Machine Learning prediction model for Dynamic Scheduling of Hybrid Flow-Shop based on Metaheuristic

Abdelhakim Ghiles Hamiti, Wassim Bouazza, Arnaud Laurent, Mebarki Nasser

Machine Learning prediction model for Dynamic Scheduling of Hybrid Flow-Shop based on Metaheuristic

Abdelhakim Ghiles Hamiti * **Wassim Bouazza** *
Arnaud Laurent * **Nasser Mebarki** * **Mohamed Kenani** *

* *Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes, France*

(emails: abdelhakim-ghiles.hamiti@univ-nantes.fr, Wassim.bouazza@univ-nantes.fr, Arnaud.Laurent@univ-nantes.fr, Nasser.Mebarki@univ-nantes.fr)

Abstract: Optimizing scheduling and allocation strategies in dynamic production environments, notably in Hybrid Flow-Shops, presents significant challenges. This study focuses on resource assignment within dynamic contexts. It proposes an approach that use Genetic Algorithm (GA) to generate data and train machine Learning (ML) to predict near optimal allocations. Through experiments across various scenarios, the accuracy of prediction of different ML models for resource allocation is evaluated. Our findings highlight the potential of ML techniques to improve decision-making in dynamic and flexible manufacturing systems (FMS), contributing to efforts to enhance reactive scheduling strategies. Future work will assess the impact of these decisions on mean completion time, providing deeper insights into on-line scheduling efficiency.

Keywords: Intelligent Manufacturing Systems, Optimization and Control, Genetic Algorithms.

1. INTRODUCTION

The flow-shop scheduling problem is an optimization problem involving resources, operations, and constraints. Each job j comprises a sequence of n_j operations, with each operation i as part of exactly one job, each executed on a specific stage k within a given processing time p_{ij}. The objective is to determine a feasible schedule, denoted by S_{ij}, that optimizes various performance measures such as completion times or lateness.

The Hybrid Flow-Shop scheduling problem (HFS), a variant of the flow-shop model, involves multiple parallel machines for at least one stage (Pinedo (2016)). This particularity involves an allocation decision for each operation of the products on each stage. In flexible HFS, jobs may also bypass certain stages, presenting unique challenges in resource assignment and operation scheduling.

This article specifically addresses the resource assignment problem within dynamic manufacturing environments. While traditional methods such as Dispatching Rules have been employed, recent advancements have seen the application of data mining and Machine Learning (ML) techniques to tackle these challenges. ML, in particular, has garnered interest due to its effectiveness in optimization tasks, albeit requiring substantial amounts of data for training.

Our approach offers a novel solution to the resource assignment problem. Genetic Algorithms (GA) and simulation are leveraged to generate data for training ML models for predictive purposes. This methodology aims to overcome the data scarcity challenge often encountered in ML applications within manufacturing settings.

2. LITERATURE OVERVIEW

Scheduling flexible manufacturing systems (FMS), such as flexible job-shops and Hybrid Flow-Shops, has become a focal point across various contexts, especially in dynamic environments where rapid optimizing production schedules is crucial. Initial research primarily centered on Dispatching Rules (DRs) to find a single performant one applicable to all scenarios. DRs are simple scheduling heuristics that gradually construct schedules by determining the next task to be assigned to an available machine based on priority evaluation (Đurasević and Jakobović, 2018). Already in 1977, Panwalkar (1977) referenced over 100 different DRs from the literature. The authors highlight that diverse researchers may use different terminologies to describe identical principles, and occasionally, new designations are introduced for these DRs, making it more challenging to quantify them. Nevertheless, Since then many other rules have been added by researchers. An example on dynamic flexible flow-shop scheduling is given in (Kianfar et al., 2009). The authors explore methods for efficiently organizing tasks in dynamic environments, aiming to minimize delays and prioritize critical tasks.

While empirical rules can be relatively effective, it may be more advantageous to create methods capable of automatically selecting or generating rules according to the problem being addressed. The main characteristic of such algorithms, called Hyper-Heuristics (HHs), is that they explore a search space of heuristics rather than a search space

of solutions (Ochoa and Özcan, 2010).A certain number of contributions seek to select the best rule among a set of candidate ones, while other methods focus on the automatic generation of specific rules. Furthermore, the work Bouazza et al. (2021) introduce a product-driven approach using HHs, enhancing adaptability and performance in dynamic scheduling of FMS. The system performs a reactive DRs switching depending of the decisional context to enhance the global performance of the system. Other DRs selection can be found in the literature as in (Drake et al., 2020). The authors specifically focused on selection HHs, offering critical discussions and highlighting current research trends.

Concerning the HHs for DRs generation, algorithms as Genetic Programming (GP) are also widely employed by researchers. In (Braune et al., 2022), the study examined both job assignment and machine sequencing using a single-tree representation, comparing it to a multi-tree approach. Computational experiments showed an advantage over existing priority rules in scenarios with unrelated parallel machines and larger instances. Another approach using GP is given by Jaklinović et al. (2021). The paper aims to assess GP's potential in constructing DRs for constrained problems by adapting schedule generation schemes and introducing supplementary terminal nodes. Results indicate that automatically generated DRs outperform manually designed ones adapted for constrained problems, with additional terminals enhancing DR construction for certain constraints.

Recent progress has involved combining different techniques as simulation-optimization, data mining, or statistical process control charts to improve scheduling strategies. These strategies consider various factors like new job insertion, machine breakdowns, operator unavailability, worker fatigue, and skill levels. Moreover, approaches merging data mining, discrete event simulation, and dispatching rules have shown substantial reductions in Makespan and notable enhancements in machine utilization (Zhao et al., 2022). An example using simulation to build DRs is proposed in (Li et al., 2022). The work introduces a new method called Combined DRs to schedule tasks in the complex environment of semiconductor manufacturing, which involves coordinating equipment maintenance and various production factors. Another example using simulation is given in (Pergher and de Almeida, 2018). The authors introduced a multi-attribute, rank-dependent utility model to select the best DR for dynamic job-shop environments. It integrates simulation, a method for quantifying the decision-maker's preferences regarding various outcomes, with a theoretical framework that examines multiple dimensions of decision-making.

Furthermore, ML techniques can also be a powerful tool to enhance the performance of dynamic scheduling. In order to improve decision-making in real-time scheduling within dynamic environments, agent-based techniques in combination with Deep Reinforcement Learning (DRL) algorithms have been put forth by Liu et al. (2023). It highlights the significance of production scheduling and the potential disruptions caused by unforeseen events, with a particular emphasis on HFS scheduling. A combined framework of DRL and Multi-Agent Systems (MAS), termed DRL-MAS, is introduced. Numerical experiments and case studies showcase the superior performance of the DRL models when compared to existing scheduling strategies. In (Marchesano et al., 2021), a DRL mechanism is also used for dynamic scheduling in a flow-shop, more precisely a Deep Q-Network (DQN). The system possesses a collection of established DRs for each machine's queue, from which the most suitable one is dynamically selected based on the system's current state. The paper's novelty lies in the modeling of the reward function, state representation, and action space used by the DQN. In (Zhang et al., 2022), the authors highlight another application of RL to optimize machine allocation and task dispatching in smart factories.They proposed a framework based on Reinforcement Learning (RL).

In conclusion, substantial advancements have been achieved in tackling manufacturing scheduling challenges in dynamic environments. While Dispatching Rules have traditionally played a pivotal role in flexible production systems, the rise of ML techniques, especially RL, has opened up novel opportunities for enhancement. However, it's noteworthy that many ML methods primarily concentrate on choosing the optimal DR from a pool of candidates, with RL demonstrating significant potential in adjusting rules based on system feedback. Based on this finding, the next section will detail our approach, which aims to be different by proposing to train a model directly from the solutions generated by a metaheuristic.

3. PROPOSED APPROACH

This section provides a detailed overview of the key elements comprising the approach. It explains the fundamental components essential to the methodology.

3.1 Resolution approach and motivation

This section explores the motivation behind our approach. Here, the scheduling problem is adapted to focus solely on allocation, disregarding sequencing. A unique sequencing rule is used. Indeed, in this paper, once allocated to resource, the products within the queue are processed in the First-In, First-Out (FIFO) order according to arrival in the queue.

The goal is to develop a model capable of predicting a near-optimal allocation decisions within a dynamic context. To achieve this objective, a dataset that accurately reflects the allocation choices is required. The GA is then used to construct such a dataset, . Similar to an oracle, its solutions are considered to be the among the best possible choices and are used into dataset. The GA knows the arrival times and characteristics of products before they occur, transforming our dynamic context into a static one for the training purpose.

In this regard, a three-step approach is proposed as shown in Fig. 1:

(1) First, data are generated by feeding sets of instances to the GA on an offline context. (see 3.2 Data generation phase)
(2) Then, this collected data are processed to retain only pertinent information for learning and predicting in a dynamic context. (see 3.3 Data preprocessing phase)

(3) Finally, ML models are trained and evaluated on this database. (see 3.4 ML models training phase)

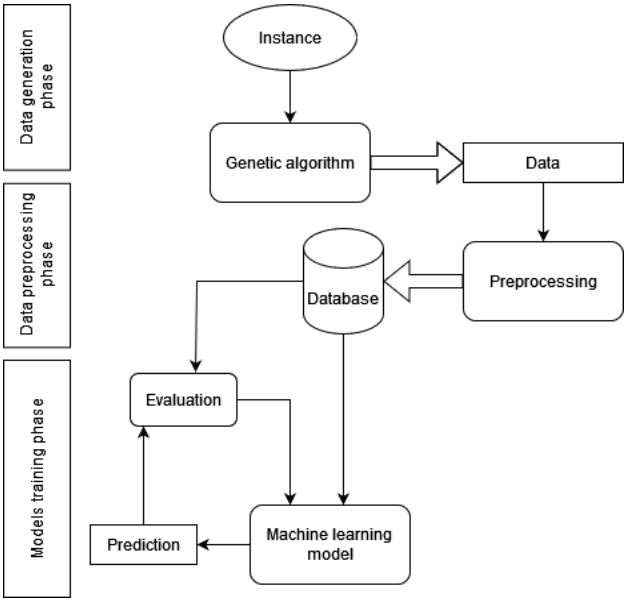

Fig. 1. Summary schema of the proposed approach

In the following details about each phase are given as well as an exploration of each component in the Fig. 1 .

3.2 Data generation phase

During this phase, the GA is used in an offline context to generate data from a set of instances.

Genetic Algorithm : The GA, inspired by natural selection, iteratively generates and evaluates potential solutions. By applying genetic operators like selection, crossover, and mutation, it aims to converge towards optimal solutions by favoring high-quality solutions' reproduction. In this context, the GA is considered as an Oracle, constructing solutions close to optimal based on the assumption of known product arrivals in advance. The Fig. 2 represents the 4 main stages of the GA proposed :

(1) Using operational data of a specific production system, a set of scenarios are constructed to describe the product arrivals and their associated characteristics across temporal intervals.
(2) For each scenario, a population of solutions is generated, with each individual representing a distinct resource allocation scheme.
(3) For each individual, a computational simulation is run to unfold the allocation scheme represented by the solution.
(4) Following simulation completion, performance metrics, such as Makespan or Mean Completion Time in our case, are compiled. If the termination condition of the GA is not met, a new simulation cycle begins with the new population obtained.

Description of the set of scenarios : The instance under study is composed of three levels, each of which contains a variable number of resources. The first stage contains two resources, the second one has three, and the last one

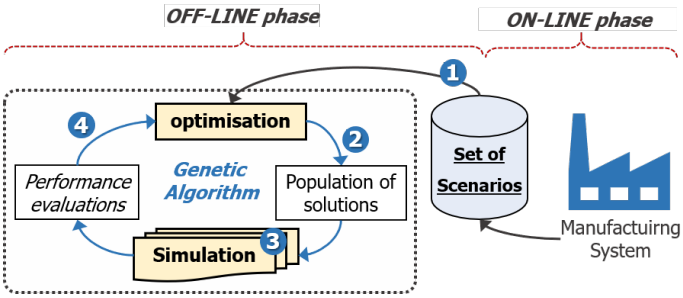

Fig. 2. The optimization process via simulation through the GA to generate off-line near optimal solutions

contains two. These resources are designed to handle a selection of five product families.

The instance is the input of the GA according to Fig. 1. It is referred to as Scenario. It's an ordered list of events describing the products arrival times and their characteristics as their families.

Description of a Solution : A solution is a list that determine the allocation choices for processing each product.

Solution Evaluation : The solution's evaluation involves calculating the mean completion time obtained after simulating the prodcuts allocation in the system.

in the following subsection we focus on the size of the solution space to assess the complexity of the problem.

Solutions Space : Here is a small recall of the used annotations in the following and what they refer to:

- n: number of products / jobs
- m_j: number of operations of the product j
- R_i: number of resources on the i-th stage

The solution space encompasses all potential allocation choices for each product within the batch. Its magnitude is determined by the formula:

$$\prod_{j=1}^{n} \left(\prod_{i=1}^{m_j} R_i \right)$$

In scenarios where each product performs an equal number of tasks ($\forall i, j \leq n, m_i = m_j$), the formula simplifies to:

$$\left(\prod_{i=1}^{m_j} R_i \right)^n$$

When the number of resources per stage remains constant ($\forall i, j \leq n, R_i = R_j$), the formula further simplifies to:

$$\prod_{j=1}^{n} (R^{m_j})$$

In cases where both the number of tasks per product and the number of resources per stage remain constant, a simplified formula is proposed :

$$R^{m^n} = R^{m \times n}$$

Hence, the complexity of the problem is demonstrated as follows.$O(R^{m \times n})$

Generated data : At each product input, the system's state is recorded, which constitutes the covariates, and the choice made by the GA, which constitutes the label. The system state refers to both the condition of resources and the condition of cells.

3.3 Data preprocessing phase :

Upon collection, the data undergoes preprocessing, which begins with the preselection of covariates based on several criteria. This phase aims to end up with a dataset that contains only useful information for learning and predicting in a dynamic context.

First, **dependent** or **redundant** variables are identified and eliminated. This step targets variables that are calculated as rates using other variables, thereby removing redundancy and preventing unwanted dependencies. For example : $ccr_i = \frac{cc_i}{\sum_i^n cc_i}$

with ccr_i : current charge rate of the ressource i, and cc_i : current charge of the ressource i.

Secondly, variables with **low variance** are discarded. This process removes variables that have minimal impact on classification accuracy.

Furthermore, variables whose relevance is specific to **sequencing** are excluded from consideration. The focus remains solely on variables pertinent to assignment within the context of our study.

The variables retained in the final database include

- T_j : arrival time of the product j in the system
- f_j : family of the product j
- pf_k : family of last product that went through the ressource k
- $curr\_c_k$: the current charge of the ressource k, includes the process and setup times for all the products on the ressource queue.
- exp_c_k : the expected charge of the ressource k if the product is allocated to it.
- $exp\_gross_k^i$: the added charge to the ressource k that belongs to the stage i if the product is allocated. This charge includes both the process time and the setup time of the added product.
- $eff\_gross_k^i$: the ratio of the effective gross for each ressource k belonging to stage i compared to the most effective resource of the stage, calculated as follows :

$$eff\_gross_k = \frac{exp\_gross_k^i}{\min_x(exp\_gross_x^i)}$$

3.4 ML models training phase :

Upon completion of data preprocessing, ML models are employed to extract valuable insights. Each model provides a unique method for classifying allocation decisions within the hybrid flexible production system. Below are the selected models:

(1) **Random Forests:** Random Forests are ensembles of decision trees, where each tree votes for the majority class. They offer great flexibility and are robust against complex datasets.

Table 1. Main parameters of the Genetic Algorithm

Parameter	**Value**
Population size	30,000 solutions
Replacement rate	0.7
Mutation rate	0.01
Crossover rate	0.5
Selection method	Stochastic universal sampling
Elitism	True
Initialization method	Random
Stop condition 1	Maximum number of generations = 1000 iterations
Or Stop condition 2	Generations without improvement = 300 iterations
Fitness function	Mean Completion Time Minimization

(2) **Bagging Classifier (Bootstrap Aggregator):** The Bagging Classifier combines multiple identical learning models to enhance accuracy and stability. It employs the bootstrap technique, randomly sampling with replacement to create several training subsets.
(3) **Gradient Boosting Classifier:** The Gradient Boosting Classifier constructs decision trees sequentially, correcting errors from previous models. This allows dynamic adaptation to residuals, producing powerful models.
(4) **Artificial neural networks (NN):** Artificial Neural Networks are models inspired by the functioning of the human brain. They consist of layers of interconnected neurons capable of learning complex relationships in data.
(5) **Artificial neural network with Bayesian optimization:** This variant of Artificial Neural Networks uses Bayesian optimization to find optimal hyperparameters, thereby enhancing the efficiency of model learning.

4. EXPERIMENTS AND RESULTS

4.1 Input data

Genetic Algorithm parameters: The following Table 1 outlines the main parameters of our Genetic Algorithm. These parameters control how the algorithm works and affects its ability to find the best or close-to-best solutions for optimization problems.

Scenario Instances: The GA was executed on 200 scenarios sharing all the following statistics :

(1) number of products between 90 and 110, uniform distribution.
(2) arrival times between 1 and 600 time units, uniform distribution.
(3) product families distribution uniform.

4.2 Results

Each stage on the flow-shop is associated with a module dedicated to resource allocation decisions, making a total of three different models.

To evaluate the performance of these models, three separate tables are used, each illustrating the error associated with different types of trained models on a specific stage.

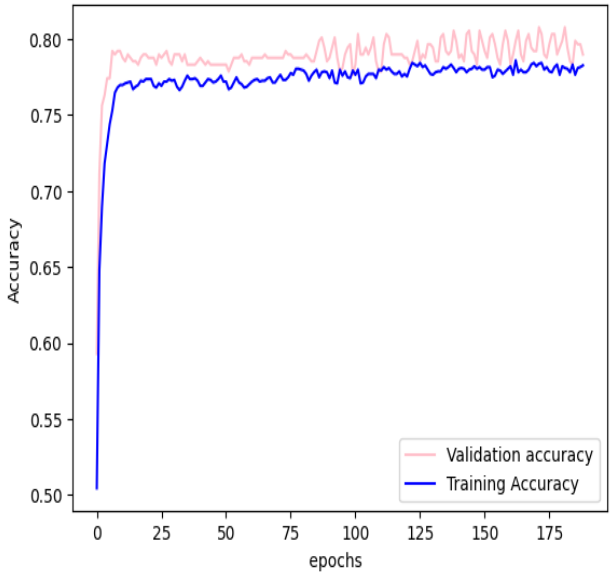

Fig. 3. accuracy evolution through training for neural network with bayesian optimisation, stage 1

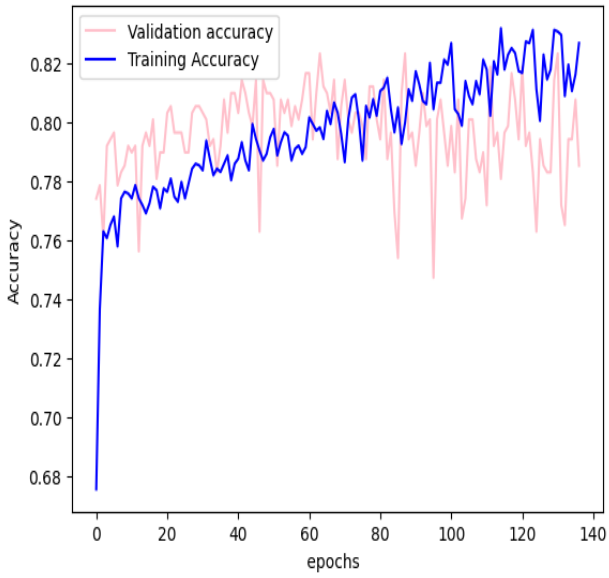

Fig. 4. accuracy evolution through training for neural network without bayesian optimisation, stage 1

Table 2. Performances of stage 1 models

Model	Accuracy
Random forests	0.818
Bagging classifier	0.796
Gradient boosting classifier	0.798
Dense neural network	0.79
Dense NN + Bayesian optimisation	0.814

Table 3. Performances of stage 2 models

Model	Accuracy
Random forests	0.65
Bagging classifier	0.647
Gradient boosting classifier	0.656
Dense neural network	0.638
Dense NN + Bayesian optimisation	0.659

Table 4. Performances of stage 3 models

Model	Accuracy
Random forests	0.957
Bagging classifier	0.9535
Gradient boosting classifier	0.9517
Dense neural network	0.9427
Dense NN + Bayesian optimisation	0.9516

4.3 Discussion

Figures 3 and 4 depict the training and validation accuracy evolution over epochs for both the neural network with and without Bayesian optimization in the first stage.

Variations in model performance across different stages of the flow-shop can be observed based on the provided tables. It's important to note that the unit of measurement here is accuracy, not mean completion time, as simulations with the models' predictions have not yet been executed. Let's analyze the results for each stage:

- *Stage 1 (2 resources):* The Random Forests model achieved the highest accuracy of 0.818, followed closely by the Dense NN + Bayesian Optimization model with an accuracy of 0.814. The Bagging Classifier, Gradient Boosting Classifier, and Dense Neural Network also demonstrated respectable accuracies ranging from 0.796 to 0.79.
- *Stage 2 (3 resources):* Across all models, the accuracies observed for stage 2 were generally lower compared to stage 1. The Gradient Boosting Classifier exhibited the highest accuracy of 0.656, followed closely by the Dense NN + Bayesian Optimization model with an accuracy of 0.659. While the Random Forests, Bagging Classifier, and Dense Neural Network also showed competitive accuracies ranging from 0.65 to 0.638.
- *Stage 3 (2 resources):* Notably, the accuracies observed for stage 3 were substantially higher compared to the other stages. The Random Forests model demonstrated the highest accuracy of 0.957, followed closely by the Bagging Classifier and Dense NN + Bayesian Optimization model with accuracies of 0.9535 and 0.9516 respectively. The Gradient Boosting Classifier and Dense Neural Network also exhibited high accuracies ranging from 0.9517 to 0.9427.

In line with our expectations, higher accuracy is observed in stages characterized by fewer resources.In simpler production environments, resource allocation decisions are relatively straightforward. As the complexity of the stage increases with a greater number of resources, the accuracy of the models appears to decrease slightly, indicating the additional challenges posed by optimizing resource allocation in more intricate production scenarios.

5. CONCLUSION

In this article, the dynamic context of the hybrid flow-shop scheduling problem, with a particular focus on resource assignment, has been delved into. While various methodologies, including DRs, heuristics, data mining, and ML, have been explored to tackle this challenge, a universally optimal approach remains elusive.

The proposed approach introduces a novel method that employs GA and ML for resource assignment in dynamic environments. The aim is to train ML models for predictive purposes by generating data through GA and simulation. This approach presents potential advantages in adaptability and efficiency, crucial in modern manufacturing settings where scheduling decisions must swiftly adapt to changing conditions.

Furthermore, detailed notations, configuration, and key elements of our approach have been outlined, with an emphasis on the importance of preprocessing data for effective ML models. The performance of various ML models across multiple stages of the flow-shop through experiments conducted on a range of scenarios, considering accuracy as the primary metric has been evaluated.

For future work, simulations of the selected configuration, utilizing the choices made by the trained models, to assess their performance in terms of mean completion time will be conducted. While precision serves as an initial evaluation metric, understanding the impact on mean completion time will provide deeper insights into the effectiveness of our approach in optimizing production schedules.

This preliminary phase demonstrated the potential of ML techniques to meet the challenges of dynamic scheduling. Further research is warranted to optimize model performance and scalability, paving the way to improve decision-making processes in dynamic manufacturing systems.

In conclusion, our study contributes to ongoing efforts in enhancing scheduling strategies, offering insights and methodologies that can bolster decision-making processes in dynamic manufacturing environments. We appreciate the reviewer's valuable feedback and eagerly anticipate refining our manuscript to address the points raised.

REFERENCES

Bouazza, W., Sallez, Y., and Trentesaux, D. (2021). Dynamic scheduling of manufacturing systems: a product-driven approach using hyper-heuristics. *International Journal of Computer Integrated Manufacturing*, 34(6), 641–665.

Braune, R., Benda, F., Doerner, K.F., and Hartl, R.F. (2022). A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems. *International Journal of Production Economics*, 243, 108342.

Drake, J.H., Kheiri, A., Özcan, E., and Burke, E.K. (2020). Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, 285(2), 405–428.

Jaklinović, K., Đurasević, M., and Jakobović, D. (2021). Designing dispatching rules with genetic programming for the unrelated machines environment with constraints. *Expert Systems with Applications*, 172, 114548.

Kianfar, K., Ghomi, S.M.T.F., and Karimi, B. (2009). New dispatching rules to minimize rejection and tardiness costs in a dynamic flexible flow shop. *The International Journal of Advanced Manufacturing Technology*, 45, 759–771. doi:10.1007/s00170-009-2015-x.

Li, S., Li, L., Yu, Q., and Iung, B. (2022). Combined dispatching rules based on flexible maintenance and multi-constraints. 3869–3874. IEEE. doi: 10.1109/CAC57257.2022.10055350.

Liu, Y., Fan, J., Zhao, L., Shen, W., and Zhang, C. (2023). Integration of deep reinforcement learning and multi-agent system for dynamic scheduling of re-entrant hybrid flow shop considering worker fatigue and skill levels. *Robotics and Computer-integrated Manufacturing*.

Marchesano, M.G., Guizzi, G., Santillo, L.C., and Vespoli, S. (2021). Dynamic scheduling in a flow shop using deep reinforcement learning. In *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems: IFIP WG 5.7 International Conference, APMS 2021, Nantes, France, September 5–9, 2021, Proceedings, Part I*, 152–160. Springer.

Ochoa, G. and Özcan, E. (2010). Special issue on hyper-heuristics in search and optimization. *Journal of Heuristics*, 16, 745–748. doi:10.1007/s10732-010-9147-x.

Panwalkar, I. (1977). A survey of scheduling rules. *Operations Research*, 25, 45–61. doi:10.1287/opre.25.1.45.

Pergher, I. and de Almeida, A.T. (2018). A multi-attribute, rank-dependent utility model for selecting dispatching rules. *Journal of Manufacturing Systems*, 46, 264–271. doi:10.1016/j.jmsy.2018.01.007.

Pinedo, M. (2016). *Scheduling: Theory, Algorithms, and Systems*. Springer International Publishing.

Đurasević, M. and Jakobović, D. (2018). A survey of dispatching rules for the dynamic unrelated machines environment. *Expert systems with applications*, 113, 555–569.

Zhang, Y., Zou, Y., and Zhao, X.L. (2022). Manufacturing resource scheduling based on deep q-network. *Wuhan University Journal of Natural Sciences*. doi: 10.1051/wujns/2022276531.

Zhao, A., Liu, P., Gao, X., Huang, G., Yang, X., Ma, Y.Y., Xie, Z., and Li, Y. (2022). Data-mining-based real-time optimization of the job shop scheduling problem. *Mathematics*. doi:10.3390/math10234608.