A two-stage hybrid flow shop problem with dedicated machine and release date

Nabli Zouhour[1], Soulef Khalfallah[2] and Ouajdi Korbaa[1]

[1] Higher Institute of Computer Science and Telecom (ISITCom), MARS (Modeling of Automated Reasoning Systems) Research Lab LR17ES05, University of Sousse, Tunisia

[2] Higher Institute of Management, University of Sousse, Tunisia

nablizouhour@yahoo.fr

Abstract. This paper presents a mathematical model, three heuristics and two lower bounds for the resolution of the hybrid flow shop scheduling problem with m_e parallel machines at the first stage and two dedicated machines at the second stage. Each job has a release date at the first stage, which cannot start before it. The objective is to minimize the makespan.

Keywords: Hybrid Flow-Shop, Dedicated Machines, Parallel Machines, Release Date, Mathematical model, Heuristics.

1 Introduction

The hybrid flow shop problem HFS is an extension of the classical flow shop problem. It is composed of S stage. The stage e (e = 1, ... S) is composed of m_e parallel machines (identical or not identical) where me≥2 in at least one stage. There are a number of variants HFS, all of which have most of the following common characteristics: the number of stages S is at least 2, jobs must be processed on a single machine on each stage, the order of stages is the same for all jobs (stage 1, stage 2, ..., stage S). The problem is to find a schedule (assigning the jobs to the machines on the different stages) in order to optimize a given objective function. According to[1] This problem was first identified by Salvador [2] who suggested a Branch-and-Bound approach to solve a production system in the synthetic fibres industry as a no-wait HFS, the objective is to minimize the makespan. Similarly, Haouari & al. [3] presented an exact Branch-and-Bound algorithm for the two-stage hybrid flow shop problem and a set of identical parallel machines in each stage, the objective was the minimization of the makespan. Tang and al. [4] presented the mathematical formulation of the hybrid flow shop with identical parallel machines. More recently, Cui & Gu, [5] modeled HFS problem by vector representation, and proposed an improved discrete artificial bee colony (IDABC) algorithm to minimize the makespan. .All approaches proposed by the authors, lead to an improvement in terms of producing more jobs on time while minimizing the makespan compared to the decision rules used generally by industries. There is several extension of the problem of the HFS, in this paper we will present

and solve a particular case HFS with dedicated machine at S2. This type of problem is characterized by the following configuration: one or more stages contain specialized machines for processing of certain operations. The problem of the flow shop with dedicated machine also contains different types of organization, among which we can mention: Riane & al. [6] treat a problem of scheduling n jobs on a three stages hybrid flow shop of particular structure (one machine in the first and third stages and two dedicated machines in stage two). The objective is to minimize the makespan. They propose two heuristics procedures to cope with realistic problems, and represent the problem formulation, with a mixed integer linear program model MIP that, when relaxed to an ordinary linear program, serves to generate lower bounds. Yang [7] consider the problem of minimizing total completion time in a two-stage HFS scheduling with dedicated machines at stage and one machine at stage 1. Lin [8] considers the manufacturing model that has a common machine at stage one and two parallel dedicated machines at stage two. He elaborates several published works on makespan minimization which are not known to other streams of recent works. He develops a linear-time algorithm concerning the case where two sequences of the two job types are given a priori. In a previous work, we have studied the problem of the HFS with m_e parallel machines at the first stage and two dedicated machines at the second stage. In [9] we presented and compared two MIP with lower bounds, in [10] we presented four heuristics and two lower bounds, and we compared the results obtained with the results obtained in [9]. In this paper, we will propose a MIP, three heuristics and two lower bounds, to solve the same problem while taking in to consideration the release date constraint: each job has a release date at the first stage, which cannot start before it. Following the α/β/γ notation of graham et al. [11], this problem is denoted $F2(P,2)|r_j|Cmax$.

2 Mathematical model

In [9] we proposed two MIP (M1 and M2). The first is based on time indexed variables and the second is based on linear ordering variables. When, we compare the computational time performances of M1 and M2, we find that M2 (linear ordering variable) is faster than M1 (time index variables). So, we will adapt M2 to take into consideration the release date constraint.
The mathematical formulation of the problem is as follows:

- **Sets**

J set of jobs, indexed j = 1,…, n ;
M set of machines, indexed m=1,…, Ma ;
Me set of machine at stage e indexed m=1,…, me ;
E set of stage indexed e=1,…,Et
l position of job on machine, indexed 1,…, n ;

- **Parameters**

P_{ej} processing time of job j at stage e,
r_j: release date of job j at stage 1,

BM a large number (i.e, "big-M"), BM≥$\sum_j p_j$

$Y_{jem}=\begin{cases}1 \text{ if job j is préaffected to machine m à l'étage e}\\ 0, otherwise\end{cases}$

- **Decision variables**

C_{ej} completion time of job j at stage e,

$d_{elj} = \begin{cases}\mathbf{1} \text{ if job } \boldsymbol{j} \text{ precedes job } \boldsymbol{l} \text{ at stage e;}\\ \mathbf{0}, otherwise\end{cases}$

$z_{ejm} = \begin{cases}\mathbf{1} \text{ if job } \boldsymbol{j} \text{ est positioned on machine } \boldsymbol{m} \text{ at stage e,}\\ \mathbf{0}, \text{sinon}\end{cases}$

$u_{elj} = \begin{cases}\mathbf{1} \text{ if job } \boldsymbol{j} \text{ and } \boldsymbol{l} \text{ are not scheduled on same}\\ \text{machine } \boldsymbol{m} \text{ at stage e;}\\ \mathbf{0}, \text{sinon}\end{cases}$

- **The model**

Objective functions: min $Cmax$

$d_{elj} + d_{ejl} + u_{elj} = 1$	e ∈ E $l \in J$ $j \in J, l < j$	(1)
$d_{elj} + d_{ejk} + d_{ekl} \leq 2$	e ∈ E $l \in J$ $j \in J$ $k \in J$, $l < j < k$	(2)
$z_{elm} + z_{ejm} + u_{elj} \leq 2$	e ∈ E $l \in J$ $j \in J, l < j$ $m \in m_e$	(3)
$\sum_{m \in me} z_{elm} = 1$	e ∈ E $l \in J$	(4)
$C_{ej} \geq r_j + p_{ej} z_{ejm}$	$j \in J$ $m \in m_e$	(5)
$c_{ej} \geq c_{el} + p_{ej}(d_{elj} + z_{elm} + z_{ejm} - 2) - BM(1 - d_{elj})$	e ∈ E $l \in J$ $j \in J$ $m \in m_e$	(6)
$c_{ej} \geq c_{e-1j} + p_{ej}$	e ∈ $E, e = 2$ $j \in J$	(7)
$Cmax \geq C_{2j}$	j ∈ J	(8)
$y_{jem} = 0 : z_{ejm} = 0$	$e = 2$ j ∈ J $m \in m_e$	(9)

(1) For each stage e, 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 e, calculates the u_{ejm} variable (assignment of jobs to machines)
(4) For each e stage, each job is positioned on a machine.
(5) Completion time C_{ej} 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_{ej} 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 two lower bounds. We add some useful notations as follow;

P1: The sum of processing times of jobs at stage 2 that are dedicated to machine 1: $P_1 = \sum_{j \in N1} P_{2j}$

P2: T The sum of processing times of jobs at stage 2 that are dedicated to machine 2: $P_2 = \sum_{j \in N2} P_{2j}$

The LB could not be smaller than the longest processing time (LPT), since the job with the longest processing time should be complete. LB_1 is based on the lower bound proposed by Carlier [12], to solve the identical parallel machines scheduling problem. This lower bound particular performance for the case where the processing time on stage 1 is strictly greater than the processing time of jobs on stage 2.

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

When the processing times in stage 2 are smaller than the processing times in stage 1 the performance of LB_1 decreases. To solve this problem we have proposed an LB where we relax the job's waiting time between stages, LB_2 is defined as follow:

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

Evidently, at stage 1 no job j can finish its processing earlier than the smallest value of the sum of processing time at stage 1 plus the release date. For stage 2 where we have two dedicated machine, we have to wait until the machine with maximum value of sum processing time complete treatment.

4 Heuristics

In this part we will adapt the local search heuristics and two priority rules presented in [10] to solve the HFS problem with parallel machines in the first stage and two dedicated machines in the second stage with release date constraint. For all heuristics, the release date will be taken into consideration when calculating the start time of the jobs. For stage 2, the jobs will be processed in non-decreasing order according to completion time at stage 1 (FIFO).

4.1 Local Search

The principle of this heuristic is to generate random permutations and apply the local search method on each permutation, while taking into consideration the release date of jobs in the calculation of the dates of the start and the end of processing of the jobs. This heuristic is assimilated to a multi-start ascending procedure. At each iteration, we compare the best result obtained with the current on. The better one becomes current. So, this heuristic allows us to visit a large number of possible solutions and improve them. The generation of permutations concerns only stage 1, the job assignment is based on first available. The different steps of this heuristic are as follows:

1- Generate a set of permutations of jobs randomly on stage 1.

For each permutation:

2- Assign the jobs one by one on the first available machine on stage 1, in non-decreasing order of starting time on S1. taking into consideration the release date of jobs in the calculation of the start time of the jobs.
3- Assign jobs to machines at stage 2, according to the completion time of jobs (First Completion Time) at stage 1.
4- Make swap permutations on the sequence of stage 1 and retain the permutation improved Cmax. Stop the swap permutation when we have not had any improvements in the current.

4.2 LPT (longest processing time) on stage 2

The principle of this priority rule is to order the jobs (in non-increasing order) according to their processing time on stage 2. So, jobs that have the maximum processing time on the stage 2 will be treated first. The objective of this priority rule is to occupy the machines on stage 2 as much as possible at the beginning until the jobs in stage 1 are processed which minimizes the waiting time on stage 2. The different steps of this priority rule are as follows:

1- Group the jobs on stage 1, according to the machine on which they will be processed on stage 2.
2- Sort the jobs within each group according to the processing time of jobs on stage 2.
3- Sort the jobs on stage 1 in a table by taking a job from group 1 and a job from group 2.
4- Execute the jobs one by one on the first machine available on stage 1, while taking into consideration the release date of jobs.
5- Process the jobs on stage 2 according to the first completion time on stage 1.

Table 1 presents an example of processing time of job j at stage 1 and stage 2, and job pre-affectation to machine at stage 2:

Table 1: Processing time of job j at S1 and S2, and job pre-affectation to machine at S2.

job	r_j	P_{1j}	P_{2j}	M_{21}	M_{22}
j1	3	11	11	1	0
j2	0	9	6	1	0
j3	8	14	7	1	0
j4	3	15	12	0	1
j5	4	5	14	0	1

The steps to solve this example with LPT priority rule are as follows:

1- Step 1: G1={j1,j2,j3} G2={j4,j5}

2- Step 2: G1={j1,j2,j3} G2={j5,j4}
3- Step 3: {j1, j5,j2,j4,j3}

Figure 1 (fig1.) represents Gantt chart of the example after applying the LPT priority rule.

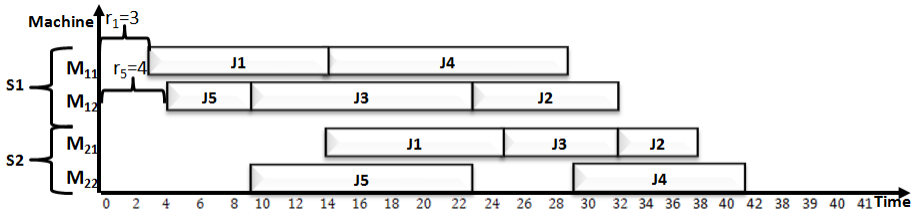

Fig. 1. Gantt chart

4.3 SPT (Smallest Processing Time) on stage 1

The principle of this priority rule is to order jobs (in non-decreasing order), according to their processing times on stage 1 and their release dates. In this way, the machines on stage 2 begin processing as soon as possible. Then, we apply an improvement on the results obtained by making swap permutations. The different steps of this priority rule are as follows:

1- Sort the jobs in non-decreasing order of starting time on S 1, according to their processing times and their release date on stage 1.
2- Process the jobs one by one on the first available machine on stage 1.
3- Process the jobs on stage 2, according to the first completion time on stage 1.
4- Make swap permutations on the sequence of stage 1 and retain the permutation that improved result. Stop the swap permutation when we have not had any improvements in the iteration.

5 Computational analysis

5.1 Instances generation

In this work, regarding processing times, we generate data based on yang [13]. We searched in the literature for benchmark problems (input) for the HFS with this configuration, but all articles do not include their input data, they only explain how they generate their processing times. We will test problems under different conditions of *m*, *n*, *n1*, *n2* that we call problem family where n is the number of jobs, n1 number of jobs processed on machine 1 at stage 2, n2 number of jobs processed on machine 2 at stage 2, and m the number of identical parallel machines at stage 1. There are 45 families of problems. For each family, the processing times are generated randomly based on a uniform distribution; $P_{ej} \in$ [PLB, PUB] where PLB is a lower limit of the processing time and PUB is an upper limit. To test the effects of the number of jobs, three different values of n are considered: 10, 50, and 100 as in [9]. To determine the effect of job assignment (The number of jobs dedicated to the first and second ma-

chine at stage 2), we consider three different combinations of *n* and *n1*. In the fist we assume that the number of jobs are equal (*n1* = 0.5**n*), in the others, we assume that the number of jobs pre-assigned to a machine is greater than the number of jobs pre-assigned to the other. (*n1* = 0.6**n* and *n1* = 0.7* *n*). We consider three different distributions for processing times:

P_{ej} : ∈ [1; 99]; with standard deviation 29.42.

P_{ej} : ∈ [25; 75]; with standard deviation 14.93.

P_{ej} : ∈ [40; 60]; with standard deviation 6.17.

We added two other distributions for which the processing time of stage 1 and stage 2 are different:

P_{1j} : ∈ [10 ; 49]; P_{2j}[50; 99]; with standard deviation 25.58.

P_{1j} : ∈ [50 ;99]; P_{2j}[10;49]; with standard deviation 25.64 (Closer to real settings).

Finally, to test the effect of numbers of parallel machines on stage 1, three values are considered: 2, 5, and 10. In these cases, we fixed n to 50 jobs. For each family of problems where n≤50, we generated 5 instances. 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: Local Search, SPT E1: Smallest Processing Time on stage 1, LPT E2: longest processing time on stage 2, STPT: Shortest total processing time). 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. 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, firstly we compare the best lower bound max (LB1 and LB2) to the result of MIP which is either an UB (Upper bound) or an optimal solution. Secondly, we compare the results obtained by the LS heuristic and the priority rules (SPT E1 and LPT E2) with the available bound (optimal solution or best lower bound). We tested the heuristics over 5 instances; Each instance is composed of 15 problems for 5 different distributions for P_{ej} .

Comparison of the best lower bound with MIP

In table 2; we compare the best lower bound max (LB_1 and LB_2) to the result of MIP model which is either an UB (Upper bound) or an optimal solution for problems with 10, 50 and 100 jobs, with 2 parallel machines at stage1. Where n=10; the MIP is solved to optimality; with computational time do not exceed 2 seconds. For all problems the LB corresponds to the optimal solution for the 5 instances. Where n=50; we compare the UB with the best lower bound. For the first 12 problems the lower bound LB1 gave the best results. And for problem 13, 14 and 15; when processing time in stage 2 are less than the processing time in stage 1, we used LB2, since it is better adapted for these cases.

Table 2. Comparison of the best lower bound with the upper bound for 10, 50 and 100 jobs with 2 parallel machines.

			10 jobs				50 jobs			
Inst	DU	(n_1,n_2)	ARD	Max RD	Min RD	AVT (s)	ARD	Max RD	Min RD	AVT (s)
pb1	[1,99]	(5,5)	0%	0%	0%	1,83	6%	12%	0%	6494
pb2		(6,4)	0%	0%	0%	1,83	2%	6%	0%	6497
pb3		(7,3)	0%	0%	0%	1,79	1%	5%	0%	5842
pb4	[25,75]	(5,5)	0%	0%	0%	1,83	14%	25%	2%	7088
pb5		(6,4)	0%	0%	0%	1,76	8%	22%	1%	7211
pb6		(7,3)	0%	0%	0%	1,77	2%	5%	0%	7212
pb7	[40,60]	(5,5)	0%	0%	0%	1,83	18%	29%	14%	7208
pb8		(6,4)	0%	0%	0%	1,82	1%	3%	0%	6347
pb9		(7,3)	0%	0%	0%	1,80	3%	7%	0%	7214
pb10	[10,49] [50,99]	(5,5)	0%	0%	0%	1,76	4%	16%	0%	4655
pb11		(6,4)	0%	0%	0%	1,74	1%	4%	0%	6771
pb12		(7,3)	0%	0%	0%	1,81	1%	2%	0%	5994
pb13	[50,99] [10,49]	(5,5)	0%	0%	0%	1,85	1%	2%	1%	7217
pb14		(6,4)	0%	0%	0%	1,85	1%	3%	0%	5770
pb15		(7,3)	0%	0%	0%	1,87	3%	5%	1%	7067

In table 3; we compare the best lower bound max (LB_1 and LB_2) to the result of MIP which is either an UB (Upper bound) or an optimal solution. We set n to 50 jobs for different number of parallel machines at stage 1; m = 5 and 10. We tested two cases: the first when the processing times P_{ej}: ϵ[1;99] and the second when $P_{1j} \in$ [50; 99] and $P_{2j} \in$ [10; 49]. In case 1 problems; the results obtained by the lower bounds are very close to the values obtained by the MIP with a maximum ARD of 1%. In case 2 problems; the results obtained by the lower bounds are close to the values obtained by the MIP with a maximum ARD of 12% for 5 parallel machines and a maximum ARD of 2% for 10 parallel machines.

Table 3. Comparison of the best lower bound with the upper bound for 10, 50 and 100 jobs, with m parallel machine.

50 jobs							
Inst	m	DU	(n_1,n_2)	ARD	Max RD	Min RD	AV time M1
pb1	5	[1,99]	(25,25)	1%	6%	0%	7217,88
pb2			(30,20)	0%	1%	0%	7224,53
pb3			(35,15)	0%	1%	0%	7231,75
Pb4	10		(25,25)	1%	1%	0%	7238,71
Pb5			(30,20)	0%	0%	0%	7221,83
Pb6			(35,15)	0%	1%	0%	7226,23
Pb7	5	[50,99] [10,49]	(25,25)	12%	26%	5%	7205,87
Pb8			(30,20)	3%	7%	0%	7237,99
Pb9			(35,15)	1%	3%	0%	7207,43
Pb10	10		(25,25)	2%	5%	0%	7228,32
Pb11			(30,20)	2%	8%	0%	7210,72
Pb12			(35,15)	0%	1%	0%	6563,03

Comparisons of heuristics with the best bound

Problem of 10 jobs and 2 parallel machines: In table 4; where n = 10, the MIP is solved to optimality. Indeed, we compared the heuristics with the optimal solution. The results obtained show that the heuristic LS gives results that are very close or equal to the optimal, for most of the instances with a maximum ARD of 0.7% and a computational time (CT) lower than 4 second. In the second place, the heuristic LPT_E2 gave a maximum average deviation of 8.1% with computational time less than that of the heuristic LS. In the third place, the SPT_E1 rule gave a maximum average deviation of 10.3% with computational time do not exceed 1 second.

Table 4. Comparisons of heuristics with the best bound for 10 jobs

inst	DU	(n_1,n_2)	RL				SPT_E1				LPT_E2			
			ARD	Max RD	Min RD	AVT (s)	ARD	Max RD	Min RD	AVT (s)	ARD	Max RD	Min RD	AVT (s)
pb1	[1,99]	(5,5)	0%	1%	0%	3,6	7,7%	13,8%	0%	0,0	8,1%	16,3%	0%	0,0
pb2		(6,4)	0,2%	0,7%	0%	3,4	8,0%	11,9%	0%	0,0	3,9%	10,0%	0%	0,0
pb3		(7,3)	0%	0%	0%	3,3	2,1%	10,6%	0%	0,0	2,3%	11,0%	0%	0,0
pb4	[25,75]	(5,5)	0,7%	2,1%	0%	3,3	4,2%	8,6%	0%	0,0	6,5%	9,1%	3,0%	0,0
pb5		(6,4)	0%	2%	0%	3,5	1,7%	5,0%	0%	0,1	5,4%	8,0%	3%	0,0
pb6		(7,3)	0%	0%	0%	3,4	0,5%	2,4%	0%	0,0	6,3%	11,7%	2%	0,0
pb7	[40,60]	(5,5)	0,3%	0,7%	0%	3,3	10,3%	18,2%	1,3%	0,0	2,2%	5,5%	0%	0,0
pb8		(6,4)	0%	0%	0%	3,4	0,3%	1,3%	0%	0,0	0,9%	3,2%	0%	0,1
pb9		(7,3)	0%	0%	0%	3,5	0,0%	0,0%	0%	0,0	0,2%	0,8%	0%	0,0
pb10	[10,49]	(5,5)	0%	0%	0%	3,3	0,0%	0,0%	0%	0,0	3,5%	8,7%	0%	0,0
pb11		(6,4)	0%	0%	0%	3,4	0,0%	0,0%	0%	0,0	3,1%	7,4%	0%	0,0
pb12		(7,3)	0%	0,0%	0%	3,5	0,0%	0,0%	0%	0,0	2,6%	5,9%	0%	0,0
pb13	[50,99]	(5,5)	0%	1%	0%	3,5	6,3%	14,2%	0,8%	0,0	5,7%	10,0%	3,3%	0,0
pb14		(6,4)	0%	1%	0%	3,6	5,5%	14,2%	0,8%	0,0	3,9%	7,3%	1,7%	0,0
pb15		(7,3)	0%	0%	0%	3,5	5,6%	14,2%	1,3%	0,0	4,2%	8,1%	1,1%	0,1

Problem of 50 jobs and 2 parallel machines: In table 5; where n = 50, the MIP was not able to give the optimal solution for most of the instances in acceptable time. So we compared the heuristic and the priority rules with the best lower bound. The results obtained show that the LS heuristic generates results equal or very close to the lower bound for most instances with a maximum ARD of 1% and maximum computational time of 8 seconds. The ARD of the priority rules LPT E2, and SPT_E1 is a little higher compared to the LS heuristic. On the other hand, we note that even when n = 50, these heuristics do not exceed 1 second of computational time. In fact, priority rules have $O(n \log n)$ complexity.

Table 5. Comparisons of heuristics with the best bound for 50 jobs

inst	DU	(n_1,n_2)	RL				SPT_E1				LPT_E2			
			ARD	Max RD	Min RD	AVT (s)	ARD	Max RD	Min RD	AV T (s)	ARD	Max RD	Min RD	AVT (s)
pb1	[1,99]	(25,25)	1%	3%	0%	7,6	2,5%	8,6%	0%	0,0	3,6%	6,0%	1%	0,0
pb2		(30,20)	0,1%	0,4%	0%	7,5	0,2%	0,4%	0%	0,0	2,3%	5,0%	1%	0,2
pb3		(35,15)	0%	0%	0%	7,1	0,1%	0,3%	0%	0,2	2,5%	4,4%	1%	0,0
pb4	[25,75]	(25,25)	0,2%	0,4%	0%	7,7	1,3%	2,4%	0%	0,0	0,9%	1,3%	0,4%	0,0
pb5		(30,20)	0%	0%	0%	8,4	0,0%	0,0%	0%	0,1	0,9%	1,6%	0%	0,1
pb6		(35,15)	0%	0%	0%	6,9	0,0%	0,0%	0%	0,1	0,8%	1,3%	0%	0,1
pb7	[40,60]	(25,25)	1%	2,1%	0%	6,3	4,5%	7,4%	0,4%	0,0	0,5%	0,8%	0%	0,0
pb8		(30,20)	0%	0%	0%	6,7	0,6%	2,9%	0%	0,1	0,5%	1,1%	0%	0,0
pb9		(35,15)	0%	0%	0%	6,5	0,5%	2,4%	0%	0,1	0,4%	1,0%	0%	0,1
pb10	[10,49]	(25,25)	0%	0%	0%	7,5	0,3%	0,7%	0%	0,0	1,0%	1,5%	0%	0,0
pb11		(30,20)	0%	0%	0%	7,4	0,1%	0,5%	0%	0,0	1,2%	1,7%	1%	0,1
pb12		(35,15)	0%	0,0%	0%	7,5	0,0%	0,0%	0%	0,1	1,0%	1,5%	0%	0,1
pb13	[50,99]	(25,25)	0%	0%	0%	6,9	2,2%	3,2%	1,2%	0,0	1,0%	1,5%	0,4%	0,0
pb14		(30,20)	0%	0%	0%	6,5	1,8%	2,2%	1,2%	0,0	1,4%	1,8%	0,8%	0,1
pb15		(35,15)	0%	0%	0%	6,9	2,2%	3,1%	1,2%	0,0	1,5%	2,6%	0,8%	0,1

Problem of 100 jobs and 2 parallel machines In table 6; Where n = 100, the MIP was not able to give the optimal solution for most of the instances. We compared the heuristics with the best lower bound. LS heuristic give results equal or very close to the lower bound for most instances with a maximum ARD of 1.3% and maximum computational time of 15.2 seconds. In the second place, LPT E2 rule gave a maximum ARD of 1.5% with computational time less than 1 second. In the third place, the ARD of the heuristics SPT E1 gave a maximum ARD of 3% with computational time less than 1 second.

Table 6. Comparisons of heuristics with the best bound for 100 jobs

inst	DU	(n_1,n_2)	RL				SPT_E1				LPT_E2			
			ARD	Max RD	Min RD	AVT (s)	ARD	Max RD	Min RD	AVT (s)	ARD	Max RD	Min RD	AVT (s)
pb1	[1,99]	(50,50)	0%	1%	0%	14,9	1,3%	3,8%	0%	0,0	1,2%	2,2%	0%	0,0
pb2		(60,40)	0%	0,2%	0%	14,9	0,0%	0,2%	0%	0,0	1,5%	2,2%	1%	0,1
pb3		(70,30)	0%	0%	0%	14,8	0,0%	0,2%	0%	0,1	1,2%	1,9%	1%	0,0
pb4	[25,75]	(50,50)	0,4%	0,9%	0%	14,5	1,3%	3,4%	0%	0,0	0,9%	1,6%	0,2%	0,0
pb5		(60,40)	0%	0%	0%	15,1	0,0%	0,2%	0%	0,0	0,8%	1,4%	0%	0,0
pb6		(70,30)	0%	0%	0%	14,9	0,0%	0,1%	0%	0,1	0,6%	1,2%	0%	0,0
pb7	[40,60]	(50,50)	1,3%	1,9%	0%	15,2	3,0%	4,2%	2,0%	0,0	0,7%	1,1%	0%	0,0
pb8		(60,40)	0%	0%	0%	14,8	0,6%	1,5%	0%	0,1	0,2%	0,3%	0%	0,1
pb9		(70,30)	0%	0%	0%	14,4	0,4%	1,1%	0%	0,1	0,2%	0,4%	0%	0,1
pb10	[10,49]	(50,50)	0%	0%	0%	14,4	0,1%	0,3%	0%	0,0	0,6%	1,0%	0%	0,0
pb11		(60,40)	0%	0%	0%	14,8	0,0%	0,1%	0%	0,1	0,5%	0,8%	0%	0,0
pb12		(70,30)	0%	0,1%	0%	14,8	0,0%	0,1%	0%	0,1	0,4%	0,7%	0%	0,1
pb13	[50,99]	(50,50)	0%	0%	0%	15,0	1,0%	1,3%	0,8%	0,0	0,4%	1,0%	0,2%	0,0
pb14		(60,40)	0%	0%	0%	15,2	1,0%	1,3%	0,8%	0,1	0,4%	0,5%	0,2%	0,0
pb15		(70,30)	0%	0%	0%	14,9	1,0%	1,4%	0,8%	0,1	0,6%	1,1%	0,2%	0,1

Problem of 50 jobs and m parallel machines In table 6; we compared the heuristics with the best lower bound. We set n to 50 jobs for different number of parallel machines at stage 1; m = 5 and 10. We tested two cases: the first when the processing times Pej: ∈[1;99] and the second when P1j∈ [50; 99] and P2j ∈ [10; 49]. In case 1, the results show that the LS give the same results as the LB in a computational time that does not exceed 4.2 seconds. In the second place, the SPT_E1 rule with maximum ARD of 0.4%. In the third place, the LPT_E2 rule with a maximum ARD of 1.8% with computational time less than 1 second also. In case 2, the results confirm the performance of the LS heuristic with a maximum ARD of 1% and computational time that does not exceed 4.2 seconds. In the second place, LPT_E2 rule with a maximum mean deviation of 1.5%. . In the third place, the SPT_E1 rule with maximum ARD of 2.8%.

Table 7. Comparisons of heuristics with the best bound for 50 jobs

Inst	m	DU	(n_1,n_2)	RL				SPT_E1				LPT_E2			
				ARD	Max RD	Min RD	AVT (s)	ARD	Max RD	Min RD	AVT (s)	ARD	Max RD	Min RD	AVT (s)
pb1	5	[1,99]	(25,25)	0%	1%	0%	4,2	0,4%	1,2%	0%	0,0	1,8%	4,2%	0,4%	0,0
pb2			(30,20)	0%	0,4%	0%	4,1	0,1%	0,4%	0%	0,0	1,0%	1,4%	0,6%	0,0
pb3			(35,15)	0%	0,3%	0%	4,2	0,1%	0,3%	0%	0,0	0,8%	1,2%	0,3%	0,1
pb4	10		(25,25)	0%	1%	0%	4,0	0,4%	1,2%	0%	0,0	1,2%	1,5%	0,4%	0,0
pb5			(30,20)	0%	0,4%	0%	4,1	0,1%	0,4%	0%	0,1	0,5%	1,1%	0,0%	0,0
pb6			(35,15)	0%	0,3%	0%	4,0	0,1%	0,3%	0%	0,1	0,4%	1,0%	0,0%	0,0
pb4	5	[50,99] [10,49]	(25,25)	1%	4%	0%	4,1	2,8%	5,4%	0%	0,0	1,5%	4,7%	0,0%	0,0
pb5			(30,20)	0%	0,2%	0%	3,9	0,0%	0,2%	0%	0,2	0,8%	2,6%	0,0%	0,1
pb6			(35,15)	0%	0,2%	0%	4,1	0,0%	0,2%	0%	0,1	0,9%	2,3%	0,0%	0,1
pb7	10		(25,25)	0%	0%	0%	4,1	0,2%	0,5%	0%	0,0	0,4%	1,0%	0,0%	0,0
pb8			(30,20)	0%	0,2%	0%	3,9	0,0%	0,2%	0%	0,1	0,4%	0,9%	0,0%	0,0
pb9			(35,15)	0%	0,2%	0%	4,2	0,0%	0,2%	0%	0,0	0,5%	0,8%	0,0%	0,0

6 Conclusion

In this paper we adapted a MIP, a local search heuristic and two priority rules for hybrid flow shop scheduling with parallel machines at the First stage and two dedicated machines at the second Stage with release date constraint. Firstly, we compared the MIP and two lower bounds. Then, the results obtained are compared with the result of the heuristic and the priority rules .Numerical results show that superiority of LS heuristic compared to priority rules.

References

1. Ribas, I., Leisten, R., & Framiñan, J. M. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. Computers & Operations Research, 37(8), 1439-1454 (2010).
2. Salvador, M. S.: A solution to a special class of flow shop scheduling problems. In Symposium on the theory of scheduling and its applications (pp. 83-91). Springer Berlin Heidelberg, (1973).
3. Haouari, M., Hidri L., Gharbi,: A. Optimal scheduling of a two-stage hybrid flow shop. Mathematical Methods of Operations Research, 64(1), 107-124 (2006).

4. Tang, L., Xuan, H., Liu, J.: A new Lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time. Computers & Operations Research, 33(11), 3344-3359 (2006).
5. Cui, Z., & Gu, X.: An improved discrete artificial bee colony algorithm to minimize the makespan on hybrid flow shop problems. Neurocomputing, 148, 248-259 (2015).
6. Riane, F., Artiba, A., & Elmaghraby, S. E. A hybrid three-stage flowshop problem: efficient heuristics to minimize makespan. European Journal of Operational Research, 109(2), 321-329 (1998).
7. Yang, J.: Minimizing total completion time in a two-stage hybrid flow shop with dedicated machines at the first stage. Computers & Operations Research, 58, 1-8 (2015).
8. Lin, B.M.: Two-stage flow shop scheduling with dedicated machines. International Journal of Production Research, 53(4), 1094-1097 (2015).
9. Nabli, Z., Korbaa, O., & Khalfallah, S.: Mathematical programming formulations for hybrid flow shop scheduling with parallel machines at the first stage and two dedicated machines at the second stage. In Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on (pp. 004389-004393). IEEE (2016).
10. Nabli, Z., Khalfallah, S., & Korbaa, O.: Heuristics for the hybrid flow shop scheduling problem with parallel machines at the first stage and two dedicated machines at the second stage. Accepted paper in the International Conference on Intelligent Systems Design and Applications. Springer (2017).
11. Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of discrete mathematics, 5, 287-326 (1979).
12. Carlier, J.: Scheduling jobs with release dates and tails on identical machines to minimize the makespan. European Journal of Operational Research, 29(3), 298-306 (1987).
13. Yang, J.: Hybrid Flow Shop with Parallel Machines at the First Stage and Dedicated Machines at the Second Stage. Industrial Engineeering & Management Systems, vol. 14, no 1, p. 22-31 (2015).