MAPPO will show whether it can meet all production requirements.

## References

1. Baer, S., Turner, D., Mohanty, P.K., Samsonov, V., Bakakeu, J.R., Meisen, T.: Multi agent deep q-network approach for online job shop scheduling in flexible manufacturing. ICMSMM 2020: International Conference on Manufacturing System and Multiple Machines (2020)
2. Cadavid, J.P.U., Lamouri, S., Grabot, B., Fortin, A.: Machine learning in production planning and control: A review of empirical literature. IFAC-PapersOnLine **52**(13), 385–390 (2019). https://doi.org/10.1016/j.ifacol.2019.11.155
3. Fera, M., Fruggiero, F., Lambiase, A., Martino, G., Elena, M.: Production scheduling approaches for operations management. In: Schiraldi, M. (ed.) Operations Management. InTech (2013). https://doi.org/10.5772/55431
4. Gabel, T., Riedmiller, M.: Scaling adaptive agent-based reactive job-shop scheduling to large-scale problems. 2007 IEEE Symposium on Computational (2007). https://doi.org/10.1109/SCIS.2007.367699
5. Gyulai, D., Pfeiffer, A., Nick, G., Gallina, V., Sihn, W., Monostori, L.: Lead time prediction in a flow-shop environment with analytical and machine learning approaches. IFAC-PapersOnLine **51**(11), 1029–1034 (2018). https://doi.org/10.1016/j.ifacol.2018.08.472
6. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments, http://arxiv.org/pdf/1706.02275v4
7. OroojlooyJadid, A., Hajinezhad, D.: A review of cooperative multi-agent deep reinforcement learning, http://arxiv.org/pdf/1908.03963v3

8. Roesch, M., Linder, C., Bruckdorfer, C., Hohmann, A., Reinhart, G.: Industrial load management using multi-agent reinforcement learning for rescheduling. In: 2019 Second International Conference on Artificial Intelligence for Industries (AI4I). pp. 99–102. IEEE (92019). https://doi.org/10.1109/AI4I46381.2019.00033
9. Schuh, G., Stich, V.: Produktionsplanung und -steuerung 2. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25427-7
10. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms, http://arxiv.org/pdf/1707.06347v2
11. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016). https://doi.org/10.1038/nature16961
12. Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., Kyek, A.: Optimization of global production scheduling with deep reinforcement learning. Procedia CIRP **72**, 1264–1269 (2018). https://doi.org/10.1016/j.procir.2018.03.212
13. Zanakis, S.H., Evans, J.R., Vazacopoulos, A.A.: Heuristic methods and applications: A categorized survey. European Journal of Operational Research **43**(1), 88–110 (1989). https://doi.org/10.1016/0377-2217(89)90412-8