Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2021) 000–000

www.elsevier.com/locate/procedia

54th CIRP Conference on Manufacturing Systems

Simultaneous Production and AGV Scheduling using Multi-Agent Deep Reinforcement Learning

Jens Popper[a,*], Vassilios Yfantis[b], Martin Ruskowski[a,b]

[a]*German Research Centre for Artificial Intelligence, Trippstadter Straße 122, 67663 Kaiserslautern, Germany*

[b]*TU Kaiserslautern, Chair of Machine Tools and Control Systems, Gottlieb Daimler Straße 42, 67663 Kaiserslautern, Germany*

* Corresponding author. Tel.: +49-631-205753421; E-mail address: jens.popper@dfki.de

Abstract

Increasing demand for customized products in the wake of the 4th Industrial Revolution is placing ever increasing demands on the flexibility of manufacturing systems. Furthermore, the increasing usage of automated guided vehicles (AGV) adds another layer of flexibility and also complexity to the overall production system. The resulting Flexible Job Shop Scheduling Problem (FJSSP), including the coordination of the AGVs, is NP-hard and therefore hard to optimize. To address this problem, a Reinforcement Learning Multi Agent (MARL) system is proposed, in which job scheduling and vehicle planning is done cooperatively. This concept is described and prototypically implemented.

Keywords: Flexible Job Shop Scheduling, Deep Reinforcement Learning, Multi Agent System, Flexible Manufacturing System

1. Introduction

The continuing trend toward customized products described by Industry 4.0 confronts manufacturing companies with major challenges. To be able to meet customer demand, the principle of flexible production systems is becoming increasingly popular. These flexible production systems enable the production of a greater number of variants, without necessitating an increase in the number of available machines. This is particularly evident with the introduction of matrix production systems. Here, a great deal of emphasis is placed on coordinating the transportation system between manufacturing cells.

The planning of such complex production systems, as well as the increasingly important planning of autonomous transport systems, is a time-consuming and cost-intensive process. Usually these planning problems are NP-hard, i.e. they cannot be planned optimally in practice. To cope with this, heuristics and metaheuristics are often used, which forgo an exact solution of the planning problem in favor of faster computation times.

However, even heuristic and metaheuristic methods often pose problems in practice. For example, the planning problem, as well as the relationships of the elements within the planning problem, must be well known. Alternatively, these are simplified approximately. In addition, production scheduling and transportation planning are usually performed sequentially. Inclusion of the location of the transportation means is mostly omitted in planning and is a new trend in production planning research [6].

To address this problems, a Multiagent Reinforcement Learning (MARL) based method is created and prototypically implemented in a Unity3D environment in this contribution. In this method, machines with their (production) capacities, as well as transport units, are modeled as independent agents that communicate with each other. This implementation solves a series of planning instances, which are compared to common heuristics used in production.

2. State of the art and related work

In Deep Reinforcement Learning research, many breakthroughs have been achieved in recent years. Arguably, Deep-Q Learning has received the most attention due to the work of Deep Mind in the game Go [8]. Further developments in Deep Reinforcement Learning include the Trust Region Policy Optimization (TRPO) algorithm presented in [9]. This algorithm updates policies by taking the largest step possible to improve performance, while satisfying a special constraint on how close the new and old policies are allowed to be based on the KL-divergence. OpenAI introduced the Proximal Policy Optimization (PPO) algorithm in [2]. This algorithm introduces an adaptive KL-penalty and is especially characterized by its robustness to hyperparameters and to actions in the continuous domain. Also adapted to action spaces in the continuous domain is the Deep Deterministic Policy Gradient (DDPG) presented in [10]. This represents an actor-critic, model-free algorithm with a deterministic policy gradient.

Other well-known algorithms that are widely used in Deep Reinforcement learning and were considered in this research are the Soft Actor-Critic (SAC) [11], the Twin Delayed DDPG (TD3) [12] and Advantage Actor-Critic (A2C) [13].

The use of Deep Reinforcement Learning in multiagent systems is also an active field of research. For example, in [14], a mixed competitive and cooperative physics-based environment was presented in which agents compete in a simple game of hide-and-seek. Particularly noteworthy is the emergent behavior of the agents, which use the environment itself to their advantage.

The distributed solving of JSSP by the means of reinforcement learning has been shown in [15]. Here, the combination of shallow neural networks with a multiagent architecture – called neurodynamic programming – has been used to solve a generic JSSP. The authors highlight the ability of this approach to achieve a good solution far faster than algorithms such as branch and bound, which asymptotically needs more time when the problem grows.

MARL systems with Deep Reinforcement Learning have also been used in the area of production planning. [4] formulates a MARL approach for online reactive scheduling of flexible job shops. Here, unpredictable events, such as machine breakdowns and reconfigurations, are included. In this approach, one or multiple jobs are assigned to an agent, which directs them through the process flow. This point distinguishes this approach from many others in which agents are assigned to machines.

Such an approach was explored in [5], which also considers FJSP. The problems studied consist of jobs in different batch sizes and producing units, where each unit consists of a collection of identical machines. Moreover, to account for setup times, the machines within a production unit have different settings that can be changed with time. For scheduling of randomized lot sizes of a range of diverse jobs, [5] was able to train a MARL such that the results do not outperform heuristic methods, but correspond to expert knowledge.

In another publication, see [3], a MARL architecture is proposed to solve classical JSP. In this approach, agents decide which jobs to process next on their assigned machines based on the local state. Local sensing of the state, on the one hand, increases the difficulty of solving the problem optimally, and on the other hand leads to decentralized control. This is an approach that, based on the results, is also pursued in the methodology proposed in this research work.

[3] was able to show that this approach achieves reactive scheduling, where the agents adjust their behavior to a global goal. After the agents generalized their strategies, results were obtained that outperformed simple priority rules and, in some cases, more complex procedures.

The integration of other planning problems in a production system in the context of job scheduling was already identified as a research topic in [6], but there is still little practical work on this topic in the area of multiagent or machine learning research. The basic approach of reinforcement learning, that the explicit objective function does not have to be known a-priori, is used in the proposed method to address this open research question.

3. Proposed method

To solve the unknown objective function problem in this optimization problem, a MARL algorithm is proposed. This decentralized approach is intended to make the planning more resilient to disturbances. Two basic types of agents are defined. The first type of agent is assigned to production machines. These are able to request orders from a job registry in order to process them. Likewise, these agents have an overview of other agents in the production plant, including requested products, feasible production operations and the current progress in the production of the current product. In addition, they receive information about available transport units, their loading and current location.

The second type of agent is assigned to transport units. These transport the requested products to the respective machines. For this purpose, they observe all requested products, the machines, the current processing progress and the current position of all other transport units.

In addition to the agents assigned to the machines and transport units, there is a central coordination layer. This layer contains an overview of all orders and the production steps to be completed on them. The agents request information from this layer about all jobs that they can process according to the operations assigned to them. In doing so, the coordination layer ensures that the agents' decisions do not overlap due to messages arriving too late. The components of the scheduling system are shown in Figure 1. Here, jobs requested by the AGVs, i.e. the transport units, represent a subset of jobs that are requested by a machine agent. The internal time specification is process-oriented. This means that the system behavior is represented by processes, which can be understood

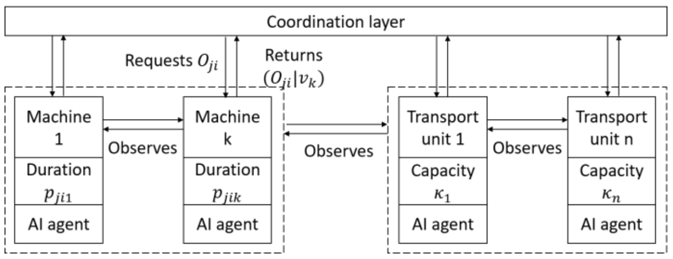

Figure 1 Proposed MARL decision architecture for integrated machine and transport unit scheduling.

as a sequence of events with the associated sequence of activities [7].

A major problem in using reinforcement learning for scheduling problems is the variable number of products to be manufactured. When modeling as an observation space, the number of orders is reflected in the size of the input vector. This can be chosen as large as possible to allow a wide range of possible planning problems. However, this also leads to a large number of possible actions, which in turn leads to poorer convergence and longer training times.

To circumvent this problem, the observation space is constantly limited to one possible order, and thus one possible action as output vector. For each machine, the agent iterates through all orders that this machine can process at this moment and assigns a continuous value v between [0,1] to each order. Once all possible orders have been evaluated, each order with the highest value is assigned to the corresponding machine. In case several machines have the same value, the order is assigned in sequence of the requesting machines. This avoids the problem of different sized order lists for a production system.

The AGVs have a different observation space. They observe their current distance to all machines, their current loading state, the destination of their current loading, the current location of all products, the products requested by machines and the current processing state of all machines. They can also observe the current destination of other transport systems. The transport systems select the order they will transport next. The local path finding itself is planned by means of an A* algorithm based on the plant topology.

In order to establish a reference between already made decisions, the previously made observation, including the made action, is fed to the agents as a stacked vector.

In the concept, the production machines as well as the transport units receive Deep Learning based reinforcement learning agents. Due to the actions in the continuous domain, a proximal policy optimization algorithm was chosen [1]. This is particularly characterized by its adaptive KL penalty, to control the change of the policy at each iteration. It is characterized by its objective function (1).

$$L_{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)] \quad (1)$$

Here, θ represents the policy parameter, $\hat{E}_t$ notes the empirical expected value over a time series, and r_t indicates the rate of probability under the respective new and old policies. $\hat{A}_t$ is the expected advantage at a time t and ε represents a hyperparameter to be adjusted.

4. Experiment setup

The experiments were conducted in a simulation environment created using Unity. The production units, in the form of 4 production cells, and 2 AGVs had to solve a randomly generated production planning problem. This plan always consisted of 13 jobs, which were composed of the job types 1-3. The theoretical minimum production time, including final transport to the outgoing warehouse, is 554 time units. This production time assumes that all products are immediately transported to the next machine or the buffer store, i.e. that transport capacities are unlimited. Furthermore, changeover times are set to a constant value. This theoretical value is not practically achievable due to the restrictions on transport capacities and compliance with all due dates, but it provides a starting point for evaluating the performance of the used heuristics and reinforcement learning techniques.

Target values of the evaluation are met deadlines and total completion time. In order to reduce the possible bias due to randomly generated orders, a test series of 100 runs is created. The average total completion time and the average negative latency are compared.

The optimization objectives according to which the agents' rewards were designed are listed in Table 1. At the end of a planning run, all agesnts receive a positive reward based on the achieved $max_j(C_j)/554$ and negative rewards for all products arriving after the deadline.

Table 1: Optimization objectives considered in the setup for the experiments of the MARL in the proposed method

Notation	Description	Meaning
C_{max}	$\max_j (C_j)$	Makespan/Max. completion time
$\bar{L}$	$\frac{1}{n}\sum L_j$	Mean Lateness
L_{max}	$\max (L_j)$	Max. Lateness

The agents consist of identically constructed networks with 5 deep layers and 512 completely connected neurons per layer. As learning rate $3 \cdot 10^{-4}$ was chosen. The agents were first trained sequentially over 8 million steps in isolation. In a second step the pre-trained networks were trained together in a multi-agent system for another 7 million steps. The training took 8 hours on the machine described in section 0.

4.1. Production plant setup

The problem can be described as a FJSSP. In a FJSSP an operation requires certain functions that one or more machines provide. That is, an operation can be processed on multiple machines if they meet the requirements. Thus, FJSSPs are a generalization of JSSPs, where the ith operation of job j O_{ji} can be assigned to a set of machines $M_{ji} \subseteq M$. Here, the process duration $p_{jik} \in \mathbb{R}^+$ depends on the machine $\mu_k \in M_{ji}$ on which the operation is processed.

In the test setup, there are $M = \{M_1, M_2, M_3, M_4\}$ machines to which the operations O_i, with process durations p_{jik} as listed in Table 2, are assigned.

There are 3 basic types of products that need to be manufactured in the test scenarios. These have different operations that must be performed on them. The final production on machine μ_4, which forms a bottleneck for comparison purposes, is always uniform.

In addition, changeover times are incurred during changeover (the switch between feasible operations O_{ji}). In this experiment, the changeover times were set to a constant value of 20.

Besides the machines, there are also transport units $T_n \subseteq \mathrm{T}$ with transport capacities $\kappa_n \in \mathbb{N}$. However, the transport times $\gamma_n(s_t) \in \mathbb{R}^+$ result dynamically from the current position s of the transport unit T_n at time t, and thus cannot be determined in advance. For the test setup, the transport units $T = (T_1, T_2)$ with capacities $\kappa_1, \kappa_2 = 1$ were chosen.

Table 2 Exemplary FSJP instance for three jobs in the experimentation setup

		Machine			
Job	Operation	M_1	M_2	M_3	M_4
J_1	$O_{1,1}$	10	15	-	-
	$O_{1,2}$	15	10	-	-
	$O_{1,3}$	-	-	20	-
	$O_{1,4}$	-	-	-	30
J_2	$O_{2,1}$	15	10	-	-
	$O_{2,2}$	-	-	20	-
	$O_{2,3}$	15	10	-	-
	$O_{2,4}$	-	-	-	30
J_3	$O_{3,1}$	10	15	-	-
	$O_{3,2}$	-	-	20	-
	$O_{3,3}$	10	15	-	-
	$O_{3,4}$	-	-	-	30

4.2. Technical setup

The described system was programmed in C# and Python. In the RL problem, the environment, i.e. the production plant and the transport system, is programmed in C# in Unity. An implementation of the PPO algorithm is used as the learning algorithm. Academy was used as the basis of the agent's communication with the environment in Unity based on the work described in [2]. Through this, the observations, actions

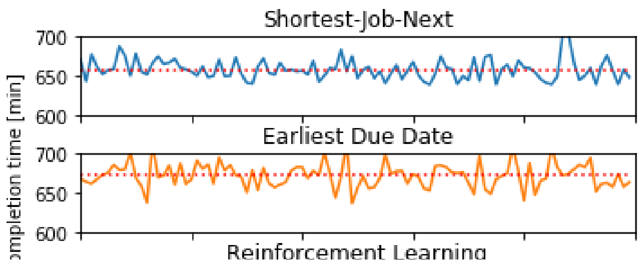

Figure 3 Visualization of the experimentation results for the agent pairs SJN [$M^{SJN} \times T^{FCFS}$] in blue, EDD[$M^{EDD} \times T^{FCFS}$] in orange and RL[$M^{RL} \times T^{RL}$] with their respective mean total makespan $\bar{C}_{max}$ in red.

and rewards are passed between the learning algorithm and the environment. The hardware used was a desktop PC with an AMD Ryzen 3600 and an Nvidia RTX2070.

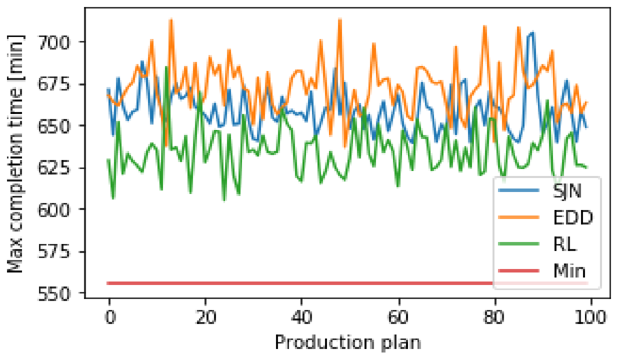

Figure 2 Overall view of the experimentation results for the agent pairs SJN [$M^{SJN} \times T^{FCFS}$] in blue, EDD[$M^{EDD} \times T^{FCFS}$] in orange and RL[$M^{RL} \times T^{RL}$]. The theoretical minimum Min is added in red.

5. Experiment results and discussion

5.1. Experiment results

As a reference to the multiagent system, according to the two target variables of the evaluation, the Shortest-Job-Next (SJN) and Earliest Due Date (EDD) heuristics are taken for the machines. For the transport units, a First Come First Serve (FCFS) heuristic and an RL agent were taken. This is noted as $\mathrm{M}^{\text{Decision rule}}$ for the machines and $\mathrm{T}^{\text{Decision rule}}$ for the transport units. The setup for the experiments are $M^{SJN} \times T^{FCFS}$, $M^{EDD} \times T^{FCFS}$, $M^{RL} \times T^{RL}$. Results of the experiments are shown in Figure 2. Also integrated in the diagram is the theoretical minimum of 554, if leaving out transport capacities.

It can be seen that the RL based agent system performs up to 100 time units better than the heuristics. The average values of the different decision rules are shown in Figure 3.

In a second experiment, all decision algorithms were cross paired with another agent for the transport units, respectively the combinations $M^{SJN} \times T^{RL}$, $M^{EDD} \times T^{RL}$, $M^{RL} \times T^{FCFS}$.While it did not change the results for M^{EDD} and M^{SJN} by a significant amount, the RL based machine realized longer completion times, as shown in Figure 4.

Table 3 shows an overview of all measured targets for the described experiments.

Table 3: Overall experimentation results for the agent pairs $M^{SJN} \times T^{FCFS}$, $M^{EDD} \times T^{FCFS}$, $M^{RL} \times T^{RL}$, $M^{RL} \times T^{FCFS}$ with the achieved mean total makespan $\bar{C}_{max}$, mean lateness $\bar{\mathrm{L}}$ and maximum lateness $\mathrm{L}_{\max}$.

Decision rule	$\bar{C}_{max}$	$\bar{L}$	L_{max}
$M^{SJN} \times T^{FCFS}$	658,2	5,24	25,19
$M^{EDD} \times T^{FCFS}$	672,09	0	0
$M^{RL} \times T^{FCFS}$	658	0	0
$M^{RL} \times T^{RL}$	634,03	0	0

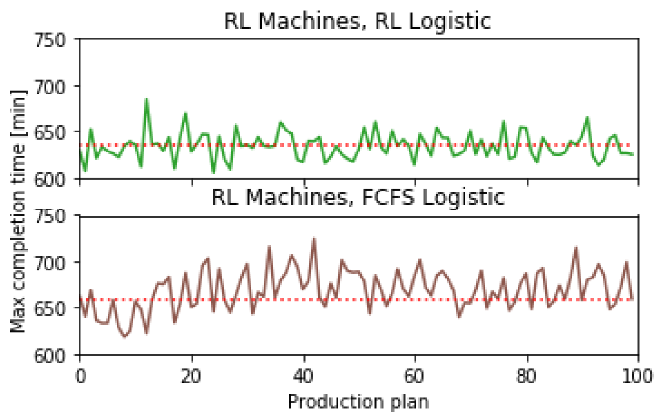

Figure 4 Experimentation results for the agent pairs $M^{RL} \times T^{RL}$ in green $M^{RL} \times T^{FCFS}$ in brown with their respective mean total makespan $\bar{C}_{\max}$ in red.

Over the 100 test planning instances, it was found that the MARL system consisting of agents for the machines and the transport units can achieve shorter total completion times than other heuristics and can match the EDD heuristic in terms of lateness. It is particularly interesting that the combination of a RL controlled machines in the multiagent system in combination with RL controlled transport units achieves better results than a combination with a FCFS controlled transport system. The heuristics EDD and SJN, on the other hand, did not perform significantly worse or better in this case.

5.2. Discussion

The results achieved are well above the minimum of 554 time units for all the processes used. However, since this value does not take into account constraints such as transport capacity limits and the physical location of the transport units, the limiting factor here is the coordination between the machines and the logistics. Here, a simultaneous planning of the machine scheduling and the transport units could show that such a multi-agent system, based on deep reinforcement learning, is able to optimize such problems under multiple objectives. In terms of total completion time, the system was even able to undercut heuristics specifically designed for this purpose. It is particularly interesting that the RL logistics system achieves better results than an FCFS heuristic only when paired with an RL controlled machine.

These results indicate that simultaneous planning of the machines and transport units offers further potential for other optimization criteria due to the interactions between them.

The simple modeling of an optimization problem with such an approach is also interesting for practice. No formalized relationships between the components of a production system have to be derived. Thus, an explicit objective function does not have to be formulated. Only restrictions, like deadlines, and optimization variables, like the minimization of the completion time, must be manually set up. However, due to the implementation, this can be done by non-experts. Furthermore, additional event-driven optimization targets, like minimizing collisions between transport units, can be included. The main question here is the design of the positive and negative rewards.

6. Summary and conclusion

In this paper, a concept for simultaneous machine job scheduling with transport planning in a flexible job shop using a MARL algorithm was presented. The results obtained with it showed that MARL systems are able to optimize scheduling problems considering multiple objectives. In particular, the actual inclusion of the location of transport units, as it occurs with autonomous transport units, increases the transferability of such generated problems to real production systems.

Future research will investigate how the integration of further planning areas affects the quality of the planning. The goal is thus a holistically planned production plant. Furthermore, the MARL system will be examined for reactive aspects, such as the failure of a machine. It is interesting to consider whether such systems reflect patterns known from lean manufacturing or whether new patterns in the scheduling of production systems are emerging.

Acknowledgements

This work has been partially funded by the European Commission through the H2020 ICT-38 Project MAS4AI - Multi-Agent Systems for Pervasive Artificial Intelligence for assisting Humans in Modular Production (No. 957204).

References

[1] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal Policy Optimization Algorithms. 2018, available online: https://arxiv.org/abs/1707.06347, last visited 23.12.2020.

[2] Juliani A, Berges VP, Teng E, Cohen A, Harper J, Elion C, Goy C, Gao Y, Henry H, Mattar M, Lange D. Unity: A General Platform for Intelligent Agents. 2018, available online: https://arxiv.org/abs/1809.02627, last visited 18.11.2020.

[3] Gabel T, Riedmiller M. Scaling adaptive agent-based reactive job-shop scheduling to large-scale problems. 2007 IEEE Symposium on Computational Intelligence in Scheduling, pp. 259–266.

[4] Baer S, Bakakeu J, Meyes R, Meisen T. Multi-agent reinforcement learning for job shop scheduling in flexible manufacturing systems. 2019 IEEE Second International Conference on Artificial Intelligence for Industries (AI4I), pp. 22–25.

[5] Waschneck B, Reichstaller A, Belzner L, Altenmüller T, Bauernhansl T, Knapp A, Kyek A. Optimization of global production scheduling with deep reinforcement learning.2018 Procedia CIRP, vol. 72, pp. 1264–1269.

[6] Li X, Xie J, Peng, K, Li H, Gao L.. Review on flexible job shop scheduling. 2019 IET Collaborative Intelligent Manufacturing. 1. 10.1049/iet-cim.2018.0009.

[7] VDI 3633. Simulation von Logistik-,Materialfluss- und Produktionssystemen - Begriffe. 2018.

[8] Silver D, Huang A, Maddison CJ, 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 TP, Leach M, Kavukcuoglu K, Graepel T, Hassabis D. Mastering the game of Go with deep neural networks and tree search. 2016 Nature, 529 (7587), pp. 484–489.

[9] Schulman J, Levine S, Moritz M, Jordan M, Abbeel P. Trust region policy optimization. 2015 Proceedings of the 32nd International Conference on

International Conference on Machine Learning - Volume 37 (ICML'15). JMLR.org, 1889–1897..

[10] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continous Control with Deep Reinforcement learning. 2015 Available online: https://arxiv.org/abs/1509.02971. Last visited: 05.01.2021.

[11] Haarnoja, T., Zhou, A., Abbeel, P. &amp; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. 2018 Proceedings of the 35th International Conference on Machine Learning, 80:1861-1870.

[12] Fujimoto S, van Hoof H, Meger D. Addressing Function Approximation Error in Actor-Critic Methods. 2018 Proceedings of the 35th International Conference on Machine learning, 80:1587:1596.

[13] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap TP, Harley T, Silver D, Kavukcuoglu K. Asynchronous Methods for Deep Reinforcement Learning. 2016 Proceedings of the 33rd International Conference on Machine Learning, 48:1928-1937..

[14] Baker B, Kanitscheider I, Markov T, Wu Y, Powell G, McGrew B, Mordatch I. Emergent Tool Use From Multi-Agent Autocurricula. 2020. Available online: https://arxiv.org/abs/1909.07528, last visited: 05.01.2021.

[15] Monostori L., Csáji B.Cs., Kádár B., Adaptation and Learning in Distributed Production Control, CIRP Annals, Volume 53, Issue 1, 2004, Pages 349-352, ISSN 0007-8506, https://doi.org/10.1016/S0007-8506(07)60714-8.