Scheduling AIV Transporter Using Simulation-Based Supervised Learning: A Case Study on a Dynamic Job-Shop with Three Workstations

Arman Hosseini, Zakaria Yahouni, Mohammad Feizabadi

Scheduling AIV transporter using simulation-based supervised learning: A case study on a dynamic job-shop with three workstations

Arman Hosseini * **Zakaria Yahouni** **
Mohammad Feizabadi ***

* *Univ. Grenoble Alpes, CNRS, Grenoble INP G-SCOP, 38000 Grenoble, France (e-mail: seyed-arman.hosseini-aliabad@grenoble-inp.org).*
** *Univ. Grenoble Alpes, CNRS, Grenoble INP G-SCOP, 38000 Grenoble, France (e-mail: zakaria.yahouni@grenoble-inp.fr)*
*** *Univ. Grenoble Alpes, CNRS, Grenoble INP G-SCOP, 38000 Grenoble, France (e-mail: seyedmohammadghasem.feizabadi@etu.univ-grenoble-alpes.fr)*

Abstract: Dynamic job shop scheduling consists of scheduling jobs dynamically with different routing on a set of machines. A feasible and quick solution can be computed using heuristics. One well-known heuristic in job shop problems is selecting the priority dispatching rule (such as giving priority to the job with the Shortest Processing Time called SPT). Nowadays, with the application of industry 4.0 technologies such as sensors, Intelligent robots, etc., workshop data are more accessible and can be exploited to find the appropriate dispatching rule depending on the state of the shop. This work proposes a data-driven methodology for scheduling an AIV (Autonomous Intelligent Vehicle) that supplies three workstations. Our approach is based on data collected from an Arena simulation model fed to a supervised learning algorithm. This one helps identify one among five dispatching rules for each scheduling decision.

Keywords: Job shop scheduling, AIV transporter, simulation-based supervised learning, flexible layout

1. INTRODUCTION

Scheduling methods are well-known processes aiming to improve the performance of manufacturing systems. In this paper, our focus is on Job shop scheduling problems (JSSP), which consist of scheduling jobs (products) that need to be processed on a set of machines. Each job has its own execution path. A feasible solution to this scheduling problem can be achieved by scheduling all jobs on machines in a specific order that adheres to the predetermined machines-execution precedence constraints. Afterward, different performance indicators can be measured, such as maximum lateness of jobs or maximum completion time of all jobs (Azadeh et al., 2012). This last indicator is referred to as *makespan* and is used for our research.

In real situations, jobs arrive dynamically, and decisions must be made in real-time (Ramasesh, 1990). Such context is referred to as dynamic scheduling. Moreover, in a dynamic job shop, processing times are stochastic and unexpected disruptions may occur. Disruptions and interruptions can arise from a variety of sources, such as machine breakdowns, material shortages, supplier delays, changes in product specifications, and fluctuations in demand (Farajzadeh et al., 2019). These unexpected events can cause delays, production inefficiencies, and additional costs. Rescheduling is one strategy to minimize the impact of these disruptions and maintain production efficiency (Yahouni, 2017; Uhlmann and Frazzon, 2018). The combination of the dynamic behavior exhibited by the Job Shop Scheduling Problem (JSSP) and its classification as NP-hard makes it challenging to attain a feasible schedule with optimal performance(Mohan et al., 2019). That is why optimization methods such as branch and bound are not feasible in real situations, significantly if the number of jobs and resources increases (Wang, 2021). To cope with this issue, heuristics can be used. One of the most used ones is Priority Dispatching Rules (PDRs), such as the Shortest Processing Time (SPT). The advantage of PDRs is the ease of implementation and quick execution, which gives credit to their ability to respond to dynamic environments (Pickardt et al., 2013).

However, the drawback of these PDRs is they do not guarantee an optimal solution. Hence, there is always a capacity to improve the solution. Researchers showed that a combination of different dispatching rules outperforms the use of only one for all processes (Pierreval and Mebarki, 1997). Various methods have been applied to

⋆ This work is supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-15-IDEX-02).

find a suitable combination. Most current approaches focus on using machine learning in JSSP (Seeger et al., 2022). Supervised and Reinforcement learning are the most used methods in predicting the best dispatching rule combinations (Usuga Cadavid et al., 2020).

Selecting the dispatching rule depends on different criteria, such as machines' current state, transportation tasks, etc. This last criterion plays a crucial role in improving the schedule performance. Several studies have been done about optimizing the transport of materials, and some of them considered the AGV (Automated guided vehicles) scheduling such as Mousavi et al. (2017). Unlike AGVs, AIVs (Autonomous Intelligent Vehicles) are more connected and do not require a movement zone. They are more flexible and use navigation technology with mapping and environmental recognition to go through their destination (Martin et al., 2021). AIVs are intractable with industry 4.0 equipment such as IoT and can enroll in a real-time decision-making process. Based on distances and obstacles, they can select which job has priority to be transferred first to its destination (Usuga Cadavid et al., 2020). This priority in transferring jobs substantially impacts the scheduling of a shop and, therefore, its performance.

In this paper, we address the problem of scheduling a single AIV in a dynamic job shop environment. The proposed methodology is applied to a case study made of three workstations. This approach schedules jobs to be transferred to machines (supposing that the AIV can only transfer one job at a time). Such a decision impact the makespan (when the number of jobs and their characteristics are supposed to be known). Our main contribution is using a data-driven approach to select the appropriate dispatching rule at each decision process dynamically. We use simulation to generate artificial data that are fed to a supervised learning AIV scheduler algorithm. The scheduler predicts the PDR to be used at the system's current state.

The remainder of the paper is organized as follows: In section 2, relevant studies are reviewed. In section 3, the methodology of simulation-based supervised learning to select the best combination of dispatching rules is proposed. In section 4, a case study of a dynamic job shop scheduling problem is introduced, and results are discussed. Finally, the conclusion and future research gaps are provided in the last section.

2. LITERATURE REVIEW

2.1 Machine learning in job shop scheduling problems

With the advancement of industry 4.0 technologies, the manufacturing process's real-time data has become more accessible. This opened new opportunities for data-driven decision-making and machine-learning techniques in JSSP. Machine learning methods are suitable tools to predict the best dispatching rules or jobs sequence on machines. The mostly applied machine learning methods for scheduling are supervised and reinforcement learning (Seeger et al., 2022).

Reinforcement learning methods get considerable attention in production scheduling (Wang and Usher, 2004). For instance, Belmamoune et al. (2022) proposed a Q-learning-based approach that selects one out of four dispatching rules in a job shop problem. The Q-learning agent uses machine loads for calculating makespan-related rewards. Another similar method is proposed in Lang et al. (2020). The authors use a deep Q Networks approach with two agents that allocate jobs to resources and select a sequence in each. This approach can be useful in a dynamic scheduling environment where real-time decisions have to be made. However, unlike supervised/unsupervised learning, they do not rely on historical data.

Supervised learning has gotten widespread attention in JSSP (Seeger et al., 2022). Training these learning models requires data. The correctness of data has a significant impact on the learning accuracy of the model. That is why several previous studies first solved the JSSP with optimization methods and then used the data of optimal schedules as a learning base for the supervised methods.

Such an approach was taken in Ingimundardottir and Runarsson (2011), where GNU linear programming is used to find the optimal schedule. A logistic regression model based on the optimal model is trained to find the best dispatching rule in each decision instance. Furthermore, Jun et al. (2019) took a similar optimization approach to a flexible job shop scheduling problem (FJSSP). They first used mixed-integer linear programming (MILP) to find the optimal solution as a learning base for the training set. After that, a random forest classifier was introduced to predict the priority in each pair of jobs. However, training the dataset based on an optimization model is not possible for all cases. Solving high-dimension JSSP (with many jobs and machines) with optimization is not always feasible, and simulation can be used to cope with this problem.

2.2 Simulation-based approaches

Dealing with JSSP in real manufacturing processes has always been a challenge for factories as they have dynamic behavior and stochastic process times. That is where the importance of simulation in JSSP emerges. Da Silva et al. (2014) considered random arrival of jobs and stochastic processing time using simulation optimization approaches to find the optimal PDR for the system. Also, Zahmani et al. (2015) took the same path but selected the best PDR for each machine and proved that using a combination of rules (PDRs) outperforms using a single rule. However, simulation optimization can be time-consuming as it should sometimes try all the possible combinations of PDRs to select the best one. Hence, it is not always feasible for real-time decision-making.

Many studies have tried to cope with real-time decision-making. Turker et al. (2019) proposed a simulation decision support system that dynamically changes the PDR based on the number of jobs waiting in the queues. If this number falls within a critical value, the order of jobs is changed using simulation.

Other approaches combined simulation and supervised learning techniques. For instance, Mouelhi-Chibani and Pierreval (2010) integrated a simulation optimization with a neural network classifier to minimize the total flow time in a flow shop scheduling workshop. This method does not need a training dataset as parameters of the neural network are selected over simulation optimization. The

drawback of this model is the time required to run the simulation, especially for more complex problems. Furthermore, Doh et al. (2014) presented simulation optimization on a flexible job shop with multiple process plans to optimize two objectives; total flow time and total tardiness. Based on the simulation, they found the best dispatching rule in a particular process plan. After that, they used the simulation result to train the decision tree classifier to predict the best PDR in each process plan.

Shahzad and Mebarki (2016) combined simulation, optimization, and decision tree to minimize the maximum lateness in a dynamic job shop scheduling problem. They introduced Tabu search, a metaheuristic method, to find the best schedule and then trained the decision tree based on that. The decision tree classifier can predict the precedence of pair of jobs on a specific machine using binary labels.

Studies above presented supervised classifiers to predict the best dispatching rules. Instead of classifiers, Heger et al. (2015) introduced Gaussian regression to see the system's performance with all the PDRs and selected the optimum PDR in each state. They minimized the Mean Tardiness for stochastic and dynamic job-shop scheduling. However, the optimum global decisions are neglected in this study as the authors were focused on the local optimum solution of the current state. A similar approach is taken in our research as regression is selected over classifiers to predict the best PDR. The selection of PDR depends on many factors, such as transportation activities (location of the transporter, transportation time, etc.).

2.3 AGV/AIV scheduling

Unlike the studies above that all considered job scheduling, few studies have been done regarding transportation scheduling inside the shop floors. Different PDRs are frequently used to schedule and assign jobs to AGVs, such as first come, first served (FCFS), shortest traveling distance (STD), longest waiting time (LWT), and nearest vehicle first (NVF), etc. (Hu et al., 2020).

Farahvash and Boucher (2004) introduced an agent-based architecture for AGV transporters. AGV agents were incorporated with three other agents on the shop floor: cell agents, material handling, and scheduling agents. They proved that the performance of the routing (scheduling) method relies on the current state of the AGVs.

Popper et al. (2021) presented a concurrent machine job scheduling and AGV planning. They minimized the makespan and total lateness using multi-agent reinforcement learning (MARL). The state-action pair in the Q-learning approach allows making a decision based on the state's data in each decision instance. Xue et al. (2018) proposed a Q-learning method for multi-AGV flow shop scheduling. The objective is to minimize the average job delays and makespan. Once an AGV is assigned to transport a job to a machine, it will select the job that waits for the longest. Table-based Q-learning algorithms can be effective in simple environments with a small number of discrete states and actions. As the number of states and actions in the environment increases, the state space becomes exponentially larger, which can lead to a dimensionality problem. This means that it becomes increasingly difficult to explore and learn optimal policies in high-dimensional state spaces (Wang et al., 2021).

To deal with the dimensionality problem of traditional table-based Q learning, in Hu et al. (2020), deep reinforcement learning for the real-time scheduling of AGVs in a flexible shop is proposed. The objective is to minimize makespan and delays. The algorithm's actions are a vector that specifies the scheduling PDR and the vehicle to be dispatched by the selected job. Five PDRs are proposed: FCFS, STD, Earliest due date (EDD), LWT, and Nearest Load Point (NLP). Comparison is made with other methods, such as Q-learning and single dispatching rules. Deep Q-networks, however, have a limitation regarding explainability, as the Q-values are acquired through a complex neural network. This makes it challenging to comprehend how the network reaches its decisions (Kayhan and Yildiz, 2021). Moreover, the scalability of action-based reinforcement learning methods is restricted due to their inability to train on small problems with limited workshop layouts and generalize effectively to larger problems with more jobs and varying layout configurations (Kayhan and Yildiz, 2021).

This study proposes simulation-based supervised learning to address explainability and scalability issues in the previous works and increase the workshop layout flexibility.

3. METHODOLOGY

In this section, details of our approach are demonstrated in two phases: (1) Simulation and data generation, (2) using multiple linear regression model to select the PDR that optimizes the performance.

3.1 Data Generation

The first step in applying supervised learning is generating data. The generated data should be large enough to contain most of the possible and random situations of the system. Our approach uses simulation to create the data, as collecting big data from a real system is time-consuming. Furthermore, simulation allows one to consider the dynamic behavior of JSSP by entering stochastic inputs for the system variables.

Two concepts of *state* and *decision* are explained; Each *state* is defined as when AIV receives one/several requests to transfer products and has to make a *decision* which is using a dispatching rule to select which product has priority. States are arrays of data that contain information such as the number of remaining jobs, next processing time of each job, transportation time of each transfer, etc. The main state's data is composed of two types of variables, as shown in Table 1. The first ones are the static or descriptive problem variables, such as the number of jobs. The second type of variables are the dynamic variables; they usually depend on the first type of variables and the current state of the shop, such as AIV location.

The decision is the priority dispatching rule (PDR) in each state, leading to the next job being transferred. Dispatching rules (PDRs) are given randomly as inputs through simulation, and makespan is measured. We propose five

PDRs selected based on previous literature (Haupt, 1989). Priority is with the job :

(1) With Shortest Processing Time (SPT)
(2) With Shortest Remaining Processing Time (SRPT)
(3) Goes to the Shortest Queue Length workstation (SQL)
(4) Goes to the Lowest Mean Utility workstation (LMU)
(5) With Shortest Transfer Time (STT)

For simplicity, the makespan that corresponds to each decision is measured. However, to avoid considering only the actual (local) makespan at each state, the final makespan is also considered and is calculated when the problem simulation is finished. This value is assigned to all the states (decisions) of the problem that led to it. Therefore, at the end of the data generation process, we propose for each state to have three possible evaluations called *Targets*:

- Target 1 = Current (local) Makespan (after taking the current decision)
- Target 2 = Final Makespan (after taking all decisions),
- Target 3 = Target 1 + Target 2

Using this approach, many simulations can be executed. Each simulation instance represents one type of problem (number of jobs, processing times, etc.), and each line of data represents a state (decision) of a problem. Each state contains the corresponding variables (descriptive and dynamic ones), a random PDR, and the three corresponding target values.

3.2 Data Preparation

After generating data, data preprocessing is necessary before applying machine learning methods. Data were generated by simulation, so there are no missing values. Besides, there is no duplicate data as the design of the experiment is applied to create unique problems and states. As the selected machine learning method is linear regression, one-hot encoding should be applied to categorical columns. Finally, the data is classified into five different datasets based on the PDRs. The reason is that the prediction should mainly be influenced by PDRs. It will be helpful to compare the results of applying different PDRs in the same decision instance.

3.3 Multiple Linear Regression

For each PDR dataset, a linear regression (LR) model is trained to predict our proposed targets. The reason for choosing linear regression over other complex regressors, such as neural networks and the random forest, is its advantage in explainability and interpretability (Burkart and Huber, 2021). Furthermore, linear regression is preferred over tree-based models as the relationship between features and target is approximately linear. Features of the linear models are all state data, and the target value is the makespan of the production. Fig. 1 illustrates the linear regression models in a block diagram.

Table 1. Generated data

Problem description	Total number of jobs and machines
	Processing time of each workstation
	Distances between all workstations
Dynamic Variables	Remaining number of Jobs on each workstation
	Current Completion time of each workstation (Ci)
	Current AIV location and transfer time
	Current queue length of each workstation
	Current mean utility of each workstation

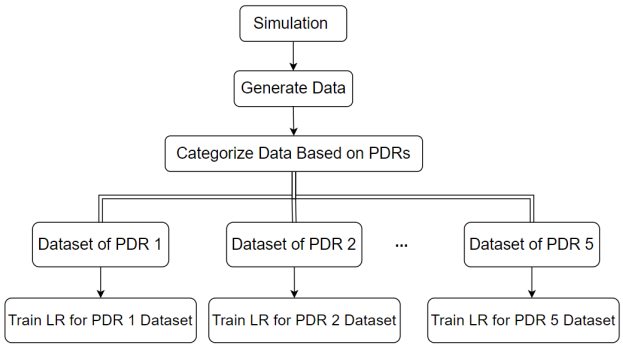


Fig. 1. The block diagram of the proposed simulation-based linear regression method

This will allow us, in real situations, whenever a decision (state) is presented, to use the linear regression model and predict the performance (target) of each PDR at each state. The PDR that presents the best target value is selected for the state. Fig. 2 indicates the process of finding the best PDR once the model is proposed.

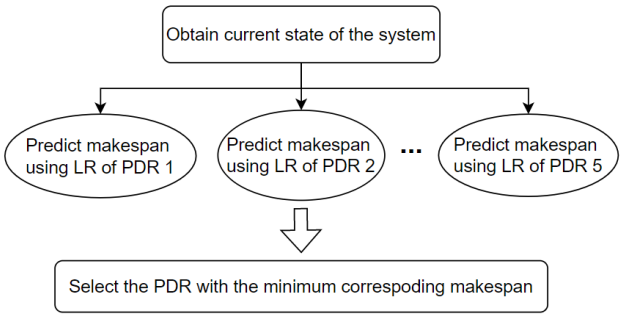


Fig. 2. Process of finding best dispatching rule in each state using Linear Regression (LR) model

4. CASE STUDY

This case study is proposed to test the approach presented in the previous section in a simple workshop with a small number of machines and tasks. It proposes a dynamic stochastic job shop scheduling problem. The job shop consists of three different workstations (W1, W2, Q) and a storage space for raw materials. W1 and W2 are assembling workstations, and Q is a quality check workstation. Two different types of jobs (Job1, Job2) are considered. The first job has two operations: from the storage to assembly W1 and then to the Quality check station (Q). The second job goes from storage to W2 and then to Q. The speed of AIV is assumed to follow a normal distribution with the following parameters (μ = 4m/s, σ = 0.4m/s). A distance matrix D is defined as the distance between all stations. Values of the D matrix can define the layout

of the job shop. There are several numbers of Job1 and Job2, and the workstations' processing times are normally distributed (μ, $\sigma = \frac{\mu}{10}$). Fig. 3 indicates the model of the case study.

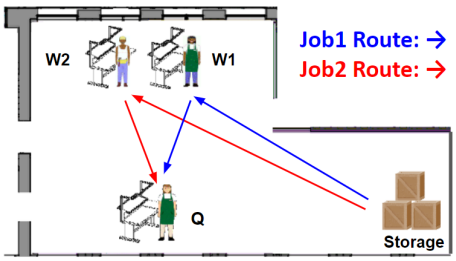


Fig. 3. Routing of Job1 and Job2 on workstations

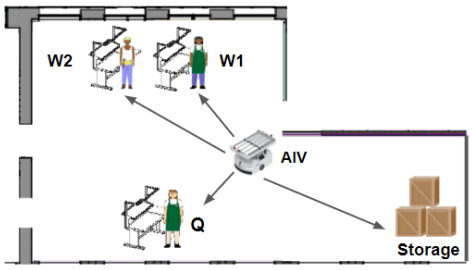


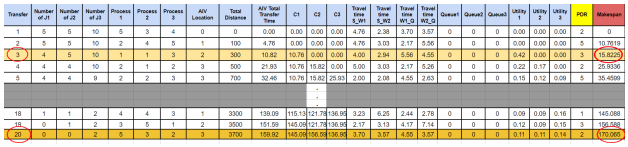
Fig. 4. Decisions of AIV

All the transfers are done by one AIV robot that can handle one product at a time. AIV transmits raw materials of Job1 and Job2 from storage to W1 and W2, respectively. So whenever W1 and W2 require new raw materials for Job1 and Job2 to assemble, they send their request to the AIV robot. Furthermore, whenever W1 and W2 finish their assembly, they will send a transfer request for finished jobs to the Quality Control station. At some point, there will be several transfer requests simultaneously (Fig. 4), and the question is which workstation to serve first. Our objective here is to assist the AIV scheduler in his decision to optimize the system's performance (makespan).

4.1 Data Generation using Arena

Simulation of this case study is made using Rockwell Arena software with VBA blocks. Whenever AIV is free to transfer, there is a state-decision pair. Making decisions in a state leads to a new state. The decision is selecting a PDR in each state, which means choosing the next job to be transferred, which is taken randomly at each state. Thanks to the simulation, 5500 distinct problems are generated using a design of experiment based on the problem description variables (Uy and Telford, 2009). Each problem has a specified number of jobs, processing times, and distances between workstations. The number of arrival jobs ranged randomly between 2 to 50 for job1 and job2, and the mean of process times of the three workstations ranged from 2 to 7 minutes for W1, W2, and Q. Distance between stations ranged from 5m to 100m. The mean speed of the AIV is assumed to be 4m/s for all the problems.

Each simulation creates several rows of state data based on the total number of AIV transfers. In total, more



Transfer	Number of J1	Number of J2	Number of J3	Process 1	Process 2	Process 3	AIV Location	Total Distance	AIV Total Transfer Time	C1	C2	C3	Travel time S_W1	Travel time S_W2	Travel time W1_Q	Travel time W2_Q	Queue1	Queue2	Queue3	Utility 1	Utility 2	Utility 3	PDR	Makespan
1	5	5	10	5	3	4	0	0	0.00	0.00	0.00	0.00	4.76	2.38	3.70	3.57	0	0	0	0.00	0.00	0.00	2	0
2	5	5	10	2	4	5	1	100	4.76	0.00	0.00	0.00	4.76	3.03	2.17	5.56	0	0	0	0.00	0.00	0.00	5	10.7619
3	4	5	10	1	1	3	2	300	10.82	10.76	0.00	0.00	4.00	2.94	5.56	4.55	0	0	0	0.42	0.00	0.00	3	15.8225
4	4	4	10	2	1	2	3	500	21.93	10.76	15.82	0.00	8.00	3.03	2.17	5.26	0	0	0	0.22	0.17	0.00	2	25.9336
5	4	4	9	2	2	3	3	700	32.46	10.76	15.82	25.93	2.00	2.06	4.55	2.63	0	0	0	0.15	0.12	0.09	5	35.4599
18	1	1	2	4	4	3	1	3300	139.09	115.13	121.78	136.95	3.23	6.25	2.44	2.78	0	0	0	0.09	0.09	0.16	1	145.088
19	0	1	2	3	5	1	2	3500	151.59	145.09	121.78	136.95	2.17	3.13	4.17	7.14	0	0	0	0.12	0.09	0.15	3	156.588
20	0	0	2	5	3	2	3	3700	159.92	145.09	156.59	136.95	3.70	3.57	4.55	3.57	0	0	0	0.11	0.11	0.14	2	170.065

Fig. 5. Data of a simulation replication with an example of State 3: Current state with local makespan, State 20: Final state with final makespan

than 1,500,000 rows of state data are generated. Fig. 5 represents an example of the recorded data of one problem out of 5500 problems. In this example, the simulation generated 20 states (decisions), and each state's performance was calculated. For instance, in state 3 (Fig. 5), Target1 (current Makespan) is 15.8225. The final Makespan (Target2) is then added when all decisions are taken (state 20 is the last decision) and equal 170.065. In this case, Target3 equals 15.8225 + 170.065 for the third state. The PDRs were chosen randomly at each state.

4.2 Result of linear regression and Comparison

The Simulation run of 5500 distinct problems in the Arena took around two hours in a PC intel I7. After simulation, multiple linear regression models are trained on the whole generated data to be able to learn various problems related to our job shop configuration in one learning process. This approach increased the scalability of our predictors and significantly reduced the computation times. This study emphasizes the ability of linear models to solve a big scale of variant problems for a specific job shop configuration. In this particular case study, our model can only be used for this specific configuration of one AIV, two types of jobs (job1 and job2), and three workstations plus one storage.

The evaluation of the model is done using three performance metrics:

(1) Accuracy of the prediction model (on 20% of the data),
(2) Comparison between the results of the proposed model and using a constant PDR such as SPT for all decisions or a random mix of PDRs,
(3) Reliability of the prediction model in different sizes of problems (scalability of the model)

For the first phase of evaluation, R-squared and adjusted R-squared are used. Mean values (of the five linear regression models) are represented in Table 2. Adjusted-R^2 is applied to check if adding some features have a significant impact on the prediction (Vittinghoff et al., 2006). Table 2 also indicates that the accuracy of predictions for all proposed models is greater than 90%. Having an evaluation for *Target 1* with a 99% score is logical as the prediction for local makespan is not that difficult once the parameters are given. However, these evaluations are still based on the data and do not guarantee a good performance of the model in real situations. That is why the second evaluation is executed.

For the second phase of evaluation, the performance of the three variants of the proposed method (Target 1, 2, and 3) are compared with existing heuristics in a set of 100 distinct problems of the case study with a different number of jobs, processing times and machines layout (distance

between workstations). These heuristics consist of using a constant PDR for all decisions of a problem. We used the five previously proposed PDRs plus a random combination of these PDRs. In total, 9 strategies were compared. For each one of the 100 problems and each strategy, the final performance (makespan) is measured. Fig. 7 represents the mean of 100 makespans for each strategy.

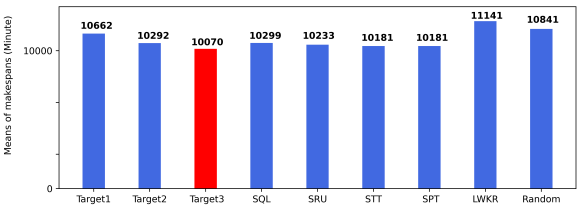


Fig. 6. comparison between the results of the proposed model and using each of the five PDRs

The result indicates that the proposed linear model with Target 3 (local + global makespan) outperforms all the other strategies. Moreover, it indicates the compatibility of the Target 3 model in job shops with flexible layouts as it trained on limited layouts and made decisions for problems with new layouts.

In the third evaluation, the model's effectiveness is tested regarding the size of the problem (number of jobs). For one specific problem with fixed categories, such as process times and distances between workstations, a hundred problems with different numbers of jobs (2 to 300 jobs in total) are tried. This approach compares our best model (Target 3) and two famous constant PDRs (SPT and STT). Target 3 outperforms the two other strategies. The significant point is by increasing the number of jobs, the difference between the performance of Target3 and the two other strategies (SPT, STT) will be augmented. That verifies the scalability of the model Target 3 to be trained on smaller-sized problems and predict larger-sized ones.

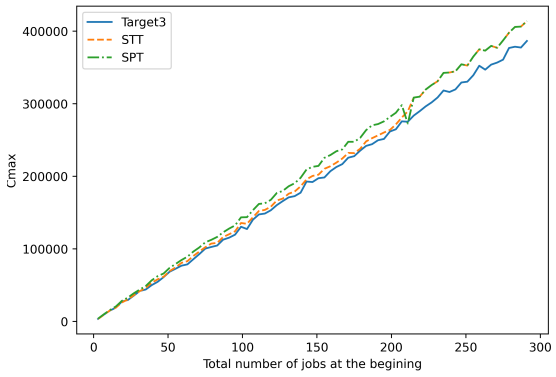


Fig. 7. reliability of the prediction model in different sizes of problems

Table 2. R^2 and Adjusted-R^2 for predictions of makespan

	Target 1	Target 2	Target 3
R^2	0.99	0.92	0.97
Adjusted-R^2	0.993	0.921	0.972

5. CONCLUSION

This paper addresses a dynamic JSSP where there is one AIV transporter to be scheduled on three workstations. A simulation-based supervised learning method is applied to minimize the makespan. Our approach consists of two phases: Generating data by simulation, then using linear regression to predict the best dispatching rule for a current state. In this way, the model compares the selection of different dispatching rules in a particular state and chooses the one that leads to the minimum makespan. A simple case study is developed to apply three different models and compare them with five heuristics. The result shows that one model *Target 3*, outperforms the others.

The advantage of such an approach is that the linear regression models are trained only once on a different range of parameters for a specific type of problem. After that, the model predicts the dispatching rule in a short time. Moreover, there is no need for an optimal schedule to train supervised learning, which gives credit to the model's capability to be applied in dynamic JSSP with many jobs.

However, this approach has its limitations, and different perspectives of this study can be highlighted. The first one consists of testing other models, such as random forest, neural networks, etc., and using different objective functions such as maximum lateness of jobs and energy consumption of AIV. It is interesting to compare these models with a stochastic approach, such as Markov Chain, to evaluate its behavior in unstable environments. The second perspective is to generalize the problem for different job shop configurations with many machines and multiple AIVs. In this case, for each state, two decisions must be taken; selecting the AIV, then choosing the workstation to be served.

ACKNOWLEDGEMENTS

This work is supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-15-IDEX-02)

REFERENCES

Azadeh, A., Negahban, A., and Moghaddam, M. (2012). A hybrid computer simulation-artificial neural network algorithm for optimisation of dispatching rule selection in stochastic job shop scheduling problems. *International Journal of Production Research*, 50(2), 551–566.

Belmamoune, M.A., Ghomri, L., and Yahouni, Z. (2022). Solving a job shop scheduling problem using q-learning algorithm. In *To be apear in the proceeding of 12th international workshop on service oriented, holonic and multi-agent manufacturing systems for industry of the future (SOHOMA)*.

Burkart, N. and Huber, M.F. (2021). A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70, 245–317.

Da Silva, E.B., MGC, M.F.D.S.D., and Silva, F.H.P. (2014). Simulation study of dispatching rules in stochastic job shop dynamic scheduling. *World Journal of Modelling and Simulation*, 10(3), 231–240.

Doh, H.H., Yu, J.M., Kwon, Y.J., Shin, J.H., Kim, H.W., Nam, S.H., and Lee, D.H. (2014). Decision tree based

scheduling for flexible job shops with multiple process plans. *International Journal of Industrial and Manufacturing Engineering*, 8(3), 621–627.

Farahvash, P. and Boucher, T.O. (2004). A multi-agent architecture for control of agv systems. *Robotics and computer-Integrated manufacturing*, 20(6), 473–483.

Farajzadeh, F., Moadab, A., Valilai, O., and K. Moghaddam, S. (2019). Developing a mutual advanced resource planning and reconfigurable manufacturing model: Fulfilling industry 4.0 paradigm.

Haupt, R. (1989). A survey of priority rule-based scheduling. *Operations-Research-Spektrum*, 11(1), 3–16.

Heger, J., Hildebrandt, T., and Scholz-Reiter, B. (2015). Dispatching rule selection with gaussian processes. *Central European Journal of Operations Research*, 23(1), 235–249.

Hu, H., Jia, X., He, Q., Fu, S., and Liu, K. (2020). Deep reinforcement learning based agvs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. *Computers Industrial Engineering*, 149, 106749. doi: https://doi.org/10.1016/j.cie.2020.106749.

Ingimundardottir, H. and Runarsson, T.P. (2011). Supervised learning linear priority dispatch rules for job-shop scheduling. In *International conference on learning and intelligent optimization*, 263–277. Springer.

Jun, S., Lee, S., and Chun, H. (2019). Learning dispatching rules using random forest in flexible job shop scheduling problems. *International Journal of Production Research*, 57(10), 3290–3310.

Kayhan, B.M. and Yildiz, G. (2021). Reinforcement learning applications to machine scheduling problems: a comprehensive literature review. *Journal of Intelligent Manufacturing*, 1–25.

Lang, S., Behrendt, F., Lanzerath, N., Reggelin, T., and Müller, M. (2020). Integration of deep reinforcement learning and discrete-event simulation for real-time scheduling of a flexible job shop production. In *2020 Winter Simulation Conference (WSC)*, 3057–3068. doi: 10.1109/WSC48552.2020.9383997.

Martin, L., González-Romo, M., Sahnoun, M., Bettayeb, B., He, N., and Gao, J. (2021). Effect of human-robot interaction on the fleet size of aiv transporters in fms. In *2021 1st International Conference On Cyber Management And Engineering (CyMaEn)*, 1–5. IEEE.

Mohan, J., Lanka, K., and Rao, A.N. (2019). A review of dynamic job shop scheduling techniques. *Procedia Manufacturing*, 30, 34–39.

Mouelhi-Chibani, W. and Pierreval, H. (2010). Training a neural network to select dispatching rules in real time. *Computers & Industrial Engineering*, 58(2), 249–256.

Mousavi, M., Yap, H.J., Musa, S.N., Tahriri, F., and Md Dawal, S.Z. (2017). Multi-objective agv scheduling in an fms using a hybrid of genetic algorithm and particle swarm optimization. *PloS one*, 12(3), e0169817.

Pickardt, C.W., Hildebrandt, T., Branke, J., Heger, J., and Scholz-Reiter, B. (2013). Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. *International Journal of Production Economics*, 145(1), 67–77.

Pierreval, H. and Mebarki, N. (1997). Dynamic scheduling selection of dispatching rules for manufacturing system. *International Journal of Production Research*, 35(6), 1575–1591.

Popper, J., Yfantis, V., and Ruskowski, M. (2021). Simultaneous production and agv scheduling using multi-agent deep reinforcement learning. *Procedia CIRP*, 104, 1523–1528.

Ramasesh, R. (1990). Dynamic job shop scheduling: a survey of simulation research. *Omega*, 18(1), 43–57.

Seeger, P.M., Yahouni, Z., and Alpan, G. (2022). Literature review on using data mining in production planning and scheduling within the context of cyber physical systems. *Journal of Industrial Information Integration*, 100371.

Shahzad, A. and Mebarki, N. (2016). Learning dispatching rules for scheduling: A synergistic view comprising decision trees, tabu search and simulation. *Computers*, 5(1), 3.

Turker, A.K., Aktepe, A., Inal, A.F., Ersoz, O.O., Das, G.S., and Birgoren, B. (2019). A decision support system for dynamic job-shop scheduling using real-time data with simulation. *Mathematics*, 7(3). doi: 10.3390/math7030278.

Uhlmann, I.R. and Frazzon, E.M. (2018). Production rescheduling review: Opportunities for industrial integration and practical applications. *Journal of manufacturing systems*, 49, 186–193.

Usuga Cadavid, J.P., Lamouri, S., Grabot, B., Pellerin, R., and Fortin, A. (2020). Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. *Journal of Intelligent Manufacturing*, 31(6), 1531–1558.

Uy, M. and Telford, J.K. (2009). Optimization by design of experiment techniques. In *2009 IEEE Aerospace conference*, 1–10. IEEE.

Vittinghoff, E., Glidden, D.V., Shiboski, S.C., and McCulloch, C.E. (2006). Regression methods in biostatistics: linear, logistic, survival, and repeated measures models.

Wang, L., Pan, Z., and Wang, J. (2021). A review of reinforcement learning based intelligent optimization for manufacturing scheduling. *Complex System Modeling and Simulation*, 1(4), 257–270.

Wang, Y.C. and Usher, J.M. (2004). Learning policies for single machine job dispatching. *Robotics and Computer-Integrated Manufacturing*, 20(6), 553–562.

Wang, Y. (2021). Flexible job shop scheduling rules mining based on random forest. In *2021 2nd International Conference on Computing and Data Science (CDS)*, 220–226. IEEE.

Xue, T., Zeng, P., and Yu, H. (2018). A reinforcement learning method for multi-agv scheduling in manufacturing. In *2018 IEEE International Conference on Industrial Technology (ICIT)*, 1557–1561. IEEE.

Yahouni, Z. (2017). *Le meilleur des cas pour l'ordonnancement de groupes : Un nouvel indicateur proactif-réactif pour l'ordonnancement sous incertitudes.* Theses, École centrale de Nantes ; Université Abou Bekr Belkaid (Tlemcen, Algérie).

Zahmani, M.H., Atmani, B., Bekrar, A., and Aissani, N. (2015). Multiple priority dispatching rules for the job shop scheduling problem. In *2015 3rd International Conference on Control, Engineering & Information Technology (CEIT)*, 1–6. IEEE.