Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 246 (2024) 10–19

Procedia Computer Science

www.elsevier.com/locate/procedia

28th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2024)

# Resolving a Hybrid Flow Shop Problem with Dedicated Machines: Incorporating Blocking and Release Date Constraints

ZouhourNabli[a,*], Soulef Khalfallah[b], Ouajdi Korbaa[a]

[a]*Higher Institute of Computer Science and Communication Technologies (ISITCom), MARS (Modeling of Automated Reasoning Systems) Research Lab LR17ES05, University of Sousse, Tunisia*
[b]*Higher Institute of Management, MARS (Modeling of Automated Reasoning Systems) Research Lab LR17ES05, University of Sousse, Tunisia*

**Abstract**

In this paper, we address the problem of two stages Hybrid Flow Shop with release dates and blocking constraints. The objective is to minimize the makespan. We consider m parallel machines at the first stage and two dedicated machines at the second stage. However, the Hybrid Flow Shop Scheduling Problem with dedicated machines, blocking and release date constraint simultaneously has not yet been well studied. Incorporating these, constraints align more closely with the real configurations found in many industries, such as the pasta industry. To address this problem, we will introduce a mathematical model and contrast the outcomes against two lower bounds discussed in the literature. In addition, we will deal with this problem through three heuristics.

*Keywords:* Hybrid Flow Shop; Dedicated Machines; Blocking constraint; Release date; Mathematical Model; Heuristics

## 1. Introduction

In tackling scheduling challenges within industrial settings, the prevailing emphasis in current research is on Hybrid Flow Shop (HFS) configurations incorporating parallel and/or dedicated machines. However, practical scenarios often entail additional constraints that are overlooked. In our approach, we aimed to extract insights from real-world scenarios within the pasta industry to formalize and elaborate on these constraints. We examine a production system

[*] Corresponding author. Tel.: +216-22-385-504.
*E-mail address:* nablizouhur@yahoo.fr

comprising a process that includes drying cabins and packaging machines. Each product undergoes four sequential steps in the production process. In Step 1, the production setup comprises c drying cabins, each accommodating o trolleys. Each trolley contains j trays, all holding the same product. Figure 1 illustrates the arrangement of a drying cabin. The trolleys assigned to cabin i have individual release dates, denoted as $r_i$, which may vary among cabins. Step 2 involves a drying operation on the cabins, with each operation lasting a fixed duration without interruption. Following drying, in Step 3, the trolleys from the cabins are manually placed in a limited space, awaiting further processing. In Step 4 of the process, the allocation of cabin contents occurs based on the inherent attributes of the product, leading to their transfer to either an F1 line or an F2 line. The F1 line, characterized as semi-automatic, involves a collaborative effort by a team of workers who concurrently oversee batches of trays containing uniform products. Conversely, the F2 line, categorized as automatic, assigns a single worker the responsibility of sequentially emptying trays, each housing identical products, into assigned tanks.

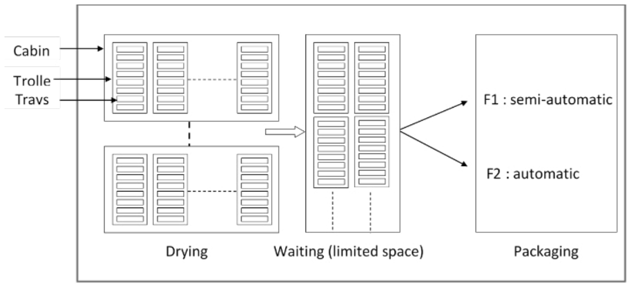

Fig. 1: Representation of the pasta production system

In theory, we examine a production system comprising a set of jobs. In this specific problem scenario, each job represents the content of a cabin. Every job must undergo processing on one machine chosen from a set of identical parallel machines. Subsequently, it proceeds to be processed on one of the two pre-assigned machines. Storage capacity between machines is constrained. The objective of the production system is to minimize the total production time, also known as the makespan. The scientific classification of this problem can be described as a hybrid flow-shop scheduling problem with blocking and release dates, following the α/β/λ classification introduced by [1].

$$HF2(P, DP2)|rj, block|Cmax$$

The problem can be formulated as follows: A set of *n* jobs, denoted as *j* = {*1, ..., n*}, must has to be executed in two stages sequentially, denoted as s= {1,2}. The first stage comprises a single set of identical parallel machines, while the second stage consists of two non-identical machines. A job *j* can commence its execution on machine *m* in stage 1 (S1) only if the job is available. Each job can be processed on only one machine per stage without interruption. Additionally, each machine can handle only one operation, denoted as $O_{ij}$, at a time. Furthermore, a machine in S1 remains blocked by a job until the job's operation on that machine is completed, and the job transitions to processing on the dedicated machine in stage 2. For S2, each job is pre-assigned to a specific machine m.

In the extensive literature on the hybrid flow shop problem, several notable contributions have been made. For instance, [2] presented and compared two Mixed Integer Programming (MIP) formulations and associated lower bounds (LB) for the HFS problem featuring dedicated machines. Similarly, [3] introduced four heuristics and two LB for this problem, conducting a comparative analysis with their previous work. Another notable study by [4] proposed a genetic algorithm designed to address the HFS scheduling problem with unrelated machines and machine eligibility constraints, aiming to minimize total tardiness. Recently, [5] delved into the HFS problem with dedicated machines at both stages, offering a constraint programming approach, a heuristic based on priority rules, and Tabu search procedures. In their study, [6] delve into the blocking HFS problem with sequence-dependent setup times, aiming to minimize total tardiness and earliness. Their investigation focuses on scenarios with uniform parallel machines under this constraint. They propose six algorithms grounded in migratory bird optimization and water wave optimization methodologies. Additionally, [7] present a mathematical model tackling the blocking hybrid flow shop problem with an energy-efficient criterion. They introduce a modified Iterative Greedy algorithm to address this variant. In this manuscript, we adopt the mathematical model and three heuristics outlined by [8]. While their approach closely resembles ours in configuration, it does not account for the blocking constraint. Our study focuses on the hybrid flow shop scheduling problem with parallel machines at the first stage and two dedicated machines at the second stage. Each job is assigned a release date at the first stage, with the objective of minimizing the makespan. We will employ the same LBs as proposed in their study for our comparative analysis.

Our contribution comprises several key elements:

1- We tackle the hybrid flow shop problem, considering specific constraints inspired by the configuration of a pasta production workshop.
2- We propose a mathematical model that utilizes a linear ordering variable approach. While [8] addressed a similar problem; their model did not incorporate the blocking constraint. Therefore, our contribution involves modifying the model to accommodate this crucial constraint, considering its significance due to space limitations in many production workshops.
3- Furthermore, we adapt three heuristics originally presented by [8] for solving the hybrid flow shop problem with release dates, modifying them to incorporate the blocking constraint.

## 2. Mathematical model

In this section, we will add the blocking constraint for the MIP model presented in [8], to solve the problem of *HF2(P,PD2)|rj,block |Cmax*. In this context, we added the following decision variable $b_l$, this variable represent the difference between the start date of job l on S2 and its completion time on S1. In addition, we added the two constraints (9) and (10), the complete model presented as follows:

**Sets**

| | |
|---|---|
| $N$ | set of jobs, indexed $j = 1,\dots, n$ |
| $M$ | set of machines, indexed $m=1,\dots, Ma$ |
| $Ms$ | set of machines at stage $s$ indexed $m=1,\dots, m_s$ |
| $S$ | set of stage indexed $s=1,\dots, \lvert S\rvert$ |
| $l$ | position of job on machine, indexed $1,\dots, n$ |

**Parameters**

| | |
|---|---|
| $p_{sj}$ | processing time of job $j$ at stage $s$ |
| $r_j$ | release date of job $j$ at stage 1 |
| $BM$ | a large number (i.e, “big-M”), $BM \geq \sum_j p_j$ |
| $Y_{jsm}$ | Binary variable that is equal to 1 if job j is "pre-affected" to machine m in stage s and 0 otherwise |

**Decision variables**

| | |
|---|---|
| $C_{sj}$ | completion time of job $j$ at stage $s$ |
| $d_{slj}$ | Binary variable that is equal to 1 if job $j$ precedes job $l$ at stage $s$ and 0 otherwise |
| $z_{sjm}$ | Binary variable that is equal to 1 if job j is positioned on machine m at stage s and 0 otherwise |
| $u_{slj}$ | Binary variable that is equal to 1 if job j and l are not scheduled on same machine m at stage s and 0 otherwise |
| $u_{slj}$ | Binary variable that is equal to 1 if job j and l are not scheduled on same machine m at stage s and 0 otherwise |
| $b_l$ | Difference between the start date of job l on S2 and its completion time on S1 |

**The model**

Objective functions: min $Cmax$

$$d_{slj} + d_{sjl} + u_{slj} = 1 \qquad s \in S \quad l,j \in N \qquad l < j \tag{1}$$

$$d_{slj} + d_{sjk} + d_{skl} \leq 2 \qquad s \in S \quad l,j,k \in N \quad l<j<k \tag{2}$$
$$z_{slm} + z_{sjm} + u_{slj} \leq 2 \qquad s \in S \quad l,j \in N \quad l<j \quad m \in ms \tag{3}$$
$$\sum_{m \in ms} z_{slm} = 1 \qquad s \in S \quad l \in N \tag{4}$$
$$C_{sj} \geq p_{sj}\, z_{sjm} \qquad j \in N \quad m \in ms \tag{5}$$
$$c_{sj} \geq c_{sl} + p_{sj}(d_{slj} + z_{slm} + z_{sjm} - 2) - BM(1 - d_{slj}) \qquad s \in S \quad l,j \in N \quad m \in ms \tag{6}$$
$$c_{sj} \geq c_{s-1j} + p_{sj} \qquad s \in S \; and \; s = 2 \quad j \in N \tag{7}$$
$$Cmax \geq C_{2j} \qquad j \in N \tag{8}$$
$$b_l = C_{2l} - p_{2l} - C_{1l} \qquad \forall\, l \in N \tag{9}$$
$$C_{1j} \geq C_{1l} + b_l + p_{1j}(d_{1lj} + z_{1lm} + z_{1jm} - 2) - BM\left(1 - d_{1lj}\right) \qquad l,j \in N \quad l \neq j \quad m \in ms \tag{10}$$
$$y_{jsm} = 0 : z_{sjm} = 0 \qquad s \in S \; and \; s = 2 \quad j \in N \quad m \in ms \tag{11}$$

(1) For each stage *s*, the job is positioned before job *j* or vice versa (job *j* is positioned before job *l*), provided the two jobs are scheduled on the same machine.
(2) Transitivity constraints that provide a linear order between three jobs.
(3) For each stage *s*, calculates the $u_{sjm}$ variable (assignment of jobs to machines)
(4) For each *e* stage, each job is positioned on a machine.
(5) Completion time $C_{sj}$ of a job *j* must be greater than or equal to the release date of the job plus its processing time. The completion time $C_{sj}$ is given by the constraints (6) and (7).
(8) The completion time of all jobs is greater than or equal to the completion time of the last job at stage 2.
(9) For the stage 2 a job *j* cannot be processed on a machine that his not pre-affected.

## 3. Lower bounds

In this section, we present the lower bounds (LBs) that will serve as benchmarks for comparison with both the Mixed Integer Programming (MIP) solution and the heuristics. These two LBs are identical to those outlined in [8] for addressing the *HF2(P,PD2)|rj|Cmax* problem.

LB₁ is specifically designed for solving the identical parallel machines scheduling problem. This lower bound performs particularly well when the processing time at S1 is strictly greater than the processing time of jobs at S2.

$$\boldsymbol{LB_1}=\left\{min_{j\in N}(r_j) + \sum_{j=1}^{n} p_{1j}/m_s + min_{j\in N}\{p_{2j}\}\right\}$$

When the processing times in S2 are shorter than those in S1, the effectiveness of $LB_1$ diminishes. To address this issue, they introduced an alternative lower bound that relaxes the waiting time of jobs between stages. $LB_2$ is defined as follows:

$$\boldsymbol{LB_2}=min_{j\in N}(p_{1j} + r_j)+max\{P_1, P_2\}$$

Clearly, during S1, no job *j* can complete its processing before the smallest value of the sum of processing times at S1 and the release date. In S2, where two dedicated machines are available, we need to wait until the machine with the maximum value of the sum of processing times completes its operation.

## 4. Heuristics

In this section, we detail the main adaptations applied to the LS (Local Search) heuristic and the priority rules, specifically LPT (Longest Processing Time) on S2 and SPT (Smallest Processing Time) on S1, initially proposed by [8]. These methodologies were originally designed for resolving the *HF2(P,2)|rj|Cmax* problem. However, we have tailored them to suit the *HF2(P,2)|rj, block|Cmax* problem. It is essential to emphasize that the fundamental principles of these heuristics remain unchanged despite the introduction of the blocking constraint. The difference lies in the calculation of job completion times. We will subsequently explain the principle used for computing this metric. The calculation of completion times for this problem is divided into two parts:

- The first part involves calculating the completion times of job processing at S1, while considering the job availability dates, denoted as $C_{1\Pi\,[j]}$. Furthermore, a machine at S1 becomes available for processing the next operation only after its current operation finishes, and the job transitions to the dedicated machine at S2.
- The second part involves calculating the completion times of treatment at S2, denoted as $C_{2\Pi\,[j]}$.

The completion time of job j on machine at S1 is determined as follows:

$$C_{1\Pi\,[j]} = \begin{cases} r_{\Pi\,[j]} + P_{1\Pi\,[j]} & \mathit{If}\ \Pi\,[j] = 1 \\ \max\{r_{\Pi\,[j]}, D_{2\Pi[j-1]}\} + P_{1\Pi[j]} & \mathit{otherwise} \end{cases}$$

If $\Pi$ [$j$] and $\Pi$ [$j-1$] pass on the same machine at S1.

The completion time of job j on machine at S2 is as follows:

$$C_{2\Pi\,[j]} = \begin{cases} C_{1\Pi[j]} + P_{2\Pi[j]} & \mathit{If}\ \Pi\,[j] = 1 \\ \max(C_{1\Pi[j]}, C_{2\Pi[j-1]}) + P_{2j} & \mathit{otherwise} \end{cases}$$

If *j* and *j*-1 pass on the same machine on S2.

## 5. Computational analysis

### *5.1. Instances generation*

In this work, regarding processing times, we used the same data used on [8]. We will conduct tests on problems under various conditions of *m*, *n*, *n1*, *n2* referred to as "*problem families*", where n is the total number of jobs, *n1* is the number of jobs processed on machine 1 at stage 2, *n2* is the number of jobs processed on machine 2 at stage 2, and *m* is the number of identical parallel machines at stage 1. There are 45 families of problems in total.

For each family, processing times are generated randomly from a uniform distribution, denoted as $p_{sj} \in [PLB, PUB]$ where *PLB* is the lower limit of the processing time and *PUB* is the upper limit. To test the effects of the number of jobs, we consider three different values of *n*: 10, 50, and 100, as in [8].

To assess the impact of job assignment (number of jobs dedicated to the first and second machines at S2), we examine three different combinations of *n* and *n1*. In the first combination, we assume that the number of jobs is equal for both machines ($n1 = 0.5*n$); in the other two combinations, we assume that the number of jobs pre-assigned to one machine is greater than the number pre-assigned to the other ($n1 = 0.6*n$ and $n1 = 0.7*n$). We consider three different distributions for processing times:

We generate six groups of data instances. For the first three groups, the intervals of processing times on S1 and S2 are generated in the same manner.

$P_{sj}$: ∈[1; 99]; with standard deviation 29.42.
$P_{sj}$: ∈[25; 75]; with standard deviation 14.93.
$P_{sj}$: ∈[40; 60]; with standard deviation 6.17.

For groups 4 to 6, we introduce three additional distributions where the processing times at S1 and S2 differ:

$p_{1j}$: ∈ [10 ;49]; $p_{2j}$: ∈ [50;99]; with standard deviation 11.65 at *S1* and 13.85 at *S2*.
$p_{1j}$: ∈ [50 ;99]; $p_{2j}$: ∈ [10;49]; with standard deviation 15 at *S1* and 12.22 at *S2*.

$p_{1j}$: ∈ [5 ;100]; $p_{2j}$: ∈ [1;20]; with standard deviation 22,37at *S1* and 4,81 at *S2*.

To evaluate the impact of the number of parallel machines on stage 1, we considered three values: 2, 5, and 10. In these instances, we maintained a fixed number of jobs, either 50. For each problem family, we generated 30 instances for thorough analysis and comparison. To evaluate the performance of the heuristics we measure the relative deviation *RD=(HE-MB)/MB*, where *HE* is the result generated by the heuristics (*LS*, *SPT_ S1* and *LPT_S2*). *MB* is equal to the solution generated by the *MIP* if it is optimal else *MB* is equal to the value of the best lower bound *LB_B*. The stopping criteria are either reaching an optimal solution or reaching a running time of 2 hours. The Average Relative Deviation (ARD) is calculated as follows:

$$ARD = \frac{\sum_{i \in instance}(RD_i)}{number\ of\ instances}$$

## *5.2. Comparison of results*

In this section, we initially compare the two LBs (LB1 and LB2) to the outcome of the MIP, which yields either an upper bound (UB) or an optimal solution. Subsequently, we compare the results obtained by the LS and the priority rules (SPT_S1, LPT_S2) with the best available bound (optimal solution or best lower bound).

- **Comparison of the best lower bound with MIP**

In Table 1, it is noted that for problem instances involving 10 and 50 jobs, all instances were successfully solved using the mathematical model when *n* = 10, or an upper bound was generated when *n* = 50. Where *n*=10, we compare LB_B to the result of the mathematical model, which provides an optimal solution. For instances ranging from pb9 to pb12, LB_B corresponds to the optimal solution for all cases. However, for instances pb13 to pb18, where $p_{1j}$ ranges from [50.99] and $p_{2j}$ from [10.49], as well as when $p_{1j}$ ranges from [5,100] and $p_{2j}$ from [1,20], LB_B closely aligns with the upper bound (UB) but does not correspond to the optimal solution in any case. For instances pb1 to pb8, the results of LB_B are either equal to or very close to LB_B for most instances, with an ARD not exceeding 2.97%. where *n* = 50, the ARD are higher. This is because the MIP stops after 2 hours of execution without generating an optimal solution, indicating that it has produced an upper bound (UB). Consequently, the results will depend on both the quality of the UB and the quality of the LB. However, it is worth noting that an optimal solution was reached for 20% of instances in some problems.

Table 1. Comparison of the best lower bound with the upper bound for 10, and 50 jobs with 2 parallel machines

| | | | **10 jobs** | | | | | **50 jobs** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Inst** | **DU** | **n1** | **ARD** | **MaxRD** | **MinRD** | **AVT(s)** | **% optimal** | **ARD** | **MaxRD** | **MinRD** | **AVT(s)** | **% optimal** |
| pb1 | | 0.5*n | 2,97% | 15,03% | 0% | 5 | 37% | 11,57% | 16,12% | 8,36% | 7205 | 0% |
| pb2 | [1,99] | 0.6*n | 2,28% | 8,45% | 0% | 8 | 40% | 9,48% | 17,04% | 4% | 7203 | 0% |
| pb3 | | 0.7*n | 1,39% | 8,18% | 0% | 7 | 63% | 0,90% | 3,19% | 0% | 7203 | 20% |
| pb4 | | 0.5*n | 1,33% | 5,90% | 0% | 24 | 40% | 11,46% | 19,19% | 4,23% | 7208 | 0% |
| pb5 | [25,75] | 0.6*n | 0,39% | 4,81% | 0% | 16 | 80% | 6,12% | 12,10% | 1% | 7203 | 0% |
| pb6 | | 0.7*n | 0,01% | 0,32% | 0% | 10 | 97% | 2,29% | 4,97% | 0,85% | 7202 | 0% |
| pb7 | | 0.5*n | 0,66% | 2,45% | 0% | 10 | 47% | 14,53% | 30,18% | 3,80% | 7209 | 0% |
| pb8 | [40,60] | 0.6*n | 0,16% | 2,50% | 0% | 17 | 90% | 0,96% | 2,19% | 1% | 7204 | 0% |
| pb9 | | 0.7*n | **0%** | 0% | 0% | 10 | **100%** | 7,95% | 22,85% | 0,44% | 7210 | 0% |
| pb10 | | 0.5*n | **0%** | 0% | 0% | 3 | **100%** | 5,90% | 11,74% | 0% | 7208 | 20% |
| pb11 | [10,49] [50,99] | 0.6*n | **0%** | 0% | 0% | 4 | **100%** | 0,14% | 0,23% | 0% | 7207 | 20% |
| pb12 | | 0.7*n | **0%** | 0% | 0% | 4 | **100%** | 0,35% | 0,67% | 0,04% | 7211 | 0% |
| pb13 | | 0.5*n | 1,30% | 3,21% | 0,40% | 7052 | 0% | 2,32% | 4,41% | 0,60% | 7204 | 0% |
| pb14 | [50,99] [10,49] | 0.6*n | 1,41% | 3,47% | 0,40% | 6856 | 0% | 3,64% | 5,77% | 1% | 7203 | 0% |
| pb15 | | 0.7*n | 1,70% | 4,59% | 0,14% | 3749 | 0% | 3,70% | 8,61% | 0,39% | 7210 | 0% |
| pb16 | | 0.5*n | 0,91% | 5,36% | 0,17% | 104 | 0% | 3,91% | 10,18% | 0,63% | 7203 | 0% |
| pb17 | [5,100] [1,20] | 0.6*n | 1,02% | 5,36% | 0,17% | 122 | 0% | 1,82% | 3,24% | 0% | 7204 | 0% |
| pb18 | | 0.7*n | 1,08% | 5,95% | 0,17% | 111 | 0% | 1,82% | 3,24% | 0,30% | 7202 | 0% |

In Table 2, for instances ranging from pb1 to pb3, when the number of parallel machines is equal to 5, the results obtained by MIP closely align with LB_B. In this scenario, the results of MIP are equal to LB_B for 15% of instances, compared to 33% of instances when m = 10. For instances pb13 to pb15, when m = 5 or 10, the relative deviation values of MIP are higher for most problems. However, optimal solutions were achieved for 40% of instances in pb14. For instances pb16 to pb18, when m = 5 or 10, the relative deviation values of MIP are very high for all problems and do not correspond to the optimal solution in any case.

Table 2. Comparison of the best lower bound with the upper bound for 50 jobs, with m parallel machine.

| 50 jobs | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Inst** | **m** | **DU** | **n1** | **ARD** | **Max RD** | **Min RD** | **AV time MIP** | **% optimal** |
| pb1 | | | 0.5*n | 0,59% | 1,24% | 0% | 7218 | 25% |
| pb2 | 5 | | 0.6*n | 4,32% | 19,15% | 0% | 7225 | 20% |
| pb3 | | [1,99] | 0.7*n | 0,72% | 2,04% | 0,11% | 7232 | 0% |
| pb1 | | | 0.5*n | 0,17% | 0,37% | 0% | 7206 | 40% |
| pb2 | 10 | | 0.6*n | 0,20% | 0,59% | 0% | 7238 | 40% |
| pb3 | | | 0.7*n | 0,31% | 0,68% | 0% | 7207 | 20% |
| pb13 | | | 0.5*n | 20,85% | 37,01% | 5,91% | 7202 | 0% |
| pb14 | 5 | | 0.6*n | 13,29% | 21,88% | 4% | 7205 | 0% |
| pb15 | | [50,99] | 0.7*n | 2,37% | 4,76% | 0,77% | 7203 | 0% |
| pb13 | | [10,49] | 0.5*n | 31,81% | 150,57% | 0,12% | 7239 | 0% |
| pb14 | 10 | | 0.6*n | 0,57% | 1,87% | 0% | 7222 | 40% |
| pb15 | | | 0.7*n | 0,46% | 0,93% | 0,17% | 7226 | 0% |
| pb16 | | | 0.5*n | 23,93% | 41,44% | 12,16% | 7228 | 0% |
| pb17 | 5 | | 0.6*n | 26,60% | 55,00% | 10% | 7211 | 0% |
| pb18 | | [5,100] | 0.7*n | 27,37% | 43,96% | 10,63% | 6563 | 0% |
| pb16 | | [1,20] | 0.5*n | 68,43% | 75,00% | 56,09% | 7202 | 0% |
| pb17 | 10 | | 0.6*n | 48,31% | 55,93% | 27% | 7202 | 0% |
| pb18 | | | 0.7*n | 25,51% | 32,71% | 12,16% | 7202 | 0% |

- **Comparisons of heuristics with the best bound**

To assess the performance of the heuristics, we suggest comparing them with LB_B and the MIP results.

**Problem of 10 jobs and 2 parallel machines:** In table 3; where n = 10, MIP consistently provides the optimal solution. Therefore, we will compare the heuristics with the optimal solution. The obtained results demonstrate that the LS yields optimal solutions for all instances from pb5 to pb12. For other problems, the Max RD does not exceed 5.71%, and the ARD remains below 0.36%. Additionally, the AVT remains constant across all instances at 0.3 seconds. On the other hand, the SPT_S1 heuristic achieves optimal solutions for 31% of instances, with a Max RD not exceeding 23% and an ARD below 7%. In contrast, the performance of the LPT_S2 heuristic is comparatively low compared to other heuristics, with an execution time of less than 0.01 seconds for both priority rules.

Table 3. Comparisons of heuristics with the best bound for 10 jobs, with 2 parallel machines

| | | | **RL** | | | | | **SPT_S1** | | | | | **LPT_S2** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| inst | DU | n1 | ARD | MaxRD | MinRD | AVT(s) | % optimal | ARD | Max RD | Min RD | AVT(s) | % optimal | ARD | Max RD | Min RD | AVT(s) | % optimal |
| pb1 | | 0.5*n | **0,36%** | 5,71% | 0% | 0,3 | 73% | 7,74% | 23,33% | 0% | <0,01 | 23% | 11,64% | 31,43% | 2,57% | <0,01 | 0% |
| pb2 | [1,99] | 0.6*n | **0,21%** | 1,99% | 0% | 0,3 | 77% | 6,59% | 18,39% | 0% | <0,01 | 23% | 9,30% | 23,33% | 1,59% | <0,01 | 0% |
| pb3 | | 0.7*n | **0,12%** | 1,20% | 0% | 0,3 | 87% | 4,93% | 20,19% | 0% | <0,01 | 33% | 7,74% | 16,73% | 0% | <0,01 | 7% |
| pb4 | | 0.5*n | **0,01%** | 0,31% | 0% | 0,3 | 97% | 3,52% | 12,26% | 0% | <0,01 | 13% | 4,58% | 13,29% | 0% | <0,01 | 7% |
| pb5 | [25,75] | 0.6*n | **0%** | 0% | 0% | 0,3 | **100%** | 2,57% | 8,71% | 0% | <0,01 | 33% | 4,60% | 15,56% | 0% | <0,01 | 7% |
| pb6 | | 0.7*n | **0%** | 0% | 0% | 0,3 | **100%** | 1,33% | 8,68% | 0% | <0,01 | 53% | 3,28% | 17,04% | 0% | <0,01 | 17% |
| pb7 | [40,60] | 0.5*n | **0%** | 0% | 0% | 0,3 | **100%** | 1,71% | 4,17% | 0% | <0,01 | 27% | 2,00% | 3,95% | 0% | <0,01 | 10% |
| pb8 | | 0.6*n | **0%** | 0% | 0% | 0,3 | **100%** | 1,03% | 5,00% | 0% | <0,01 | 23% | 1,54% | 5,36% | 0% | <0,01 | 27% |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| pb9 | | 0.7*n | **0%** | 0% | 0% | 0,3 | **100%** | 0,52% | 2,68% | 0% | <0,01 | 43% | 1,06% | 5,04% | 0% | <0,01 | 40% |
| pb10 | | 0.5*n | **0%** | 0% | 0% | 0,3 | **100%** | 0,04% | 0,64% | 0% | <0,01 | 93% | 2,00% | 5,32% | 0% | <0,01 | 30% |
| pb11 | [10,49] [50,99] | 0.6*n | **0%** | 0% | 0% | 0,3 | **100%** | 0,05% | 0,67% | 0% | <0,01 | 90% | 1,73% | 7,06% | 0% | <0,01 | 30% |
| pb12 | | 0.7*n | **0%** | 0% | 0% | 0,2 | **100%** | 0,05% | 0,56% | 0% | <0,01 | 90% | 1,66% | 6,06% | 0% | <0,01 | 27% |
| pb13 | | 0.5*n | **0,03%** | 0,87% | 0% | 0,3 | 97% | 3,12% | 7,16% | 0% | <0,01 | 7% | 0,77% | 2,63% | 0% | <0,01 | 17% |
| pb14 | [50,99] [10,49] | 0.6*n | **0,02%** | 0,58% | 0% | 0,3 | 97% | 4,76% | 10,24% | 1,16% | <0,01 | 0% | 2,12% | 6,10% | 0% | <0,01 | 3% |
| pb15 | | 0.7*n | **0,03%** | 0,58% | 0% | 0,3 | 93% | 5,42% | 9,83% | 1,16% | <0,01 | 0% | 2,77% | 9,56% | 0,47% | <0,01 | 0% |
| pb16 | | 0.5*n | **0,15%** | 1,03% | 0% | 0,3 | 73% | 3,50% | 8,25% | 0,98% | <0,01 | 0% | 2,76% | 10,04% | 0% | <0,01 | 3% |
| pb17 | [5,100] [1,20] | 0.6*n | **0,10%** | 0,89% | 0% | 0,3 | 80% | 4,70% | 9,64% | 1,47% | <0,01 | 0% | 2,97% | 7,14% | 0% | <0,01 | 7% |
| pb18 | | 0.7*n | **0,26%** | 1,54% | 0% | 0,3 | 60% | 5,21% | 8,55% | 1,64% | <0,01 | 0% | 4,26% | 12,99% | 0,66% | <0,01 | 0% |

**Problem of 50 jobs and 2 parallel machines:** In table 4; where n = 50, MIP no longer provides the optimal solution for most instances. Therefore, we compared the heuristics with LB_B. The results indicate that LS achieves optimal results for 33% of problems, with a maximum ARD of 10.46%. Notably, for instances pb9, pb11, and pb12, LS provided the optimal solution for 100% of cases. Additionally, the average execution time (AVT) of the LS does not exceed 11 seconds.

Table 4. Comparisons of heuristics with the best bound for 50 jobs, with 2 parallel machines

| | | | **RL** | | | | | **SPT_S1** | | | | | **LPT_S2** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| inst | DU | n1 | ARD | MaxRD | MinRD | AVT(s) | % optimal | ARD | Max RD | Min RD | AVT(s) | % optimal | ARD | Max RD | Min RD | AVT(s) | % optimal |
| pb1 | | 0.5*n | 10,46% | 16,18% | 0,29% | 11,8 | 0% | 20,81% | 34,20% | 7,54% | <0,01 | 0% | 21,93% | 33,48% | 4,67% | <0,01 | 0% |
| pb2 | [1,99] | 0.6*n | 4,44% | 16,45% | 0% | 11,4 | 7% | 13,39% | 30,79% | 1,37% | <0,01 | 0% | 12,85% | 31,33% | 0,25% | <0,01 | 0% |
| pb3 | | 0.7*n | 0,55% | 3,41% | 0% | 10,8 | 37% | 6,04% | 20,04% | 0% | <0,01 | 3% | 7,38% | 14,83% | 1,64% | <0,01 | 0% |
| pb4 | | 0.5*n | 5,66% | 9,13% | 0,46% | 10,7 | 0% | 10,15% | 16,31% | 3,04% | <0,01 | 0% | 10,04% | 15,46% | 3,76% | <0,01 | 0% |
| pb5 | [25,75] | 0.6*n | 0,59% | 3,77% | 0% | 10,1 | 37% | 5,02% | 13,41% | 0,77% | <0,01 | 0% | 3,13% | 11,59% | 0,06% | <0,01 | 0% |
| pb6 | | 0.7*n | 0,01% | 0% | 0% | 8,9 | 97% | 1,60% | 6,78% | 0% | <0,01 | 20% | 1,12% | 3,13% | 0% | <0,01 | 3% |
| pb7 | | 0.5*n | 3,45% | 4,88% | 0,51% | 11,0 | 0% | 5,32% | 7,23% | 2,43% | <0,01 | 0% | 5,33% | 7,26% | 2,43% | <0,01 | 0% |
| pb8 | [40,60] | 0.6*n | 0,16% | 0,74% | 0% | 10,4 | 33% | 2,09% | 5,38% | 0,19% | <0,01 | 0% | 1,39% | 3,63% | 0,46% | <0,01 | 0% |
| pb9 | | 0.7*n | 0% | 0% | 0% | 9,0 | 100% | 0,75% | 2,13% | 0,06% | <0,01 | 0% | 0,66% | 1,69% | 0% | <0,01 | 3% |
| pb10 | | 0.5*n | 0,02% | 0,16% | 0% | 8,5 | 83% | 0,21% | 1,56% | 0% | <0,01 | 47% | 0,91% | 2,11% | 0% | <0,01 | 3% |
| pb11 | [10,49] [50,99] | 0.6*n | 0% | 0% | 0% | 7,3 | 100% | 0,13% | 1,45% | 0% | <0,01 | 57% | 0,64% | 1,70% | 0% | <0,01 | 3% |
| pb12 | | 0.7*n | 0% | 0% | 0% | 6,0 | 100% | 0,13% | 2,55% | 0% | <0,01 | 60% | 0,52% | 1,45% | 0% | <0,01 | 10% |
| pb13 | | 0.5*n | 0,09% | 0,21% | 0% | 9,4 | 3% | 1,14% | 2,29% | 0,10% | <0,01 | 0% | 0,39% | 1,55% | 0,05% | <0,01 | 0% |
| pb14 | [50,99] [10,49] | 0.6*n | 0,12% | 0,26% | 0% | 9,5 | 3% | 1,74% | 3,06% | 0,74% | <0,01 | 0% | 0,81% | 1,52% | 0,38% | <0,01 | 0% |
| pb15 | | 0.7*n | 0,20% | 0,45% | 0,08% | 9,9 | 0% | 1,65% | 2,94% | 0,58% | <0,01 | 0% | 1,02% | 4,81% | 0,39% | <0,01 | 0% |
| pb16 | | 0.5*n | 0,15% | 0,48% | 0,04% | 10,7 | 0% | 1,16% | 2,57% | 0,39% | <0,01 | 0% | 0,89% | 2,58% | 0,18% | <0,01 | 0% |
| pb17 | [5,100] [1,20] | 0.6*n | 0,17% | 0,40% | 0,04% | 10,8 | 0% | 1,43% | 2,42% | 0,60% | <0,01 | 0% | 1,21% | 2,33% | 0,35% | <0,01 | 0% |
| pb18 | | 0.7*n | 0,26% | 0,64% | 0,08% | 10,9 | 0% | 1,36% | 2,65% | 0,67% | <0,01 | 0% | 2,21% | 5,39% | 0,58% | <0,01 | 0% |

**Problem of 100 jobs and 2 parallel machines:** In table 5; Where n = 100, The results obtained indicate that the LS achieves optimal results for 100% of instances in problems pb11 and pb12, and for 80% of instances overall. The ARD exceeds 5% for problems pb1, pb2, pb4, and pb7. However, for the remaining instances, the results are very close to LB_B, with a maximum ARD typically less than 1%. The maximum Average Execution Time (AVT) recorded is 48 seconds. The other three heuristics generate noteworthy ARDs, but achieve very few optimal solutions. The AVT of these three priority rules is less than 0.01 seconds. Additionally, for all problem families, whenever there is a load imbalance, the heuristics tend to be less efficient.

Table 5. Comparisons of heuristics with the best bound for 100 jobs, with 2 parallel machines

| | | | **RL** | | | | | **SPT_S1** | | | | | **LPT_S2** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| inst | DU | n1 | ARD | MaxRD | MinRD | AVT(s) | % optimal | ARD | Max RD | Min RD | AVT(s) | % optimal | ARD | Max RD | Min RD | AVT(s) | % optimal |
| pb1 | | 0.5*n | **12,26%** | 15,99% | 5,15% | 46 | **0%** | 22,48% | 30,62% | 9,96% | <0,01 | **0%** | 21,12% | 28,62% | 12,50% | <0,01 | **0%** |
| pb2 | [1,99] | 0.6*n | **6,63%** | 12,81% | 0,60% | 45 | **0%** | 16,46% | 31,93% | 3,45% | <0,01 | **0%** | 13,97% | 24,93% | 4,48% | <0,01 | **0%** |
| pb3 | | 0.7*n | **1,97%** | 11,58% | 0,08% | 42 | **0%** | 8,82% | 30,35% | 2,64% | <0,01 | **0%** | 8,89% | 20,75% | 3,43% | <0,01 | **0%** |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| pb4 | | 0.5*n | **8,26%** | 9,81% | 6,12% | 42 | **0%** | 11,54% | 14,29% | 6,74% | <0,01 | **0%** | 11,16% | 14,65% | 7,81% | <0,01 | **0%** |
| pb5 | [25,75] | 0.6*n | **1,86%** | 4,03% | 0,22% | 41 | **0%** | 7,36% | 12,33% | 2,78% | <0,01 | **0%** | 3,98% | 8,07% | 1,36% | <0,01 | **0%** |
| pb6 | | 0.7*n | **0,08%** | 0,48% | 0% | 36 | **33%** | 3,12% | 7,30% | 1,14% | <0,01 | **0%** | 1,34% | 2,54% | 0,28% | <0,01 | **0%** |
| pb7 | | 0.5*n | 5,06% | 6,24% | 4,05% | 47 | **0%** | 5,16% | 6,31% | 3,11% | <0,01 | **0%** | **4,90%** | 6,35% | 2,40% | <0,01 | **0%** |
| pb8 | [40,60] | 0.6*n | **0,53%** | 1,17% | 0% | 44 | **3%** | 2,46% | 4,74% | 0,52% | <0,01 | **0%** | 1,46% | 2,83% | 0,42% | <0,01 | **0%** |
| pb9 | | 0.7*n | **0,01%** | 0,09% | 0% | 38 | **80%** | 0,88% | 2,64% | 0% | <0,01 | **3%** | 0,40% | 0,81% | 0,11% | <0,01 | **0%** |
| pb10 | [10,49] | 0.5*n | 0,73% | 1,95% | 0% | 33 | **3%** | 0,60% | 3,00% | 0% | <0,01 | **33%** | **0,41%** | 0,95% | 0% | <0,01 | **3%** |
| pb11 | [50,99] | 0.6*n | **0%** | 0% | 0% | 29 | **100%** | 0,39% | 3,60% | 0% | <0,01 | **40%** | 0,34% | 0,79% | 0,04% | <0,01 | **0%** |
| pb12 | | 0.7*n | **0%** | 0% | 0% | 22 | **100%** | 0,13% | 2,65% | 0% | <0,01 | **40%** | 0,34% | 0,70% | 0,04% | <0,01 | **0%** |
| pb13 | [50,99] | 0.5*n | **0,05%** | 0,11% | 0% | 38 | **3%** | 0,62% | 1,60% | 0,21% | <0,01 | **0%** | 0,11% | 0,26% | 0% | <0,01 | **3%** |
| pb14 | [10,49] | 0.6*n | **0,06%** | 0,17% | 0,03% | 39 | **0%** | 0,83% | 1,27% | 0,47% | <0,01 | **0%** | 0,31% | 0,67% | 0,19% | <0,01 | **0%** |
| pb15 | | 0.7*n | **0,14%** | 0,21% | 0,03% | 40 | **0%** | 0,91% | 1,73% | 0,54% | <0,01 | **0%** | 0,95% | 4,94% | 0,24% | <0,01 | **0%** |
| pb16 | [5,100] | 0.5*n | **0,09%** | 0,21% | 0% | 48 | **7%** | 0,70% | 1,74% | 0,25% | <0,01 | **0%** | 0,58% | 1,95% | 0,04% | <0,01 | **0%** |
| pb17 | [1,20] | 0.6*n | **0,12%** | 0,25% | 0,04% | 48 | **0%** | 0,82% | 1,55% | 0,26% | <0,01 | **0%** | 0,93% | 2,12% | 0,11% | <0,01 | **0%** |
| pb18 | | 0.7*n | **0,22%** | 0,59% | 0,04% | 48 | **0%** | 0,79% | 1,30% | 0,36% | <0,01 | **0%** | 1,81% | 3,81% | 0,39% | <0,01 | **0%** |

**Problem of 50 jobs and m parallel machines:** In table 6; we presents a comparison of the heuristics with LB_B for problem instances involving 50 jobs, considering m= 5 and m = 10. This comparison includes instances from pb1 to pb3 and from pb13 to pb18. Observations for problem instances from pb1 to pb3, with m=5, reveal that out of the 90 instances across the three problems, the LS generates optimal solutions for 89% of instances, with an ARD not exceeding 0.03% and a maximum AVT of 8 seconds. Additionally, the SPT_S1 heuristic yields results matching LB_B for 67% of instances, with an ARD not exceeding 1.58% and an AVT less than 0.01 seconds. However, LPT_S2 is less efficient, resulting in fewer optimal solutions compared to LS and SPT_S1. Similarly, for m= 10, LS and SPT_S1 heuristics are the most efficient. LS provides results matching LB_B for 100% of instances, with a maximum AVT of 10 seconds. Meanwhile, SPT_S1 achieves results matching LB_B for 98% of instances, with an ARD not exceeding 0.2%. For problem instances from pb13 to pb15, it is evident that the heuristics perform better when m= 10. Specifically, LS achieves optimal results for 41% of instances when m=5, but this figure increases to 99% when m=10. Similarly, the SPT_S1 heuristic matches LB_B for 6% of instances when m= 5, but achieves this for 100% of instances when m=10. Conversely, LPT_S2 is comparatively less efficient in both scenarios. For problems from pb16 to pb18, when m =5, the results indicate that no heuristic generates results equal to LB_B. Furthermore, the performance of the LS deteriorates in this scenario. When m =10, the SPT_S1 heuristic matches LB_B for 20% of instances in problem pb18. Additionally, the LS achieves the same results as LB_B for 3% and 7% of instances in problems pb17 and pb18, respectively.

Table 6. Comparisons of heuristics with the best bound for 50 jobs, with m parallel machines

| | | | | **LS** | | | | | **SPT_S1** | | | | | **LPT_S2** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **inst** | **m** | **DU** | **n1** | **ARD** | **MaxRD** | **MinRD** | **AVT(s)** | **% optimal** | **ARD** | **Max RD** | **Min RD** | **AVT(s)** | **% optimal** | **ARD** | **Max RD** | **Min RD** | **AVT(s)** | **% optimal** |
| pb1 | 5 | [1,99] | 0.5*n | **0,03%** | 0,46% | 0% | 8 | 87% | 1,04% | 6,69% | 0% | <0,01 | 53% | 5,47% | 11,70 % | 0% | <0,01 | 3% |
| pb2 | | | 0.6*n | **0,01%** | 0,17% | 0% | 7 | 97% | 1,58% | 11,92 % | 0% | <0,01 | 63% | 3,03% | 12,45 % | 0% | <0,01 | 7% |
| pb3 | | | 0.7*n | **0%** | 0% | 0% | 7 | 83% | 0,45% | 9,15% | 0% | <0,01 | 83% | 2,34% | 7,63% | 0% | <0,01 | 10% |
| pb1 | 10 | | 0.5*n | **0%** | 0% | 0% | 10 | 100% | 0% | 0% | 0% | <0,01 | 100% | 3,12% | 7,15% | 0% | <0,01 | 7% |
| pb2 | | | 0.6*n | **0%** | 0% | 0% | 9 | 100% | 0% | 0% | 0% | <0,01 | 100% | 1,83% | 4,48% | 0% | <0,01 | 7% |
| pb3 | | | 0.7*n | **0%** | 0% | 0% | 9 | 100% | 0,20% | 5,95% | 0% | <0,01 | 93% | 1,78% | 4,64% | 0% | <0,01 | 10% |
| pb13 | 5 | [50,99] [10,49] | 0.5*n | **7,54%** | 14,74% | 0,31% | 10 | 0% | 9,89% | 19,08 % | 2,08% | <0,01 | 0% | 17,34 % | 29,18 % | 7,01% | <0,01 | 0% |
| pb14 | | | 0.6*n | **1,25%** | 7,75% | 0% | 10 | 30% | 5,72% | 17,22 % | 0,10% | <0,01 | 0% | 7,48% | 21,03 % | 0,50% | <0,01 | 0% |
| pb15 | | | 0.7*n | **0,01%** | 0,30% | 0% | 9 | 93% | 2,75% | 9,48% | 0% | <0,01 | 17% | 3,95% | 12,50 % | 0% | <0,01 | 3% |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| pb13 | | | 0.5*n | **0%** | 0,13% | 0% | 11 | 97% | 0% | 0% | 0% | <0,01 | 100% | 3,82% | 11,38 % | 0% | <0,01 | 3% |
| pb14 | 10 | | 0.6*n | **0%** | 0% | 0% | 10 | 100% | 0% | 0% | 0% | <0,01 | 100% | 2,30% | 5,78% | 0% | <0,01 | 10% |
| pb15 | | | 0.7*n | **0%** | 0% | 0% | 9 | 100% | 0% | 0% | 0% | <0,01 | 100% | 1,97% | 4,70% | 0% | <0,01 | 7% |
| pb16 | | | 0.5*n | 13,84% | 19,68% | 8,92% | 10 | 0% | **9,96%** | 15,06 % | 6,46% | <0,01 | 0% | 26,65 % | 39,58 % | 19,52 % | <0,01 | 0% |
| pb17 | 5 | | 0.6*n | 14,42% | 20,68% | 9,63% | 10 | 0% | 11,48% | 16,54 % | 8,39% | <0,01 | 0% | **2,30%** | 45,51 % | 18,68 % | <0,01 | 0% |
| pb18 | | [5,100] | 0.7*n | 17,38% | 24,50% | 12,00% | 10 | 0% | 11,56% | 17,27 % | 6,63% | <0,01 | 0% | **1,97%** | 44,48 % | 18,52 % | <0,01 | 0% |
| pb16 | | [1,20] | 0.5*n | 29,25% | 41,43% | 17,02% | 12 | 0% | **23,84 %** | 34,66 % | 12,46 % | <0,01 | 0% | 55,17 % | 78,72 % | 31,61 % | <0,01 | 0% |
| pb17 | 10 | | 0.6*n | 17,31% | 29,97% | 2,82% | 12 | 0% | **13,79 %** | 26,91 % | 1,08% | <0,01 | 0% | 43,92 % | 76,27 % | 29,28 % | <0,01 | 0% |
| pb18 | | | 0.7*n | 7,08% | 16,03% | 0,23% | 12 | 0% | **4,%** | 14,29 % | 0% | <0,01 | 20% | 29,71 % | 48,84 % | 12,44 % | <0,01 | 0% |

## Conclusion

In this study, we addressed the FH2(P,PD2)|rj,block|Cmax problem, an extension of the hybrid flow shop problem. This variant entails a setup with multiple parallel machines at stage 1 and two dedicated machines at stage 2. A critical aspect is that job execution on a machine in stage 1 is contingent upon job availability. Moreover, once a job occupies a machine in stage 1, it holds that machine until its operation concludes, transitioning thereafter to stage 2's dedicated machines. We tailored a mathematical model to suit this specific problem, accounting for the blocking constraint. Utilizing two established lower bounds from prior literature—originally devised for the FH2(P,PD2)|$r_j$|Cmax problem we adjusted them to accommodate the added blocking constraint. Consequently, we presented a novel formula for calculating the makespan, crucial in this adaptation. Our investigation revealed that, despite the inclusion of the blocking constraint, Local Search (LS) outperformed classical priority rules. Moving forward, our research will explore alternative methodologies, including constraint programming and exact procedures like Branch and Bound, to further enhance our understanding and address the complexities of this problem.

## References

[1] Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In Annals of discrete mathematics (Vol. 5, pp. 287-326). Elsevier.

[2] Nabli, Zouhour, OuajdiKorbaa, and SoulefKhalfallah. (2016) "Mathematical programming formulations for hybrid flow shop scheduling with parallel machines at the first stage and two dedicated machines at the second stage"*Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*(pp. 004389-004393). IEEE.

[3] Nabli, Zouhour, SoulefKhalfallah, and OuajdiKorbaa,(2017)"Heuristics for the hybrid flow shop scheduling problem with parallel machines at the first stage and two dedicated machines at the second stage."In: Intelligent Systems Design and Applications. ISDA 2017. Advances in Intelligent Systems and Computing, vol 736. Springer

[4] Yu, C., Semeraro, Q., & Matta, A. (2018). A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility. Computers & Operations Research, 100, 211-229.

[5] Hajji, M. K., Hadda, H., & Dridi, N. (2023). Makespan Minimization for the Two-Stage Hybrid Flow Shop Problem with Dedicated Machines: A Comprehensive Study of Exact and Heuristic Approaches. Computation, 11(7), 137.

[6] Aqil, S., & Allali, K. (2021). Two efficient nature inspired meta-heuristics solving blocking hybrid flow shop manufacturing problem. Engineering Applications of Artificial Intelligence, 100, 104196.

[7] Qin, H. X., Han, Y. Y., Zhang, B., Meng, L. L., Liu, Y. P., Pan, Q. K., & Gong, D. W. (2022). An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem. Swarm and Evolutionary Computation, 69, 100992.

[8] Nabli, Z., Khalfallah, S., & Korbaa, O. (2018). A two-stage hybrid flow shop problem with dedicated machine and release date. Procedia Computer Science, 126, 214-223.