Int j simul model 21 (2022) 1, 5-16
Original scientific paper

TWO-STAGE HYBRID FLOWSHOP SCHEDULING PROBLEM WITH INDEPENDENT SETUP TIMES

Jemmali, M.[*,**,***]; Hidri, L.[****] & Alourani, A.[*,#]
[*] Department of Computer Science and Information,
College of Science at Zulfi, Majmaah University, AL-Majmaah 11952, Saudi Arabia
[**] Mars Laboratory, University of Sousse, Sousse, Tunisia
[***] Department of Computer Science, Higher Institute of Computer Science and Mathematics of Monastir, Monastir University, Monastir, 5000, Tunisia
[****] Industrial Engineering Department, College of Engineering,
King Saud University, 11421 Riyadh, Saudi Arabia
E-mail: a.alourani@mu.edu.sa ([#] Corresponding author)

Abstract
In this paper, a two-stage hybrid flowshop scheduling problem with independent setup times is examined. The complexity of the studied problem is NP-Hard in the strong sense since it generalized the two-stage hybrid flowshop scheduling problem. Several real-life problems in different areas are modelled using the addressed problem, as in parallel computing and manufacturing processes. Solving the current scheduling problem necessitates the construction of adequate algorithms, providing near-optimal solutions within a satisfactory computing time. In this study, a genetic algorithm with specific features and three other heuristics are developed. These three heuristics are based on the optimal solution of the parallel machine scheduling problem with release dates and delivery times. In order to assess the performance of the meta-heuristic and the heuristics, a family of three lower bounds is proposed. An exhaustive numerical study is performed over a total of 1920 test problems. The obtained results provide strong evidence of the efficiency and the performance of the proposed procedures.
(Received in August 2021, accepted in December 2021. This paper was with the authors 6 weeks for 2 revisions.)

Key Words: Two-Stage Hybrid Flowshop, Independent Setup Times, Genetic Algorithm, Heuristics, Lower Bounds

1. INTRODUCTION

Many real-life applications in manufacturing processes, industries, and parallel computing are modelled throughout the Hybrid flowshop scheduling problems (HFSP). In this context, several processes in the car industry, chemical industry, electronic, and semiconductors are modelled using the HFSP [1-4]. The HFSP shop is composed of a set of serial stages, where each one contains identical parallel machines. The processed jobs must visit all the stages from stage one to the last one in that sense. This study focuses on a particular case of HFSP, which is the two-stage HFSP, where the number of stages is limited to two. Plenty of research works are conducted for this scheduling problem and some of its variants. Indeed, this is due to the wide range of practical applications that it models, in addition to its theoretical challenging side. Several types of research are carried out to solve this scheduling problem and its variants. In this context, solution approaches as heuristics, meta-heuristics, and exact methods are proposed.

In order to narrow the gap between real-life applications and theory, some characteristics of either tasks or machines are included. In this context, the two-stage HFSP considering the transportation times between stages is examined in [5]. In this work, the exact method, several lower bounds, and heuristics are developed. An exhaustive experimental study shows that the proposed procedures are efficient. The availability constraint is taken into account for the two-stage HFSP in [6], where the solution is obtained using an exact algorithm. It is worth mentioning that in the later work, the first stage contains only one machine. In addition, the number of unavailability periods on each machine is restricted to one. The genetic algorithm

(GA) is used in order to provide a near-optimal solution for the two-stage HFSP with no-wait constraint in [7]. An efficient GA is proposed in the latter research work.

In [8] the authors study the HFSP problem with batch processing machines and release times. In the latter work a meta-heuristic algorithm based on variable neighbourhood search (VNS) is elaborated. The HFSP with total completion time minimization and with parallel dedicated machines is examined in [9]. A set of two efficient heuristics are presented in order to provide a near optimal solution within an acceptable computation time. In the latter problem the first stage contains only one machine while in the second stage there are two machines.

In [10] a two-stage HFSP with multi-objective function is addressed. The two considered objectives are the maximum completion time and the total consumed energy.

A mutant firefly meta-heuristic is proposed to solve a two-stage HFSP problem in [11]. Two simultaneous objectives are considered in the latter work.

The setup times use to be neglected in comparison with processing times, in some applications. In this study, the two-stage HFSP with setup times is considered. Considering the setup times within the preparation phase of production schedules permits to have accurate production planning. Authors in [12] addressed the HFSP with sequence-dependent setup time. An exact algorithm is proposed to provide optimal solution. In addition, three heuristics are proposed where one of them is based on the Hungarian method. Experimental study provides a strong evidence of the efficiency of the proposed procedures. Authors in [13] addressed the distributed HFSP with sequence-dependent setup times. A discrete artificial bee colony meta-heuristic is proposed to provide in near optimal solution. An experimental study carried out on 780 benchmarks test problem shows the efficiency of the proposed meta-heuristic. The HFSP with sequence-dependent setup times is studied in [14] where a near optimal solution is obtained by applying a simulated annealing algorithm.

Recently, authors in [15, 16] treated the two-machine flowshop scheduling problem blocking constraints. The algorithms used in the latter works can be utilized to extend the presented work. In addition, hybrid flowshop using simulation-optimization approach are developed in [17]. Other flow shop scheduling problem can be considered to utilize the proposed algorithms to be adopted with the studied problem such as problems detailed in [18-20].

The two-stage HFSP with independent setup times is examined in this paper. This problem is NP-Hard in the strong sense since it is a generalization of the well-known two-stage HFSP problem. In order to provide efficient solutions within satisfactory computation time, several solution approaches are presented. In this context, a GA-based meta-heuristic and a set of new heuristics are proposed. The core of the proposed heuristics is the optimal solution of the parallel machines scheduling with heads (release) and tails (delivery times). In order to assess the performance of the proposed approximation solution asset of lower bounds is presented. In this paper, an exhaustive experimental study is carried out on a total of 1920 test problems. Compared to the existing literature, one can observe that the large size problem is rarely treated. In this paper, the number of jobs reaches 200. As it will be shown in the experimental section, the proposed procedures (GA and the heuristics) are able to solve large-size problems within an acceptable computational time.

The remainder of this paper is organized as follows. Section 2 is reserved for the problem statement and the presentation of its relevant proprieties. In Section 3, the lower bounds are presented. The proposed GA and the heuristics are presented in Section 4. Section 5 is intended for the experimental study as well as the interpretation of the results. The findings and the future research directions are summarized in the conclusion.

2. PROBLEM STATMENT AND RELEVANT PROPRIETIES

The problem definition, as well as its relevant proprieties, are presented, in this section. The two-stage hybrid flowshop scheduling problem with independent setup times is defined as follows. A shop composed of two serial stages S_1 and S_2, containing each one m_i $(i = 1, 2)$ parallel and identical processors. In addition, we are given a set $J = \{J_1, J_2, \dots, J_n\}$ of n tasks for processing in the stages S_1 and S_2. The processing of each task J_j $(1 \leq j \leq n)$ is performed as follows. An available processor of S_1 is setup (prepared) to be ready for processing task J_j during a setup time $s_{1,j}$. Once the setup is finished, the task J_j is processed during $p_{1,j}$ unites of time on this processor. Completing the processing in the first stage, the task J_j is moved to the second stage S_2 where an available processor is setup during $s_{2,j}$. After that, the task J_j is treated on that processor during $p_{2,j}$. Once the processing is completed, the task J_j exits the system. The processing of the tasks is performed with the conditions as follows. The tasks are available to be processed from time 0. The processors are available for processing from time 0. A processor can handle one task at maximum for the same time. The preemption (interruption) is not permitted. The $s_{1,j}$, $s_{2,j}$, $p_{1,j}$, and $p_{2,j}$ are assumed to be integral and deterministic. The buffer's capacity between the two stages is infinite. If $t_{2,j}$ denotes the starting time for processing the task J_j in a feasible schedule for stage S_2, then its completion time is $c_{2,j} = t_{2,j} + p_{2,j}$. In this case, the maximum completion time (makespan) is: $C_{max} = \max_{1 \leq j \leq n}(c_{2,j})$.

The objective of this study is the determination of a feasible schedule that minimizes the maximum completion time. Following the three fields notation [21], the current studied scheduling problem is denoted $F(P_{m_1}, P_{m_2})m|s_{1,j}, s_{2,j}|C_{max}$.

Example 1: Considering $n = 10$ jobs to be processed in S_1 and S_2 with two processors in each stage ($m_1 = m_2 = 2$). The tasks' characteristics (processing and setup times) are presented in Table I.

Table I: Instance data for Example 1.

J_j	1	2	3	4	5	6	7
$s_{1,j}$	6	3	7	10	8	11	9
$p_{1,j}$	5	9	4	12	7	9	13
$s_{2,j}$	9	10	5	5	8	15	6
$p_{2,j}$	13	11	16	9	5	5	13

Fig. 1 presents a feasible solution for the Example 1. The maximum completion for this feasible schedule is $C_{max} = 76$. It is worth mentioning that the setup time and the processing time in the first stage S_1 are connected, then the setup time might be considered as a part of the processing time. In contrast, in the second stage S_2 it occurs that the setup time is independent from the processing time. This occurs for tasks $\{7, 2\}$. Indeed, task 7 finish its processing in stage S_1 at 22. For this reason, we can't start the execution of task 7 before 22 and the setup of task 7 is performed within the time interval $[0, 6]$. The task 2 finishes processing in stage S_1 at 12, therefore its setup time is scheduled in the second stage from time 0 to 10. The setup times of tasks $\{7, 2\}$ in the second stage are not a part of the processing time.

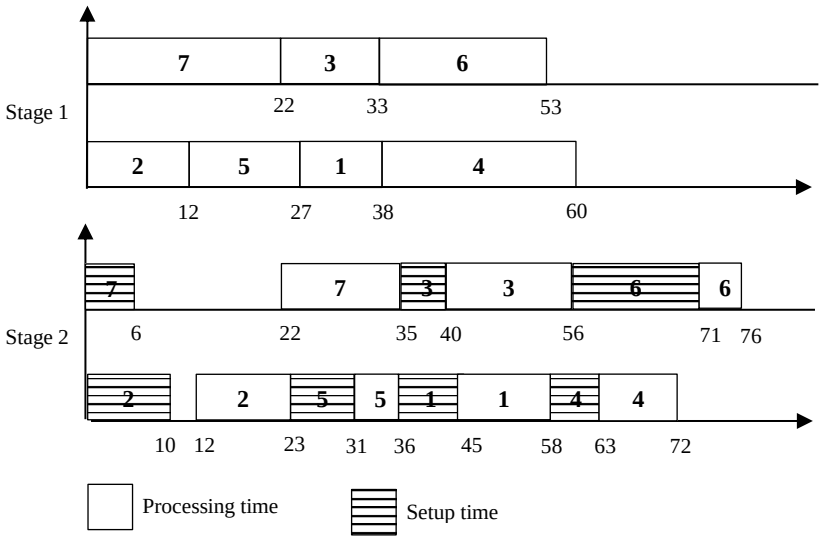

Figure 1: Two-stage schedule for the Example 1.

3. PROPOSED LOWER BOUNDS

Three new lower bounds are proposed in this section. These lower bounds are based on the capacity relaxation of the stages. In other terms, these lower bounds are obtained when the number of processors is supposed to be infinite in a stage.

3.1 Second stage relaxation capacity based lower bound (*SSR*)

If the number of processors in the second stage is relaxed to be infinite, then an arriving task to this stage is not waiting. The duration of the stay of this task is exactly $s_{2,j} + p_{2,j}$. The relaxed problem in the stage 1 is a parallel machine scheduling problem $P_{m_1}|r_j, q_j|C_{max}$ with release time r_j and delivery time q_j, where: $r_j = s_{1,j}$ for all $J_j \in J$ $(1 \leq j \leq n)$, $p_j = p_{1,j}$ for all $J_j \in J$ $(1 \leq j \leq n)$, $q_j = p_{2,j}$ for all $J_j \in J$ $(1 \leq j \leq n)$, and $m = m_1$.

Corollary: If LB is a lower bound for $P_{m_1}|r_j, q_j|C_{max}$ then it is a valid lower bound for $F(P_{m_1}, P_{m_2})m|s_{1,j}, s_{2,j}|C_{max}$.

Proof: The problem $P_m|r_j, q_j|C_{max}$ is a relaxation of the problem $F(P_{m_1}, P_{m_2})m|s_{1,j}, s_{2,j}|C_{max}$.

Lemma: If $\bar{r}_j$ and $\bar{q}_j$ denote the j^{th} r_j and j^{th} q_j sorted in the non-decreasing order then $SSR = \left\lceil \frac{\sum_{j=1}^{m_1} \bar{r}_j + \sum_{j=1}^{n} p_j + \sum_{j=1}^{m_1} \bar{q}_j}{m_1} \right\rceil$ is a valid lower bound for the problem $(P_{m_1}, P_{m_2})m|s_{1,j}, s_{2,j}|C_{max}$.

Proof: SSR is a lower bound for $P_{m_1}|r_j, q_j|C_{max}$. Based on the latter corollary, SSR is lower bound for $F(P_{m_1}, P_{m_2})m|s_{1,j}, s_{2,j}|C_{max}$.

3.2 First stage relaxation capacity based lower bound (*FSR*)

If the number of processors in the first stage is relaxed to be infinite, then an arriving task to this stage is not waiting. The duration of the stay of this task is exactly $s_{1,j} + p_{1,j}$. The obtained relaxed problem in the second stage is a parallel machine scheduling problem $P_{m_2}|r_j, q_j|C_{max}$ with release date r_j and delivery time q_j, where: $r_j = \max(s_{1,j} + p_{1,j}, s_{2,j})$ for all $J_j \in J$ $(1 \leq j \leq n)$, $p_j = p_{2,j}$ for all $J_j \in J$ $(1 \leq j \leq n)$, $q_j = 0$ for all $J_j \in J$ $(1 \leq j \leq n)$ and $m = m_2$.

The same reasoning as for *SSR*, allows to obtain a new lower bound *FSR* = $\left\lceil \frac{\sum_{j=1}^{m_2} \bar{r}_j + \sum_{j=1}^{n} p_j + \sum_{j=1}^{m_2} \bar{q}_j}{m_2} \right\rceil$ where C_{max}^2 is the optimal solution of the previous problem $P_{m_2}|r_j, q_j|C_{max}$.

3.3 Relaxation on First stage using parallel machine (*RFP*)

In this lower bound we relax the capacity of the first stage and m_1 is supposed to be infinite. In this assumption, for any task a machine is setup at zero time and executed it without delay in stage 1. Thus, the resulted scheduling problem in stage 2 is a parallel machine problem with release date r_j and delivery time q_j denoted $P_m|r_j, q_j|C_{max}$ where: $r_j = \max(s_{1,j} + p_{1,j}, s_{2,j})$ for all $J_j \in J$ $(1 \leq j \leq n)$, $p_j = p_{2,j}$ for all $J_j \in J$ $(1 \leq j \leq n)$, $q_j = 0$ for all $J_j \in J$ $(1 \leq j \leq n)$, and $m = m_2$. The Branch & Bound (B&B) exact solution for $P_m|r_j, q_j|C_{max}$ is utilized to obtain the optimal solution, and then the third lower bound denoted *RFP*.

Remark: To solve optimally the NP-Hard problem $P_m|r_j, q_j|C_{max}$ an exact procedure is used. Clearly, its optimal value is a lower bound for the studied problem as it is stated in the previous lemma. It might occur that the optimal solution of the NP-Hard problem $P_m|r_j, q_j|C_{max}$ is not obtained within a prefixed time.

Example 2: The same data as in Example 1 are used. The resulted data for the problem $P_{m_2}|r_j, q_j|C_{max}$ as defined above are presented in Table II.

Table II: Release date, delivery time and processing time values for *RFP*.

j	1	2	3	4	5	6	7
r_j	11	12	11	22	15	20	22
p_j	13	11	16	9	5	5	13
q_j	0	0	0	0	0	0	0

The optimal schedule of $P_{m_1}|r_j, q_j|C_{max}$ in the first stage is given as follows. The tasks $\{1, 7, 5, 6\}$ are assigned to processor one, and tasks $\{3, 2, 4\}$ are assigned to processor two (see Fig. 2). The optimal makespan is 47, and then *RFP* = 47.

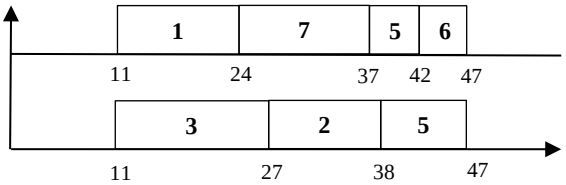

Figure 2: Optimal schedule for $P_{m_1}|r_j, q_j|C_{max}$ applying *RFP*.

4. APPROXIMATION SOLUTIONS

Since the studied problem is NP-Hard in the strong sense, finding a near-optimal solution within a satisfactory computation time is the primary goal. A meta-heuristic and three heuristics are proposed in this section.

4.1 Genetic algorithm (GA)

The genetic algorithms (GAs) is first proposed by Holland in [22]. The GA is based on reproducing the natural evolution. Indeed, the chromosomes are the main component in this

process. These chromosomes are the location where all the transformations, mutations are performed. In this study the adopted GA has the following standards elements:

1. Chromosome representation: A feasible schedule is formulated over a permutation of jobs. The decoding is conducted by scheduling the first unscheduled task in the permutation in the most available process in the first stage S_1. The selected process is then prepared during a setup time and the task is then processed. The tasks are sorted according to the increasing order of their finishing processing in the first stage. In the second stage, the tasks are scheduled according to the order of their finishing processing in the first stage.

2. Initial population: At the beginning of the GA an initial population is randomly generated. In addition, one particular solution is generated with respect to the Shortest Processing Time (SPT) in the first stage. This population is the starting population on which all the operations are carried out.

3. Selection operator: The selection step concerns the setting of a rule that allows to pick the most valuable solutions. This selection permits to accelerating the convergence toward an optimal solution.

4. Reproduction: The next generation is produced over the reproduction phase. This phase is intended to preserve the highest scored individuals to be apart from the new generation.

5. Crossover operator: This step is oriented to enrich the existing population. This is performed by choosing a percentage rate p_c ($p_c \in [0,1]$) of the existing individuals to be mutated. In this study, the position-based crossover operator (POX) is selected to perform the mutation.

6. Mutation: Each element (gene) of a chromosome (individual) has a probability p_m to be modified (mutated).

7. Replacement strategies: During this phase, some individuals are eliminated and they are no longer a part of the existing population. These individuals are replaced by others according to an established rule. The replacement approaches might be deterministic or stochastic. In practice, the population is fixed to 150. The pair (crossover rate, probability of mutating) selected in the GA are fixed as following: (0.3, 0.3), (0.3, 0.6), (0.3, 0.9), (0.6, 0.3), (0.6, 0.6), (0.6, 0.9), (0.9, 0.3), (0.9, 0.6), and (0.9, 0.9).

4.2 Optimal procedure heuristic (OP)

This heuristic is based on solving exactly a sequence of two parallel machines with head and tails. First, a parallel machine scheduling problem $P_m|r_j, q_j|C_{max}$ is solved in stage 1. The parameters are: $r_j = 0$ for all $J_j \in J$ $(1 \le j \le n)$, $p_j = s_{1,j} + p_{1,j}$ for all $J_j \in J$ $(1 \le j \le n)$, $q_j = p_{2,j}$ for all $J_j \in J$ $(1 \le j \le n)$, and $m = m_1$.

Once the above scheduling problem ($P_m|r_j, q_j|C_{max}$) is solved and an optimal solution is obtained, the completion time of a job $J_j \in J$ is denoted $c_{1,j}$. Second, in the second stage, a parallel machine scheduling problem $P_m|r_j, q_j|C_{max}$ is optimally solved. The characteristics of the latter problem are: $r_j = \max(c_{1,j}, s_{2,j})$ for all $J_j \in J$ $(1 \le j \le n)$, $p_j = p_{2,j}$ for all $J_j \in J$ $(1 \le j \le n)$, $q_j = 0$ for all $J_j \in J$ $(1 \le j \le n)$, and $m = m_2$.

In the third step, we observe that the returned solution is not sure to be a feasible one since it may occur that a job has no setup time in the stage 2. This problem of infeasibility can be corrected by a given adjusted solution. The adjustment is as follows. If a job J_j is the first scheduled job in a machine of stage 2, imply two cases must be tested:

1) if $s_{2,j} \ge c_{1,j}$ imply $c_{2,j} = s_{2,j} + p_{2,j}$.
2) if $s_{2,j} < c_{1,j}$ imply $c_{2,j} = c_{1,j} + p_{2,j}$.

If a job J_j is preceded by a job J_i on the same machine imply:

1) if $c_{2,i} + s_{2,j} \ge c_{1,j}$ imply $c_{2,j} = c_{2,i} + s_{2,j} + p_{2,j}$.

2) if $c_{2,i} + s_{2,j} < c_{1,j}$ imply is $c_{2,j} = c_{1,j} + p_{2,j}$.

The concatenation of the two feasible solutions in two stages gives a feasible solution for the presented problem.

Example 3: For this example, we use the same instance displayed in Table I. The schedule of this instance allying the optimal procedure heuristic is given in Fig. 3. This latter figure shows that $C_{max} = 78$.

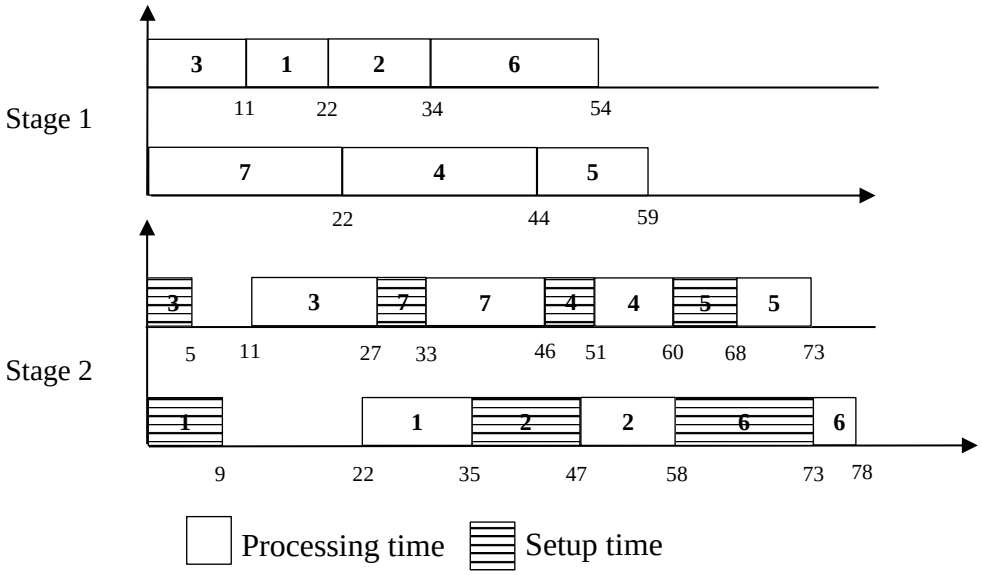

Figure 3: Optimal procedure heuristic schedule for the Example 3.

4.3 Optimal procedure with largest-first stage heuristic (*OLF*)

Firstly, we select the job that having the largest processing time on the first stage $p_{1,j}$. We preserve the setup times and the processing times on the two stages on a list. After that, we call the OP heuristic on the $n-1$ remaining jobs. Finally, we schedule the preserved job after the last scheduled jobs on stage one and stage two. The obtained result will constitute the *OLF* value.

4.4 Optimal procedure with smallest-first stage heuristic (*OSF*)

Firstly, we select the job that having the smallest processing time on the first stage $p_{1,j}$. We preserve the setup times and the processing times on the two stages on a list. After that, we call the OP heuristic on the $n-1$ remaining jobs. Finally, we schedule the preserved job after the last scheduled jobs on stage one and stage two. The obtained result will constitute the *OSF* value.

4.5 Optimal procedure with largest-second stage heuristic (*OLS*)

Firstly, we search the job that having the largest processing time on the second stage $p_{2,j}$. We preserve the setup times and the processing times on the two stages on a list. After that, we call the OP heuristic on the $n-1$ remaining jobs. Finally, we schedule the preserved job after the last scheduled jobs on stage one and stage two. The obtained result will constitute the *OLS* value.

4.6 Optimal procedure with smallest-second stage heuristic (*OSS*)

Firstly, we search the job that having the smallest processing time on the second stage $p_{2,j}$. We preserve the setup times and the processing times on the two stages on a list. After that, we call the OP heuristic on the $n-1$ remaining jobs. Finally, we schedule the preserved job after the last scheduled jobs on stage one and stage two. The obtained result will constitute the *OSS* value.

5. COMPUTATIONAL RESULTS

This section focuses on the analyses of the results obtained after implementation. All proposed lower bounds and upper bounds for the presented problem were coded in C++. The workstation that run all developed programs in C++ is an Intel(R) Xeon(R) CPU E5-2687W v4 @ 3.00 GHz and 64 GB RAM.

5.1 Test problems

To assess the performance of the proposed lower bounds and the upper bounds of the studied problem, four classes were developed. These classes were based on the uniform distribution denoted by $U(.)$. The classes depend on the manner that the setup times and processing times were generated. These classes are given as following:

- Class 1: $s_{1,j}$ in $U(1,30)$, $p_{1,j}$ in $U(1,50)$, $s_{2,j}$ in $U(1,30)$, $p_{2,j}$ in $U(1,50)$.
- Class 2: $s_{1,j}$ in $U(1,50)$, $p_{1,j}$ in $U(1,50)$, $s_{2,j}$ in $U(1,50)$, $p_{2,j}$ in $U(1,50)$.
- Class 3: $s_{1,j}$ in $U(1,30)$, $p_{1,j}$ in $U(1,30)$, $s_{2,j}$ in $U(1,30)$, $p_{2,j}$ in $U(1,30)$.
- Class 4: $s_{1,j}$ in $U(1,50)$, $p_{1,j}$ in $U(1,30)$, $s_{2,j}$ in $U(1,50)$, $p_{2,j}$ in $U(1,30)$.

In each stage, the number of parallel machines were selected in {2, 4, 6, 8}. There are 16 combinations of pair (m_1, m_2).

The number of tasks $n \in \{10, 30, 60, 100, 130, 150\}$. For each type of class each (m_1, m_2) and each n value, five instances were generated. The total number of tested instances is $16 \times 4 \times 6 \times 5 = 1920$.

5.2 Performance measures

To evaluate the performance of the proposed upper bounds and lower bounds, the following performance measures are applied:

- $G = \frac{U^* - L^*}{L^*}$, where U^* the minimum value obtained by all upper bounds and L^* the maximum value returned by all lower bound.
- $G_L = \frac{L^* - L}{L^*}$, where L is the presented lower bound value.
- $G_U = \frac{U - U^*}{U^*}$, where U is the presented upper bound value.
- $Perc$: Percentage of instances when $U = U^*$ ($L = L^*$) and $Time$: average CPU time in TU (time unit).

5.3 Results and discussion

Table III illustrates the behaviour of the *G* according to *n*. As illustrated in this table when the number of tasks increase, the *G* value increase. The minimum *G* of 0.419 is obtained when *n* = 10.

Table III: *G* according to *n*.

n	***G***
10	0.419
30	0.888
60	0.985
100	1.017
130	1.040
150	1.037

Table IV shows that the minimum *G* value of 0.823 is reached when $m_1 = 8$. The minimum G value of 0.867 is reached when $m_2 = 8$. We observe that the *G* value decreases when m_1 increases. This phenomenon is not applicable for m_1.

Table IV: *G* according to m_1 and m_2.

m_1	*G*	m_2	*G*
2	0.991	2	0.957
4	0.909	4	0.896
6	0.869	6	0.867
8	0.823	8	0.871

Table V presents the Gap values according to $Class$. The minimum gap value of 0.513 is obtained when $Class = 1$. However, the maximum gap value of 1.368 is obtained when $Class = 4$.

Table V: *G* value according to $Class$.

Class	*G*
1	0.513
2	0.853
3	0.856
4	1.368

Now, we display the experimental results regarding the lower bounds results. Table VI shows that *RFP* reached almost cases the maximum value. Indeed, the percentage of instances that *RFP* reached the minimum value is 60.5 % in an average time of 1.922 TU and an average gap of 0.149. This is shows that the best proposed lower bound is *RFP*. The second best lower bound is *SSR* with 40 %.

Table VI: Overall lower bounds comparison.

	SSR	*FSR*	*RFP*
G_L	0.221	0.185	0.149
$Time$	0.010	0.002	1.922
$Perc$	40.0 %	12.8 %	60.5 %

Table VII represents the variation of G_L according to *n*. For *RFP* the minimum G_L value of 0.034 is reached when *n* = 10 and the maximum G_L value is obtained when *n* = {130, 150}. The latter table shows that for *RFP* the G_L value increases when *n* increase.

Table VII: G_L and $Time$ according to *n.*

n	*SSR*		*FSR*		*RFP*	
	G_L	$Time$	G_L	$Time$	G_L	$Time$
10	0.287	-	0.222	-	0.034	0.022
30	0.221	-	0.164	-	0.147	7.654
60	0.215	0.057	0.168	0.009	0.163	2.983
100	0.202	-	0.183	-	0.181	0.217
130	0.200	-	0.186	-	0.184	0.394
150	0.200	-	0.186	-	0.184	0.264

Table VIII displayed the G_L and $Time$ behaviour according to m_1 and m_2 for all lower bounds. The lower bound that consumed more time compared with others is *RFP*. Indeed, when $m_2 = 8$, the average running time reached 5.295 TU.

Table VIII: G_L and $Time$ according to m_1 and m_2 for all lower bounds.

m_1	SSR		FSR		RFP	
	G_L	$Time$	G_L	$Time$	G_L	$Time$
2	0.019	0.038	0.432	0.006	0.402	2.344
4	0.167	-	0.190	-	0.153	1.377
6	0.292	-	0.077	-	0.039	2.122
8	0.405	-	0.041	-	0.001	1.848

m_2	SSR		FSR		RFP	
	G_L	$Time$	G_L	$Time$	G_L	$Time$
2	0.474	-	0.012	-	0.007	0.132
4	0.232	-	0.137	-	0.108	0.509
6	0.115	0.038	0.245	0.006	0.198	1.755
8	0.062	-	0.346	-	0.283	5.295

The variation of Gap_L and $Time$ according to $Class$ for all lower bounds is illustrated in Table IX.

Table IX: G_L and $Time$ according to $Class$ for all lower bounds.

$Class$	SSR		FSR		RFP	
	G_L	$Time$	G_L	$Time$	G_L	$Time$
1	0.213	0.000	0.189	0.000	0.157	3.530
2	0.224	0.038	0.182	0.006	0.146	2.696
3	0.218	0.000	0.188	0.000	0.153	0.752
4	0.229	0.000	0.181	0.000	0.140	0.712

The assessment of the proposed heuristics is presented in tables X and XI. Table X shows that the best heuristic is GA with 70.8 % compared with all other heuristics that the percentage of everyone does not exceed 13 %. The GA heuristic reaches the result in an average time of 4.666 TU and an average gap of 0.005. We observe that the other heuristics are characterized by their time-consuming. Indeed, the minimum average time for heuristics excluding GA is 40.206 TU.

Table X: Overall results for all heuristics.

	GA	OP	OLF	OSF	OLS	OSS
G_U	0.005	0.064	0.118	0.084	0.124	0.083
$Time$	4.666	42.760	40.206	42.259	40.311	42.092
$Perc$	70.8 %	12.4 %	11.5 %	4.9 %	3.2 %	8.8 %

Table XI shows that for GA heuristic the minimum G_U value of 0.003 is reached two times when n = {130, 150}. It is important to note that the G_U value increases when n increase for all heuristics excluding OP.

Table XI: G_U and $Time$ variation according to n for all heuristics.

n	GA		OP		OLF		OSF		OLS		OSS	
	G_U	$Time$	G_U	$Time$	G_U	$Time$	G_U	$Time$	G_U	$Time$	G_U	$Time$
10	0.006	0.061	0.096	0.853	0.301	0.195	0.190	0.215	0.327	0.229	0.173	0.246
30	0.009	0.389	0.098	54.541	0.166	49.508	0.122	54.392	0.171	51.927	0.120	54.090
60	0.006	1.660	0.066	51.043	0.095	50.240	0.070	48.487	0.096	49.168	0.074	48.113
100	0.004	4.715	0.048	49.883	0.059	45.855	0.051	51.791	0.058	46.980	0.051	48.463
130	0.003	8.742	0.040	49.937	0.046	49.158	0.040	48.374	0.048	50.329	0.041	52.137
150	0.003	12.428	0.037	50.302	0.040	46.282	0.036	50.297	0.043	43.234	0.039	49.504

The minimum average time of 0.061 TU is obtained by GA heuristic when $n = 10$. However, the maximum average time of 54.541 TU is obtained by OP heuristic when $n = 30$.

6. CONCLUSION

In this paper, the two-stage hybrid flowshop scheduling problem with independent setup times is addressed. A family of three valid lower bounds is proposed. These lower bounds are based on the capacity relaxation of one of the stages. Then a parallel machine scheduling problem with release date and delivery times is obtained. A lower bound for the latter problem is indeed a lower bound for the studied one. In addition, several heuristics and a genetic algorithm are developed. The heuristics are composed of two families, where the first one is based on dispatching rules as LPT. The main component of the second family of heuristics is the optimal solution of the parallel machine scheduling problem with release date and delivery time. A genetic algorithm is also proposed. An exhaustive experimental study is carried out to assess the performance of the proposed procedures. Two metrics are adopted for the evaluation, the relative gap and the computational time. The numerical results show the efficiency of the proposed algorithms. The genetic algorithm is outperforming all the other heuristics in terms of relative gap and computational time. Proposing an efficient exact procedure is future research. The proposed procedures in this study could provide a solid background for these for the exact solutions. Furthermore, new meta-heuristics might be adapted for the current studied problem.

ACKNOWLEDGEMENT

The authors extend their appreciation to the Deanship of Scientific Research at Majmaah University for funding this work under project number R-2021-283.

REFERENCES

[1] Alcaraz, J.; Maroto, C. (2001). A robust genetic algorithm for resource allocation in project scheduling, *Annals of Operations Research*, Vol. 102, No. 1-4, 83-109, doi:10.1023/A:1010949931021

[2] Shi, L.; Guo, G.; Song, X. (2021). Multi-agent based dynamic scheduling optimisation of the sustainable hybrid flow shop in a ubiquitous environment, *International Journal of Production Research*, Vol. 59, No. 2, 576-597, doi:10.1080/00207543.2019.1699671

[3] Han, W.; Deng, Q.; Gong, G.; Zhang, L.; Luo, Q. (2021). Multi-objective evolutionary algorithms with heuristic decoding for hybrid flow shop scheduling problem with worker constraint, *Expert Systems with Applications*, Vol. 168, Paper 114282, 17 pages, doi:10.1016/j.eswa.2020.114282

[4] Wang, Y.; Wang, S.; Li, D.; Shen, C.; Yang, B. (2021). An improved multi-objective whale optimization algorithm for the hybrid flow shop scheduling problem considering device dynamic reconfiguration processes, *Expert Systems with Applications*, Vol. 174, Paper 114793, 14 pages, doi:10.1016/j.eswa.2021.114793

[5] Hidri, L.; Elkosantini, S.; Mabkhot, M. M. (2018). Exact and heuristic procedures for the two-center hybrid flow shop scheduling problem with transportation times, *IEEE Access*, Vol. 6, 21788-21801, doi:10.1109/ACCESS.2018.2826069

[6] Allaoui, H.; Artiba, A. (2006). Scheduling two stage hybrid flow shop with availability constraints, *Computers & Operations Research*, Vol. 33, No. 5, 1399-1419, doi:10.1016/j.cor.2004.09.034

[7] Wang, S.; Liu, M. (2013). A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem, *Computers & Operations Research*, Vol. 40, No. 4, 1064-1075, doi:10.1016/j.cor.2012.10.015

[8] Tan, Y.; Mönch, L.; Fowler, J. W. (2018). A hybrid scheduling approach for a two-stage flexible flow shop with batch processing machines, *Journal of Scheduling*, Vol. 21, No. 2, 209-226, doi:10.1007/s10951-017-0530-4

[9] Yang, J. (2011). Minimizing total completion time in two-stage hybrid flow shop with dedicated machines, *Computers & Operations Research*, Vol. 38, No. 7, 1045-1053, doi:10.1016/j.cor.2010.10.009

[10] Wang, S.; Wang, X.; Chu, F.; Yu, J. (2020). An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production, *International Journal of Production Research*, Vol. 58, No. 8, 2283-2314, doi:10.1080/00207543.2019.1624857

[11] Fan, B.; Yang, W.; Zhang, Z. (2019). Solving the two-stage hybrid flow shop scheduling problem based on mutant firefly algorithm, *Journal of Ambient Intelligence and Humanized Computing*, Vol. 10, No. 3, 979-990, doi:10.1007/s12652-018-0903-3

[12] Wang, S.; Wang, X.; Yu, L. (2020). Two-stage no-wait hybrid flow-shop scheduling with sequence-dependent setup times, *International Journal of Systems Science: Operations & Logistics*, Vol. 7, No. 3, 291-307, doi:10.1080/23302674.2019.1575997

[13] Li, Y.; Li, X.; Gao, L.; Zhang, B.; Pan, Q.-K.; Tasgetiren, M. F.; Meng, L. (2021). A discrete artificial bee colony algorithm for distributed hybrid flowshop scheduling problem with sequence-dependent setup times, *International Journal of Production Research*, Vol. 59, No. 13, 3880-3899, doi:10.1080/00207543.2020.1753897

[14] Mirsanei, H. S.; Zandieh, M.; Moayed, M. J.; Khabbazi, M. R. (2011). A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times, *Journal of Intelligent Manufacturing*, Vol. 22, No. 6, 965-978, doi:10.1007/s10845-009-0373-8

[15] Agrebi, I.; Jemmali, M.; Alquhayz, H.; Ladhari, T. (2021). Metaheuristic algorithms for the two-machine flowshop scheduling problem with release dates and blocking constraint, *Journal of the Chinese Institute of Engineers*, Vol. 44, No. 6, 573-582, doi:10.1080/02533839.2021.1933600

[16] Jemmali, M.; Agrebi, I.; Alquhayz, H.; Ladhari, T. (2021). Optimal algorithm for a two-machine flowshop scheduling problem with release dates and blocking constraints, *Journal of the Chinese Institute of Engineers*, Vol. 44, No. 5, 440-447, doi:10.1080/02533839.2021.1919560

[17] Ištoković, D.; Perinić, M.; Borić, A. (2021). Determining the minimum waiting times in a hybrid flow shop using simulation-optimization approach, *Technical Gazette*, Vol. 28, No. 2, 568-575, doi:10.17559/TV-20210216132702

[18] Xu, L. Z.; Xie, Q. S.; Yuan, Q. N.; Huang, H. S. (2019). An intelligent optimization algorithm for blocking flow-shop scheduling based on differential evolution, *International Journal of Simulation Modelling*, Vol. 18, No. 4, 678-688, doi:10.2507/IJSIMM18(4)CO16

[19] Uzun Araz, O.; Eski, O.; Araz, C. (2019). A reactive scheduling approach based on fuzzy inference for hybrid flowshop systems, *International Journal of Simulation Modelling*, Vol. 18, No. 1, 5-18, doi:10.2507/IJSIMM18(1)448

[20] Istokovic, D.; Perinic, M.; Vlatkovic, M.; Brezocnik, M. (2020). Minimizing total production cost in a hybrid flow shop: a simulation-optimization approach, *International Journal of Simulation Modelling*, Vol. 19, No. 4, 559-570, doi:10.2507/IJSIMM19-4-525

[21] Graham, R. L.; Lawler, E. L.; Lenstra, J. K.; Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, Vol. 5, 287-326, doi:10.1016/S0167-5060(08)70356-X

[22] Holland, J. H. (1984). Genetic algorithms and adaptation, Selfridge, O. G.; Rissland, E. L.; Arbib, M. A. (Eds.), *Adaptive Control of Ill-Defined Systems*, 317-333, Springer, Boston, doi:10.1007/978-1-4684-8941-5_21