Using Decision Trees and Reinforcement Learning for the Dynamic Adjustment of Composite Sequencing Rules in a Flexible Manufacturing System

Thomas Voß*, Jens Heger, Mazhar Zein El Abdine

Leuphana University Lüneburg, Inst. of Product & Process Innovation, Universitätsallee 1, 21335 Lüneburg, Germany; *voss@leuphana.de

SNE 32(3), 2022, 169-175, DOI: 10.11128/sne.32.tn.10617
Selected ASIM SPL 2021 Postconf. Publication: 2022-01-27;
Received English version: 2022-05-29; Accepted: 2022-06-03
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Integrating machine learning methods into the scheduling process to adjust priority rules dynamically can improve the performance of manufacturing systems. In this paper, three methods for adjusting the k-values of the ATCS sequencing rule are analyzed: neural networks, decision trees and reinforcement learning. They are evaluated in a static and a dynamic scenario. The required dataset was synthetically generated using a discrete event simulation of a flow shop environment, where product mix and system utilization were varied systematically. Across all scenarios, it is shown that all three methods can improve the performance. On par, RL and NN can reduce the mean tardiness by up to 15% and compensate for unplanned product mix changes.

Introduction

Finding a good sequence of operations on a machine can be difficult under changing conditions, such as machine failure. Since the use of centralized and static solution methods is not suitable in complex and uncertain scenarios, decentralized sequencing rules are a viable option. These rules use locally available information for fast decision making. However, no rule exists that outperforms all others under varying system performance.

For this reason, a hyperheuristic is developed to dynamically select and adjust weighting values of a composite sequencing rule, selecting the next job to be processed based on the system state. Based on a variety of training scenarios considering several dynamic influences, such as stochastically distributed arrival times or changing proportions of product families in the product mix, the benefits of dynamically adjusting the k-factors of the rule is presented.

To estimate the performance of the system based on the current state, different machine learning models have provided very good results depending on the selection of the weights of the composite rule (Heger 2014; Mouelhi-Chibani and Pierreval 2010; Shiue et al. 2018). When using these methods, however, there is not only the question of the amount of training points, but also the aspect of transferability of the acquired knowledge in new scenarios and the suitability, generalizability and traceability of the methods used (Priore et al. 2006; Priore et al. 2018; Usuga Cadavid et al. 2020).

The knowledge and understanding of actions and decisions taken during the process is crucial and is increasingly preferred as opposed to simple prediction and black box optimization (Nunes and Jannach 2017; Rehse et al. 2019). At this point, the usage of hyperheuristics for the selection and adjustment of different sequencing rules in combination with comprehensible learning methods (e.g. decision trees) can prove useful. This contribution elaborates on the usage of three different methods to dynamically adjust the behavior with regards to performance and comprehensibility.

1 State-of-the-Art

Due to their ease of comprehension and very short computation time, the use of priority rules for sequencing, i.e. selecting the next job to be processed by the machines, is very popular in the industry. It should be noted that more than 100 rules are known, which perform differently depending on the scenario (Panwalkar and Iskander 1977). Over the years, priority rules which look at multiple job attributes simultaneously have been developed to improve system performance.

For example, the „Apparent Tardiness Cost“ rule, which, in addition to the weighted process time, also includes the planned completion time and a weighting value (k_1) (Vepsalainen and Morton 1987). With regard to setup times, the rule was then extended to include a setup time term and has been since described as "Apparent Tardiness Cost with Setups" (ATCS). The additional term denotes the ratio between sequence-dependent setup time and average setup time multiplied by the second weighting value (k_2). The rule is used in the form shown in equation (1) for this study. The combination of three attributes and the use of two weighting values make it possible to achieve good performance across a wide range of scenarios when properly tuned (Lee et al. 2002).

$$Z_i^t = \frac{w_i}{p_i} exp\left(-\frac{(d_i - t - p_i)^+}{k_1 \overline{p}}\right) exp\left(-\frac{s_{i,l}}{k_2 \overline{s}}\right) \quad (1)$$

Knowing that the system performance strongly depends on the correct selection of the k-values to match the system workload, they are required to be dynamically adjusted to the situation on the shop floor. Consequently, the dynamic adaptation is a hyperheuristic. However, to build the knowledge base about the relationship between the k-values and the resulting performance, all possible combinations of k-values, product mix and system state would have to be known.

Because of the complexity in real systems, not all possible combinations of influencing factors can be simulated. For that reason, a defined combinations of system states is simulated and the unknown situations are estimated by a regression procedure

In the current state of the literature, the use of neural networks (NN) represents the standard to forecast system behavior. Specifically, the usage of NN for the prediction of system performance was considered in detail with regard to the dependence on the k-values and the system status in multiples works (Heger 2014; Heger et al. 2016; Mönch et al. 2006; Mouelhi-Chibani and Pierreval 2010). However, despite good results, it should be noted that NNs are basically used as a black box and do not allow us to infer any information about the influence of certain factors. To this end, the use of NNs as a baseline was previously compared with the use of decision trees and reinforcement learning (Rai, 2020).

Decision trees have recently received significant attention in the context of Explainable AI (Puiutta and Veith 2020; Rai 2020).

Unlike complex methods, such as deep NNs, which produce non-interpretable black-box models, decision trees are rule-based methods that provide the user with an intuitive representation of rules and processes. At each node of a decision tree, a particular objective function is tested. The result provides the path to the new node. The structure repeats until a particular condition is met. Human comprehensible rules can be derived from paths through the decision tree.

Due to their structure, decision trees can be used for both classification and regression tasks. Thus, being generally suitable for dynamic selection of priority rules, they are prone to perform worse in unknown scenarios (Shahzad and Mebarki 2016). Other tree-based methods, such as Random Forest and XGBoost, based on a combination of decision trees and have forfeited a certain degree of interpretability in order to achieve better accuracy and generalization. Nevertheless, they are increasingly equipped with further functionalities to improve interpretability (Lundberg et al. 2020)

The use of reinforcement learning has already achieved good results as a hyperheuristic in the dynamic selection of sequencing rules. Studies show that the inherent advantages of reinforcement learning, as opposed to supervised learning methods, are in the direct interaction with the system. The agent learns the correct behaviors based on the observed behavior and the feedback received. Specifically, the use case entailing the selection of priority rules for all machines in the system (Heger and Voss 2020, 2021) has shown good results. The authors show that based on the observed system workload and queues, performance can be improved by dynamically selecting sequencing rules. Similarly, other authors show that dynamic adaptation of machine-specific rules enables significant performance improvements across different scenarios (Shiue et al. 2018, 2020).

This paper examines the extent to which the three aforementioned methods can be useful in supporting the selection of appropriate k-values for the ATCS sequencing rule. Specifically, the extent to which the use of comprehensible actions leads to a reduction in performance is to be examined. The interaction effects between performance and explainability is examined in more detail in the context of the presented scenario. In addition, using the trained hyperheuristics, it is to be tested whether they are still able to select and dynamically adjust the k-values according to the system state in an unknown scenario, thereby achieving a more robust performance

.

2 Simulation and Scenario

The study is used and conducted in the context of a manufacturing system with sequence-dependent setup times. Unplanned and unknown changes, such as product mix changes and workload fluctuations, are added to be able to look at a behavior of the different methods in unknown scenarios. The scenario is described in detail below:

System	Machines: 10 Machine Groups: 5 Structure: Flow shop
Job Parameter	Product types: 4 Distribution of product types: based on Product Mix Operations per Order: 10 Distribution of Interarrival Times: Poisson Process times: 1 – 99 Distribution of Process times: uniform Due Date: TWK Method
Simulation	Warm Up: 2500 Jobs Duration of Simulation: 12500 Jobs
KPIs	Average Mean Tardiness

Table 1: Detailed description of the flow shop scenario.

The particular focus in this contribution is to consider the impact of the sequence-dependent setup times, which are shown below. Depending on the product mix, the proportions of the four product types are different, which leads to different ratios of setup time. Thus, it can be assumed that a product mix with the first three product families requires significantly less setup time than a product mix containing all four product families. Here, the matrix in (2) can be read as follows: the setup time from family 1 to family 2 is 5 minutes; the setup time from family 1 to family 4 is 25 minutes. With an average processing time of 50 minutes, the setup time ratio can have a massive impact on the performance of the system.

$$\begin{pmatrix} 0 & 5 & 10 & 25 \\ 5 & 0 & 10 & 25 \\ 5 & 5 & 0 & 25 \\ 5 & 5 & 10 & 0 \end{pmatrix} \quad (2)$$

To measure performance, the average mean tardiness and the average lead time are documented. The average tardiness results from the sum of all deviations of planned completion time and the actual completion divided by the number of observations.

It should be noted that orders completed too early are assessed with a delay of zero. The due date, considered as the planned completion time, is calculated from the sum of the start time and the average planned processing time for all steps multiplied by a due date factor. The due date factor is selected so that a certain setup-, maintenance- and transport-time between machines is acceptable.

For the creation of regression models, training data is generated in an extensive parameter study using the discrete-event simulation model. The use of the simulation makes it possible to determine the length and width of the data set itself. The generated data forms the basis for the knowledge-based approaches to dynamic adjustment described later. In this case, the simulation model is available as a training environment for reinforcement learning as well.

Figure 1 describes the multiple steps of the procedure; starting at the bottom center is the simulation model. Through the parameter variation experiment it is possible to examine the behavior of the performance depending on different system state combinations, thus making it possible to create training data for the different regression models.

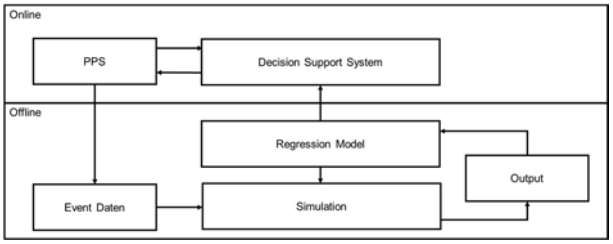

Figure 2: The regression model is trained offline based on the simulation and utilized online after training as a decision support system.

Figure 2 shows a schematic of the dynamic adaptation procedure during the online application. It considers the performance of the system over time and under changing states as well as the last selection of k-values. Depending on the selected performance indicator, the goal is to minimize or maximize this performance value; in this contribution, the minimization of the average tardiness is considered.

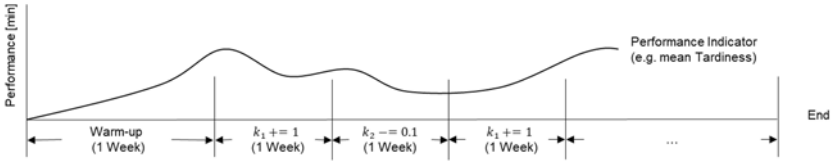

Figure 1: Generalized behavior of the adjustment of k-value pairs with RL.

At defined points in time, regardless of the particular method being used, decision support utilizes the regression model to make a statement about the most appropriate k-values for the situation to improve performance. The values are used for a defined period of time and re-evaluated afterwards

It is necessary to evaluate whether and to what degree the variation occurs due to the inherent stochastic uncertainties of the simulation model. The simulation study can further evaluate how the frequency of adjustment affects performance.

A parameter analysis is performed in this context to find out which observations have significant influence, and which do not. Data pre-processing such as standardization, one-hot encoding, and a combination of these are performed independently of the method used, but due to the simulation focus, their consequences are not evaluated in detail.

3 Evaluation

For the training data set, a parameter study was performed recording all possible combinations of k_1-values from 1 to 10 and k_2-values from 0.01 to 1.01 under 7 different workloads from 85 % to 95 % as well as 12 different product mixes with different setup proportions.

For each of the 9240 individual parameter combinations, 5 replications were performed. The data set used thus comprises 46200 samples. The observations from the system were the average mean tardiness, average lead time, product mix, and average machine utilization. In this contribution, the k-values, machine utilization, and performance indicators are considered as continuous variables, while the product mix is considered as a categorical variable.

In the raw data, it can be seen that product mixes perform differently at the same utilization rate and using the same static k-values, depending on the setup ratio of the mix, shown for two product mixes in Figure 3. A small k_2-value, which is beneficial for product mixes with high setup ratio (e.g. [30,40,20,10]), would lead to a 30% degradation in performance for product mixes with a lower setup ratio. In this study, a machine utilization of 85 % is considered. It should be mentioned that the raw data shows an increase of up to 5 % utilization depending on the different setup time proportions per product mix, for the same planned utilization. Further, low utilization levels mean that no potential for improvement is possible. This situation must be examined over a number of product mixes in order to improve performance.

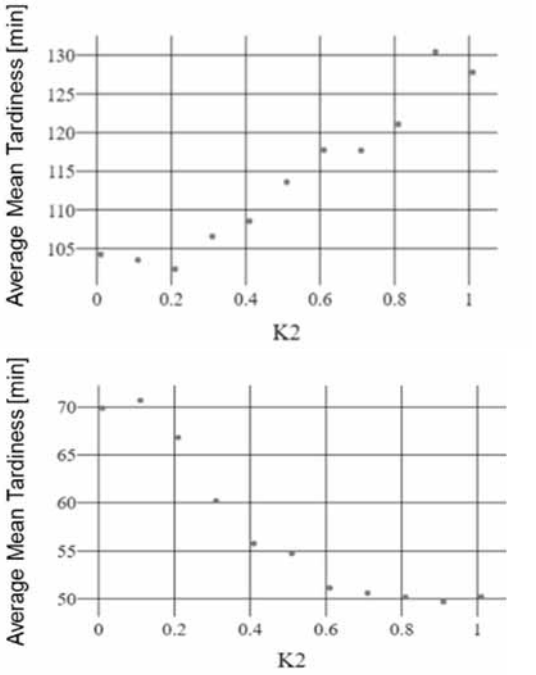

Figure 3: Based on the k_2-values, the performance of product mix [30,40,20,10] (left) and product mix [10,70,10,10] (right) are different given the planned utilization of 85 %.

In addition to the first data set with 46200 data points, a second data set with 13860 data points (corresponding to 30% of the first data set) is generated. Subsequently, the NNs as well as the decision trees (DTs) were trained on both sets to be able to make a statement about the performance with more data points. The parameters for the NNs as well as the DTs were determined using a grid search procedure

The NN was implemented as a multi-layer perceptron in Python using the scikit learn library. The resulting two-layer network with 10 neurons in the first layer and 30 neurons in the second layer had the activation function "relu". In combination with the solver "adam", a mini-batch size of 500 samples showed good results. The initial learning rate was set to 0.01. An L2 regularization was performed.

The DT was implemented using scikit learn as a decision tree regressor in Python. In this framework, it was found that using a maximum depth of 5, with at least 4 samples per leaf led to good results.

The RL agent was trained using the Pathmind software-as-a-service platform with 12000 simulation runs. For training, the discrete-event simulation model was exported as a stand-alone Java file and trained on the platform, independent of local resources for 12 hours.

During this process, various hyperparameter configurations were automatically evaluated as part of population-based training and the best configuration for the scenario was found. Pathmind uses Ray and RLlib for training the agent. The strategy of the agent was trained by Proximal Policy Optimization

As part of the evaluation, the three methods are tested in online use in the context of the event discrete simulation. This involves documenting performance for a known scenario with a static workload and a known product mix. In the following, the scenario is a workload of 85% and product mix [30,40,20,10] from above. Figure 4 shows an example of the agent's behavior trained with reinforcement learning. As seen above, a low k_2-value is beneficial for the product mix [30,40,20,10]. It can be clearly seen that when the average delay (left Y-axis) in the system varies over time (X-axis), the used k_2-values of the ATCS rule (right Y-axis) is adjusted.

Over 30 replications, for a static utilization and for a known product mix, dynamic adjustment of k-values with RL has a positive impact but is not significantly different from static selection of k-values.

Of particular interest (see Figure 5) is the poor performance of the NNs; it is reasonable to assume that the dynamic selection and adjustment of the rules by the NNs has a negative impact on performance in the static scenario (Priore et al. 2006)

In the second evaluation scenario for the dynamic adjustment, a static utilization with changing product mixes is evaluated. In the scenario, a new product mix (in this case product mix [10,70,10,10]) is considered in the system over ¼ of the simulation time. It can be seen in Figure 6 that the selection of good static k-values already leads to good performance. In comparison, the DTs, the NNs as well as the RL agent bring an additional significant improvement of up to 15 %. Additionally, the RL agent is still 3 % better than the DTs. In contrast to the static scenario, the NN can show its advantages regarding generalizability of behavior. The comparable performance of RL and NN is understandable since RL uses NNs to estimate the reward.

Over the evaluation in both scenarios, it is shown that DTs can reproduce known system behavior very well and can describe dynamic behavior to some extent.

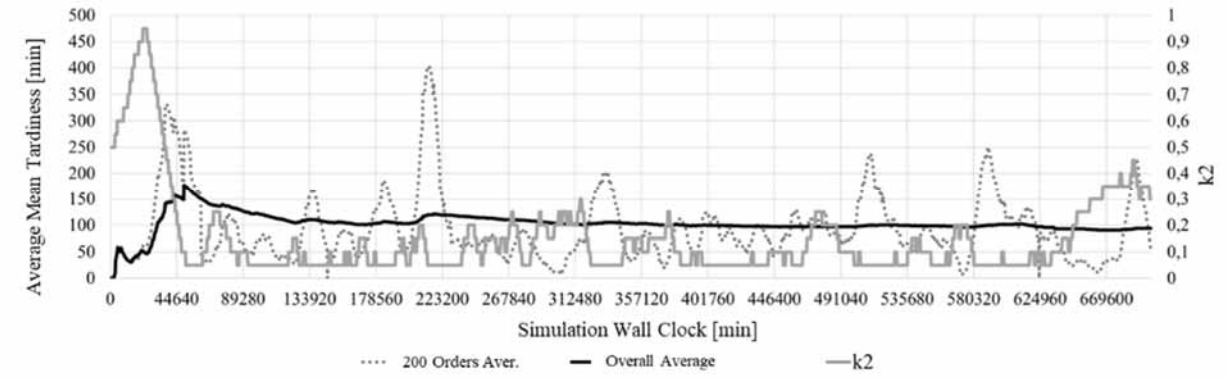

Figure 4: The RL-agent adjusts the k_2-value based on the system status dynamically.

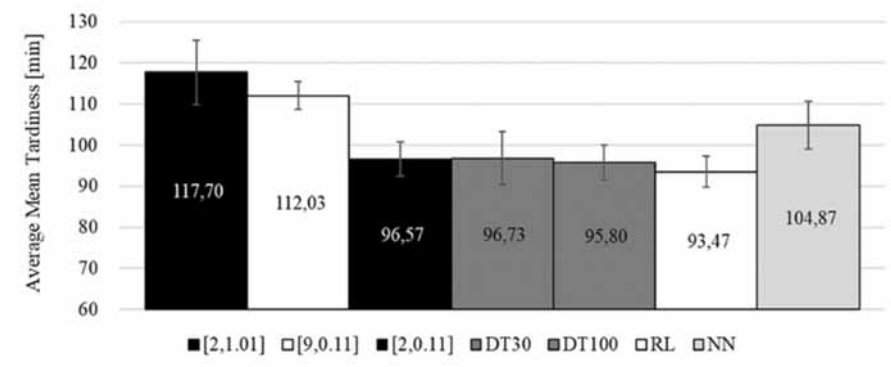

Figure 5: The direct comparison shows the performance with static as well as dynamically adjusted k-values by the DT, NN and the RL approach in a known scenario.

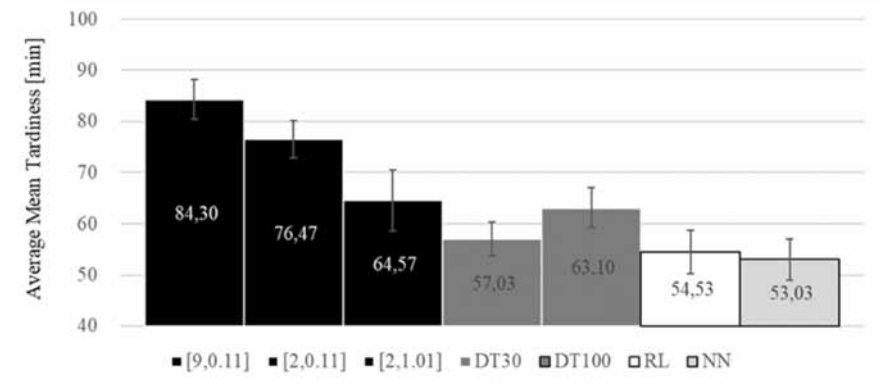

Figure 6: The direct comparison shows the performance with static as well as dynamically adjusted k-values by the DT, NN and the RL approach in an unknown scenario.

The use of NNs and RL is especially advantageous in scenarios with unknown behavior and can lead to an improvement in performance of up to 15%.

4 Summary and Outlook

Dynamic adaptation of priority rules using various machine learning methods can lead to improved performance. In this paper, three methods for adjusting the k-values of the ATCS rule were trained and evaluated over two scenarios. A data set which includes the relationships between product mix, k-values, and system utilization was created using a flow shop manufacturing environment and an extensive parameter study. This was then used as the training basis for DT and NN, while the discrete-event model was used as the training environment for the RL agent.

The comparison within the static scenario shows that DT and RL can reproduce the performance of the static k-values. During training, it was shown that the use of DTs can help in making qualitative statements regarding performance. In the dynamic scenario, it was shown that all three methods can improve the performance. On par, RL and NN can reduce average delay by 15% and compensate for unplanned product mix changes. In the next step, a deep and detailed analysis of the dynamic adjustment over multiple product mixes and unknown scenarios will be performed

References

[1] Heger J. Dynamische Regelselektion in der Reihenfolgeplanung. Springer Fachmedien Wiesbaden, Wiesbaden, 2014.

[2] Heger J, Voss T. Dynamically Changing Sequencing Rules with Reinforcement Learning in a Job Shop System with Stochastic Influences. In: Bae K-H, Feng B, Kim S, Lazarova-Molnar S, Zheng, Z, Roeder T, Thiesing R (eds). Proceedings of the Winter Simulation Conference, 2020, pp 1608–1618.

[3] Heger J, Voss T. Dynamically adjusting the k-values of the ATCS rule in a flexible flow shop scenario with reinforcement learning. International Journal of Production Research, 2021. DOI 10.1080/00207543.2021.1943762

[4] Heger J, Branke J, Hildebrandt T, Scholz-Reiter B. Dynamic adjustment of dispatching rule parameters in flow shops with sequence-dependent set-up times. International Journal of Production Research 54:6812–6824, 2016. DOI 10.1080/00207543.2016.1178406

[5] Lee YH, Jeong CS, Moon C. Advanced planning and scheduling with outsourcing in manufacturing supply chain. 2002, Computers & Industrial Engineering 43:351–374.

[6] Lundberg SM, Erion G, Chen H, DeGrave A, Prutkin JM, Nair B, Katz R, Himmelfarb J, Bansal N, Lee S-I. From Local Explanations to Global Understanding with Explainable AI for Trees. 2020, Nat Mach Intell 2:56–67. DOI 10.1038/s42256-019-0138-9

[7] Mönch L, Zimmermann J, Otto P. Machine learning techniques for scheduling jobs with incompatible families and unequal ready times on parallel batch machines. 2006, Engineering Applications of Artificial Intelligence 19:235–245.

[8] Mouelhi-Chibani W, Pierreval H. Training a neural network to select dispatching rules in real time. 2020, Computers & Industrial Engineering 58:249–256.

[9] Nunes I, Jannach D. A systematic review and taxonomy of explanations in decision support and recommender systems. 2017, User Model User-Adap Inter 27:393–444. DOI /10.1007/s11257-017-9195-0

[10] Panwalkar SS, Iskander W. A Survey of Scheduling Rules.1977. Operations Research 25:45–61.

[11] Priore P, La Fuente D de, Puente J, Parreno J. A comparison of machine-learning algorithms for dynamic scheduling of flexible manufacturing systems. 2006, Engineering Applications of Artificial Intelligence 19:247–255. DOI 10.1016/j.engappai.2005.09.009

[12] Priore P, Ponte B, Puente J, Gómez A. Learning-based scheduling of flexible manufacturing systems using ensemble methods. 2018, Computers & Industrial DOI 10.1016/j.cie.2018.09.034

[13] Rai A. Explainable AI: from black box to glass box. 2020, J. of the Acad. Mark. Sci. 48:137–141. DOI 10.1007/s11747-019-00710-5

[14] Rehse J-R, Mehdiyev N, Fettke P. Towards Explainable Process Predictions for Industry 4.0 in the DFKI-Smart-Lego-Factory. 2019, Künstl Intell 33:181–187. DOI 10.1007/s13218-019-00586-1

[15] Shahzad A, Mebarki N. Learning Dispatching Rules for Scheduling: A Synergistic View Comprising Decision Trees, Tabu Search and Simulation. 2016, Computers 5:3. DOI 10.3390/computers5010003

[16] Shiue Y-R, Lee K-C, Su C-T. Real-time scheduling for a smart factory using a reinforcement learning approach. 2018, Computers & Industrial Engineering 125:604–614. DOI 10.1016/j.cie.2018.03.039

[17] Shiue Y-R, Lee K-C, Su C-T. A Reinforcement Learning Approach to Dynamic Scheduling in a Product-Mix Flexibility Environment. 2020, IEEE Access 8:106542–106553. DOI 10.1109/ACCESS.2020.3000781

[18] Usuga Cadavid JP, Lamouri S, Grabot B, Pellerin R, Fortin A. Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. 2020, J Intell Manuf 31:1531–1558. DOI 10.1007/s10845-019-01531-7

[19] Vepsalainen APJ, Morton TE. Priority rules for job shops with weighted tardiness costs. 1987, Management Science 33:1035–1047