Permutation in shop scheduling problems with FIFO considerations

Randa Ouchene[1], Djamal Rebaine[1], Pierre Baptiste[2]

***Abstract*— Traditionally, permutation scheduling in Flow Shop problems have been used as maintaining the same task order across all the machines. However, this definition becomes ambiguous when operations are missing [16]. Alternative definitions, such as the FIFO-based definition introduced by Pinedo [18], are more suitable in those contexts. The present study first analyzes Pinedo's definition in Flow Shop problems with missing operations, then extends this approach to Job Shop scheduling, examining its implications with respect to the quality of the solutions.**

I. Introduction

Shop scheduling problems occupy a pivotal role in both industrial [15], [24], [6], [11], [25], [21], [9], [22] and service sectors [5], [4], [13], [26]. These problems involve scenarios where tasks, composed of multiple operations, must be executed across various machines. In a Flow Shop, each job follows the same predetermined machine sequence. Conversely, in a Job Shop environment, the routing is job-specific: each job may follow a different sequence of machines depending on its individual requirements.

In both industrial and service settings, it is common for certain jobs to skip some machines, leading to the presence of missing operations. For instance, in hospital services, one patient might undergo multiple diagnostic tests, while another diagnosed with a different condition might proceed directly to a specialized service. Similarly, in appointment scheduling systems [26], patients follow personalized paths based on their medical needs, resulting in heterogeneous routing and partially defined job structures.

There are two well-known ways to interpret missing operations. The first, proposed by Potts et al. [19], treats missing operations as tasks with negligible but non-zero duration, requiring jobs to be assigned to every machine. The second interpretation, which we adopt in this work and which aligns with [27], [23], [28], [17], considers a missing operation to be entirely absent from the job's routing.

In classical Flow Shops with missing operations, permutation schedules are traditionally defined as those that preserve a global ordering of jobs across all machines. However, alternative definitions have been proposed. One approach is based on the "no-overtaking" principle: if job A precedes job B on one machine, it must also precede it on all other machines they both visit. Another interpretation, introduced by Pinedo [18], defines permutation scheduling according to a FIFO (First-In, First-Out) discipline: a job that finishes earlier on one machine should be scheduled earlier on the next.

In service environments such as healthcare and public administration, FIFO rules play a critical role in managing resources fairly and transparently. By preserving the order of arrival, they help ensure equity among users and contribute to higher levels of satisfaction [7], [20]. Moreover, minimizing waiting time is a crucial objective in these settings, as it directly affects user experience and operational efficiency ([1]). However, when jobs follow personalized and incomplete routing paths, as is often the case in service systems, enforcing FIFO discipline across shared machines presents significant modeling and scheduling challenges.

This paper addresses two related but distinct objectives:

- to analyze the theoretical properties of FIFO-based permutation schedules in Flow Shops with missing operations, and
- to extend this FIFO-based permutation concept to Job Shop environments, through mathematical modeling and experimental analysis.

We first explore, in Section II, the applicability and implications of Pinedo's FIFO definition in Flow Shops with missing operations. Although limited to a specific machine structure, this analysis offers valuable insights into the behavior of partially routed jobs under FIFO constraints.

In Section III, we generalize this perspective to the Job Shop setting, where routing flexibility introduces new modeling complexities. We propose a MIP formulation that explicitly incorporates FIFO constraints between machines. This model allows us to evaluate the impact of FIFO enforcement on performance indicators such as makespan and waiting time.

Finally, Section IV summarizes the findings and outlines potential directions for future research.

II. Pinedo permutation for Flow Shop with missing operations

Permutation flow shop scheduling is traditionally defined by imposing the same task sequence across all machines. However, as highlighted by [16], this classical definition becomes ambiguous when considering missing operations. Consequently, two variants arise:

- A **strong permutation** imposes a global ordering of tasks across all the machines, irrespective of missing

[1]Département d'informatique et de mathématique, Université du Québec à Chicoutimi, Saguenay (Québec), Canada. CONTACT Randa Ouchene. Email: randa.ouchene1@uqac.ca, CONTACT Djamal Rebaine. Email: drebaine@uqac.ca

[2]Département de mathématiques et de génie industriel, École Polytechnique Montréal, Montréal (Québec), Canada. CONTACT Pierre Baptiste. Email: pierre.baptiste@polymtl.ca

operations (illustrated in Figure 1, where the global order is 2, 1, 3).

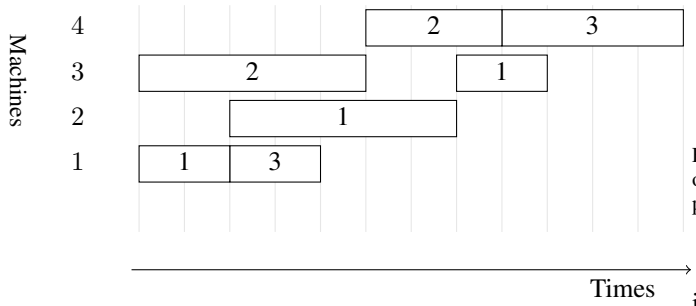

Fig. 1. Illustration of strong permutation scheduling

- A **weak permutation** is defined by a non-overtaking rule: if task i precedes task j on one machine, it must also precede task j on every subsequent machine. Figure 2 illustrates a weak permutation scenario that does not satisfy the strong permutation definition (in this instance, job 1 precedes job 3, job 2 precedes job 1, and job 3 precedes job 2).

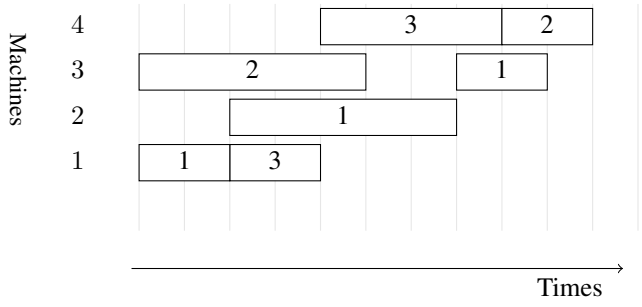

Fig. 2. Illustration of weak permutation scheduling

Moreover, Pinedo (see [18]) introduces a permutation definition based on a FIFO (First In, First Out) discipline for managing machine queues. This FIFO-based definition coincides with the classical permutation definition in flow shops without missing operations, but it may diverge when certain jobs skip some machines.

In this work, the FIFO rule ensures that if a job finishes its operation earlier on a preceding machine, it must also be executed earlier on the next common machine. This ordering constraint can be formalized as follows. Let J_1 and J_2 be two jobs, and k_1, k_2, and h be machines such that:

- J_1 is processed on machine k_1, then on machine h,
- J_2 is processed on machine k_2, then on machine h,
- h is the next immediate machine for both jobs in their respective routes.

The FIFO rule states that if job J_1 completes on machine k_1 before job J_2 completes on machine k_2, then job J_1 must be completed before job J_2 on machine h.

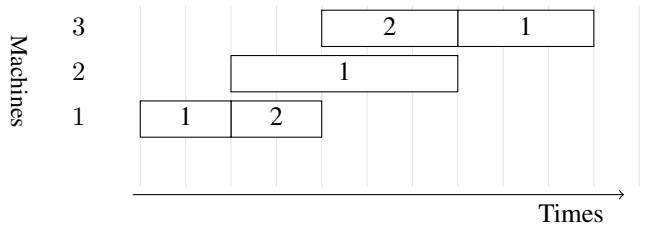

Fig. 3. Pinedo's permutation scheduling for a Flow Shop with missing operations, illustrating a scenario that is neither a weak nor a strong permutation.

Figure 3 illustrates an optimal solution for an instance involving three machines and two jobs, where the objective is to minimize the makespan. Notably, this solution adheres neither to a weak nor a strong permutation, indicating that, under either the weak or strong permutation definitions, permutation scheduling does not dominate[1], even in a three-machine scenario. In the following discussion, we will demonstrate that when considering Pinedo's definition, permutation becomes dominant for three machines. This highlights the divergence among the various definitions of permutation.

In the following, we illustrate several properties of permutation flow shop scheduling as defined by Pinedo, specifically within contexts where certain operations may be missing.

Theorem 2.1: In a permutation flow shop problem with missing operations, scheduling tasks according to a FIFO discipline on the last machine M_m is dominant under the makespan criterion.

Proof: Consider a flow shop system with missing operations. When focusing specifically on scheduling tasks on the last machine M_m, the problem reduces to a single-machine scheduling problem with release dates, $1 \mid r_i \mid C_{\max}$, where the release date r_i of each task corresponds to its completion time on preceding machines.

The optimal solution for the $1 \mid r_i \mid C_{\max}$ problem involves sequencing tasks in ascending order of their release dates. Thus, scheduling tasks on machine M_m following the FIFO (First In, First Out) discipline does not increase the makespan and is therefore dominant.

■

Corollary 2.2 (Two-machine case): In a two-machine permutation flow shop scheduling problem with missing operations, sequencing tasks according to Pinedo's FIFO discipline on the second machine is dominant under the makespan criterion.

Theorem 2.3: In a two-machine permutation flow shop problem minimizing makespan, schedules respecting FIFO discipline on the second machine are dominant.

Proof: Tasks arrive in the queue of the second machine either from machine 1 or directly if machine 2 corresponds to

[1]A given permutation definition is said to be dominant if, for any feasible schedule that does not satisfy it, there exists a schedule that satisfies this definition and achieves a makespan that is less than or equal.

their first operation. To ensure that the queue on the second machine respects the FIFO discipline, we must show that if a task j finishes after a task i on machine 1, then task i must start processing before task j on machine 2.

Consider the first pair of tasks (i, j) such that task i precedes task j on machine 1, but task j precedes task i on machine 2. Suppose that tasks $i_1, \ldots, i_k$ lie between i and j on machine 1. To correct this ordering, we remove task i from the queue on machine 1, shift tasks $i_1, \ldots, i_k$ and task j forward, and then insert task i immediately after task j. This operation restores the correct order (i, j) on both machines without loss of feasibility or increase in makespan.

Repeating this procedure for all improperly ordered task pairs, we obtain a schedule that respects the FIFO discipline on machine M_2, thereby proving the dominance of this scheduling approach. ■

Theorem 2.4: For a three-machine flow shop problem with missing operations with respect to the makespan criterion, Pinedo permutation schedules, which enforce FIFO discipline on machines 2 and 3, are dominant.

Proof: In a three-machine flow shop setting, ensuring a FIFO queue discipline on machines 2 and 3 aligns with dominance conditions demonstrated in Theorems 2.1 and 2.3. Combining these two results confirms that FIFO-based schedules on M_2 and M_3 are dominant regarding makespan. ■

Remark: In the case without missing operations, Theorem 2.4 aligns with the classical result presented in [8], which states that permutation schedules are dominant in three-machine flow shop problems under the makespan criterion.

In this section, we examined the impact and relevance of imposing the permutation constraint defined by Pinedo (based on the FIFO rule) in the general flow shop setting with missing operations. We now extend our analysis to a more general framework: the job shop scheduling problem, where FIFO constraints are integrated into the mathematical model. We propose a mathematical formulation that incorporates these constraints and investigate their impact on the makespan and waiting time.

III. Adapting Pinedo's permutation concept to the Job Shop environment

The objective of this section is to analyze how FIFO constraints can be enforced in job shop scheduling problems, and to evaluate their impact on makespan performance. Although job shops differ structurally from flow shops, the FIFO rule can still be applied to explore its influence on schedule quality. This rule is particularly important in service-oriented contexts and appointment management, where respecting the order of arrival is essential to ensure fairness and customer satisfaction.

The definition introduced by Pinedo for Flow Shop systems offers a compelling theoretical framework that can be tailored to Job Shop contexts. Pinedo defines permutation in Flow Shop scheduling (FIFO permutation) as a system where *all task queues adhere to the FIFO (First In, First Out) discipline*, ensuring that no job can bypass another once it is queued [18]. Building on this, we suggest that applying this FIFO constraint could provide a theoretical basis for redefining permutation within Job Shop scheduling, drawing inspiration from Pinedo's original formulation for Flow Shop environments.

To illustrate this, we consider a Job Shop example involving three jobs. Suppose jobs 1, 2, and 3 have processing times of $(3, 2, 1)$, $(2, 1, 2)$, and $(1, 3, 5)$ respectively on three machines.with the machine order for each job being $(1, 3, 2)$ for job 1, $(3, 2, 1)$ for job 2, and $(2, 1, 3)$ for job 3.

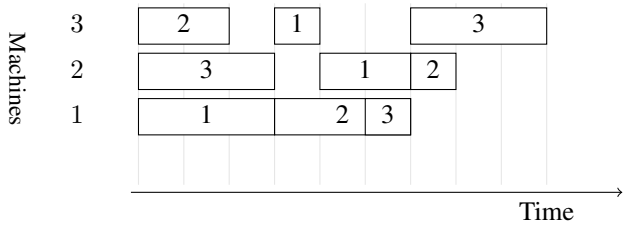

Fig. 4. Scheduling with respect to FIFO in a job shop problem

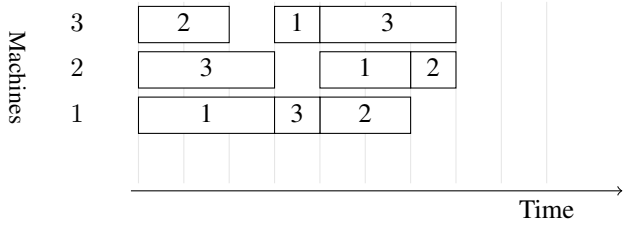

Fig. 5. Scheduling without the FIFO constraint in a job shop problem

The FIFO (First-In, First-Out) rule is one of the most commonly used in service systems. In this context, other performance criteria, such as the minimization of the average waiting time, are also relevant in addition to the makespan. The objective of this section is to study the impact of the FIFO constraint in a Job Shop Scheduling Problem (JSP), focusing on both makespan and waiting time. We first present a mathematical formulation that integrates the FIFO constraint, followed by an extended model aiming to minimize waiting time.

A. Mathematical Model

Initially, we define the PJSP as a Job Shop model augmented with FIFO constraints. To integrate this FIFO constraint, we adapt the classical disjunctive JSP model presented in [14] by introducing additional parameters, variables, and constraints.

Before presenting the detailed model, we first introduce the essential notation:

Parameters

- i: Index for jobs, $i \in J = \{1, \ldots, n\}$, with n representing the total number of jobs.

- j: Index for machines, $j \in M = \{1, \ldots, m\}$, with m representing the total number of machines.
- k: Index denoting the k-th operation within job i.
- p_{ij}: Processing time of job i on machine j.
- m_{ik}: Machine assigned to the k-th operation of job i.
- L: A sufficiently large constant (Big-M).

Decision Variables

- S_{ij}: Start time of job i on machine j.
- $a_{i_1,i_2,j}$: Binary sequencing variable; $a_{i_1,i_2,j} = 1$ if job i_1 precedes job i_2 on machine j, and 0 otherwise.

To explicitly integrate the FIFO constraint into the JSP, we introduce an additional parameter: pr_{ij} denoting the machine assigned to the operation immediately preceding the operation scheduled on machine j for job i. For the first operation of job i, $\text{pr}_{i,m_{i1}} = -1$. With this parameter, the FIFO constraint is explicitly modeled as follows:

$$S_{i_1,\text{pr}_{i_1 j}} + p_{i_1,\text{pr}_{i_1 j}} + a_{i_1,i_2,j} L \quad \geq \quad S_{i_2,\text{pr}_{i_2 j}} + p_{i_2,\text{pr}_{i_2 j}}, \quad \forall i_1 \neq i_2 \in N, \forall j \in M,$$

where $\text{pr}_{i_1 j}, \text{pr}_{i_2 j} \neq -1$. This constraint ensures that for any two jobs i_1 and i_2 processed on machine j, if job i_2 precedes job i_1 ($a_{i_1,i_2,j} = 0$), job i_2 must finish its preceding operation before job i_1 completes its own preceding operation.

Next, we extend both the classical JSP and the PJSP models to explicitly account for waiting times between successive operations of a job. While traditional scheduling models primarily focus on optimizing criteria such as makespan, they often overlook the impact of idle periods experienced by individual jobs. In many service-oriented environments, these waiting times can significantly affect the perceived quality of service. Minimizing waiting time is therefore a crucial objective in such settings, as it directly influences both user satisfaction and the overall operational efficiency of the system [1]. Incorporating waiting times into the objective function allows for a more balanced evaluation of schedules, especially in contexts where fairness, responsiveness, and client-oriented performance are key concerns. We introduce additional constraints and variables aimed at minimizing the average maximum waiting time. The extended models are referred to as the WJSP, which corresponds to the classical JSP with waiting time minimization, where the objective is to minimize the total waiting time while bounding the makespan by the optimal value of the classical JSP. Similarly, the WPJSP extends the PJSP (JSP with FIFO constraints) by minimizing the total waiting time while bounding the makespan by the optimal value obtained for the PJSP.

Additional Decision Variables

- w_{ij}: Waiting time between operation O_{ij} and the subsequent operation O_{ij+1} of job i.
- y_i: Maximum waiting time observed for job i across all its operations.

Additional Constraints for WJSP and WPJSP

- Waiting time calculation for each job i and each operation j (except the last):

$$S_{i,O_{ij}} + p_{i,O_{ij}} + w_{i,O_{ij}} = S_{i,O_{ij+1}}, \qquad \forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, m-1\}$$

- Maximum waiting time constraint per job:

$$y_i \geq w_{ij}, \quad \forall i, \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, m\}$$

The complete mathematical formulation of the WJSP model is presented below. In this formulation, $C_{\max}^{\text{JSP}}$ represents the optimal makespan value computed from the classical Job Shop Scheduling Problem.

$$\text{WJSP} \quad \begin{cases} \min \frac{1}{n} \sum y_i & (1) \\ C_i \geq S_{ij} + p_{ij} \quad \forall i, j & (2) \\ C_{\max} \geq C_i \quad \forall i & (3) \\ S_{ij} \geq 0 \quad \forall i, j & (4) \\ S_{i,m_{ik}} + p_{i,m_{ik}} + w_{i,m_{ik}} = S_{i,m_{i(k+1)}} \quad \forall i, k < m & (5) \\ S_{i_1,j} + p_{i_1,j} \leq S_{i_2,j} + L(1 - B_{i_1 i_2 j}) \quad \forall i_1 \neq i_2, \forall j & (6) \\ B_{i_1 i_2 j} + B_{i_2 i_1 j} = 1 \quad \forall i_1 \neq i_2, \forall j & (7) \\ y_i \geq w_{ij} \quad \forall i, j & (8) \\ C_{\max} \leq C_{\max}^{\text{JSP}} & (9) \end{cases}$$

By adding the FIFO constraint to the WJSP model, we obtain the formulation of the WPJSP model, presented below. In this formulation, $C_{\max}^{\text{PJSP}}$ represents the optimal makespan value computed from the classical Job Shop Scheduling Problem with FIFO constraints.

$$\text{WPJSP} \quad \begin{cases} \min \quad \frac{1}{n} \sum y_i & (1) \\ \text{subject to} \\ (2), \quad (3), \quad (4), \quad (5), \quad (6), \quad (7) \text{ and } \quad (8), \\ S_{i_1,\text{pr}_{i_1 j}} + p_{i_1,\text{pr}_{i_1 j}} + B_{i_1 i_2 j} L \geq S_{i_2,\text{pr}_{i_2 j}} + p_{i_2,\text{pr}_{i_2 j}}, & (10) \\ \forall i_1 \neq i_2 \in N, \forall j \in M, \text{pr}_{i_1 j}, \text{pr}_{i_2 j} \neq -1. \\ C_{\max} \leq C_{\max}^{\text{PJSP}} & (11) \end{cases}$$

It is worth noting that the constraint on the makespan differs between the two models: in WJSP, the upper bound is the optimal makespan obtained from the classical JSP without FIFO, whereas in WPJSP, the bound corresponds to the optimal makespan under FIFO constraints, with $C_{\max}^{\text{PJSP}} \geq C_{\max}^{\text{JSP}}$.

B. Computational study

In this section, we present the experimental study we conducted using the IBM CPLEX 20.1 solver on a cluster of 11 Dell PowerEdge R740 machines, each featuring 512GB of RAM and Intel Xeon Gold 6258R processors clocked at 2.70 GHz.

To evaluate the performance of our approach, we tested the models against a collection of well-established benchmark instances from the literature. The job sizes considered were $\{6, 10, 15\}$, paired with machine configurations of $\{5, 6, 10, 15\}$. We conducted a comparative analysis by benchmarking our results against optimal solutions derived from standard Job Shop scheduling problem instances [2], [3], [10], [12], [29]. Only the instances that reached optimality within 3600 seconds were retained for the remainder of the study.

In Table 1, note that PJSP refers to the Optimal Solution of the Permutation Job Shop, while BKSJSP denotes the Best Known Solution of the Job Shop. The GAP metric is computed using the following formula:

$$\text{GAP } (\%) = \left(\frac{\text{BKSJSP} - \text{PJSP}}{\text{BKSJSP}} \right) \times 100.$$

The results reveal that the difference (labeled GAP in the table) between the classical Job Shop and the Permutation Job Shop is minimal, averaging 1.43%. Furthermore, 80% of the instances exhibit a gap below 2%. Figure 6 shows that

TABLE I

RESULTS ON THE CLASSICAL BENCHMARK INSTANCES

Jobs	Machines	Instances	PJSP	BKSJSP	GAP (%)
Adams, Balas and Zawack [2]					
10	10	abz5	1242	1234	0.65
10	10	abz6	945	943	0.21
Fisher and Thompson [10]					
6	6	ft06	55	55	0.00
10	10	ft10	942	930	1.29
Lawrence [12]					
10	5	la01	666	666	0.00
10	5	la02	671	655	2.44
10	5	la03	618	597	3.52
10	5	la04	591	590	0.17
10	5	la05	593	593	0.00
10	10	la16	953	945	0.85
10	10	la17	787	784	0.38
10	10	la18	854	848	0.71
10	10	la19	865	842	2.73
10	10	la20	907	902	0.55
15	10	la22	946	927	2.05
15	10	la24	962	935	2.89
15	10	la25	998	977	2.15
15	15	la36	1278	1268	0.79
15	15	la39	1250	1233	1.38
15	15	la40	1236	1222	1.15
Applegate and Cook [3]					
10	10	orb01	1069	1059	0.94
10	10	orb02	890	888	0.23
10	10	orb03	1021	1005	1.59
10	10	orb04	1021	1005	1.59
10	10	orb05	898	887	1.24
10	10	orb06	1030	1010	1.98
10	10	orb07	404	397	1.76
10	10	orb08	907	899	0.89
10	10	orb09	937	934	0.32
10	10	orb10	968	944	2.54
Taillard [29]					
15	15	ta01	1256	1231	2.03
15	15	ta02	1265	1244	1.69
15	15	ta03	1248	1218	2.46
15	15	ta04	1197	1175	1.87
15	15	ta05	1247	1224	1.88
15	15	ta07	1245	1227	1.47
15	15	ta08	1253	1217	2.96
15	15	ta09	1299	1274	1.96
15	15	ta10	1273	1241	2.58

in the majority of instances, applying the FIFO constraint (PJSP) leads to a reduction in the average waiting time per job. On average, the improvement is about 19.8% compared to the classical JSP model.

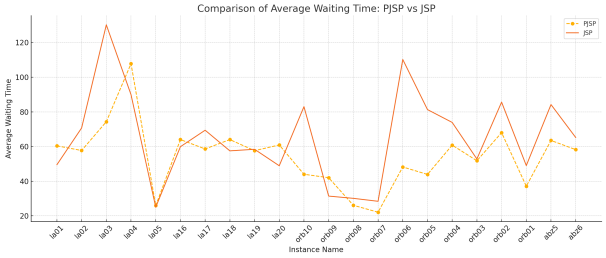

Fig. 6. Comparison of average waiting time for JSP and PJSP

Thus, this section demonstrates that for the tested instances, the impact of FIFO on the makespan is not very significant, and more importantly, enforcing FIFO can significantly reduce job waiting times.

IV. CONCLUSION

In this study, we analyzed the impact of enforcing FIFO constraints in two scheduling environments: the Flow Shop with missing operations and the Job Shop scheduling problem.

In the Flow Shop context, we examined how different definitions of permutation, such as strong, weak, and FIFO-based, can diverge when operations are missing. This theoretical analysis demonstrated that the choice of permutation definition can significantly affect the structure of feasible schedules and the dominance relationships between permutation and non-permutation solutions.

For the Job Shop problem, we adapted the classical disjunctive model to study how the FIFO discipline influences scheduling efficiency and solution quality. Our experiments on benchmark instances revealed that the PJSP (Permutation Job Shop based on FIFO) model results in only a negligible degradation in makespan. Fixing the makespan to $C_{\max}^{\text{JSP}}$ in the WJSPmodel (without FIFO constraints) leads to higher average waiting times than in the WPJSP model, where FIFO constraints are enforced and the makespan is set to $C_{\max}^{\text{PJSP}}$. The WPJSP model yields an average reduction of 19.8% in job waiting times, highlighting the practical benefits of incorporating FIFO constraints, especially in service-oriented applications where waiting time is a key factor in user satisfaction.

Future research could explore how FIFO enforcement affects delays in job shop environments where makespan minimization is the primary objective. In such cases, certain operations may be prioritized and processed earlier in machine queues. Strict FIFO enforcement, however, can introduce additional delays, as lower-priority operations may be forced to wait despite machine availability, solely to preserve queue order. These delays, especially in service-driven contexts, could negatively impact customer satisfaction by keeping resources idle rather than using them efficiently.

The proposed models were tested on instances of small to moderate size, involving up to 15 jobs and 15 machines, which reflect the complexity found in certain service-oriented applications. To generalize the findings and address a broader range of problem sizes, future research could explore alternative exact solution approaches built upon the proposed

models, such as Lagrangian relaxation or branch-and-price techniques.

ACKNOWLEDGMENT

This research was partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC)

REFERENCES

[1] S. Agnihothri, P. Cappanera, M. Nonato, and F. Visintin, "Appointment scheduling in surgery pre-admission testing clinics," *Omega*, vol. 123, p. 102994, 2024.

[2] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, vol. 34, no. 3, pp. 391–401, 1988.

[3] D. Applegate and W. Cook, "A computational study of job-shop scheduling," *ORSA Journal of Computing*, vol. 3, no. 2, pp. 149–156, 1991.

[4] N. Saadani, Z. Bahroun, and A. Bouras, "A linear mathematical model for patients' activities scheduling on hospital resources," in *Proc. 2014 Int. Conf. on Control, Decision and Information Technologies (CoDIT)*, IEEE, pp. 74–80, 2014.

[5] D. Conforti, F. Guerriero, R. Guido, M. M. Cerinic, and M. L. Conforti, "An optimal decision-making model for supporting week hospital management," *Health Care Management Science*, vol. 14, pp. 74–88, 2011.

[6] B. D. Corwin and A. O. Esogbue, "Two-machine flow shop scheduling problems with sequence-dependent setup times: A dynamic programming approach," *Naval Research Logistics Quarterly*, vol. 21, no. 3, pp. 515–524, 1974.

[7] I. Vermeulen, S. Bohte, K. Somefun, and H. La Poutré, "Multi-agent Pareto appointment exchanging in hospital patient scheduling," *Service Oriented Computing and Applications*, vol. 1, pp. 185–196, 2007.

[8] S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Ellis Horwood Ltd., Chichester, UK, 1982.

[9] Y. Fu, J. Ding, H. Wang, and J. Wang, "Two-objective stochastic flowshop scheduling with deteriorating and learning effect in Industry 4.0-based manufacturing system," *Applied Soft Computing*, vol. 68, pp. 847–855, 2018.

[10] H. Fisher and G. L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, pp. 225–251, 1963.

[11] X. Jiang, Z. Tian, W. Liu, Y. Suo, K. Chen, X. Xu, and Z. Li, "Energy-efficient scheduling of flexible job shops with complex processes: A case study for the aerospace industry complex components in China," *Journal of Industrial Information Integration*, vol. 27, p. 100293, 2022.

[12] S. Lawrence, "Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (Supplement)," Carnegie-Mellon University, 1984.

[13] B. Liang, A. Turkcan, M. E. Ceyhan, and K. Stuart, "Improvement of chemotherapy patient flow and scheduling in an outpatient oncology clinic," *International Journal of Production Research*, vol. 53, no. 24, pp. 7177–7190, 2015.

[14] A. S. Manne, "On the job-shop scheduling problem," *Operations Research*, vol. 8, no. 2, pp. 219–223, 1960.

[15] K. Namiki, H. Koga, S. Aida, and N. Honda, "An approach to the production line of automobiles by man-computer system," in *Case Studies in Automation Related to Humanization of Work*, Elsevier, 1979, pp. 97–106.

[16] R. Ouchene, D. Rebaine, and P. Baptiste, "Ambiguity of the definition of permutation flow shops in the presence of missing operations," *Computers & Industrial Engineering*, vol. 182, p. 109387, 2023.

[17] R. Ouchene, D. Rebaine, and P. Baptiste, "The impact of missing operations on the flow shop: permutation vs. non permutation modes," *RAIRO - Operations Research*, vol. 59, no. 3, pp. 1273–1293, 2025.

[18] M. Pinedo, *Scheduling: Theory, Algorithms and Systems*, 3rd ed., Springer, New York, 2002.

[19] C. N. Potts, D. B. Shmoys, and D. P. Williamson, "Permutation vs. non-permutation flow shop schedules," *Operations Research Letters*, vol. 10, no. 5, pp. 281–284, 1991.

[20] J. Marynissen and E. Demeulemeester, "Literature review on multi-appointment scheduling problems in hospitals," *European Journal of Operational Research*, vol. 272, no. 2, pp. 407–419, 2019.

[21] A. J. Ruiz-Torres, J. H. Ablanedo-Rosas, and L. D. Otero, "Scheduling with multiple tasks per job—the case of quality control laboratories in the pharmaceutical industry," *International Journal of Production Research*, vol. 50, no. 3, pp. 691–705, 2012.

[22] J. C. Serrano-Ruiz, J. Mula, and R. Poler, "Development of a multidimensional conceptual model for job shop smart manufacturing scheduling from the Industry 4.0 perspective," *Journal of Manufacturing Systems*, vol. 63, pp. 185–202, 2022.

[23] J. Sridhar and C. Rajendran, "Scheduling in a cellular manufacturing system: a simulated annealing approach," *International Journal of Production Research*, vol. 31, no. 12, pp. 2927–2945, 1993.

[24] R. A. Dudek, S. S. Panwalkar, and M. L. Smith, "The lessons of flowshop scheduling research," *Operations Research*, vol. 40, no. 1, pp. 7–13, 1992.

[25] S. Wang and J. Yu, "An effective heuristic for flexible job-shop scheduling problem with maintenance activities," *Computers & Industrial Engineering*, vol. 59, no. 3, pp. 436–447, 2010.

[26] P. Vogl, R. Braune, and K. F. Doerner, "Scheduling recurring radiotherapy appointments in an ion beam facility: considering optional activities and time window constraints," *Journal of Scheduling*, vol. 22, no. 2, pp. 137–154, 2019.

[27] M. Ritt and D. A. Rossit, "Effective heuristics for permutation and non-permutation flow shop scheduling with missing operations," *Computers & Operations Research*, vol. 170, p. 106742, 2024.

[28] M. Henneberg and J. S. Neufeld, "A constructive algorithm and a simulated annealing approach for solving flowshop problems with missing operations," *International Journal of Production Research*, vol. 54, no. 12, pp. 3534–3550, 2016.

[29] E. D. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993.