Adaptive manufacturing: dynamic resource allocation using multi-agent reinforcement learning

David Heik¹, Fouad Bahrpeyma², Dirk Reichelt³

Abstract

The global value creation networks have experienced increased volatility and dynamic behavior in recent years, resulting in an acceleration of a trend already evident in the shortening of product and technology cycles. In addition, the manufacturing industry is demonstrating a trend of allowing customers to make specific adjustments to their products at the time of ordering. Not only do these changes require a high level of flexibility and adaptability from the cyber-physical systems, but also from the employees and the supervisory production planning. As a result, the development of control and monitoring mechanisms becomes more complex. It is also necessary to adjust the production process dynamically if there are unforeseen events (disrupted supply chains, machine breakdowns, or absences of staff) in order to make the most effective and efficient use of the available production resources. In recent years, reinforcement learning (RL) research has gained increasing popularity in strategic planning as a result of its ability to handle uncertainty in dynamic environments in real time. RL has been extended to include multiple agents cooperating on complex tasks as a solution to complex problems. Despite its potential, the real-world application of multi-agent reinforcement learning (MARL) to manufacturing problems, such as flexible job-shop scheduling, has been less frequently approached. The main reason for this is most of the applications in this field are frequently subject to specific requirements as well as confidentiality obligations. Due to this, it is difficult for the research community to obtain access to them, which presents substantial challenges for the implementation of these tools.

This paper focuses on the application and comparison of single-agent RL as well as MARL algorithms for solving the problem of dynamic scheduling in the form of an intelligent resource allocation problem using a model factory as an example, where the objective is to reduce the makespan of given jobs. To reduce the entry barriers for other researchers and to ensure reproducibility, a simulation environment is provided to the research community, which was also used to perform the experiments of this study. By including redundant operations, variations in order compositions, product variants, setup times, and an automated material transportation tools, a realistic approach is made possible. Moreover, the (composite of) intelligent dispatcher(s) is faced with variations in operation times and breakdowns. Further, this study investigates the convergence behavior of trained RL and MARL models, as well as their performance in handling unknown and unforeseen scenarios compared to heuristic approaches. The experiments demonstrate that even under significant time constraints, RL, especially under the multi-agent setting with Proximal Policy Optimization (PPO) algorithm at the core, is able to outperform conventional heuristic methods when dealing with the complex problem of production scheduling under uncertainty.

¹ david@heik.science  
² bahrpeyma@ieee.org  
³ dirkreichelt@htw-dresden.de  
University of Applied Sciences Dresden, Faculty of Informatics/Mathematics, 01069 Dresden (Germany)
Keywords

Multi-Agent Reinforcement Learning, Dynamic Resource Allocation, Manufacturing Systems, Smart Production Systems, Industrial IoT Test Bed

1 Introduction

In the era of Industry 4.0, modern production facilities were characterized by a high degree of product diversity and complex material flows [12]. Furthermore, the manufacturing industry is experiencing unprecedented growth in the amount of data available. Information is gathered from intelligent sensors, directly from the production line (by means of Enterprise Resource Planning or Manufacturing Execution Systems), from the environment, as well as from machine parameterisation [3]. The increasing individualization of products necessitates the implementation of an adaptive production scheduling and management system. The accessibility of process, environmental and quality-related data offers the potential to sustainably improve process and product quality [5, 23] and can help deal with the high dimensionality and complexity of the data as well as the prevailing and dynamic nature of the environment. However, it has been recognised that this information streams can also be challenging and have a negative impact, as they can distract from the main causalities or lead to delayed or incorrect conclusions about appropriate decisions [13, 23]. For this reason, this paper examines the application of reinforcement learning (RL) to the production scheduling process and resource allocation as an attempt to learn how the RL agents perform under complex industrial settings. Since many manufacturing systems, especially in small and medium-sized enterprises, are currently characterised by a heterogeneous mixture of programmable logic controllers (PLC) and proprietary field devices of different brands, data acquisition plays an important role to the successful solution. In the same way, it is essential to ensure that the decisions made in real-time can be transferred to the production system for implementation. Our Industrial Internet of Things (IIoT) Test Bed at HTW Dresden (see Figure 1) is equipped with all of the mentioned features and is located within an industrial laboratory for smart production, which is available for both teaching and research purposes. We present an approach to reduce the overall makespan using Proximal Policy Optimization (PPO) based on this environment. Consequently, we will examine different arrangements in the actor-critic approach. Simulations will be used to train and evaluate the models.

![Figure (1): Industrial Internet of Things Test Bed at HTW Dresden](image-url)
2 Related works

As per the literature, production scheduling is considered to be an essential component of manufacturing systems [24]. Such processes have as their primary purpose the allocation of job operations to processing resources in such a way that a target function is optimized [20]. Due to the underlying decisive impact on production processes and the potential for improving efficiency and effectiveness, scheduling and planning decisions have always been considered an essential research topic in interdisciplinary fields such as industrial engineering, automation, and management sciences [24]. The optimization of the production schedule can be viewed and formulated as a combinatorial problem. As long as the general conditions are static, this problem can be solved mathematically (with exact algorithms), but is not applicable to large practical problems due to its NP-hard nature [23].

However, heuristic and meta-heuristic approaches are frequently used and well-known in practical applications. Lin et al. [14] proposed a smart manufacturing factory framework based on edge computing, in which seven heuristic methods can be selected. These include, among others: first in first out (FIFO), shortest processing time (SPT), longest processing time (LPT), most operations remaining (MOPNR) or longest operation processing time (LOPT). Mati et al. [15] employ the shortest path approach to solving a multi-resource job shop scheduling problem involving blockages in the process. The authors demonstrate how a sequence of operations can be identified in which deadlocks are avoided and the total duration, i.e. the makespan, is minimized. The relevant literature indicates that heuristic methods have been extremely popular in semiconductor manufacturing and even considered the state-of-the-art in real-world algorithmic order dispatching in this domain, since long process chains and short lead times can sometimes allow only very small deviations from the due date [12, 27].

By using meta-heuristics, practicable scheduling solutions can be achieved for smaller problems within an acceptable computation time, where the design of the search operators depends on the specific problem. When a problem becomes increasingly complex, finding a suitable solution can be a lengthy process, which makes it increasingly challenging to apply to real-life scenarios [24]. Using the five meta-heuristics: particle swarm optimisation (PSO), genetic algorithm (GA), harmony search (HS), artificial bee colony (ABC), and Jaya algorithm, Pan et al. [17] studied how these meta-heuristics can be applied and improved to solve the flow shop scheduling problem. Their initialization strategy was based on the Nawaz-Enscore-Ham (NEH) heuristic. Their analysis demonstrated that the ABC algorithm was the most competitive, and they were able to reduce the maximum completion time (makespan). Seidgar et al. [22] were able to minimize the makespan for the two-stage assembly flow shop problem using an imperialist competitive algorithm (ICA). In most cases, they found that their algorithm outperformed the performance another similar algorithm called GA. Allahverdi and Al-Anzi [1] addressed a similar via the application of the following meta-heuristics: simulated annealing (SA), self-adaptive Differential Evolutionary algorithm (SDE) and ant colony optimisation (ACO), whereby SA showed superiority in performance.

A production scheduling problem can be formulated as an RL environment, where an agent makes decisions based on the current system state and receives feedback as reward or penalty accordingly. In such a scenario, a suitable strategy or policy can be obtained through training [24]. RL offers a comparatively new method for solving scheduling problems and is able to produce promising solutions, especially when the environment is characterized by uncertain and dynamic processes, and has high real-time requirements [12]. Zhou et al. [30] used Q-learning to build a manufacturing value network for estimating state values from high-dimensional sensor data of manufacturing objects. Real-time strategies were trained to make decisions according to the states of available machines and pending orders, resulting in a significant improvement in the multi-objective performance metrics. Zhou et al. [29] utilized an extended form of Q-learning in which the observation table was replaced by a network, deep Q-Network (DQN), in order to minimize the maximum completion time of all the issued tasks (makespan). To increase flexibility and robustness in automotive production, Mayer et al. [16] incorporated PPO to modular production systems where workstations are decoupled by driverless transport systems. The results of their experiments in several modular production control environments demonstrated that the learning behavior of their solution is stable, reliable, optimal and generalizable. Schmidl et al. [21] demonstrated a method for implementing RL agents on edge devices to save
energy in production plants through direct communication with the PLC. Based on a simulation environment similar to that described in this paper, but characterized by a lower degree of complexity, Heik et al. [11] analyzed the performance of double deep Q-Networks (DDQN), PPO, and RecurrentPPO in combination with a long short-term memory (LSTM) on the scheduling problem, where the results showed that PPO outperformed other comparative algorithms (heuristics and meta-heuristics). This included parallel operations under uncertainty, but the environment was characterized by only one work plan, omitting the consideration of product families as well as different routes (use of switches and bypass routes). By contrast with the single-agent RL approaches studied to date, Wang et al. [26] introduced a MARL algorithm that used correlated equilibrium to optimize the makespan and cost to control the workflow over clouds. Wang et al. [25] present a novel multi-agent graph convolution integrated scheduler (MAGCIS) for addressing the problem of dynamic task scheduling in cloud manufacturing systems. With their approach, the researchers were able to compete with advanced methods such as QMIX (consists of agent networks representing each $Q_A$ and a mixing network [19]) and value decomposition networks (VDN). In the contribution by Baer et al. [2], the authors propose a multi-agent idea for a multi-stage approach, whereby each agent navigates a specific product to the corresponding processing machine. Another multi-agent approach to solving the job shop scheduling problem is presented by Waschneck et al. [28] In this work, a factory with several workcenters is considered, whereby each of the cooperative DQN agents determines suitable scheduling rules for a specific workcenter. Qu et al. [18] presents a framework for a formal representation of a synchronised, station-based flow shop and studies how marl can be connected to this environment resulting from the ontology in order to plan such a manufacturing system with multi-stage processes and several product types. Heik et al. [10] also investigated the use of MARL for dynamic scheduling wherein, they compare the asynchronous advantage actor-critic (A3C), PPO and RecurrentPPO (LSTM) for different composite scenarios. They observed that PPO in a single-agent configuration was able to outperform the other methods in respect in terms of the overall performance. Due to the very limited complexity within the former publication, in this paper we consider an advanced version in which we expand the number of parallel machines, parallel operations, routing options and the product variety.

3 Problem statement

3.1 Behavior and general conditions

This section describes the methodology used to consider the scheduling task and resource allocation under uncertainty, such as varying operating times or breakdowns for different routings as well as product variants and families. The simulation environment described in the following is based on an existing modular model factory, the IIoT Test Bed at HTW Dresden (see Figure 1), with the capability of manipulating and assembling workpieces via various manufacturing components by systematically applying a variety of manufacturing operations. In the mentioned real factory model, continuous processes have been discretized both in time and space, whereby one step corresponds to one second. Furthermore, the model factory consists of a number of independent stations, that can each perform several operations. With the exception of their position in the 2D-space, the physical properties of the machines are not fully formulated; instead, only their logical behavior is emulated realistically. As each station has a location, there is also an area for receiving and dispensing products. In addition, the stations are characterized by individual but constant loading and unloading times. During the manufacturing procedure, the cyber-physical workpiece handling is taken into consideration and is carried out either before or after the actual operation (or operation sequence). In the event that a product requires multiple operations from the same station, that are performed directly consecutively, no handling is required between the two operations. The operation times are subject to stochastic fluctuations and vary by $\pm 1$ second according to a uniform distribution. Automated material transport between the stations is accomplished by using a ring-shaped and unidirectional conveyor belt. The conveyor belt also has switches and bypass sections, which means the products to be manufactured can take several routes to reach their destination. The conveyor belt graph is shown in Figure 2. A special characteristic of the production line is that only one carrier can be assigned to a slot at a time, which means
that the carriers cannot overtake each other. In addition, the product to be manufactured is married
to a specific carrier from the start of the first operation to the completion of the last operation. If a
station should perform an operation on the workpiece, the carrier is held at the handover position
until the operation (or the operation sequence) is completed and thus unavoidably blocks subsequent
carriers, resulting in a jam. A carrier can therefore only be transported one slot further per time step,
provided that this subsequent slot (or critical section) is free. Alternatively, the carrier must wait for
its predecessor to be transported onwards or wait for the station to complete its process. In the simu-
lation environment, the two product families $\alpha$ and $\beta$ are considered. The associated products differ in
their dimensions, which requires retooling of the stations if the product family is changed in between.
The operations provided by the stations do not differentiate between the different product families, so
the same average reference times apply, which are shown for each station and each operation offered
there in Table 1.

Table (1): Operations offered and their average execution times

<table>
<thead>
<tr>
<th>Station id</th>
<th>Slot on conveyor belt</th>
<th>Loading and unloading time [s]</th>
<th>Set-up time [s]</th>
<th>Operation</th>
<th>Mean operation time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>15</td>
<td>A</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1</td>
<td>10</td>
<td>F</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>1</td>
<td>10</td>
<td>G</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>95</td>
<td>5</td>
<td>20</td>
<td>C</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>2</td>
<td>15</td>
<td>E</td>
<td>27</td>
</tr>
<tr>
<td>6</td>
<td>42</td>
<td>2</td>
<td>15</td>
<td>A</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>48</td>
<td>1</td>
<td>10</td>
<td>G</td>
<td>33</td>
</tr>
<tr>
<td>8</td>
<td>54</td>
<td>1</td>
<td>10</td>
<td>B</td>
<td>32</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>5</td>
<td>20</td>
<td>B</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C</td>
<td>36</td>
</tr>
<tr>
<td>10</td>
<td>66</td>
<td>2</td>
<td>15</td>
<td>E</td>
<td>27</td>
</tr>
<tr>
<td>11</td>
<td>72</td>
<td>1</td>
<td>15</td>
<td>H</td>
<td>26</td>
</tr>
<tr>
<td>12</td>
<td>78</td>
<td>1</td>
<td>12</td>
<td>F</td>
<td>32</td>
</tr>
</tbody>
</table>

There are also six different product variants proposed, which are available for both product families
$\alpha$ and $\beta$, see Figure 3. The orders are managed, prioritized and assigned to the physical carriers via a
central manufacturing execution system. In the simulation environment as well as in the real model
factory, the orders are processed according to the FIFO method, whereby each order has an individual
lot size and specifies a concrete product family and product variant. The order processing sequence is
therefore considered to be given. The execution sequence of the operations is also predetermined and
is specified for each product variant by an individual work plan, see Figure 3. The fact that similar op-
erations are offered by different machines in the manufacturing system generates redundancies. This
helps to eliminate bottlenecks and contributes to maintaining the performance of the overall system
even if operations fail (e.g. due to wear, lack of material or preventive maintenance). The performance
of the assembly line is significantly influenced by the allocation of production resources and the selection of carrier routes. In order to limit the complexity of the decision and eliminate infinitely long routes, the restrictions that each slot may be visited a maximum of once were set up, resulting in the rule that between two executed operations, a downward switch (I1 at Slot#8, I2 at Slot#14, I7 at Slot#74 or I8 at Slot#80) can be used one time or none at all and an upward switch (I3 at Slot#32, I4 at Slot#38, I5 at Slot#50 or I6 at Slot#56) can be used a maximum of once. In addition, a combination of the following queries was made so that only one decision is required per step:

- At which station should the operation be performed?
- Which route should the carrier use to reach the station?

Specifically, with each operation alternative offered, there are generally up to 19 possible route options, although not all of them are always available. For example, a product that has previously completed Operation#20 at Station#8 (Slot#54) and now requests Operation#30 only has the following options:

- Operation#30 at Station#9 (Slot#100), without the use of switches, or
- Operation#30 at Station#4 (Slot#95), without the use of switches, or
- Operation#30 at Station#4 (Slot#95), use of switch I6 at Slot#56.

The bypass points visualized in blue (in Figure 2) cannot be actively controlled. The bypass routes can only be entered indirectly if the next station destination of the carrier is within the bypass. All carriers will avoid the bypass route if their destination is outside the relevant bypass section. The directly following slots behind the respective active and passive (bypass) switches are, like the respective switches themselves, part of the critical section around each switch. In other words, a carrier can only enter the switch if the switch itself is free and the first slots of the two path alternatives are also free. Similar to the real model, the simulation environment was developed in such a way that whenever an empty carrier enters a station, the MES is contacted to ask if an order should be assigned. If this is the case, as described above, a decision is required as to where the first operation should be performed and which route the carrier should take to reach the designated station. The same applies if an operation has been completed and another operation is required to complete the product, or if the last operation in the work plan has been completed (Operation#I) and the carrier is now ready to receive a new product. The current state of the system is presented as follows before each request as a base for the decision-making process. We have chosen a binary representation for the state space. For each of the slots on the conveyor belt, 13 bits are used:

- whether the slot is responsible for the request,
- whether the slot is occupied,
- whether the product in the slot is from product family $\alpha$,
- whether the product in the slot is from product family $\beta$,
- whether Operation#A is still required until completion,
+ 8 further bits for Operation#B, #C, #D, #E, #F, #G, #H and #I.

In addition to the $13 \times 102 = 1326$ bits, there are a further 3 bits for each station (total 36) which indicates:

- whether the station is equipped with product family $\alpha$,
- whether the station is equipped with product family $\beta$,
- whether the station is currently executing an operation.
In addition, there is a bit for each of the operations offered by the stations (total 16) which indicates whether the operation is currently usable (otherwise it has failed). Consequently, the state space is represented by $1326 + 36 + 16 = 1378$ bits. As noted above, we have also considered a non-ideal behavior with respect to the temporal unavailability of operations. The probability of an error occurring is independent of other events and, as in the real model, is exponentially distributed, i.e. it increases with the duration of the error-free operating time. On arithmetic mean, errors occur every $\sim 35.5$ minutes and last on average $\sim 6.2$ minutes.

### 3.2 Problem complexity

The number of possible decision combinations can increase exponentially with respect to the number of items being produced as well as the scale of the production line. As each operation requires a combined decision, both the number of available operation alternatives and the number of possible routes to the destination have a significant impact on the decision space, as shown in Equation (1). As the number of paths is determined by the constellation of current position and destination, the decision space for 64 products (all Variant#1) varies within the limits according to Equation (2) and (3).

$$\text{DecisionSpace} = (\text{OperationAlternatives} \times \text{AlternativeRoutes}^{\text{RedundantOperations}})^{\text{OrderQuantity}}$$  
(1)

$$\text{DecisionSpace}_{\text{Min}} = ((2 \times 17)^{64} = 7.27 \times 10^{134}$$  
(2)

$$\text{DecisionSpace}_{\text{Max}} = ((2 \times 15)^{64} = 5.63 \times 10^{661}$$  
(3)

To determine the number of possible initial situations (see Equation (4) and (5)), we consider:
- the fluctuation range of the operating times (a fluctuation range of $\pm 1$ second results in 3 possible operating times),
- the initial distribution of the carriers on the conveyor belt (the calculation is made using the binomial coefficient), with the binomial coefficient, where $n = 102$ (slots) and $k = 16$ (carriers),
- and the number of products to be produced and the number of product variants.

$$\text{InitialSpace} = (\text{Margin}_{\text{Operations}}) \times \left(\frac{\text{Slots}}{\text{Carrier}}\right)^{\text{Producttypes}_{\text{Orderscope}}}$$  
(4)

$$\text{InitialSpace} = (3^9) \times \left(\frac{102}{16}\right)^{64} = 2.37 \times 10^{72}$$  
(5)

This research aims at developing a decision policy that minimizes the makespan by considering the temporal uncertainty, possible malfunctions, as well as the underlying dynamics of the system, while taking into account the existing redundancy and routing functionality of the system. The best global solution can only be determined by considering all theoretical initial conditions, to each of which all possible permutations of decisions are applied. Equation (6), (7) and (8) illustrate the corresponding mathematical relationship.

$$\text{SolutionSpace} = \text{InitialSpace} \times \text{DecisionSpace}$$  
(6)

$$\text{SolutionSpace}_{\text{Min}} = (2.37 \times 10^{72}) \times (7.27 \times 10^{134}) = 1.71 \times 10^{207}$$  
(7)

$$\text{SolutionSpace}_{\text{Max}} = (2.37 \times 10^{72}) \times (5.63 \times 10^{661}) = 1.33 \times 10^{734}$$  
(8)

### 4 Methodology

#### 4.1 Reinforcement learning

Assuming only the lower limits of the solution space and the exaggerated assumption of being able to calculate a billion experiments per second, it would still take $\sim 5.5 \times 10^{169}$ years to always make the best global decision. However, solving the problem combinatorially is deemed practically impossible. Furthermore, the global maximum is unknown even for a constant initial situation, which is the reason...
that no training datasets with labeled data are available. However, in order to come up with an appropriate solution, we will use an exploratory approach to problem solving, namely RL, which allows us to develop decision policies in an exploratory manner. The literature suggests the use of MARL in conjunction with an on-policy, policy gradient approach to solving complex problems characterized by uncertainty with reduced training effort \[4\]. In previous research, we have obtained the best results in terms of robustness and performance with algorithms that are based on a gradient descent approach and follow an on-policy strategy \[8, 9, 11, 10\]. Moreover, in an earlier contribution, which showed a significantly lower problem complexity, we have already investigated promising methods that use the gradient descent approach \[10\]. In this context, among other approaches, A3C, PPO and RecurrentPPO (implemented with a LSTM) were explored in practice. We analyzed different configurations of the actor-critic architecture to solve the scheduling problem. The best results were obtained with PPO and an architecture where each station has its own independent actor and is evaluated by a global critic. Nonetheless, single-agent approaches were superior, especially with the PPO algorithm. This paper seeks to extend previously proven experiments to a more complex environment.

4.2 Training procedure

We designed our experiments so that training was conducted in episodes. At the beginning of each episode, the environment is reset and a random situation is generated. As in our real world model, the number of carriers on the conveyor belt is set at 16, and fluctuations in operating times are limited to one second. Over the course of the experiments, the probability of operational failure and its duration remain constant. Each time a new initial situation is established, the carriers are randomly distributed, such that critical sections are not occupied. In addition, the breakdown events are determined in accordance with the distribution mentioned above. The actual simulation of the production process takes place after this initialization. A step function provided by the simulation environment is used to fast-forward until the decision is required. The simulation is then paused and the current state of the system is supplied to the problem solver in binary format. The respective problem solver has to make a decision based on the current state description and return this decision to the simulation environment in order to continue the simulation process. The decision itself is an integer number in the range between 0 and 29. Although there are 19 different route options and a maximum of 2 alternative stations that offer the operation (in total 38), a binary form of switch representation allows the answer to be compressed a little. In order to apply the decision, the route and the destination station are extracted from the response, and then applied to the carrier before the simulation continues. This process continues until the next decision is required and the step function is invoked once again. Upon completion of all orders, an episode is considered complete. Due to the fact that the makespan can only be calculated after completion of the last task, the reward for RL models can only be determined once an episode has been completed. The reward is calculated both for the episode and for each individual step and is returned to the agents in combined form, see Equation (13). For this purpose, the activity of all carriers are recorded during production and individual decisions are retraced. The simulation works discretely, so each product and each individual time step can now be assigned to a specific category: being transported (\(T\)), waiting for predecessors (\(W\)), loading into a station (\(L\)), unloading from a station (\(U\)), goods awaiting completion of station setup (\(S\)), performing an operation (\(P\)) or awaiting a breakdown resolution (\(F\)). For each decision, the cardinality of each of these categories is determined and used for calculating the \(Penalty_{\text{Step}}\) according to Equation (12). The evaluation of the \(\text{Episode}_{\text{Rewards}}\) (see Equation (11)) also takes into account how many products were actually completed (see Equation (10)), as a jam can occur due to carrier routing, which cannot be solved without additional algorithms, since each carrier waits for another.

\[
\text{Offset} = 200
\]

\[
Penalty_{\text{Episode}} = \frac{\text{Makespan}}{\text{QuantityOfProductsManufactured}}
\]

\[
\text{Reward}_{\text{Episode}} = \text{Offset} - Penalty_{\text{Episode}}
\]
PenaltyStep = (|T| * 1) + (|W| * 1) + (|L| * 1) + (|U| * 1) + (|S| * 1) + (|P| * 1) + (|F| * 3) \tag{12}

RewardStep = RewardEpisode - (PenaltyStep * 0.05) \tag{13}

4.3 Evaluation

As a means of comparing and assessing the performance of the trained RL methods, we have generated an evaluation dataset for reference, which contains 1000 random generated initial conditions under the same conditions as in training. There is a checkpoint after every 50 training episodes during the training process of the RL models. At this point, the weights of the models are exported in a serial form. In order to make a deterministic prediction with RL models, these weights may be restored for evaluation purposes. Following the completion of the training process, the evaluation checkpoint is selected based on the highest mean reward value observed. During the evaluation, the models determined are applied to the problems in the evaluation dataset, and their respective makespans are determined. Our evaluation metric is based on the observed makespans, which are recorded and averaged.

4.4 Reproducibility

To ensure the reproducibility of our results, we released the source code for the simulation environment [6] used in this paper as well as the evaluation dataset [7]. This allows interested researchers to integrate and evaluate their own methods and algorithms. Further information and implementation details can be found in the repository. We have specified the settings and hyperparameters used for the algorithms in Table 2 and Table 3.

Table (2): General settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input dimensions</td>
<td>((13) * 102) + (3 * 12) + (16) = 1378</td>
<td>((Product and slot information)<em>Slots) + (Station information</em>Stations) + (Operation information)</td>
</tr>
<tr>
<td>Action dimensions</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Maximum episodes</td>
<td>50000</td>
<td></td>
</tr>
<tr>
<td>Uncertainty in operation times</td>
<td>±18</td>
<td></td>
</tr>
<tr>
<td>Ammount of carriers</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

5 Experimental results

5.1 Training

This section discusses the results observed when applying a state-of-the-art RL technique known as PPO, which is based on the actor-critical method. Thereby, the RL method was applied to the simulation environment presented in Section 3, using both single-agent and multi-agent architectures. We limit the training horizon to 50,000 episodes to ensure that our results can be applied in practice in a
reasonable time and effort. If an episode requires an average of 7200 seconds (~2 hours) during training, this corresponds to ~11.5 years of behaviour learning on the real physical system, whereby a total of ~21,333,000 decisions are made. When training the PPO models in the different agent architectures, the previously described reward function (see Equation (13)) is used to achieve the specific objective of this research (minimising the total production time) in the introduced scenario. We present the training curve of 9 experiments each for PPO in conjunction with a single (central) agent and critic as well as PPO in conjunction with 12 agents (one for each station) and with a central critic, using the same parameters (environmental conditions, model hyperparameters and applied reward function) and with the same constraints (maximum 50,000 episodes) in Figure 4 and Figure 5. In each of these graphs, the Y-axis shows the retrieved reward (plus an offset of 200 according to Equation (9)), aiming for the highest possible score. The X-axis, on the other hand, shows the progression over time during training. The graphs were averaged over the last 50 episodes to obtain a smooth representation. For each RL method and architecture, a representative learning curve is selected (outlined in orange), discussed below, and finally the representatives of these experiment series are used for the evaluation. In this process, we never selected the best observation, but one that reflects typical behaviour. It should be noted here that peaks in the graphs cannot be analysed at any point in time, as a cyclical checkpoint was only reached every 50 episodes.

In the experiments with PPO in combination with the single-agent architecture, the plots show a continuous increase in the graphs in $\frac{2}{3}$ of the cases, until a reward of between 86 and 90 is finally achieved around episode 45,000 to 50,000. The graphs of the first two rows visualise the increase in knowledge of the models very nicely. However, we assume that the training has not yet reached saturation and only the hard time constraint has prevented further improvement of the models. In the other $\frac{1}{3}$ of the experiments, no significant increase in knowledge could be detected, the graphs rather suggest a random decision making of the agents, whereby the graph at the bottom left and bottom centre each show a collapse of the learning curve, from which the agents recover again, even if they never achieved a reward above 85 in the rolling average over the last 50 episodes.

Similarly, when PPO is implemented with a multi-agent architecture, we observed a continuous increase in the graphs in $\frac{2}{3}$ of the cases. Here, too, a local maximum is usually reached around episode 45,000 to 50,000, but the reward is much higher, between 97 and 102. Furthermore, the plots show the good exploration behaviour of the algorithm in combination with a multi-agent architecture. Our representative experiment (top left) experienced a significant drop around episode 23,000, but was able to recover from this trough with a very well-chosen decision combination. During the experiment with the best results (top centre), the learning curve shows that the agent initially made very weak decisions at the beginning of training. However, it seems that the agent was able to abstract this knowledge and avoid these situations in the future. In the other $\frac{1}{3}$ of our experiments with PPO in a MARL setting, the results are much more differentiated and no longer follow a distinct pattern. Some of the respective agents were unable to generate an increase in knowledge and tend to make decisions at random.
5.2 Evaluation

In this section, RL models are evaluated and analysed in depth. In order to better interpret the results obtained with RL, we also analysed well-known heuristic dispatching strategies. Among them is the job assignment strategy that is currently used for this scenario in the Industrial IoT Test Bed of the
HTW Dresden, a shortest path first (SPF). For comparison, we also use its counterpart, the longest path first (LPF), which never takes an active shortcut. Furthermore, we consider a random resource allocation to assess whether the agents could indeed abstract knowledge. For each selected representative, those models were evaluated (using the previously described evaluation dataset) whose checkpoints achieved the highest reward in the training process. During the evaluation, we ensured that the results were reproducible, so the exploration strategy was deactivated for the RL methods and a seed was determined for the heuristic random method so that deterministic results could be observed.

<table>
<thead>
<tr>
<th>Method</th>
<th>Heuristics</th>
<th>RL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SPF</td>
<td>LPF</td>
</tr>
<tr>
<td>Mean makespan across all orders</td>
<td>6541.54</td>
<td>9482.56</td>
</tr>
<tr>
<td>Max makespan across all orders</td>
<td>9096.00</td>
<td>12918.00</td>
</tr>
<tr>
<td>Percentage deviation in the makespan</td>
<td>±0.00</td>
<td>+44.96</td>
</tr>
</tbody>
</table>

The Table 4 lists all evaluation results in groups. In addition to the most important metric for us, the average makespan, the maximum observed makespan was also listed as a reference in order to be able to estimate the behaviour of the algorithms in extreme situations. In addition, we have included the percentage deviation from this method based on the current allocation strategy (SPF). In our evaluation, PPO in combination with a multi-agent architecture (MA PPO) achieved the best overall performance and outperformed the previously used SPF allocation strategy. Contrary to our expectations, PPO in combination with the single-agent architecture (SA PPO) achieved a significantly worse result and is even superior to the heuristic assignment strategy on average. Although we were also able to identify episodes in which PPO performed better when analysing the evaluation dataset, partial episodes were also significantly worse. This is also reflected in the line Max makespan across all orders. Here, in the worst case, MA PPO was able to complete the order 62 seconds faster than SPF, whereas SA PPO took 142 seconds longer than SPF and therefore even 204 seconds longer than MA PPO. Nevertheless, SPF partially succeeded in allocating resources in such a way that it was possible to observe a shorter processing time compared to MA PPO or SA PPO. The LPF heuristic procedure, which is very primitive compared to RL, guarantees that no jamming occurs, as the critical sections are always left from one side and thus never cause a long-term congestion. Although we were unable to detect any such jams in our scenario based on the evaluation dataset used, but based on the experience with the IIoT Test Bed it is known that they can occur and that the probability increases with the number of carriers and the use of shortcuts. When analysing the evaluation dataset, neither LPF nor the random could outperform the other methods considered. We can therefore show that the RL algorithms we analysed provide good and reliable results on average, but in some cases there is still potential for further improvement.

6 Conclusion

The challenges faced by modern manufacturing systems are manifold and range from high complexity, dynamics and high dimensionality in the data to chaotic structures in data and unpredictable uncertainties. Production planning in particular, which is an NP-hard problem and prone to stochastic fluctuations, has a significant influence on the overall performance of the manufacturing system. Given the increasing availability of data and computational resources, production can benefit from the application of reinforcement learning. However, these methods are still in the concept phase and require rigorous validation against more established approaches such as a priori planning with exact solvers or simple prioritisation rules [20].

In this work, we have therefore proposed the use of RL under single-agent and multi-agent conditions to handle dynamic scheduling considering potential operation breakdowns and unstable process times within the plant as well as fluctuating order compositions. The RL method used in this research is PPO.
For a better evaluation and assessment of the results, heuristic methods (SPF, LPF and random assignment) were also analysed as possible problem solvers, since an exact determination is not feasible due to the prevailing complexity. Against our initial expectations, the results of our experiments show that multi-agent architectures perform better than single-agent architectures. In our previous experiments, which had significantly lower complexity, the opposite was observed [10]. The models generated with PPO in combination with the multi-agent architecture achieved results that were able to outperform the other methods and architectures, despite the consideration of strict time constraints during the learning process. Nevertheless, when evaluating with the automatically generated evaluation dataset, we observed in some situations that the agents of our multi-agent architecture performed worse than the SPF and SA PPO planning strategies. Therefore, we hypothesise that better results could be observed for both SA PPO and MA PPO with an increased time constraint, as the learning curves had not yet reached saturation in our experiments. The reason that PPO was able to outperform other methods at all is mainly due to the fact that PPO has a built-in mechanism that prevents too large gradient updates with a clipped surrogate objective function. The reliability of training with MARL architectures has improved a bit compared to our previous contribution [10], because we were able to observe the start of convergence in the training process in \( \frac{3}{4} \) of the cases with both PPO architectures. Still, the lack of guarantee that every training attempt will be successful is one of the major problems that needs to be addressed in the future to eliminate the observed trade-off between models that learn very well and converge quickly and models that do not abstract any knowledge at all. Through a well-chosen form of discretisation and the selection of relevant variables for a suitable representation of the current state space, the simulation environment could be scaled up to the real scope of the IIoT Test Bed at HTW Dresden. Since the complexity of the problem increases exponentially with multiple stations, products and operations, this success is not predetermined and required many previous experiments and preliminary tests. In addition, the optimisation of hyperparameters is difficult to handle automatically, which makes access to the research area more difficult. A remaining problem for future work is also the reduction of the computational effort, either in an improvement of the simulation environment, so that the manufacturing process can be simulated with less computational resources, or an improvement of the RL mechanics, so that the RL models achieve a training success in less time. In addition, an in-depth evaluation should be conducted to analyse how the trained models react to a changed environment.

Acknowledgements

This research was funded as part of the project “Produktionssysteme mit Menschen und Technik als Team” (ProMenTaT, application number: 100649455) with funds from the European Social Fund Plus (ESF Plus) and from tax revenues based on the budget passed by the Saxon State Parliament.

References


Adaptive manufacturing: dynamic resource allocation using multi-agent reinforcement learning


[27] Bernd Waschneck, Thomas Altenmüller, Thomas Bauernhansl, and Andreas Kyek. Production scheduling in complex job shops from an industry 4.0 perspective: A review and challenges in the semiconductor industry, 2016.

