Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Minimizing makespan in a three-stage hybrid flow shop with dedicated machines

Asma Ouled bedhief[a*] and Najoua Dridi[a]

[a]*Oasis, Ecole nationale d'ingénieurs de Tunis, Université de Tunis El Manar B.P. 37 Le Belvédère 1002 Tunis, Tunisia*

C H R O N I C L E

Article history:
Received September 10 2018
Received in Revised Format
October 24 2018
Accepted October 24 2018
Available online
October 24 2018

Keywords:
Flow shop
Dedicated machines
Polynomial cases
Heuristic
Local search approach

A B S T R A C T

In recent years, many studies on scheduling problems with dedicated machines have been carried out. But, few of them have considered the case of more than two stages. This paper aims at filling this gap by addressing the three-stage hybrid flow shop scheduling problem with two dedicated machines in stage 3. Each job must be processed, consecutively, on the single machines of stages 1 and 2, and depending on its type, it will be further processed on one of the two dedicated machines of stage 3. The objective is to find an optimal schedule that minimizes the maximum completion time (makespan). Since this problem is strongly NP-hard, we first provide some basic results including solutions for several variations of the problem. Then, for the general case we adapt a set of lower bounds from the literature and propose a heuristic approach that is based on the dynamic programming technique, which uses a local search procedure. Finally, various experimentations on several problems with different sizes are conducted and the computational results of the heuristic show that the mean percentage deviation value from the lower bound was lower than 0.8 percent for some instances with 40 to 200 jobs in size.

1. Introduction

The hybrid flow shop (HFS) scheduling problem consists of optimizing the processing of a set of n jobs in m stages of machines. At least, one stage has to contain more than one machine. The HFS problem is frequently encountered in practice, essentially in the industry process where many machines are available in each stage, as well as in the flexible manufacturing environment (Gupta et al., 1971, 1997; Gupta & Stafford Jr, 2006). The HFS scheduling problems are $\mathcal{NP}$-hard, with very few exceptions (Riane et al., 1998). Hence, it has been widely studied in the literature. In this paper, we tackle a special type of HFS scheduling problem that may arise in a large class of real manufacturing environments: the hybrid flow shop with dedicated machines. In various industrial applications, dedicated machines produce different variants of the same basic product (different product types). First, all the products require the same operations of the production process. Then, they are processed on dedicated machines specific to each product type. We take, for example, the pharmaceutical industry, where a drug may be in forms of tablets or powder sachets, (or even syrup and ampoules). These two variants of such a drug

* Corresponding author
E-mail: bedhief.asma@gmail.com (A. Ouled bedhief)

2019 Growing Science Ltd.
doi: 10.5267/j.ijiec.2018.10.001

have to initially go through the same fabrication stages. Then, they must be processed in two different production lines of dedicated machines. The first one is dedicated to sachets, whereas the second line is dedicated to tablets. Similar examples can be found in label sticker manufacturing (Lin & Liao, 2003), pottery production, furniture assembly (Cheng et al., 2009), mass customization (Cheng et al., 2009) and global manufacturing firms (Yang, 2011). Many real applications may exist. In general, the production process is composed of many stages of dedicated and non-dedicated machines.

In this work, we study a HFS scheduling problem with two dedicated machines: Two product types are thus processed. Scheduling in such an environment with two or more stages tends to be $\mathcal{NP}$-hard in the strong sense (Lin, 1999). The traditional regular flow shop is a particular case of the HFS problem with dedicated machines, where only one product type is treated. Moreover, with the presence of more than one dedicated machine in a same stage, this problem can be seen as a hybrid flow shop, in which the assignment problem of jobs to machines does not arise. Scheduling problem with dedicated machines can also refer to as a particular case of an assembly flow shop, since a job of type 1 can be processed on the dedicated machine of type 2 or vice versa, with a processing time equal to zero. Therefore, due to the special configuration of our problem, we examine, in addition to the HFS with dedicated machines, the existing studies accomplished on assembly and on flow shop problems,

The two-machine flow shop problem is probably the first one to be successfully resolved, since Johnson (1954) proposed his well-known $O(nlogn)$exact algorithm. After Johnson's paper, several researchers have dealt with the m-machine flow shop problem, with $m >= 3$, which is known to be $\mathcal{NP}$-hard (Garey & David, 1976). The review of literature reveals that for a flow shop with no more than three machines, we can consider only the permutation schedules (Conway et al., 1967). Unfortunately, for more than three machines, we cannot guarantee the optimality of permutation sequences. However, Conway et al. (1967) stated that for minimizing the makespan in m-machine flow shop, we can consider only permutation schedules for which the same job order is prescribed on the first two machines and the same job order is prescribed on the last two machines. To tackle such an $\mathcal{NP}$-hard, several researchers have suggested heuristic approaches. Many of these heuristics are based on the Johnson's algorithm (Campbell et al., 1970; Chen et al., 1996). Campbell et al. (1970) proposed a heuristic (CDS) which consists essentially of splitting m-stage problem into $m - 1$ fictitious two-machine flow shop problems and solved them by Johnson's algorithm. In each case, we end up with a schedule and the best one is selected. The CDS heuristic's complexity is $O(mnlogn)$. Another approach is to give an index to every job and then schedule the sequence by sorting the jobs according to their assigned index. This idea was first used by Palmer (1965). He developed a simple heuristic in which for every job a "slope index" is defined. Then, the jobs having the greatest slope are sequenced first and so on, which leads to a computational complexity of $O(nm + n\ logn)$. Nawaz et al. (1983) designed a heuristic method (NEH) to the m-machine flow shop problem. It gives priority to jobs with high total processing time. The heuristic builds the schedule in a constructive way. At each step, the job with high priority is added to the partial solution in a way that induces the best partial makespan. The NEH algorithm, of complexity $O(mn^2)$, is one of the most popular algorithms in flow shop studies due to its high performance for the makespan minimization (Taillard, 1990; Framinan, 2004). For more details on the m-machine flow shop problem, a literature review may be found in (Ruiz & Maroto, 2005; Gupta & Stafford, 2006; Yenisey & Yagmahan, 2014).

The hybrid flow shop (HFS) problem consists of assigning jobs to parallel machines in each stage and sequencing the jobs assigned to the same machine. Although there is a vast literature on HFS, the most studied problem is the two-stage configuration, whereas very few studies have focused on the k stages. The two-stage HFS problem is $\mathcal{NP}$-hard, even if there is only one machine in the first stage and two machines in the second stage (Gupta, 1988). An extensive review of the literature on hybrid flow shops can be found in (Riane et al., 1997; Linn & Zhan, 1999; Ruiz & Rodriguez, 2010).

The first study in assembly flow shop scheduling problem was performed by Lee et al. (1993). They considered the three-machine assembly flow shop denoted as 3MAF (3-Machine Assembly Flow shop) with the objective of minimizing the makespan. In their problem, each product is assembled from two types of parts. A first machine processes the first type, whereas the second type is processed on a second machine. Finally, the two parts are assembled into a product on the third machine. This problem has been proved to be $\mathcal{NP}$ -hard (Lee et al., 1993; Potts et al., 1995). Later, Haouari and Daouas (1999) defined the inverse of 3MAF as a two-stage scheduling problem with one machine in the first stage and two machines in the second one. This problem is referred to as the 3-machine dismantling problem (3MDF). The authors showed that 3MAF and 3MDF are equivalent problems: an optimal schedule for 3MAF can be reversed to obtain an optimal schedule for 3MDF, with the same makespan. Only few studies have considered the case of more than two stages, for more details, the reader is invited to consult the existing papers (e.g. Koulamas & Kyparisis, 2001; Yokoyama & Santos, 2005; Fattahi et al., 2014). For the HFS problem with dedicated machines, most literature is focused on the two-stage configurations. Some studies have proposed exact methods (Riane et al., 2002; Mosheiov & Sarig, 2010; Hadda et al., 2014). Mosheiov and Sarig (2010) proposed an integer programming model and a polynomial dynamic programming algorithm for the two-stage HFS with *m* dedicated machines and a common job's due date. A branch and bound method has also been proposed to minimize the makespan for the case of *m* dedicated machines in the second stage (Hadda et al., 2014). Riane et al. (2002) studied the HFS with two dedicated machines and developed a dynamic program. Other studies have proposed heuristic approaches (Oguz et al., 1997; Dridi et al., 2009; Wang & Liu, 2013). Oguz et al. (1997) proposed an efficient heuristic, based on the Johnson's algorithm, to minimize the makespan in a two-stage flow shop with two dedicated machines in the first stage. Yang (2015) studied the same configuration with the objective of minimizing the total completion time. The author established the complexity of the problem and presented an optimal solution for the case where the processing times on the single machine of stage 2 are identical. For the case of *m* dedicated machines in the first stage, a set of dominance rules are provided and several polynomial cases are identified (e.g. Yang, 2013; Hadda et al., 2015). Riane et al. (2002) considered the problem with two dedicated machines in the second stage, where the objective is to minimize the makespan. They proved its strong $\mathcal{NP}$-hardness and proposed several polynomial heuristics. Several researchers have also studied the case where the processing sequences of the two types of jobs are given and fixed: a polynomial-time dynamic programming algorithm (Huang & Lin, 2013) and a linear-time algorithm (Lin, 2015) was developed to minimize the makespan and the maximum lateness, respectively. Dealing with the makespan minimization for the case of *m* dedicated machines in the second stage, several heuristics were proposed (Dridi et al., 2009; Wang & Liu, 2013). For the same problem, Hadda et al. (2012) presented a new dominance rule. The efficiency of the rule is shown through an analysis of the dominating set cardinality. As well, Yang (2011) studied the same configuration with the objective of minimizing the total completion time and provided optimal solutions for many cases of the problem.

For the case of more than two stages, Riane et al. (1998) studied the minimization of makespan in a three-stage HFS with two dedicated machines in the second stage. They developed two heuristic approaches. The first one is a dynamic programming based heuristic, whereas the second heuristic is based on a branch and bound algorithm. As for the assembly flow shop, most of the researches on HFS with dedicated machines focus on two-stage problems. To the best of our knowledge, there is only one study that considered three stages (Riane et al., 1988), in which the dedicated machines are in the second stage. In our opinion, the lack of studies considering three stages represents a step that can be taken in order to obtain a more thorough view on the problem. Furthermore, we note that many approaches, minimizing makespan in flow shops, are used to construct new solution methods for general scheduling problems, such as HFS, assembly flow shop and HFS with dedicated machines. NEH and Johnson algorithms, for example, are widely used in elaborated approaches for solving different configurations.

In this paper, we consider a three-stage HFS with two dedicated machines.We focus on the case where there exists only one machine in each of the first and second stages and two machines in the third stage

(Fig. 1). The two machines in stage 3 are dedicated (which means that the machine on which the job is processed is known in advance). Thus, the set of jobs is partitioned into two job families according to the dedicated machines. All jobs have to be processed on M_1 and M_2 consecutively, and then on the dedicated machine $D_k, k = 1,2$. The objective is to minimize the maximum completion time, $i.e$ makespan (C_{max}).

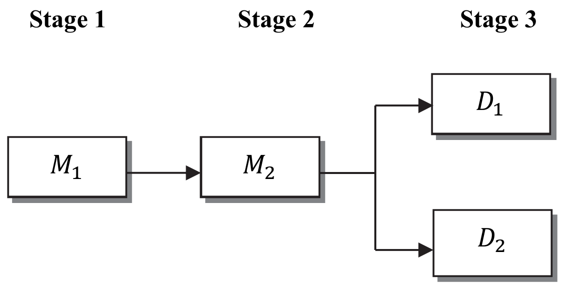


Fig. 1. Schematic of the production process

Following the notation of $\alpha|\beta|\gamma$ offered by Graham et al. (1979), we denote the three-stage HFS scheduling problem with two dedicated machines as $3FHD|1,1,2\,|C_{max}$. For notational convenience, let our problem be referred to as $\mathcal{P}$roblem $\mathcal{P}$. To the best of our knowledge, there is currently no report on heuristic methods for this special kind of problem. In this context, we introduce some basic results including solutions for several variations of the problem. We also adapt a set of lower bounds from the literature and we propose a heuristic method based on the dynamic programming procedure to solve the problem $\mathcal{P}$.

The rest of the paper is organized as follows. Section 2 introduces the notations and assumptions. In Sections 3 and 4, we develop some preliminary results and several lower bounds for the problem, respectively. In Section 5, we establish the complexity of the problem in the case where one of the machines requires a job-independent processing time: all processing times of jobs are identical on one of the machines. A description of the heuristic method is given in Section 6, while Section 7 reports the results of computational experiments to evaluate its performance. Finally, we provide a summary at the end of this paper.

2. Notations and assumptions

Throughout the paper, we will use the following notations:

M_j :	The single machine in each of the first and second stage, $j = 1,2$.
D_k :	The dedicated machine of type k in the third stage, $k = 1,2$.
n :	Number of jobs.
J :	Set of n jobs.
n_k :	Number of jobs of type k, ,$k = 1,2$.
J_k :	Subset of jobs of type k,$k = 1,2$, such that $\cup_{k=1,2} J_k = J$ and $\cap_{k=1,2} J_k = \emptyset$
p_{ij} :	Processing time of job i on M_j ,$j = 1,2$.
dp_{ik} :	Processing time of job i on D_k , $k = 1,2$.
$\pi = (\pi_1, \pi_2, \dots, \pi_n)$:	Permutation schedule where π_i is the i^{th} job.
$C_{ij}(\pi)$:	Completion time of job i on M_j in schedule π for $i \in J$and $j = 1,2$
$dC_{ik}(\pi)$:	Completion time of job i on D_k in schedule π for $i \in J_k \forall\ k = 1,2$
$C_{max}^k(\pi)$:	Completion time of the last job in schedule π on D_k, $k = 1,2$
$C_{max}(\pi)$:	Makespan of schedule π.
C_{max}^* :	Makespan of optimal schedule.

We note that all machines and jobs are available at time zero. Also, no preemption is allowed. A job cannot be processed by more than one machine at the same time and each machine processes, at most, one job at a time. The aim is to find a feasible schedule that minimizes the makespan. Note that for a given sequence π, the makespan is $C_{max}(\pi) = \max_{k=1,2}(C^k_{max}(\pi))$.

3. Preliminary results

In this section, we identify some of the optimal schedule's properties. The property below states that we can limit ourselves to the permutation schedules.

Property 1: For $\mathcal{P}$, there exists an optimal permutation schedule.
Proof: A permutation schedule is dominant on the first two machines M_1 and M_2 (Conway et al., 1967). Suppose that there exists an optimal schedule π in which the processing order on D_k , $k = 1$ or 2, differs from the order on M_2 . Then, there exists two consecutive jobs i and j on D_k such that j is sequenced just before i on M_2 . By inserting the job j before i on D_k , we can check that the solution π' obtained from π is no worse than π ($i.e$ $C_{max}(\pi') \leq C_{max}(\pi)$).

As a result of Property 1, we restrict our search for an optimal solution to permutation schedules. Also, a schedule can be fully described by the job order. We note that this result is no longer valid when there are more than three stages.

The spirit of Property 2 is based on a generalization of the idea used by Haouari and Daouas (1993) about the equivalence between the problem studied and its inverse. We define the inverse of $\mathcal{P}$ as a three-stage scheduling problem with two dedicated machines in the first stage and one machine in each of the second and third stages. The job is processed first on the dedicated machine corresponding to its type. Then, it will be processed, consecutively, on the single machines in stages 2 and 3.

Property 2: An optimal schedule for $\mathcal{P}$ can be reversed to obtain an optimal schedule for its inverse, with the same makespan.
Proof: If π is an optimal schedule for $\mathcal{P}$, then we can perform a mirror transformation of this schedule with respect to vertical line $y = C_{max}(\pi)$, (fig. 2). So, we obtain a feasible schedule $\pi^{-1} = (\pi_n, \pi_{n-1}, \ldots \pi_1)$ for the inverse problem of $\mathcal{P}$ with the same makespan. Thus, $\mathcal{P}$ and its inverse are equivalent problems. This result is still valid for the case of more than three stages.

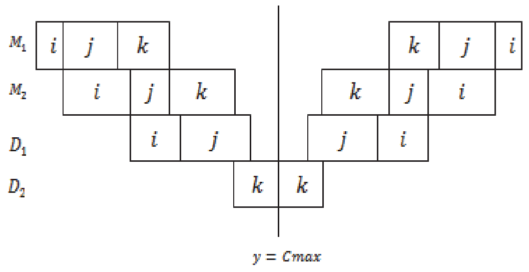


Fig.2. The Illustration of the proof for Property 2

4. Lower Bounds

In the following, we adapt four lower bounds from the literature to problem $\mathcal{P}$. These bounds will be used in the computational experiments. The first three lower bounds are derived from flow shop problems. The last one is a generalization of the lower bound previously proposed by Dridi et al. (2009) for the two-stage HFS with dedicated machines in the second stage.

The first lower bound $LB_1$is based on the observation that all jobs must be first processed in stage 1 and it is not possible to complete the last processed job in less than $\min_{\substack{i\in J_k \\ k=1,2}}(p_{i2} + dp_{ik})$.

$$LB_1 = \sum_{i\in J} p_{i1} + \min_{\substack{i\in J_k \\ k=1,2}}(p_{i2} + dp_{ik}). \tag{1}$$

For the second lower bound LB_2, no job can be started on M_2 in the second stage sooner than $\min_{i\in J} p_{i1}$ and it is not possible to complete the last processed job in less than $\min_{\substack{i\in J_k \\ k=1,2}} dp_{ik}$.

$$LB_2 = \min_{i\in J} p_{i1} + \sum_{i\in J} p_{i2} + \min_{\substack{i\in J_k \\ k=1,2}} dp_{ik}. \tag{2}$$

The third lower bound is derived from the flow shops "backwards". The jobs in subset $J_k, k = 1,2$ will be processed at a time at least equal to $\sum_{i\in J_k} dp_{ik}$ on D_k , $k = 1,2$, and it is not possible for these jobs to start on D_k in the third stage sooner than $\min_{i\in J_k}(p_{i1} + p_{i2})$. Therefore, the lower bound LB_3 may be described as follows:

$$LB_3 = \max_{k=1,2}[\min_{i\in J_k}(p_{i1} + p_{i2}) + \sum_{i\in J_k} dp_{ik}] . \tag{3}$$

$LB_4$is based on the observation that one of the dedicated machines $D_{k_0}, k_0 = 1 \text{or } 2$ cannot start its process until the second job has been completed on M_2. In other words, it is not possible to begin processing jobs on D_{k_0}sooner than the minimum completion date of the second job on M_2, which can be given as: $p_{j1} + p_{j2} + \min_{i\in J_{k_0}}(p_{i2} + max(0, p_{i1} - p_{j2}))$, where $j \in J_{k\neq k_0}$ and$p_{j1} + p_{j2} = \min_{i\in J_{k\neq k_0}} (p_{i1} + p_{i2})$. Thus, LB_4 may be described as follows:

$$LB_4 = \min_{k=1,2} \{ \min_{i\in J_{k_0\neq k}} (p_{i1} + p_{i2}) + \min_{i\in J_k}(p_{i2} + max(0, p_{i1} - p_{j2})) + \sum_{i\in J_k} dp_{ik}\} . \tag{4}$$

The desired lower bound LB is given by:

$$LB = max\ (LB_1, LB_2, LB_3, LB_4) . \tag{5}$$

We note here that the relative performance of such lower bounds depends on the workload of the different machines. If the workload of M_1(respectively M_2) is larger than that of the other machines then, LB_1(respectivelyLB_2) tends to perform best. For the case where one of the two dedicated machines is a bottleneck, it is more likely that LB is the third lower bound, LB_3. Finally, if the dedicated machines are the most-loaded, then LBmay correspond to LB_4.

5. Particular cases

In some cases, a machine may require a processing time that does not depend on the job, but rather on the nature of the operation to be performed, such as perforation, cutting operation or oven residence time. In such cases, the problem becomes much easier to study. Then, the question is: Is the problem still, in this case, $\mathcal{NP}$-hard or can we find an optimal schedule? In the following, we seek to answer this question in the cases where the machine M_j requires a job-independent processing time: $p_{ij} = p_j , j = 1,2$. Before discussing these cases, let us first introduce the following notations:

Let $F_3||C_{max}$be the three-machine flow shop problem and p_{ij} be the processing time of job i on machine $M_j, j = \{1,2,3\}$. If $J_1 = J$ or $J_2 = J$ then $\mathcal{P}$ is identical to problem $F_3||C_{max}$.
We further denote $\mathcal{P}'$ as the two-stage hybrid flow shop scheduling problem, which may be identified as $2FHD|1,2\ |C_{max}$ (Fig.3). There is one machine in the first stage and two dedicated machines in the second stage.

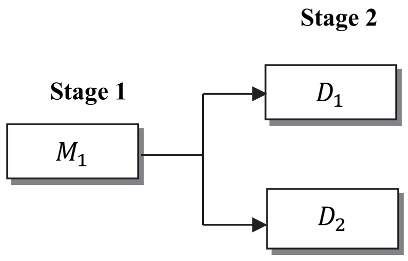


Fig.3. Schematic of the production process of $\mathcal{P}'$

Consider a given permutation schedule π , let $C'_{i1}(\pi)$ be the completion time of job i on M_1 in π for $\mathcal{P}'$ and $dC'_{ik}(\pi)$ its completion time on D_k , $k = 1,2$. When there exists no confusion, we replace $C_{ij}(\pi)$ and $dC_{ik}(\pi)$, respectively, with C_{ij} and dC_{ik} for $\mathcal{P}$; and we replace $C'_{i1}(\pi)$ and $dC'_{ik}(\pi)$ respectively with C'_{i1} and dC'_{ik} for $\mathcal{P}'$.

In Lemmas 1 and 2, we show that under certain conditions, the three-stage problem $\mathcal{P}$ is equivalent to $\mathcal{P}'$.

Lemma 1: For the particular case satisfying condition:

$$p_{i1} = p_1 \forall i \in J \text{ and } p_1 \leq \min_{i \in J} p_{i2} \qquad (C1)$$

$\mathcal{P}$ and $\mathcal{P}'$ are equivalent problems (with$\mathcal{P}'$being composed by M_2 in the first stage).
Proof: According to the hypothesis, M_2 processes all jobs without idle time since the date p_1. Hence, for a given permutation π we have: $C_{max}(\pi) = p_1 + C'_{max}(\pi)$, with $C'_{max}(\pi)$ being the value of the makespan for $\mathcal{P}'$. Therefore, an optimal schedule for the two-stage HFS with dedicated machines is also optimal for the problem $\mathcal{P}$ (and vice versa).

Corollary: For the case where $p_1 \leq \min_{i \in J} p_{i2}$, the problem $F_3||C_{max}$can be solved in polynomial time.
Proof: According to the hypothesis, $F_3||C_{max}$is equivalent to the $F_2||C_{max}$ problem composed by M_2 and M_3 (Particular case of Lemma 1). Since $F_2||C_{max}$ is successfully resolved by Johnson's algorithm, $F_3||C_{max}$ is solved in polynomial time.

Lemma 2: For the particular case satisfying condition:

$$p_{i2} = p_2 \forall i \in J \text{ and } p_2 \leq \min_{i \in J} p_{i1} \qquad (C2)$$

$\mathcal{P}$ and $\mathcal{P}'$ are equivalent problems (with $\mathcal{P}'$being composed by M_1 in the first stage).
Proof: Due to the hypothesis, we have:

$$C_{i2} = C_{i1} + p_2 \ \forall i \in J. \qquad (6)$$

For the first job processed on D_k , we have: $dC_{ik} = C_{i1} + p_2 + dp_{ik}$

$$dC_{ik} = dC'_{ik} + p_2 \quad \forall k = 1,2 \qquad (7)$$

Let π be a permutation schedule of *n* jobs processed on M_1. We have to prove, by induction, that (7) remains valid for all jobs$i \in J$.

Suppose Eq. (7) holds for all jobs from π_1 to π_k. Now, we have to check that it also holds for the $k+1^{th}$ job, π_{k+1}. Let $l = \pi_{k+1}$ be the $k+1^{th}$job and l' is the job of the same type that l, such that l' is sequenced just before $l$on D_k .

By induction of the hypothesis, we have:

$$dC_{l\,k} = max\{dC_{l'k}\ , C_{l\,2}\} + dp_{lk} = max\{dC'_{l'k}\ ,\ C_{l1}\ \} + p_2 + dp_{lk}.$$

As a consequence:

$$dC_{l\,k} = dC'_{l\,k} + p_2. \tag{8}$$

Hence, an optimal schedule for the two-stage HFS with dedicated machines is also optimal for the problem $\mathcal{P}$ (and vice versa).

The following theorems establish three polynomial cases for the problem $\mathcal{P}$.

Theorem 1: For $\mathcal{P}$, if $p_{i1} = p_1$ and $p_{i2} = p_2$ then there exists a polynomial time optimal solution.
Proof: Let us suppose that $p_{i1} = p_1$ and $p_{i2} = p_2 \forall i \in J$

We have two cases:

- (1) If $p_1 \leq p_2$ then, due to lemma 1, $\mathcal{P}$ is equivalent to $\mathcal{P}'$ such that M_2 is in the first stage, and D_k $(k = 1{,}2)$ are in the second stage.
- (2) If $p_1 \geq p_2$ then, due to lemma 2, $\mathcal{P}$ is equivalent to $\mathcal{P}'$ such that M_1 is in the first stage, and D_k $(k = 1{,}2)$ are in the second stage.

In both cases, all processing times on the single machine in the first stage of $\mathcal{P}'$ are identical. Due to the equivalence between $\mathcal{P}'$ and its inverse (Haouari & Daouas, 1999), $\mathcal{P}'$ can be solved in polynomial time (Yang, 2013). Therefore, an optimal permutation schedule can be obtained for problem $\mathcal{P}$ with constant processing times on the first and second stages.

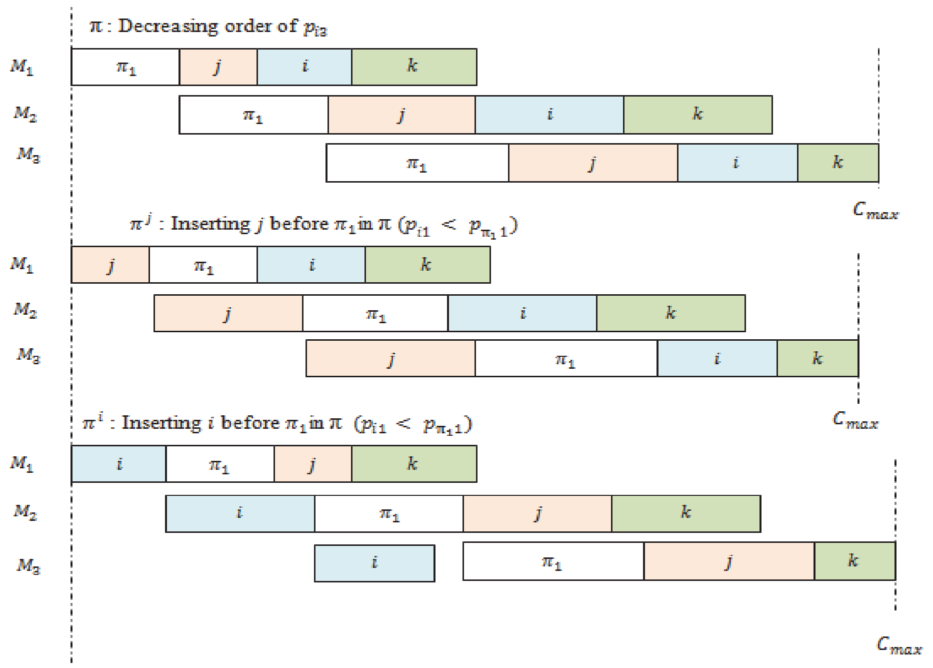
Next, we consider that only the processing times of jobs on M_2 are identical $i.e\ p_{i2} = p_2 \forall i \in J$.

For problem $F_3||C_{max}$, even though the processing times of all jobs on the second machine are identical ($p_{i2} = p_2$), it is still $\mathcal{NP}$-hard (Kise, 1991). Nevertheless, we show, in the following, that there exists a polynomial case for some values of p_2.

Lemma 3: If $p_2 \geq \max_{i \in J} p_{i1}$, then problem $F_3||C_{max}$can be solved in polynomial time.
Proof: In this case, the activity of the machine M_2 depends only on the processing time of the first job on M_1. Let $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ be the permutation obtained by the decreasing order of p_{i3}. The value of C_{max} is, then, given by:

$$C_{max}(\pi) = max\ \left(p_{\pi_1 1} + p_2 + \sum_{i=1}^{n} p_{i3}\,; p_{\pi_1 1} + np_2 + \min_i p_{i3}\ \right). \tag{9}$$

The only way of finding a better solution is the existence of a job i such that $p_{i1} < p_{\pi_1 1}$.
Let $A = \{i \in J /\ p_{i1} < p_{\pi_1 1}\ \}$ and let π^i be the permutation obtained, from π, by inserting the job i before π_1. The optimal solution is the one minimizing C_{max} among the permutations π and $\pi^i,\ i \in A$. In Fig. 4, we illustrate the case where $A = \{i, j\}$.



π : Decreasing order of p_{i3}
M_1
M_2
M_3
π_1
j
i
k
C_{max}
π^j : Inserting j before π_1 in π ($p_{j1} < p_{\pi_1 1}$)
π^i : Inserting i before π_1 in π ($p_{i1} < p_{\pi_1 1}$)

Fig.4. Illustration of the case where $A = \{i, j\}$

Theorem 2: For the case where $p_2 \geq \max_{i \in J} p_{i1}$, the problem of finding a permutation schedule for $\mathcal{P}$ is polynomially solvable.

Proof: In this case, the equivalent problem $\mathcal{P}'$ (with the machines M_2, D_1 and D_2) becomes polynomial (Yang, 2013). If π is an optimal schedule for $\mathcal{P}'$, then the makespan given by π for $\mathcal{P}$ is: $C_{max}(\pi) = p_{\pi_1 1} + C'_{max}(\pi)$, with $C'_{max}(\pi)$ being the value of the makespan for $\mathcal{P}'$. The only way to decrease the makespan is to insert another job in the first place. The rest of the proof is identical to that of **Lemma 3**.

Theorem 3: For $\mathcal{P}$, if the second machine dominates the other machines such that:

$$\max\ p_{i1} \leq \min_{i \in J} p_{i2} \text{ and } \max dp_{ik} \leq \min_{i \in J} p_{i2} \tag{10}$$

Then, $\mathcal{P}$ can be solved in polynomial time.

Proof: In such a case, for any permutation π, we have:

$$C_{max}(\pi) = p_{\pi_1 1} + \sum_{i=1}^{n} p_{i2} + dp_{lk} \text{ , with } l = \pi_n, l \in J_k, k \in \{1,2\}. \tag{11}$$

Let's suppose that: $\min_{i \in J} p_{i1} = p_{i_0 1}$ and $\min_{\substack{i \in J_k \\ k=1,2}} dp_{ik} = dp_{j_0 k}$.

If $i_0 \neq j_0$, then the optimal makespan is given by:

$$C^*_{max} = p_{i_0 1} + \sum_{i=1}^{n} p_{i2} + dp_{j_0 k} . \tag{12}$$

Otherwise ($i_0 = j_0$), let $i_1 \in J$: $p_{i_1 1} = \min_{i \neq i_0} p_{i1}$ and let $j_1 \in J$: $p_{j_1 1} = \min_{\substack{i \neq j_0 \\ k=1,2}} dp_{ik}$ $k = 1,2$

In this case, the optimal makespan is given by:

$$C^*_{max} = p_{i_0 1} + \sum_{i=1}^{n} p_{i2} + dp_{j_1 k} \tag{13}$$

Or

$$C^*_{max} = p_{i_1 1} + \sum_{i=1}^{n} p_{i2} + dp_{i_0 k} \ . \qquad (14)$$

6. Heuristic approach

In this section, we provide a heuristic method to find approximate solutions for the problem $\mathcal{P}$. This heuristic consists of three steps. The first one solves two separate flow shop problems according to the job either type J_1or J_2. The second step combines the two resulting subsequences, to produce a single ordered sequence of *n* jobs for the original problem. Finally, the third step improves the quality of the achieved solutions, using a 2-opt Exchange operator. In the following, we present in detail the three steps of this proposed heuristic.

First step: Solving two flow shop problems

It consists to consider the actual flow of jobs as two separate ones. The first flow involves the jobs of the subset J_1 and corresponds to the routing M_1- M_2- D_1. While, the second flow involves the jobs of the subset J_2 and corresponds to the routing M_1- M_2-D_2. The jobs of each flow can be scheduled using several algorithms which are proposed in the literature for flow shop problems. The main idea, in this paper, is to exploit the existing heuristics which proved their efficiency for the flow shop, in order to solve the HFS with dedicated machines. In this work, we use the NEH algorithm (Nawaz et al., 1983) to process jobs in the two subsequences denoted as σ_1 and σ_2 . This algorithm, running in $O(mn^2)$, has been commonly regarded as one of the best constructive methods for solving flow shop problems. It consists of two steps. First, the jobs are ordered in non-increasing order of their sum of processing times. Second, an iterative insertion of jobs into a partial sequence according to the job order of step 1, is used. The solution quality obtained by the NEH algorithm depends on the initial jobs order of step 1.Thus, we further propose to construct another initial order in our version of NEH, which is more appropriate to the problem we study. This order gives priority to the jobs that are expected to be at the end of the schedule. Thus, we define an index T_i to every job i. It is calculated as follows:

$$T_i = p_{i1} + p_{i2} - dp_{ik} \forall i \in J \ . \qquad (15)$$

To distinguish from the classical NEH algorithm, MNEH is used to denote the modified NEH algorithm. In such an algorithm, the jobs are firstly sorted in the non-increasing order of the index T_i. For notational convenience, we define H as the algorithm used to construct the two subsequences σ_1 and σ_2 , such as H = NEH or MNEH.

Second Step: Dynamic programming procedure

Once the two subsequences are determined, the jobs in σ_2 have to be interleaved with those of σ_1 to give a single schedule of *n* jobs denotes as σ. Many combination methods are proposed for this purpose in the literature on hybrid flow shops. For instance, Oguz et al. (1997) proposed to schedule the jobs in the non-decreasing order of their completion times on the dedicated machines for the two-stage HFS with two dedicated machines in the first stage. As well, Riane et al. (1998) studied the three-stage HFS with two dedicated machines in the second stage and developed a dynamic program to combine the resulting sequences. In this work, we use the $O(n^3)$ dynamic programming procedure of (Riane et al., 1998) to achieve the best combination of the two subsequences. Our heuristic can be referred to as H-DP (Dynamic Programming-based Heuristic). The idea of H-DP is to keep the H-order in both subsequences when the jobs in σ_2 are to be interleaved within the sequence σ_1 . Thereby, the jobs of σ_1 can be viewed as possessing"gaps" between them, in which the jobs of σ_2 must fit.

Let $S_q \subseteq \sigma_2$ be the subset of the q first jobs in σ_2 such as $S_q = \{1 \dots q\}$. We denote $f_g(S_q)$ as the minimal makespan where the subset S_q is located in the first g gaps of σ_1 . In such a dynamic program, the stages are the gaps in σ_1. The subset S_q represents the state. Let π_h denote the subset of the last h jobs in S_q placed in gap g, $0 \leq h \leq q$. The decision is, thus, the determination of the subset π_h. The dynamic program recursion equation is defined as follows (Riane et al., 1998):

$$f_g(S_q) = \min_{\pi_h \subseteq S_q} \{f_{g-1}(S_q - \pi_h) \circ M_g(\pi_h)\}, \quad \forall g = 1 \dots n_1 \text{and } q = 1 \dots n_2 \tag{16}$$

where $M_g(\pi_h)$ represents the makespan when $\pi_h \subseteq S_q$ is placed in gap g. The "$\circ$" indicates the composition of the makespan due to the assignment of the subset π_h in gap g with the optimum of the subset $S_q - \pi_h$.

In the experimental study, we compare the performance of NEH-DP and MNEH-DP with the heuristic of Riane et al. (1998) that we denote as CDS-DP. The objective of evaluating these three variants of H-DP is to prove the effect of the H-order on the performance of H-DP for the problem $\mathcal{P}$.

Third step: 2-Opt

For HFS scheduling problems, many neighborhood search approaches are used to improve the quality of solutions, and the performance is significant. In this paper, the idea is to perform the algorithm H-DP followed by a local search improvement procedure. Hence, a $O(n^2)$ 2-Opt algorithm is applied: It starts with an initial solution S, which is considered as a current solution at the first iteration. Then, we use a 2-opt exchange operator to generate a new solution S', from S. This one is compared with the current solution. If it is better, then it is accepted and is used as a current solution in the next iteration. Otherwise, the candidate solution is rejected and we carry out the next iteration with the same current solution. Furthermore, we use $n(n-1)/2$ as a maximum number of iterations. In the following, we present an illustrative flow diagram of the heuristic IH-DP.

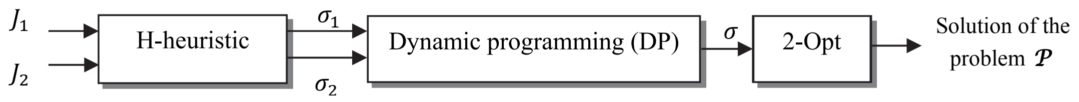


Fig. 6. Flow diagram of the proposed heuristic IH-DP

We note that the first step of the heuristic IH-DP (NEH or MNEH algorithm) is of complexity $O(mn^2)$, while the second one (Dynamic programming procedure) is a $O(n^3)$ algorithm. The third step (2-Opt algorithm) runs in $O(n^2)$ time. Hence, the heuristic approach IH-DP is of complexity $O(n^3)$.

7. Computational experiments

In this section, different computational experiments are conducted to evaluate the performance of the H-DP and IH-DP algorithms. The procedures are implemented in DEV C++ and the results are obtained in a personal computer with an Intel 2.60 GHz CPU and 1.96 GB RAM.

7.1. Data generation

We have conducted the computational experiments on four classes of test problems generated randomly. In each class, first and second-stage processing times are random integers from a uniform distribution from 1 to 20, denoted by $p_{i1} \sim U[1,20]$ and $p_{i2} \sim U[1,20]$. The ease of finding a satisfying solution for the studied problem depends on whether there is a balance between average workloads of dedicated machines and the total workload on the single machines. Hence, the processing times of jobs on the dedicated machines are generated randomly from the following uniform distributions:

CLASS 1: $dp_{ik} \sim U[1,20] \quad \forall k \in \{1,2\}$	**CLASS 3:** $dp_{ik} \sim U[20,40\ ] \ \forall k \in \{1,2\}$
CLASS 2: $dp_{ik} \sim U[1,60] \quad \forall k \in \{1,2\}$	**CLASS 4:** $dp_{ik} = p_{i2} + 5 \quad \forall k \in \{1,2\}$

Through these different classes, we seek to study the behavior of the heuristic on different types of data. In CLASS 1, the dedicated machines are the least loaded, while in CLASS 2, the dedicated machines are more loaded than M_1 and M_2 but with processing times generated from 1 to 60 units. In CLASS 3, the dedicated machines are also the most-loaded but the processing times are the most important and dominate those on M_1 and M_2. As for CLASS 4, we imposed that the increasing order of the processing times on M_2 coincides with the increasing order of the processing times on the dedicated machines. Also, each job has an integer parameter which defines its processing on the dedicated machine in stage 3. This parameter is chosen randomly between D_1 and D_2 , since there is no preference between the two dedicated machines. Several instances of various sizes are generated ($n = 40, 80, 120, 160, 200$) for each of the four classes. For each problem size, we tested 20 instances. For each instance, the quality of solution is measured by a percentage deviation PD, which is calculated for each of the three variants of H-DP (H= NEH, MNEH and CDS), as well as for the IH-DP algorithms. The final results are presented in terms of mean percentage deviation MPD.The MPD of an algorithm is defined as:

$$MPD = \sum_{i=1}^{20} \frac{PD(i)}{20}$$

where $PD(i) = \frac{[C_{max}(i) - LB(i)]}{LB(i)} \times 100.$

$C_{max}(i)$ represents the makespan value obtained by IH-DP (or H-DP) for problem instance i. As well, $LB(i)$ denotes the lower bound value of instance i.

7.2.Computational Results

The main objectives of the computational experiments are to compare the performance of CDS, NEH and MNEH used in H-DP, and to assess the improvement brought by the 2-Opt procedure in the solutions' quality. In our study, the best resulting sequence of the three variants of H-DP is considered as the initial solution of 2-Opt. The results are presented in Tables 3-7. Tables 3-6 show the mean percentage deviation for the three variants of H-DP and IH-DP algorithms, whereas Table 7 gives the percentage of instances for which IH-DP finds the optimal solution which is equal to LB out of 20 instances.

Table 3
Computational results for CLASS 1 problems

MPD: Mean percentage deviation				
CLASS 1	H-DP			IH-DP
	CDS	NEH	MNEH	
n = 40	1.00	0.31	**0.15**	**0.04**
n = 80	0.65	0.28	**0.15**	**0.01**
n = 120	0.31	0.12	**0.12**	**0.02**
n = 160	0.42	0.12	**0.07**	**0.02**
n = 200	0.42	0.03	**0.05**	**0.00**

Table 4
Computational results for CLASS 2 problems

MPD: Mean percentage deviation				
CLASS 2	H-DP			IH-DP
	CDS	NEH	MNEH	
n = 40	2.76	2.79	**2.17**	**0.08**
n = 80	2.65	3.25	**1.58**	**0.04**
n = 120	3.19	3.22	**3.04**	**0.02**
n = 160	3.39	4.55	**2.99**	**0.08**
n = 200	4.04	4.29	**3.09**	**0.05**

Table 5
Computational results for CLASS 3 problems

MPD: Mean percentage deviation				
CLASS 3	H-DP			IH-DP
	CDS	NEH	MNEH	
n = 40	3.55	**2.73**	2.91	0.17
n = 80	4.58	**3.64**	3.81	**0.11**
n = 120	5.03	3.88	**3.38**	**0.08**
n = 160	5.69	5.13	**3.13**	**0.07**
n = 200	6.17	4.03	**3.49**	**0.07**

Table 6
Computational results for CLASS 4 problems

MPD: Mean percentage deviation				
CLASS 4	H-DP			IH-DP
	CDS	NEH	MNEH	
n = 40	6.84	6.49	**2.47**	**0.80**
n = 80	5.64	4.11	**2.39**	**0.65**
n = 120	6.59	3.31	**2.38**	**0.37**
n = 160	6.82	3.50	**2.61**	0.60
n = 200	7.56	4.42	**2.25**	**0.32**

Table 7
Percentage of instances for which IH-DP finds the optimal solution which is equal to LB

$P_{C_{max}=BI}$(%)	*CLASS 1*	*CLASS 2*	*CLASS 3*	*CLASS 4*
n = 40	95	85	85	30
n = 80	95	85	75	35
n = 120	90	90	80	45
n = 160	90	65	75	45
n = 200	100	65	50	30

According to the results, the following observations about the problem $\mathcal{P}$ can be concluded:

Considering the results per problem class, we can see from Table 3 that for all the instances of CLASS 1, the MPD values of the three variants of H-DP are less than 1%. With the IH-DP, the MPD values do not even exceed 0.04%. CLASS 1 is, thus, relatively easy to solve. CLASS 2, CLASS 3 and CLASS 4, whose results are presented respectively in Tables 4, 5 and 6, are however more difficult as the MPD values are larger. We note that MPD values obtained for these classes are about 6% for CDS-DP, 4% for NEH-DP, 3% for MNEH-DP and 0.5% for IH-DP. Nevertheless, for most cases, the MPD values of CLASS 2 and CLASS 3 are smaller than those in CLASS 4. Comparing the results of three variants of H-DP, we notice that in almost all cases, NEH-DP and MNEH-DP perform much better than CDS-DP whose MPD values are larger. Although both NEH-DP and MNEH-DP are very effective to solve CLASS 1, MNEH-DP results were slightly better. Overall, for almost all the instances of CLASS 2 and CLASS 4, MNEH-DP performs better than NEH-DP, with a mean percentage deviation that does not exceed 4.47%, while it can reach 6.49% for NEH-DP. We can therefore conclude that MNEH-DP outperforms NEH-DP in finding near optimal solution. For CLASS 3, the superiority of MNEH-DP is however not so clear since the best results are being shared between NEH-DP and MNEH-DP. This can be explained by the fact that the processing times on the dedicated machines are very important in this class of problems, and the determination of the relative order of jobs with high priority can be done randomly.

Despite the great quality of these obtained solutions, we notice that IH-DP still brings a significant improvement for nearly all of them. In fact, for all cases, the mean percentage deviation (MPD) of IH-

DP solutions from the lower bound are less than 0.80%. This can also be observed from the percentage of instances for which IH-DP heuristic finds the optimal solution ($i.e.$ a solution equal to LB) as shown in Table 7. We can thus conclude that the addition of the 2-Opt moves is beneficial for the performance of IH-DP in terms of high-quality solutions. Further, the performed tests show that the quality of the initial solution has a significant impact on the outcomes of IH-DP, as the latter performs better than a 2-Opt algorithm starting from a randomly generated solution.

IH-DP heuristic is able to solve the generated medium-size instances (200 jobs) in a very reasonable CPU time ($i.e$ less than 2 minutes). This can be observed from the Fig. 5 below. Also, as the number of jobs n, increases, the average CPU time increases for almost all cases. This is an expected result due to the increasing search space of the 2-Opt procedure.

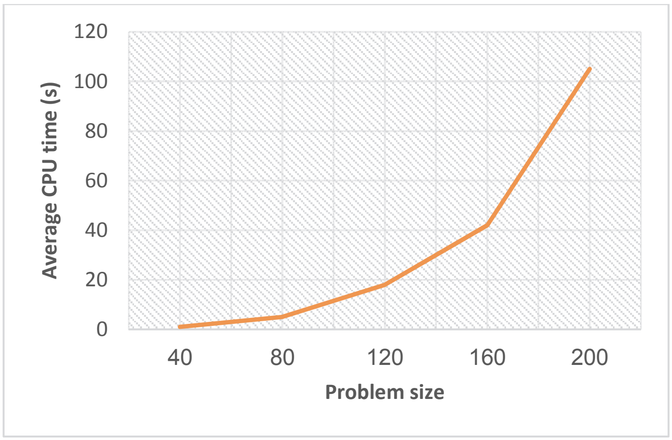


Fig. 5. Average CPU time of IH-DP according to the problem size

8. Conclusion

In this paper, we have dealt with the three-stage hybrid flow shop problem with two single machines in the first and the second stages, and two dedicated machines in stage three. Considering the $\mathcal{NP}$-hardness of the problem, several particular cases, which can be considered in many industrial activities, were studied. For some cases in which the processing times on M_j are identical, an optimal solution was proposed for the problem. As well, a heuristic method was proposed for the general case and its implementation was described. A local search approach was also included to improve the quality of the obtained solutions. Then, an extensive experimental study was conducted to evaluate the performance of the proposed method. The results emphasized its high performance and showed that the mean percentage deviation (MPD) of the reached solutions from the lower bound was less than 0.8% for all the instances. By adapting the lower bounds and restricting ourselves to permutation schedule, the proposed heuristic can still be used to solve the k-stage HFS with dedicated machines. We further note that all of the obtained results in this paper are still valid for the inverse problem. Going forward, we are interested in studying more realistic situations, such as multiple stages with dedicated machines. Different objective functions as well as other heuristic approaches and metaheuristic algorithms, such as genetic algorithm or tabu search, can also be assessed.

References

Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970). A heuristic algorithm for the n job, m machine sequencing problem. *Management science*, *16*(10), B-630.

Chen, B., Glass, C. A., Potts, C. N., & Strusevich, V. A. (1996). A new heuristic for three-machine flow shop scheduling. *Operations Research*, *44*(6), 891-898.

Cheng, T. E., Lin, B. M., & Tian, Y. (2009). Scheduling of a two-stage differentiation flowshop to minimize weighted sum of machine completion times. *Computers & Operations Research*, *36*(11), 3031-3040.

Conway, R. W., Maxwell, W. L., & Miller, L. W. (2003). *Theory of scheduling*. Courier Corporation.

Dridi, N., Hadda, H., & Hajri-Gabouj, S. (2009). Méthode heuristique pour le problème de flow shop hybride avec machines dédiées. *RAIRO-Operations Research*, *43*(4), 421-436.

Fattahi, P., Hosseini, S. M. H., Jolai, F., & Tavakkoli-Moghaddam, R. (2014). A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations. *Applied Mathematical Modelling*, *38*(1), 119-134.

Framinan, J. M., Gupta, J. N., & Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, *55*(12), 1243-1255.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, *1*(2), 117-129.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics, 5*, 287-326.

Gupta, J. N. (1971). A functional heuristic algorithm for the flowshop scheduling problem. *Journal of the Operational Research Society*, *22*(1), 39-47.

Gupta, J. N. (1988). Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society*, *39*(4), 359-364.

Gupta, J. N., Hariri, A. M. A., & Potts, C. N. (1997). Scheduling a two-stage hybrid flow shop with parallel machines at the first stage. *Annals of Operations Research*, *69*, 171-191.

Gupta, J. N., & Stafford Jr, E. F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, *169*(3), 699-711.

Hadda, H., Dridi, N., & Hajri-Gabouj, S. (2012). A note on the two-stage hybrid flow shop problem with dedicated machines. *Optimization Letters*, *6*(8), 1731-1736.

Hadda, H., Dridi, N., & Hajri-Gabouj, S. (2014). Exact resolution of the two-stage hybrid flow shop with dedicated machines. *Optimization Letters*, *8*(8), 2329-2339.

Hadda, H., Hajji, M. K., & Dridi, N. (2015). On the two-stage hybrid flow shop with dedicated machines. *RAIRO-Operations Research*, *49*(4), 795-804.

Haouari, M., & Daouas, T. (1999). Optimal scheduling of the 3-machine assembly-type flow shop. *RAIRO-Operations Research*, *33*(4), 439-445.

Huang, T. C., & Lin, B. M. (2013). Batch scheduling in differentiation flow shops for makespan minimisation. *International Journal of Production Research*, *51*(17), 5073-5082.

Kise, H. (1991). On an automated two-machine flowshop scheduling problem with infinite buffer. *Journal of the Operations Research Society of Japan*, *34*(3), 354-361.

Koulamas, C., & Kyparisis, G. J. (2001). The three-stage assembly flowshop scheduling problem. *Computers & Operations Research*, *28*(7), 689-704.

Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, *1*(1), 61-68.

Lee, C. Y., Cheng, T. C. E., & Lin, B. M. T. (1993). Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem. *Management Science*, *39*(5), 616-625.

Lin, B. M. (1999). The strong NP-hardness of two-stage flowshop scheduling with a common second-stage machine. *Computers & Operations Research*, *26*(7), 695-698.

Lin, B.M.T. (2015). Two-stage flow shop scheduling with dedicated machines. *International Journal of Production Research*, *53*(4), 1094–1097.

Lin, H. T., & Liao, C. J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. *International Journal of Production Economics*, *86*(2), 133-143.

Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: a survey. *Computers & Industrial Engineering*, *37*(1-2), 57-61.

Mosheiov, G., & Sarig, A. (2010). Minimum weighted number of tardy jobs on an m-machine flow-shop with a critical machine. *European Journal of Operational Research*, *201*(2), 404-408.

Nawaz, M., Enscore Jr, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91-95.

Oguz, C, Lin, B. M. T., & Cheng, T. C. E. (1997). Two-stage flow shop scheduling problem with a common second-stage machine. *Computers & Operations Research 24*, 1169–1174.

Palmer, D. S. (1965). Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum. *Journal of the Operational Research Society*, *16*(1), 101-107.

Potts, C. N., Sevast'Janov, S. V., Strusevich, V. A., Van Wassenhove, L. N., & Zwaneveld, C. M. (1995). The two-stage assembly scheduling problem: complexity and approximation. *Operations Research*, *43*(2), 346-355.

Riane, F., Artiba, A., and Elmaghraby, S. E. (1997). *A two-stage hybrid flow shop problem to minimize makespan*. Research Report No 310-97WR214, FUCaM, Mons, Belgium.

Riane, F., Artiba, A., & Elmaghraby, S. E. (1998). A hybrid three-stage flow shop problem: Efficient heuristics to minimize makespan. *European Journal of Operational Research 109*(2), 321–329.

Riane, F., Artiba, A., & Elmaghraby, S. E. (2002). Sequencing a hybrid two-stage flowshop with dedicated machines. *International Journal of Production Research*, *40*(17), 4353-4380.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, *165*(2), 479-494.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, *205*(1), 1-18.

Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, *47*(1), 65-74.

Wang, S., & Liu, M. (2013). A heuristic method for two-stage hybrid flow shop with dedicated machines. *Computers & Operations Research*, *40*(1), 438-450.

Yang, J. (2011). Minimizing total completion time in two-stage hybrid flow shop with dedicated machines. *Computers and Operations Research, 38*, 1045–53.

Yang, J. (2013). A two-stage hybrid flow shop with dedicated machines at the first stage. *Computers & Operation Research, 40*, 2836–2843.

Yang J. (2015). Minimizing total completion time in two-stage hybrid flow shop with dedicated machines at the first stage. *Computers & Operations Research, 58*, 1–8

Yenisey, M. M., & Yagmahan, B. (2014). Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, *45*, 119-135.

Yokoyama, M., & Santos, D. L. (2005). Three-stage flow-shop scheduling with assembly operations to minimize the weighted sum of product completion times. *European Journal of Operational Research*, *161*(3), 754-770.