AN INTELLIGENT SIMULATION MODEL FOR FLOW SHOP SEQUENCING

BIJAN JAMSHID-NEJAD[1]* , SAMIRA ALVANDI[2]

[1]School of Engineering & Technology
CQUniversity Australia,400 Kent St, Sydney NSW 2000
b.jamshid-nejad@cqu.edu.au

[2]Faculty of Engineering and IT, School of Professional Practice and Leadership,
University of Technology Sydney, Broadway, Sydney, 2000
samira.alvandi@uts.edu.au

ABSTRACT

This study presents an intelligent simulation model for permutation flow shop sequencing of 20 jobs with 5 machines. Reinforcement learning (RL) has been used as a metaheuristic to evaluate and identify optimal heuristic sequencing rules. The model is flexible and can be applied for different sequencing problems with minor programming modification. The RL-based metaheuristics compares 11 sequencing rules to find the best sequencing heuristics. The results demonstrate that heuristics based on shortest processing times outperform other heuristics. Among those high-performing heuristics, the sequencing rule based on shortest processing times of the first two machines generates minimum average makespan in the shortest simulation time. The paper emphasizes the necessity for future investigations to repeat this study for other sequencing and scheduling problems like job shop environments. Overall, this research contributes valuable insights into the application of reinforcement learning algorithms in sequencing, addressing the complexities of modern production systems.

Keywords: Simulation, Flow Shop, Sequencing, Reinforcement Learning

1 INTRODUCTION

In the era of mass customization, agile manufacturing systems should be efficiently responding to market conditions and powerful algorithms are needed to handle ever-increasing complexity of scheduling and sequencing. While mass customization makes scheduling problems more complex, advanced technologies for data collection and storage under what are called Industry 4 and cloud computing offer new rooms for improving scheduling techniques. However, using high-dimensional data for scheduling production systems with multiple objectives like minimizing makespan, reducing production costs is challenging [1]. This situation has been exacerbated with new requirements for sustainability and energy

* Corresponding Author

efficiency. Shorter life cycle for fully customizable products present major challenges to production managers [2].

Sequencing is an integral part of the scheduling process, where tasks should be organized in a specific order to enhance efficiency. It is about allocating some tasks to resources to optimize a criterion like makespan. Broadly speaking, scheduling and sequencing can be applied to a wide variety of situations in theory and practice. Resources can be personnel and tools in a project context or machinery of a manufacturing company. The application can be expanded to operation rooms in hospitals or processing facilities of data centers. Efficient sequencing and scheduling help meet demand on time, minimize waste (e.g. work in progress and waiting time), increase utilization and in general minimize production costs [3].

Because of its wide application, sequencing and scheduling is well researched by mathematical modelling. The issue is that even though ignoring np-hardness of scheduling/sequencing problems, mathematical models are difficult to apply for real cases as they are built on simplifying assumptions which are not valid in real scenarios. Practitioners prefer empirical models and heuristics over complex mathematical models to enhance efficiency and effectiveness by reducing costs, minimizing makespan, and decreasing delays. However, there remains a gap in the availability of practical, off-the-shelf heuristics.

Flow shop environment is an important area of manufacturing scheduling [3]. The sequencing problem in a flow shop environment is how to allocate some resources (hereinafter machines) to some tasks. Sometimes called permutation flow shop scheduling (PFSS) problem, it is categorized as NP-complete problem when there is more than three machines and heuristic and meta-heuristic algorithms have been developed to find near-optimum solutions [4]. The processing route is fixed, and each job should be processed by each machine in order. The processing time is different among jobs but fixed. Job processing cannot be interrupted or split. This is the very basic sequencing problem in a flow shop system.

Despite the advancements in heuristic and meta-heuristic approaches for tackling PFSS problems, there remains a significant gap in the literature in addressing the complexities and dynamic nature of real-world environments. Our research aims to fill this gap by developing a generic model for solving the PFSS problem that can handle a range of scenarios, including typical flow shop production, job-shop environments, and ultimately batch production. The core of our approach is based on a generic RL algorithm to model PFSS problem. This foundation enables our model to be applicable to larger and more complex scheduling contexts. Our approach is highly adaptable, allowing for easy modifications with minimal changes to suit specific problem cases, while also providing a robust platform for comparing different heuristics. This is especially crucial in today's mass customization landscape, where flexibility and responsiveness of companies are key.

This research is to offer one solution for flow shop sequencing problem, in which computer simulations are used to simplify an otherwise np-hard problem. The simulation model we have developed offers two benefits. Firstly, it can be used for specific cases with minor changes in the program. Secondly, it can be applied as a testing lab to compare different heuristics and find the most efficient heuristics in general. In the mass customization environment we are in, these capabilities are of paramount importance. In the following sections of this paper, first a brief review of literature of reinforcement learning (RL) application in sequencing and scheduling is presented, followed by the structure of the model and some results. The

conclusion is the last part in which the main findings and future research directions are highlighted.

2 LITERATURE REVIEW

The literature of heuristics of scheduling and sequencing is rich such as [5-9]. Some simple dispatching rules used in scheduling are first come first served, shortest processing time, earliest due date but application of machine learning in production systems is getting momentum behind modelling and visualization techniques [2]. Smart planning and scheduling are among highly researched areas of Industry 4.0 [10]. Production planning and control is the overarching domain that include facility resource planning, capacity planning, purchase and supply management, production scheduling, and inventory management [11]. Among those, RL has been applied mostly for production planning and control [3, 11]. Reinforcement learning is a class of machine learning models by which an agent learns to take actions in an environment using feedback from its actions to improve its performance. Q-learning, temporal difference and deep Q network are the three most used algorithms in RL [3]. In this study RL is used as a metaheuristic to compare other heuristics and find the best one. Metaheuristics are classes of heuristics based on a fundamental algorithm. Some of other metaheuristics applied in scheduling are simulated annealing, tabu search, genetic algorithms, ant colony optimization, and scatter search. Metaheuristics should represent a final solution, an initial solution or population generation, diversification mechanisms, acceptance criterion, and termination criterion [12].

Characteristics of application of RL in sequencing and scheduling that should be understood for implementation are method, states, actions, reward [11]. Value-based methods like Q-learning, TD(λ) algorithm, and SARSA are more employed than policy-based methods in production scheduling [11].

To name an application of RL in sequencing, Yan et al. [4] developed a deep RL model for job shop sequencing considering machine availability. They applied the developed algorithm to joint production-maintenance scheduling problem and compared the performance to alternative genetic algorithm and iterated greedy algorithms. Deep RL uses collected sensor data from the environment to make possible a flexible, real-time control of the production system.

Han, Guo, and Su [13] presented a reinforcement learning algorithm for a hybrid flow shop scheduling problem. A significant strength of this work is its validation of the reinforcement learning (RL) method through practical applications in industrial settings. Specifically, the RL approach was employed for scheduling in a metal processing workshop at an automobile engine factory, as well as for sortie scheduling of carrier aircraft in continuous dispatch. However, the study falls short of demonstrating the adaptability of their proposed RL algorithm, as it specifically addresses the challenges of hybrid flow-shop scheduling and is not generalizable to other scenarios, such as changes in production volume, machine breakdowns, or variations in job priorities.

Fonseca-Reyna et al. [14] analysed the effects of parameters on learning processes. However in their work, there is limited investigation into the adaptability of RL models when faced with changing production conditions or objectives. It is essential to develop reinforcement learning strategies that can quickly adapt to new information and unforeseen changes in a production environment.

Arviv, Stern, and Edan [15] applied collaborative reinforcement learning to a two-robot flow-shop scheduling problem, introducing a dual Q-learning algorithm. While this method may effectively address the specific problem at hand, the paper lacks a framework for adapting the algorithm to other production scenarios, such as multi-robot systems or different job types. This specificity raises concerns about the algorithm's generalizability.

Brammer, Lutz, and Neumann [16] addressed PFSS in multiple production line environment with varying demand plans and propose a reinforcement learning (RL) approach to optimize scheduling. While it has notable strengths, the model requires substantial modifications to the action space in order to make its approach adaptable to the general flow shop problem.

Our brief review of literature demonstrates that specific application of RL for PFSS problems are few and the existing works have limited application in the broad range of PFSS problems. Therefore, there is a gap in the literature in this area and this research is to fill this gap.

The structure of the model is explained in the following section. The flow shop problem we have looked at is sequencing 20 jobs with five machines.

3 MODEL STRUCTURE

The simulations model has been developed based on the structure depicted in Figure 1.

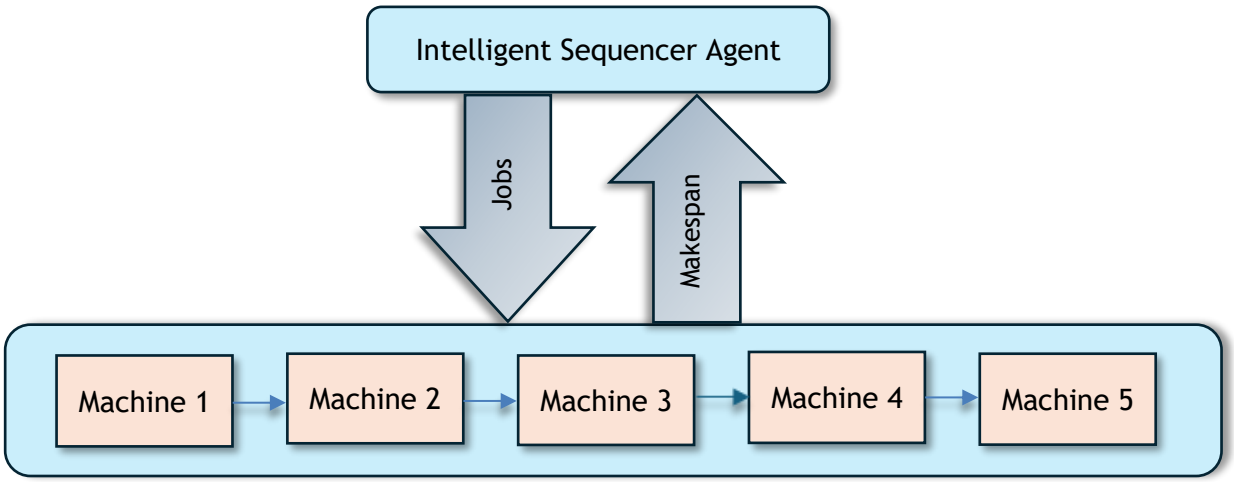

Figure 1: Structure of the Simulation Model

The pseudo code of the simulation model is as follows:

Initialization

- Input Jobs= N
- Input Machines M= the number of operations is the same as number of machines.

Main

Do:

(1) The *Sequencer* generates N random jobs.

(2) The *Machine 1* picks one of them based on a sequencing rule and send it to the *Machine*. This continues until the *Machine M* finishes a job and send it back to the *Sequencer*

(3) The *Sequencer* calculates the makespan of the allocation and updates the average makespan and its preference levels.

(4) The *Sequencer* selects a new sequencing rule based on its reinforcement learning algorithms and let *Machine 1* start a new set of jobs.

While:

The stop conditions are not satisfied (i.e. one specific sequencing rule becomes dominant.)

The sequencing problem we are solving has the following characteristics:

(1) Jobs are independent and can be processed at time zero and each machine can only process one job at the same time.

(2) Each job needs to be processed on each machine only once with no interruption. The transportation and loading/unloading times are included in the processing time.

The reward of each rule (the reward) appears in the form of a rise or decline of average of makespan, incremented by the following equation:

$$\overline{r_{t+1}} = \overline{r_t} + \alpha[r_t - \overline{r_t}] \quad (1)$$

where α is step-size parameter and r_t is the makespan of current run. Each sequencing rule has a numeric preference level ($p_t(a)$), incremented in each sequencing round according to the following equation:

$$p_{t+1}(a) = p_t(a) + \beta[r_t - \overline{r_t}] \quad (2)$$

where ß is another step-size parameter. The preference of selecting successful sequencing rules increases gradually, which in turn increases the probability of their selection according to the Gibbs distribution:

$$Pr\{a_t = a\} = \frac{e^{p_t(a)}}{\sum_{b=1}^{n} e^{p_t(b)}} \quad (3)$$

This equation shows that taking action a at time t depends on the preference of action a over the preference of all other actions.

The default parameters of the simulation are shown in

Table 1.

Table 1- Default Parameters of Simulation

Sequencing Scenarios	20 Jobs with 5 machines		
	'Low' Scenario	'Moderate' Scenario	'High' Scenario
Step-size parameter for makespan (α)	0.1	0.4	0.8
Step-size parameter for sequencing rules' preferences (β)	0.2	0.5	0.9

The rules play the role of actions in reinforcement learning here. Each rule gives feedback in the form of job makespan. The list of sequencing rules studied in this research can be found in

Table 2. Rules 1 to 8 are variations of well-known dispatching rules based on longest process times(LPT) and shortest process times(SPT).

Rule 9 and Rule 10 are designed with the analogy from assembly line balancing and resource levelling. Sequencing of jobs in our setting resembles a production line in which to minimize waiting times (which is translated into minimum makespan ultimately), it is crucial to have close processing times in each job. Similarly in resource leveling there is a metric called Resource Improvement Coefficient (RIC) which quantifies the effectiveness of resource allocation strategies by reducing peaks and troughs of resource demands. Here the machines play the role of resource in our case. Based on these analogies, rule 9 and 10 have been defined. Based on these rules, the jobs are sorted based on uniformity of their process times. The more uniform the process times are, the more balanced this production line would be or put another way, the less peaks and troughs we have in our resource (i.e. machine) demand. The uniformity measure is called Uniformity Measure (UM) and it is calculated as below:

$$UM = \frac{n*\sum_1^n {R_i}^2}{(\sum_1^n R_i)^2} \quad (4)$$

Where R_i is the process time of Machine i and n is the number of machines. Jobs with less UM are more uniform. In other words, their process times are more uniform.

Rule 11 is based on the closeness principle used in Rule 9 and Rule 10. The difference is that jobs are compared two by two based on the uniformity measure. An extension of this research would be defining new rules based on other distance measures which will be discussed in the conclusion section.

Table 2- The Sequencing Rules

Rule	Description
0	The jobs are sequenced randomly.
1	Pick the job with Longest Processing Time (LPT) on Machine 1
2	Pick the job with Shortest Processing Time (SPT) on Machine 1
3	Pick the job with LPT on Machine "1 + 2" together
4	Pick the job with SPT on Machine "1 + 2" together
5	Pick the job with LPT on Machine "1 + 2 + 3" together
6	Pick the job with SPT on Machine "1 + 2 + 3" together
7	Pick the job with LPT of Machines "1 to 5" together
8	Pick the job with SPT of Machines "1 to 5" together
9	Pick the job with maximum UM
10	Pick the job with minimum UM
11	Pick the pair of jobs with minimum UM

4 RESULTS AND DISCUSSION

To program the simulation model, REPAST (REcursive Porous Agent Simulation toolkit) platform is used. REPAST is an open-source platform in which snippets from different programming languages like JAVA and Python can be included [17]. To implement our model, we add some JAVA snippets to simulate sequencing of 20 jobs with 5 machines. The learning agent is called Sequencer in Figure 1. This agent allocates jobs to Machine 1 based on the performance of different sequencing rules and over time learns the most efficient rules. As no job interruption is allowed, Machine 1 allocation is the key decision-making factor for sequencing.

Table 3 and Figure 3 Figure show average makespan of different sequencing rules under 'Low' scenario. As seen, Rule 4 has the best performance with an average of 1407. Overall, Rule 2 and Rule 6 show some good performance as well which are all variation of SPT heuristic.

Table 3: Results of Different Rules

Rule	0	1	2	3	4	5	6	7	8	9	10	11
Average Makespan	1512	1682	1436	1649	1407	1623	1431	1518	1522	1520	1516	1516

To assess the performance of Rule 4 against other available solutions, we refer to some test problems listed in Taillard [18]. As shown in Table 4, the best makespan reported in the 20 job-5 machines test problems is 1073, whereas Rule 4 in our results has the minimum 1058 makespan.

Table 4: Comparison of RL Results with Other Heuristics

Test Problem	Lower Bound	RL Best Makespan
20 jobs and 5 machines	1073	1058 with Rule 4

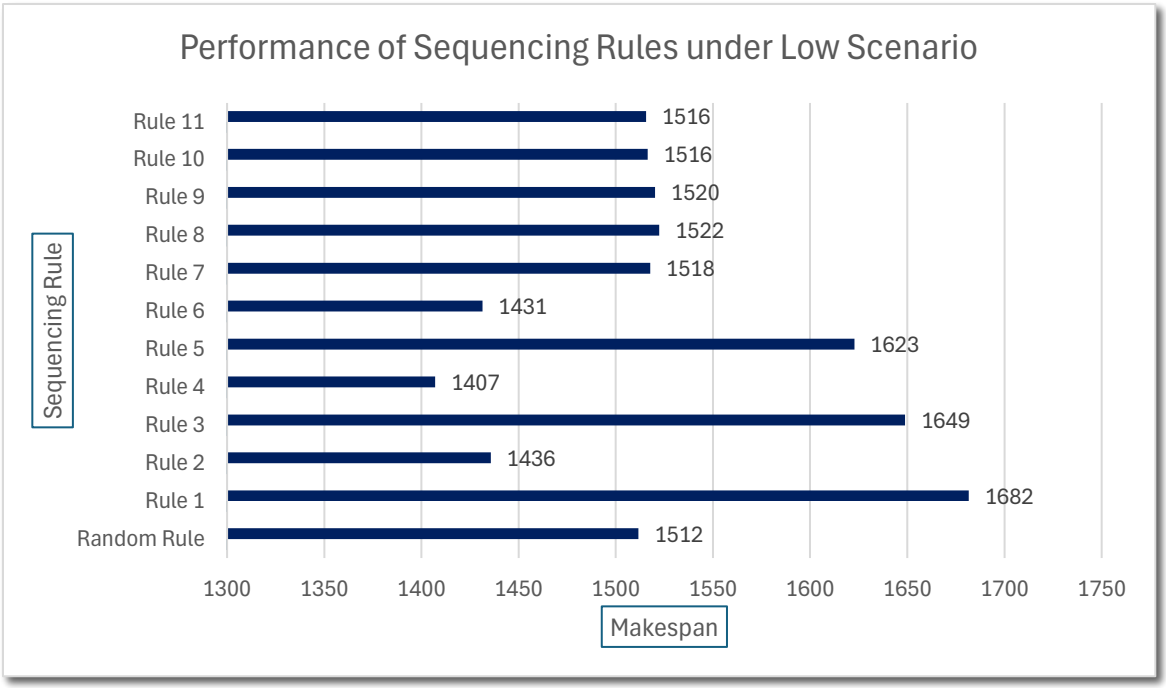

Figure 3: Average Makespan of Sequencing Rules under 'Low' Scenario

Figure 4 depicts simulation time to find best results. From the sample taken, it is seen that the average simulation time of Rule 4 is less than Rule 2 which is the only competitor.

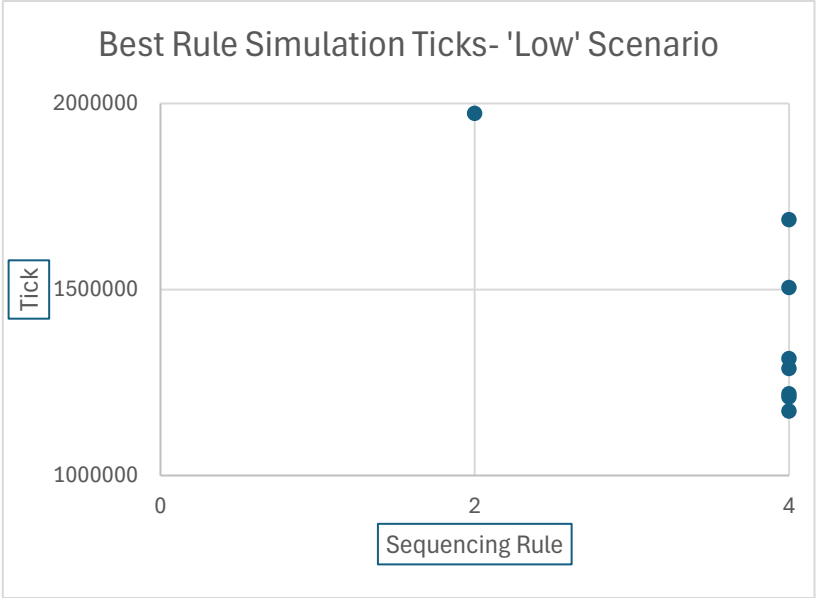

Figure 4: Best Rules' Simulation Ticks under 'Low' Scenario

The results of 'Moderate' and 'High' scenarios, as shown in Figure 5 and Figure 7 are almost the same as 'Low' scenario and Rule 4 demonstrates better performance over other sequencing rules tested. The only observed difference is that the dominance of rule 4 is less overwhelming. As regards simulation time, again the simulation model learns Rule 4 faster than others as shown in Figure 6 and Figure 8.

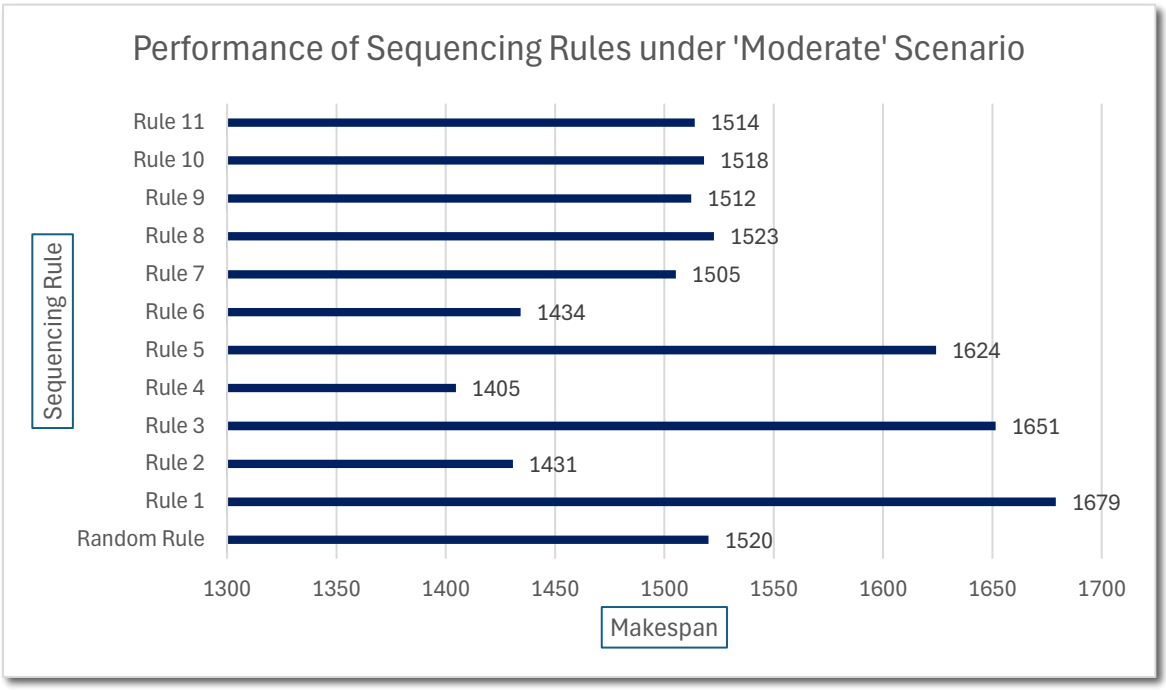

Figure 5: Performance of Sequencing Rules under 'Moderate' Scenario

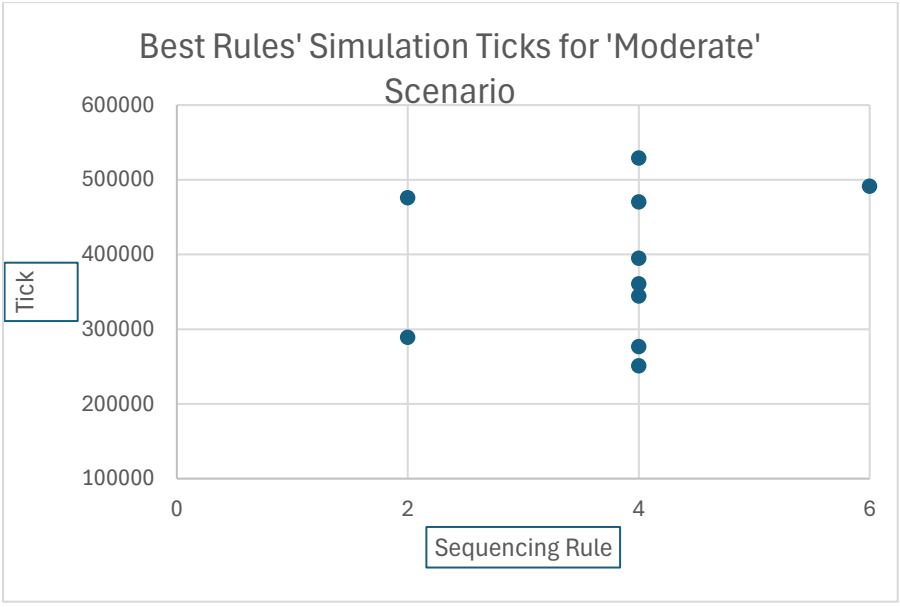

Figure 6: Best Rules' Simulation Ticks under 'Moderate' Scenario

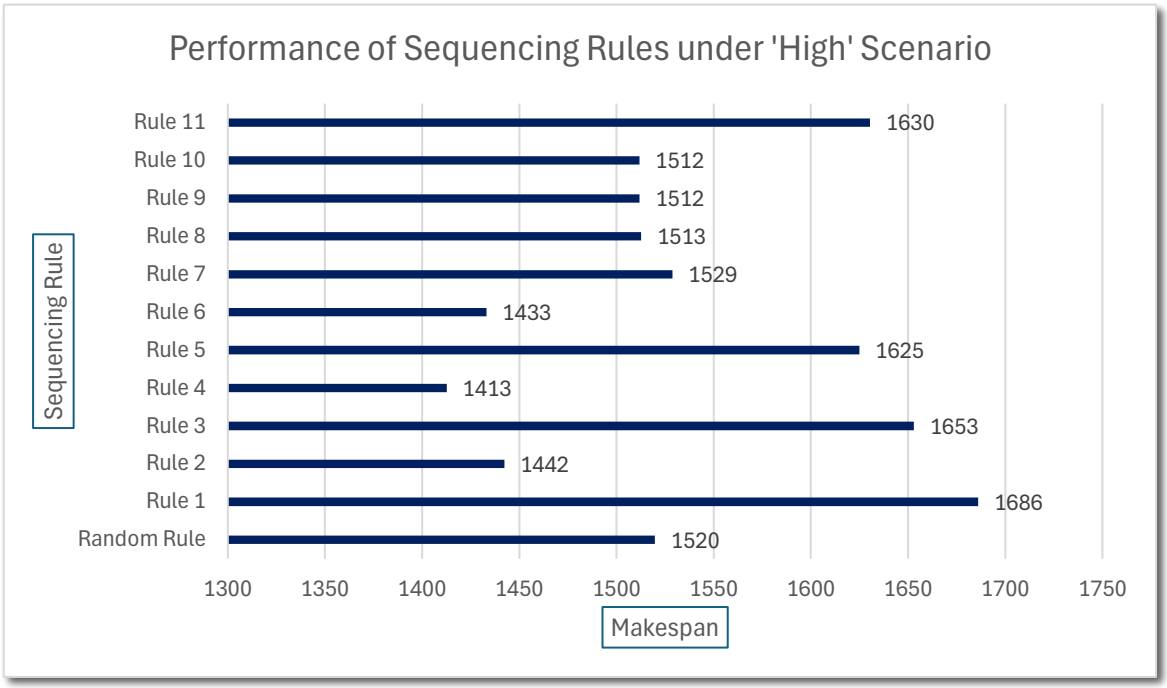

Figure 7: Performance of Sequencing Rules under 'High' Scenario

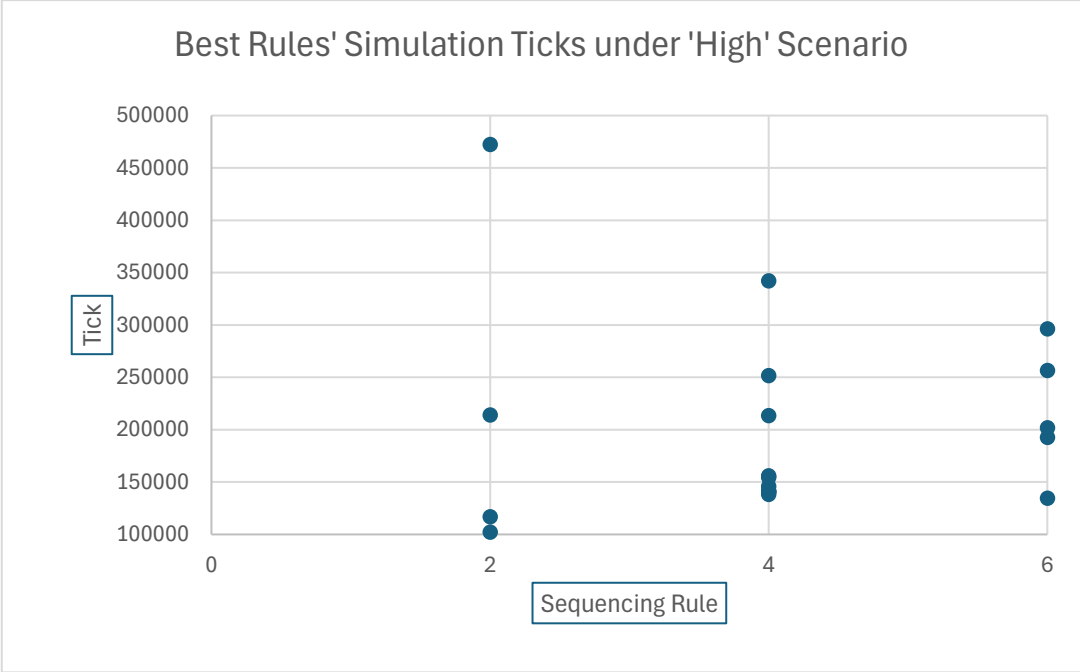

Figure 8: Best Rules' Simulation Ticks under 'High' Scenario

To validate these results, the same algorithm has been tested for the problem of 50 jobs and 5 machines. As depicted in Figure 10, Figure 9, and Figure 11. The results show that like 20 jobs- 5 machines problem, Rule 4 and Rule 6 are better rules with different learning parameters. However, Rule 4 outperforms narrowly Rule 6, and this outcome is consistent across 'Moderate' and 'High' scenarios.

5 Conclusion

This study presents an intelligent simulation model for flow shop sequencing, utilizing a reinforcement learning algorithm. The RL is used to compare and identify best sequencing heuristics among 11 sequencing rules. The main contribution of this research is that it offers a RL-based framework that can be easily adapted to compare other sequencing heuristics. Also, it can be applied for specific cases with minor program modifications.

The study demonstrates that among those 11 sequencing rules, Rule 4 is the best. Based on this rule, jobs are sorted based on the shortest processing times of the first two machines. Also, other variations of SPT (i.e. Rule 2 and Rule 6), consistently outperforms other tested rules across different scenarios. Another advantage of Rule 4 is its computational efficiency, measured by simulation time. On average, Rule 4 has a shorter simulation time across different 'Low', 'Moderate', and 'High' scenarios.

There could be some future research directions upon the results of this research. The first possibility is the application of the framework for other sequencing problems like 50 jobs-10 machines and 100 jobs-10 machines. Conducting those studies with rigorous statistical analysis, help us validate the results of this study. The second area is developing a simulation model in which heuristics are generated automatically and tested with this RL-based framework. Upon achieving these objectives, the intelligence of the framework facilitates its application for more general problems. Another extension of this research is the application of the RL-based framework for other environments like job shop scheduling and batch production to tackle real-world, multi-objective problems. An opportunity exists to enhance the current RL model to balance multiple objectives—such as minimizing tardiness while maximizing utilization.

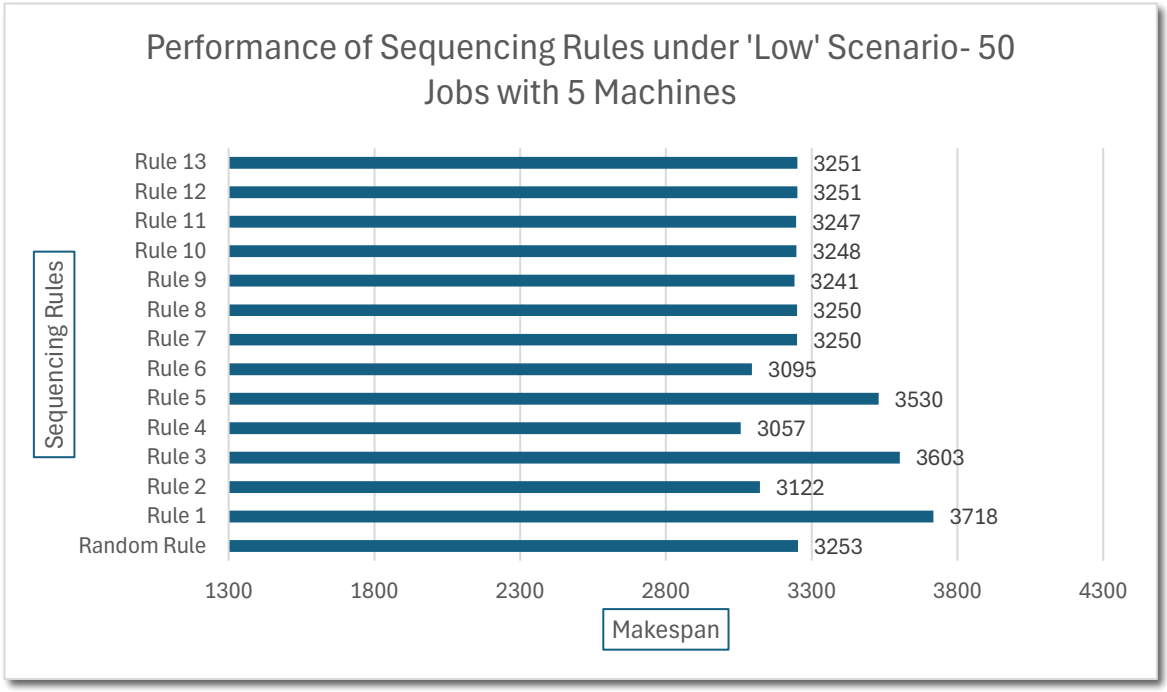

Figure 10: Performance of Different Sequencing Rules under 'Low' Scenario for 50 Jobs-5 Machines Problem

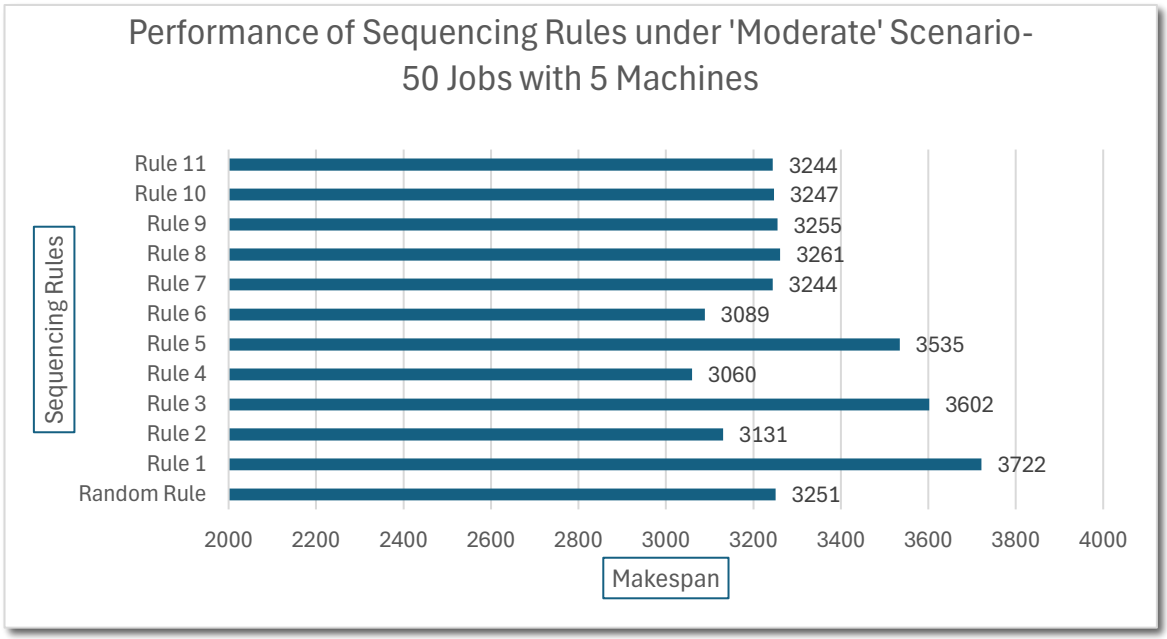

Figure 9: Performance of Different Sequencing Rules under 'Moderate' Scenario for 50 Jobs-5 Machines Problem

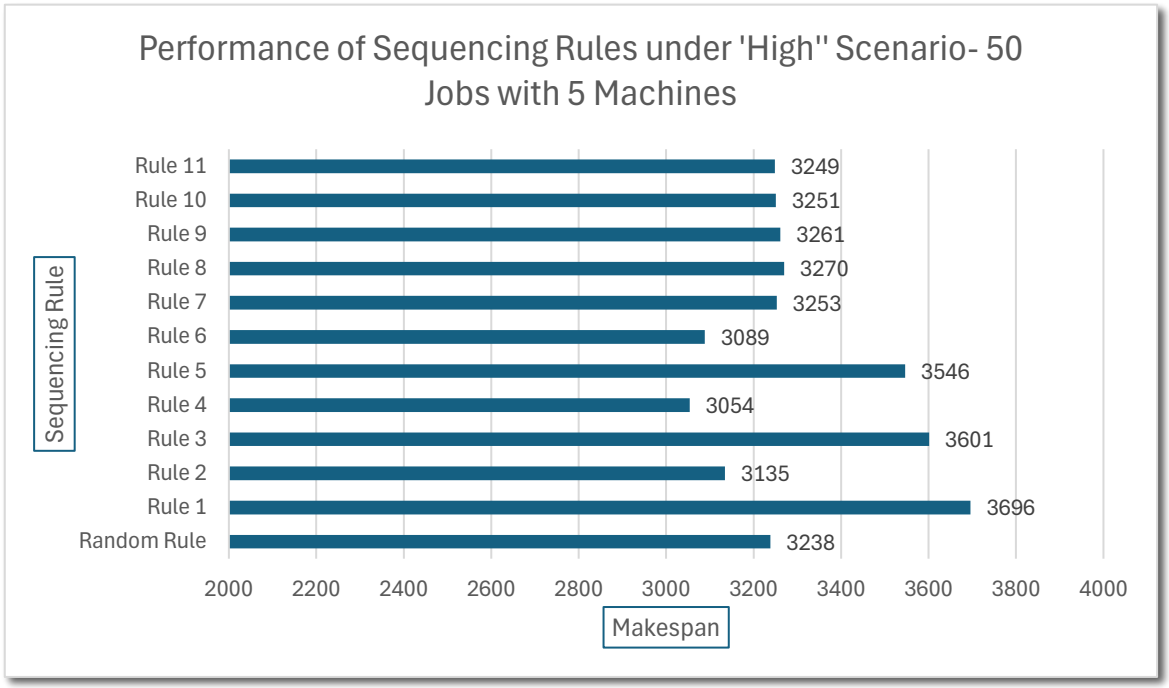

Figure 11: Performance of Different Sequencing Rules under 'High' Scenario for 50 Jobs-5 Machines Problem

6 REFERENCES

[1] T. Zhou, D. Tang, H. Zhu, and L. Wang, "Reinforcement learning with composite rewards for production scheduling in a smart factory," *IEEE Access,* vol. 9, pp. 752-766, 2020.

[2] M. Panzer, and B. Bender, "Deep reinforcement learning in production systems: a systematic literature review," *International Journal of Production Research,* vol. 60, no. 13, pp. 4316-4341, 2022.

[3] B. M. Kayhan, and G. Yildiz, "Reinforcement learning applications to machine scheduling problems: a comprehensive literature review," *Journal of Intelligent Manufacturing,* vol. 34, no. 3, pp. 905-929, 2023.

[4] Q. Yan, W. Wu, and H. Wang, "Deep reinforcement learning for distributed flow shop scheduling with flexible maintenance," *Machines,* vol. 10, no. 3, pp. 210, 2022.

[5] A. Mishra, and D. Shrivastava, "A TLBO and a Jaya heuristics for permutation flow shop scheduling to minimize the sum of inventory holding and batch delay costs," *Computers & Industrial Engineering,* vol. 124, pp. 509-522, 2018/10/01/, 2018.

[6] S.-y. Wang, L. Wang, M. Liu, and Y. Xu, "An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem," *International Journal of Production Economics,* vol. 145, no. 1, pp. 387-396, 2013/09/01/, 2013.

[7] X. Wu, and A. Che, "Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search," *Omega,* vol. 94, pp. 102117, 2020/07/01/, 2020.

[8] M. M. Yenisey, and B. Yagmahan, "Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends," *Omega,* vol. 45, pp. 119-135, 2014/06/01/, 2014.

[9] G. I. Zobolas, C. D. Tarantilis, and G. Ioannou, "Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm," *Computers & Operations Research,* vol. 36, no. 4, pp. 1249-1267, 2009/04/01/, 2009.

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

[11] A. Esteso, D. Peidro, J. Mula, and M. Díaz-Madroñero, "Reinforcement learning applied to production planning and control," *International Journal of Production Research,* vol. 61, no. 16, pp. 5772-5789, 2023.

[12] R. Ruiz, "Scheduling Heuristics," *Handbook of Heuristics*, R. Martí, P. Panos and M. G. C. Resende, eds., pp. 1-24, Cham: Springer International Publishing, 2016.

[13] W. Han, F. Guo, and X. Su, "A reinforcement learning method for a hybrid flow-shop scheduling problem," *Algorithms,* vol. 12, no. 11, pp. 222, 2019.

[14] Y. C. Fonseca-Reyna, Y. Martínez-Jiménez, and A. Nowé, "Q-learning algorithm performance for m-machine, n-jobs flow shop scheduling problems to minimize makespan," *Investigación Operacional,* vol. 38, no. 3, pp. 281-290, 2018.

[15] K. Arviv, H. Stern, and Y. Edan, "Collaborative reinforcement learning for a two-robot job transfer flow-shop scheduling problem," *International Journal of Production Research,* vol. 54, no. 4, pp. 1196-1209, 2016.

[16] J. Brammer, B. Lutz, and D. Neumann, "Solving the mixed model sequencing problem with reinforcement learning and metaheuristics," *Computers & Industrial Engineering,* vol. 162, pp. 107704, 2021.

[17] M. J. North, N. T. Collier, J. Ozik, E. R. Tatara, C. M. Macal, M. Bragen, and P. Sydelko, "Complex adaptive systems modeling with Repast Simphony," *Complex Adaptive Systems Modeling,* vol. 1, no. 1, pp. 3, 2013/03/13, 2013.

[18] E. Taillard, "Benchmarks for basic scheduling problems," *european journal of operational research,* vol. 64, no. 2, pp. 278-285, 1993.