Contents lists available at SciVerse ScienceDirect

CIRP Annals Manufacturing Technology

Journal homepage: www.elsevier.com/locate/cirp

Integrated process-system modelling and control through graph neural network and reinforcement learning

Jing Huang[a], Jianjing Zhang[b], Qing Chang[a, c, *], Robert X. Gao (1)[b]

[a] Department of Mechanical and Aerospace Engineering, University of Virginia, Charlottesville, VA, United States
[b] Department of Mechanical and Aerospace Engineering, Case Western Reserve University, Cleveland, OH, United States
[c] Department of Engineering Systems and Environment, University of Virginia, Charlottesville, VA, United States

Modern manufacturing systems are becoming increasingly complex, dynamic, and connected, and their performance is being affected by not only their constituent processes but also their system-level interactions. This paper presents an integrated modelling method based on a graph neural network (GNN) and multi-agent reinforcement learning (MARL) collaborative control for adjusting individual machining process parameters in response to system- and process-level conditions. The structural and operational dependencies among process machines are captured with a GNN. Iteratively trained with MARL, machines learn to adaptively control local process parameters, e.g., machining speed and depth of cut, while achieving the global goal of improving production yield.

Multi-level modelling, Process control, Process-system integration

1. Introduction

As manufacturing systems become increasingly complicated and interconnected, challenges arise for throughput maximization, quality assurance, and cost reduction [1-2]. Despite extensive research on process- and system-level analysis, the integration of these two aspects for global operation optimization continues to attract researchers' interest [3]. Integration of processes and systems is imperative to the overall system performance: 1) from the process perspective, local parameter adjustment without considering the system may lead to over- or under-production that overflows or depletes buffers and limits throughput; 2) from the system perspective, speeding up certain processes without ensuring that process constraints are met (e.g., part quality) may diminish the production yield. Therefore, the optimal process control strategy is deeply coupled with system-level conditions and vice versa. Seamlessly integrated process-system modelling and a holistic control scheme would significantly enhance the performance in both product quality and system productivity.

However, effective integration of processes and systems is quite challenging. Current modelling methods for manufacturing systems are mostly dissociated from the underlying processes. A process model usually considers individual process parameters to optimize the process. System-level analysis mainly addresses time-stamped material flow and current methods mainly focus on system throughput evaluation in the steady state (e.g., Markov Chain based methods [4]) or real-time production loss evaluation (e.g., recursive algorithm-based model [5]). As such, the analysis of production quantity at the system level usually ignores the impact of process-level activities on system dynamics. With the current approach of separately analyzing processes and systems, it is fundamentally difficult for multi-stage manufacturing to achieve a real-time coordinated and optimized operation.

With the increasing availability of manufacturing data enabled by extensive sensor deployment [6], learning-based methods for manufacturing process and system control have attracted interest as they provide a viable path towards process-system integration by capturing the dynamics and interdependencies embedded in the data [7-8]. Graph learning methods, as represented by GNN, have demonstrated a capability in modelling the interconnections among system elements by representing the system as a graph consisting of nodes and links. This topology allows information exchange among the connected elements [7] and achieves two-level integrated modelling so that each node is aware of both its local condition and the status at the global level.

In addition, reinforcement learning (RL), which aims to find an optimal policy through interactions with the environment [8], has opened up a new research avenue of process and system optimization without a rigid rulebook. Extensive RL-based research has been reported in the literature of manufacturing, for process optimization (e.g., deep drawing [9] and machining [10]), deficient stage identification [11], and maintenance scheduling [12]. However, due to the lack of integrated process-system modelling, there is a notable separation between process-level [9-10] and system-level [11-12] RL applications. RL for process-system integrated control is more challenging than single-level problems due to the dramatically expanded state/action space and more complicated process and system dynamics.

Motivated by these prior efforts, this paper presents a process-system integrated modelling based on GNN by representing the manufacturing system as a graph. Built on GNN modelling, a distributed adaptive control scheme is established based on the MARL paradigm in order to adjust individual process parameters and maximize system yield. The goal of this paper is to bridge the existing knowledge gap in process-system integration. Specifically, relevant machines are modelled as nodes in the GNN with their interdependencies modelled as links. Each machine in the graph is then represented by an MARL agent that makes decisions based on the information from both the process and system. Evaluation through simulations for a multi-stage grinding operation has shown that GNN-MARL framework is capable of adaptively optimizing local process parameters, e.g., machining speed and depth of cut, to achieve global maximization of the system yield.

2. Problem description and formulation

For manufacturing systems, multiple process stages are typically needed to produce a part. As shown in Fig. 1, which schematically

illustrates a simplified camshaft production line, machines process products with given process parameters and buffers temporarily hold intermediate parts that await further processing.

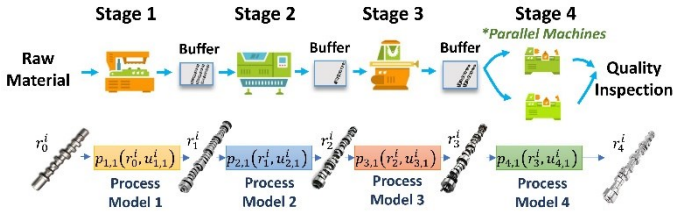

Fig. 1. Manufacturing system structure and underlying process models.

At the process level, process models often quantify the relationship between quality performance of manufacturing processes and input parameters. Such models can be used to optimize the process for improved efficiency and product quality. Let r^i denote a series of key features that characterize the quality of product i, then the process model for machine n at stage m can be represented as a function $p_{m,n}(*)$:

$$r_m^i = p_{m,n}\left(r_{m-1}^i, u_{m,n}^i\right) \tag{1}$$

where r_{m-1}^i and r_m^i are the product's key features before and after stage m respectively, and $u_{m,n}^i$ denotes the process control parameters for processing product i. For example, in a grinding process, key features r^i may refer to product geometry or surface roughness, while process parameters $u_{m,n}^i$ may include depth of cut and machining speed. Various physics models and machine learning methods have been developed to obtain $p_{m,n}(*)$ [13-14]. Finally, the quality inspection is the procedure that compares key features r_m^i of the completed product against quality standard r^* in order to determine if the product is defective or compliant.

At the system level, current throughput analysis mainly concerns aggregated parameters from the process such as average cycle time. Machine downtimes and limited inline buffers lead to complicated stochastic dynamics and nonlinear interactions within the system. For example, a machine is starved and must idle when its upstream buffers are empty due to upstream machines' inefficiencies. Let TP_{sys} denote the system throughput, then:

$$TP_{sys} = g\left(\boldsymbol{\Sigma}_{\mathbf{proc}}, \boldsymbol{\Sigma}_{\mathbf{sys}}\right) \tag{2}$$

where $\boldsymbol{\Sigma}_{\mathbf{proc}}$ are the aggregated parameters from processes, $\boldsymbol{\Sigma}_{\mathbf{sys}}$ are system-level architecture and parameters. In general, there is no closed-form representation for $g(*)$, and aggregation [4] or recursive algorithms [5] have been used to evaluate throughput.

Notably, Eq. (1) and Eq. (2) are not functionally connected. This indicates that process optimization based on process-level models and system-level throughput improvement based on system quantitative analysis are mostly separated in current methods. A key question that arises is*: if the parameter (e.g., depth of cut) of an intermediate machining process changed, how will it impact the overall system yield? Or to maximize the yield of the system, how should the process parameters be adjusted*? To formulate the research problem, two quantities that relate to the performance of both processes and system are denoted in this work:

- **Yield** y: number of compliant products among TP_{sys}.
- **Defect** d: number of defective products among TP_{sys}.

Accordingly, the problem studied in this paper is then presented as follows:

Given the manufacturing system as described, establish an integrated process-system modelling approach, and build an automated control scheme to find optimal adaptive policies for each machine to adjust process parameters for each product with the aim of maximizing the system yield, i.e.,

$$u_{m,n}^i(t) = \arg\max_{u_{m,n}^i(t)} \{y(T)\}, t \in [0,T], subject\ to: \boldsymbol{C} \tag{3}$$

where $y(T) = f\left(\left\{r_m^i, u_{m,n}^i \middle| m \in \boldsymbol{M}, n \in \boldsymbol{N}, i \in \boldsymbol{Z}^+\right\}, \boldsymbol{E}, \boldsymbol{\Sigma}_{\mathbf{sys}}\right)$ *is the total system yield during a given time horizon* T, $\boldsymbol{E}$ *is set of random downtime events during time horizon* T. *All the constraints are denoted by* $\boldsymbol{C}$, *which are defined by Eqs. (1) and (2).*

3. Process-system integration through GNN

To handle the complexity involved in optimizing Eq. (3), a GNN-based integrated modelling is proposed to connect the process model with the flow-based system model. As shown in Fig. 2, the manufacturing system is represented as a graph, where machines are treated as nodes and material flows as links. The material flow showing possible routes of products also describes how machines may interact with each other.

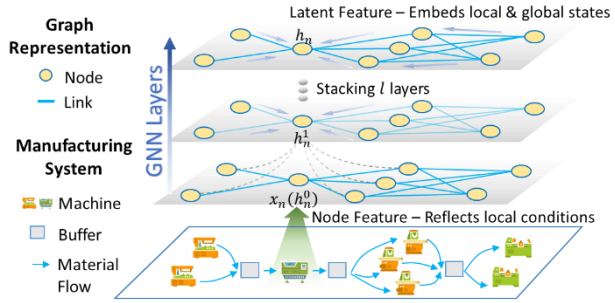

Fig. 2. Mapping a manufacturing system to a graph and extracting latent features with GNN.

In the manufacturing system graph, the node feature x_n includes local conditions for machine n, interaction between machines as represented by buffer conditions, and process model output in order to enable a direct connection between process outputs and system. Specifically, it is defined as:

$$x_n = [bl_n, bv_n, w_n, \alpha_n, r_n, u_n, m, n] \tag{4}$$

The physical meanings of the components are described as follows:

- bl_n and bv_n describe the buffer level/vacancy of machine's immediate upstream/downstream buffer, which are closely related to potential blockages or starvations.
- w_n and α_n indicate the local operation status of machine n, where w_n is machine's up/down status, and α_n is machine's processing progress on its current product.
- r_n and u_n depict the process model output conditions of product, where r_n denote the key features of the current product and u_n is the currently applied process parameters.
- m and n are included in the feature to uniquely identify machine's stage and numbering respectively for the ease of parameter sharing in neural networks.

In this work, we use a GNN to encode machine nodes and obtain each node's latent feature that reflects both the local condition of the machine and global status of the whole system. Node encoding by GNN, specifically graph convolutional network (GCN) [7] used in this work, is carried out by layers (see Fig. 2). For each layer, a machine node pulls information from its first-order neighbouring machines according to the following function:

$$h_n^{l+1} = \sigma\left(\sum_{j \in (n' \cup n)} \alpha_{nj}^l h_j^l W^l\right) \tag{5}$$

where σ is the activation function, α_{nj}^l is the learnable relation weight that determines the weight of information that machine n pulls from its neighbor machine n', W^l is learnable parameters for

layer l. Starting from local node feature $h_n^0 = x_n$, by stacking l layers, the machine node can aggregate information from the machines that are l-hops away and thereby, encoding the system dynamics in its latent feature h_n. A GNN therefore bridges the gap between process- and system-level conditions and lays the foundation for adaptive process control.

4. Distributed adaptive control through MARL

Since the online control problem in Eq. (3) is NP-hard involving complex dynamics [15], a distributed adaptive control scheme based on MARL is established. Based on the GNN-integrated modelling, each agent (i.e., each machine) in MARL makes independent but informed local process control decisions to achieve the global maximization of yield in a collaborative manner.

4.1. MARL problem formulation

To fit the control problem in Eq. (3) into the MARL framework, each machine is modelled as a distributed agent and has an independent adaptive control policy $\pi^n(u_n|h_n)$ that conditions on machine node's latent feature h_n. In other words, the latent feature h_n serves as the state/observation in MARL, and therefore machine agent could make control decisions that are adaptive to not only machine local conditions but also global system status.

Besides the state/observation, the definitions of action and reward function are also indispensable in formulating the MARL problem. The definition of action u_n for each machine depends on the specific manufacturing process. For example, the process control parameters in grinding often include machining speed and depth of cut. Then, the available actions for a grinding machine are all possible combinations of the two process parameters. The machine agent is triggered to select an action, i.e., a set of process parameters, whenever loading a product from its upstream buffer.

The reward function r_t is defined to reflect the "goodness" of the selected actions towards the collaborative common benefit. In this work, the global reward function at time step t is defined as:

$$r_t = -d_t + y_t \tag{6}$$

where d_t is the stepwise defect and y_t is the stepwise yield. Note that if the costs/profits associated with defective/compliant products are available through additional cost analysis, which is beyond the scope of this presented study, Eq. (6) can be redefined to pursue maximum profits through MARL.

4.2. MARL training paradigm and algorithm

The formulated MARL problem relies on effective iterative training to learn the optimal control policies $\pi^n(u_n|h_n)$ for each machine. In MARL, the major challenge lies in quantifying an individual agent's contribution to the system performance, which is the key to improving the agents' policies. There have been a number of relevant studies in this field. In this work, we investigate the C-COMA algorithm in [16] due to its compatibility with GNN.

C-COMA algorithm employs the Advantage Actor Critic (A2C) framework in a distributed setting by designing a tailored advantage function $A^n(s, \boldsymbol{u})$ for each agent, expressed as:

$$A^n(s, \boldsymbol{u}) = Q(s, \boldsymbol{u}) - \sum_{u_n'} \pi^n(u_n'|h_n) Q\left(s, (\boldsymbol{u}^{-u_n}, u_n')\right) \tag{7}$$

where $Q(s, \boldsymbol{u})$ is the state-action value for agents' joint action $\boldsymbol{u} = [u_1, \dots, u_n, \dots]$ in state s, $\pi^n(u_n'|h_n)$ is agent n's current policy, and $Q\left(s, (\boldsymbol{u}^{-u_n}, u_n')\right)$ is the state-action value by replacing agent n's current action u_n in $\boldsymbol{u}$ with u_n' while holding other agents' actions fixed. The second term on the right-hand side of Eq. (7) is a counterfactual state-action value for agent n by marginalizing its action over its current policy. Therefore, $A^n(s, \boldsymbol{u})$ evaluates how good an agent's current action is compared to "average", which offers a customized baseline for the agent to reason about the contribution of its current action to the global performance. Based on the advantage function, the policy gradient g_θ is obtained as:

$$g_\theta = E_\pi \left[\sum_N \nabla_\theta \log \pi(u_n|h_n) A^n(s, \boldsymbol{u}) \right] \tag{8}$$

where θ denotes all the parameters in GNN and agent's policy network. Therefore, the learnable parameters in GNN layers and MARL policy network are optimized simultaneously with gradient g_θ. More details on C-COMA algorithm can be found in [16].

5. Prototype implementation

The GNN integrated modelling and MARL-based adaptive control scheme (see Fig. 3) is implemented on a sequence of four grinding processes in simulated camshaft production, as shown in Fig. 1.

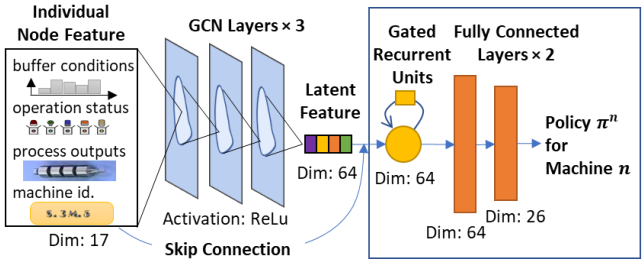

Fig. 3. GNN-MARL neural network architecture.

The system is a four-stage grinding line. Stage 1~3 have one machine each, and stage 4 has two parallel machines. Machines' random downtimes are obtained from sensor data. In simulation, we assume the time between downtime event arrivals and duration of downtime events follow exponential distributions. Buffers are placed between every two stages and all of them have a limited capacity of ten. The time horizon considered in this case is a ten-hour shift, i.e., 600 minutes.

The key feature of the product is characterized by the surface roughness of the four sequential grinding processes, i.e., $r^i = [r_1^i, r_2^i, r_3^i, r_4^i]$. The final product is deemed compliant if $\sum_{m=1}^4 r_m^i < 5.0$ μm, or defective otherwise. We adopt a simplified grinding process model [13] to relate the surface roughness to process parameters such as machining speed v_n and depth of cut a_n. Accordingly, the MARL agent's action is $u_n = \{[\mathrm{v_n}, \mathrm{a_n}]\}$, where $v_n \in \{0.30, 0.35, 0.40, 0.45, 0.50\}$ m/s, and $a_n \in \{1.20, 1.35, 1.50, 1.65, 1.80\} \times 10^{-5}$ m. The surface roughness and cycle time for the given process parameters at each stage are listed in Table 1. The resulting surface roughness at each stage follows a normal distribution with its mean and variance depending on the control parameters. The "reciprocal" relationship in terms of parameters between the surface roughness and cycle time requires the MARL to find the balance between quality and throughput in order to maximize the yield of production.

Table 1
Grinding process model and grinding cycle time given process parameters.

Stage	Surface roughness (μm)	Cycle time (min)
1	$r_1^i \sim N\left(\left(\frac{v_1 a_1}{30}\right)^{0.90}, 5.48 v_1^2 a_1^{1.40}\right) \times 10^6$	$T_1^i = \frac{1.0 \times 10^{-5}}{v_1 a_1}$
2	$r_2^i \sim N\left(\left(\frac{v_2 a_2}{30}\right)^{0.85}, 5.48 v_2^2 a_2^{1.35}\right) \times 10^6$	$T_2^i = \frac{1.0 \times 10^{-5}}{v_2 a_2}$
3	$r_3^i \sim N\left(\left(\frac{v_3 a_3}{30}\right)^{0.90}, 5.48 v_3^2 a_3^{1.50}\right) \times 10^6$	$T_3^i = \frac{0.5 \times 10^{-5}}{v_3 a_3}$
4	$r_4^i \sim N\left(\left(\frac{v_4 a_4}{30}\right)^{0.85}, 5.48 v_4^2 a_4^{1.30}\right) \times 10^6$	$T_4^i = \frac{3.0 \times 10^{-5}}{v_4 a_4}$

6. Results and discussion

6.1. Convergence analysis

In Fig. 4, the mean and standard deviation of the test returns (i.e., accumulated rewards) are plotted against training steps (left plot). Progress continues until a leap at around the four millionth steps. The algorithm convergence is confirmed after ten million steps.

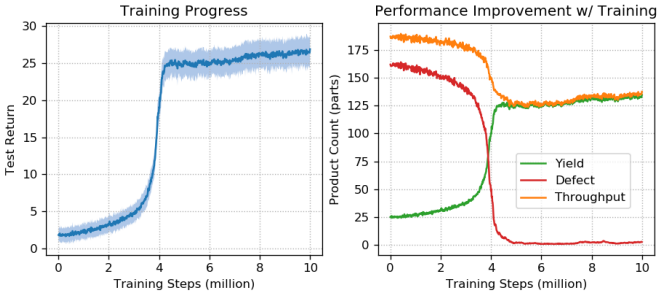

Fig. 4. GNN-MARL training progress and performance improvement.

Fig. 4 also presents the breakdowns of the system throughput during training (right plot). It shows that the system outputs a large number of products with low yield when the training begins, due to the fact that initial policies are largely random. As the training continues, agents learn to adjust the parameters and collaborate with each other to gradually reduce defect while improving yield. The similar trend between the yield and return curves confirms the effectiveness of the reward setting (Eq. (6)).

6.2. Performance evaluation and comparison

To further evaluate the effectiveness of the developed method, we compared the performance of the GNN-MARL policy with two common industrial baselines. In Baseline 1, process parameters for each stage are tuned independently to keep the surface roughness of each stage under 1.25 μm with a 99% probability (i.e., three standard deviations from the mean in normal distribution) to ensure that the final product meet the quality standard. Process parameters remain unchanged during the whole production. In Baseline 2, a local controller is adopted to adjust the processing speed on the basis of the parameters obtained in Baseline 1 according to only individual machines' local features.

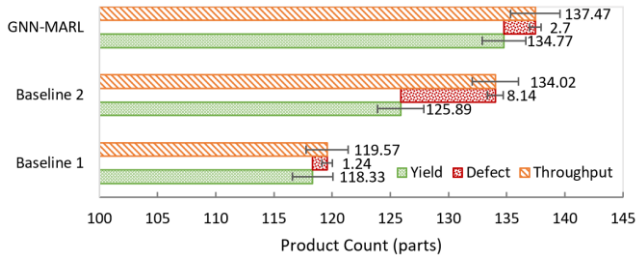

Fig. 5. GNN-MARL policy compared with industrial baselines. Each bar shows the mean and 95% confidence interval.

The system performance of the three scenarios is shown in Fig. 5. Baseline 1 has the lowest defect rate since its process parameter setting statistically guarantees a high ratio of compliant product. However, Baseline 1 fixes process parameters and hence lacks adaptions to the dynamically changing system conditions. Given its convenience and stability, this strategy has been widely adopted by the industry, despite its conservative nature.

Baseline 2 employs heuristics based on local machine conditions to change the processing speed. Although it leads to higher yield and throughput compared to Baseline 1, it results in many more defective products. This demonstrates that such local process optimization without considering the overall system dynamics is not sufficient for optimizing system-level performance.

In comparison, the GNN-MARL algorithm is shown to achieve true process-system integration through GNN and effective learning of adaptive policies by MARL. It has not only achieved the highest yield, but also a 98.04% ratio of compliant product (i.e., yield/throughput), which is much better than that of Baseline 2. This demonstrates the effectiveness of GNN-MARL scheme.

7. Conclusion

An integrated method for manufacturing processes and system modelling and distributed adaptive control are developed for a multi-stage manufacturing system, based on a GNN and MARL. The GNN encodes complex system dynamics in machine nodes by aggregating real-time information from the neighbouring machines. MARL models each machine as individual agent and learns adaptive process parameter control policy to cooperatively achieve the goal of global maximization of system yield. Evaluated in a case study of simulated camshaft production, the developed GNN-MARL has achieved significant improvement on both product quality and system productivity as compared to baselines presently adopted by the industry. Future research will focus on understanding the effects of reward settings as well as the incorporation of real-time tool condition diagnosis and remaining useful life prognosis on the properties of the control policies.

Acknowledgement

This work is supported by the National Science Foundation under CMMI-1853454. The authors sincerely appreciate help from Mr. Jianyu Su from the University of Virginia.

References

[1] Monostori L, Kádár B, Bauernhansl T, Kondoh S, Kumara S, Reinhart G, Sauer O, Schuh G, Sihn W, Ueda K (2016) Cyber-physical systems in manufacturing. CIRP Annals 65(2):621-641.

[2] Koren Y, Hu SJ, Weber TW (1998) Impact of manufacturing system configuration on performance. CIRP Annals 47(1):369-372.

[3] Colledani, M., Tolio, T., Fischer, A., Iung, B., Lanza, G., Schmitt, R., & Váncza, J. (2014). Design and management of manufacturing systems for production quality. CIRP Annals 63(2):773-796.

[4] Bai Y, Tu J, Yang M, Zhang L, Denno P (2020) A new aggregation algorithm for performance metric calculation in serial production lines with exponential machines: design, accuracy and robustness. International Journal of Production Research:1-18.

[5] Zou J, Chang Q, Arinez J, Xiao G, Lei Y (2017) Dynamic production system diagnosis and prognosis using model-based data-driven method. Expert Systems with Applications 80:200-209.

[6] Gao R, Wang L, Helu M, Teti R (2020) Big data analytics for smart factories of the future. CIRP Annals 69(2):668-692.

[7] Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907.

[8] Sutton R, Barto AG (2018) Reinforcement learning: An introduction. MIT press.

[9] Dornheim J, Link N, Gumbsch P (2020) Model-free adaptive optimal control of episodic fixed-horizon manufacturing processes using reinforcement learning. International Journal of Control, Automation and Systems 18(6):1593-1604.

[10] Xiao Q, Li C, Tang Y, Li L (2019) Meta-reinforcement learning of machining parameters for energy-efficient process control of flexible turning operations. IEEE Transactions on Automation Science and Engineering 33:1-21.

[11] Epureanu BI, Li X, Nassehi A, Koren Y (2020) Self-repair of smart manufacturing systems by deep reinforcement learning. CIRP Annals 69(1):421-424.

[12] Huang J, Chang Q, Arinez J (2020) Deep reinforcement learning based preventive maintenance policy for serial production lines. Expert Systems with Applications 160:113701.

[13] Chiu N, Malkin S (1993) Computer simulation for cylindrical plunge grinding. CIRP Annals 42(1):383-387.

[14] Zhang J, Wang P, Gao R (2020) Attention mechanism-incorporated deep learning for AM part quality prediction. Procedia CIRP 93:96-101.

[15] Arinez J, Chang Q, Gao R, Xu C, Zhang J (2020) Artificial intelligence in advanced manufacturing: current status and future outlook. Journal of Manufacturing Science and Engineering 142:111003

[16] Su J, Adams S, Beling A (2020) Counterfactual multi-agent reinforcement learning with graph convolution communication. arXiv:2004.00470.