RAIRO-Oper. Res. 49 (2015) 795–804
DOI: 10.1051/ro/2015005

RAIRO Operations Research
www.rairo-ro.org

ON THE TWO-STAGE HYBRID FLOW SHOP WITH DEDICATED MACHINES

Hatem Hadda[1], Mohamed Karim Hajji[2] and Najoua Dridi[2]

Abstract. In this paper we develop new elimination rules and discuss several polynomially solvable cases for the two-stage hybrid flow shop problem with dedicated machines. We also propose a worst case analysis for several heuristics. Furthermore, we point out and correct several errors in the paper of Yang [J. Yang, A two-stage hybrid flow shop with dedicated machines at the first stage. Comput. Oper. Res. **40** (2013) 2836−2843].

Keywords. Scheduling, hybrid flow shop, dedicated machines, worst case analysis, elimination rule.

Mathematics Subject Classification. 90B35, 90B30.

1. Introduction

This paper tackles a special case of the two-stage hybrid flow shop problem in which there are m dedicated machines M_k, $k \in \{1, \ldots, m\}$ on the first stage and a single machine M_0 on the second stage. The problem consists on minimizing the makespan ($C_{\max}$) for a set $J = \{J_1, J_2, \ldots, J_n\}$ of n jobs. The jobs belong to m disjoint types T_k, $k \in \{1, \ldots, m\}$. Each job is composed by a sequence of two operations. The first operation must be processed on M_k if the job is of type T_k ($k \in \{1, \ldots, m\}$), whereas the second operation of all jobs must be performed on M_0. The problem is strongly NP-hard [8] and we denote it by $F2(PD^m, 1)||C_{\max}$ where PD^m refers to the existence of m parallel dedicated machines on the first stage. We recall that permutation solutions are dominant and that $F2(PD^m, 1)||C_{\max}$ is

Received September 18, 2013. Accepted Ferbruary 11, 2015.

[1] LISI, INSAT, Centre Urbain Nord B.P. N676, 1080 Tunis Cedex. Tunisia.
hatem.hadda@esti.rnu.tn

[2] Unité de Recherche OASIS, ENIT, BP. 37, Le belvédère, 1002 Tunis. Tunisia.
h.mohamed.karim@hotmail.fr; najoua.dridi@enit.rnu.tn

equivalent to $F2(1, PD^m)||C_{\max}$ in which M_0 is on the first stage and the dedicated machines are on the second one [1]. In the reminder of the paper, all the results will be expressed for the problem $F2(PD^m, 1)||C_{\max}$.

In [9], authors consider $F2(PD^2, 1)||C_{\max}$ for which they propose two lower bounds and two polynomial cases. They also propose a $(\frac{3}{2})$-approximation algorithm (noted H_{OLC} hereafter). In [1] authors consider $F2(1, PD^m)||C_{\max}$ for which they identify several elimination rules and polynomial cases. They also derive new lower bounds and propose a new heuristic. Later, an efficient elimination rule is introduced in [4], and a heuristic method is developed in [11]. In [2] the author points out errors in the work given in [11]. In the same context, a worst case analysis is proposed for several simple heuristics in [12]. We note here that many of the results proposed in [12] were first proposed in [1]. More recently, a branch and bound algorithm is developed and tested in [6]. Authors also propose an empirical analysis of the makespan distribution for small sizes instances. Finally in [3], several approximations algorithms are proposed for $F2(1, PD^m)||C_{\max}$ under availability constraint.

We also emphasize that $F2(PD^m, 1)||C_{\max}$ can be considered as a particular case of the two-stage assembly problem $(A_m||C_{\max})$ in which there are m parallel machines on the first stage and a single assembly machine on the second stage. Each job is composed by $m+1$ operations. The first m operations can be processed in parallel on the first stage, then we proceed to the assembly operation on the second stage. Considering $A_m||C_{\max}$, the best known approximation algorithm guarantees an error bound of $(2-1/m)$ [10]. In the same context several heuristics with an error bound of 2 are proposed in [5] for the $A_m||C_{\max}$ problem under availability constraint.

The reminder of this paper is organized as follows. In Section 2 we introduce new elimination rules. In Section 3 we discuss some particular cases. In Section 4 we propose a worst case analysis for several heuristics. We also discuss and correct some of the results given in [12]. Finally, Section 5 concludes the paper and gives some perspectives.

2. Notation and basic properties

The following notations will be used in the subsequent analysis.

$J = \{J_1, J_2, \ldots, J_n\}$: set of all jobs;
n_k: number of jobs of type T_k, $k \in \{1, \ldots, m\}$, ($J = \bigcup_{k=1}^m T_k$ and $n = \sum_{k=1}^m n_k$);
a_i: processing time of $J_i \in T_k$ on M_k;
b_i: processing time of J_i on M_0;
$\pi = \langle \pi_1, \pi_2, \ldots, \pi_n \rangle$: permutation schedule where π_i is the job at the ith position;
$C_{ki}(\pi)$: completion time of $J_i \in J$ on M_k, $k \in \{0, \ldots, m\}$ in schedule π;
$C_{\max}(\pi)$: makespan of schedule π;
$C^\star_{\max}$: Optimal makespan.

For a given schedule, let J_u and J_v be two consecutive jobs on a dedicated machine, we designate by t_{uv} the sum of the processing times of the jobs (of other types) scheduled between J_u and J_v on M_0.

For a given type T_k, $k \in \{1, \ldots, m\}$, we denote by $\overline{T_k} = \{J_i \in T_k | a_i \leq b_i\}$ and $\underline{T_k} = \{J_i \in Q | a_i > b_i\}$. Moreover, for any given set of jobs $Q \subseteq J$ we also note by $a(Q) = \sum_{J_i \in Q} a_i$ and $b(Q) = \sum_{J_i \in Q} b_i$.

We recall that the two-machine flow shop problem ($F_2||C_{\max}$) can be solved by Johnson's Rule (JR) [7] which can be stated as follows: J_i precedes J_j if $\min\{a_i, b_j\} \leq \min\{a_j, b_i\}$.

Furthermore, for a given type T_k, $k \in \{1, \ldots, m\}$, we designate by z^k the optimal makespan if we consider an $F_2||C_{\max}$ problem with the jobs of T_k. Note that $z^k \leq C^{\star}_{\max}$, $\forall\ k \in \{1, \ldots, m\}$.

We now recall the following result.

Theorem 2.1 [4]**.** *Given a solution π for $F2(PD^m, 1)||C_{\max}$, let J_u and J_v be two consecutive jobs on M_k, $k \in \{1, \ldots, m\}$. If the relation*

$$\min\{a_u; b_v\} \leq \min\{a_v; b_u + t_{uv}\}, \tag{2.1}$$

does not hold, then the solution π' obtained from π by inserting J_v before J_u is no worse than π (i.e. $C_{\max}(\pi') \leq C_{\max}(\pi)$).

Using Theorem 2.1 it is possible to prove the existence of an optimal solution in which every couple of consecutive jobs of the same type verifies relation (2.1) [4].

When dealing with permutation solutions for $F2(PD^m, 1)||C_{\max}$, it should be understood in the sense that if two jobs of the same type succeeds each other on M_0 (possibly separated by other jobs of other types), they will appear in the same order on their respective dedicated machine. In this sense, a complete schedule is specified by a sequence of all jobs on M_0. We now introduce a simple yet interesting propriety for the $F2(PD^m, 1)||C_{\max}$ problem.

Theorem 2.2. *For $F2(PD^m, 1)||C_{\max}$, if we fix m sub-permutations for types T_k on machine M_k, $1 \leq k \leq m$, then a minimum completion time on M_0 is obtained by scheduling jobs in a non-decreasing order of their completion times on the first stage.*

Proof. When we fix the sequences on the dedicated machines, the minimization of the makespan on M_0 reduces to a $1|r_i|C_{\max}$ problem where the release date r_i of J_i corresponds to its completion time on the first stage. It is well known that an optimal solution for $1|r_i|C_{\max}$ is given by scheduling the jobs in a non-decreasing order of the release dates. □

Theorem 2.2 specifies the best way to interleave m given sub-permutations on M_0, which allows a considerable reduction of the search space. Indeed instead of searching the best possible permutation over the n jobs on M_0 (with $n!$ possible solutions), Theorem 2.2 suggests to look for the best sub-permutations on the dedicated machines (with $\Pi_{1 \leq k \leq m} n_k!$ solutions). We illustrate in Table 1 some values of the ratio $\Pi_{1 \leq k \leq m} n_k!/n!$ for several combinations of n and m (the jobs are supposed to be equally partitioned over the types). We also remark that given

TABLE 1. Ratio $\Pi_{1\leq k\leq m} n_k!/n!$.

	n		
	50	100	150
$m = 2$	8×10^{-15}	1×10^{-29}	1×10^{-44}
$m = 5$	2×10^{-32}	9×10^{-67}	2×10^{-101}
$m = 10$	2×10^{-44}	4×10^{-93}	3×10^{-142}

any solution π, it is always possible to construct a solution π' by rearranging the jobs on M_0 according to Theorem 2.2 such that $C_{\max}(\pi') \leq C_{\max}(\pi)$.

We now introduce new elimination rules.

Corollary 2.3. *For any given type T_k, $k \in \{1,\ldots,m\}$, if there exists a set $Q \subseteq \overline{T_k}$ such that $\max_{J_i \in Q}\{a_i\} \leq \min_{J_i \in T_k \setminus Q}\{a_i\}$ then there exists an optimal solution where the jobs of Q are scheduled first on M_k in a non-decreasing order of a_i.*

Proof. Let $\pi^\star$ be an optimal solution in which the jobs of Q are not scheduled first on M_k. In such a solution there are two jobs $J_u \in T_k \setminus Q$ and $J_v \in Q$ such that J_u directly precedes J_v on M_k. By definition we have $a_v \leq b_v$ and $a_v \leq a_u$, hence $\min\{a_u, b_v\} \geq \min\{a_v, b_u + t_{uv}\}$. Theorem 2.1 allows the insertion of J_v before J_u without altering the optimality of the solution. Consequently there exists an optimal solution in which the jobs of Q are scheduled first on M_k. Now let J_u and J_v be two consecutive jobs from Q such that $a_v < a_u$. Here too we have $\min\{a_u, b_v\} > \min\{a_v, b_u + t_{uv}\}$ and we can insert J_v before J_u. By applying the same argument for the rest of the jobs, it is possible to construct an optimal solution where the jobs of Q are scheduled first on M_k in a non-decreasing order of a_i. □

Corollary 2.4. *If for a given type T_k, $k \in \{1,\ldots,m\}$, we have $\underline{T_k} = \emptyset$ then there exists an optimal solution where the jobs of T_k are scheduled on M_k in a non-decreasing order of a_i.*

Proof. Similar to the proof of Corollary 2.3. □

3. Polynomially solvable cases

In this section we first introduce a simple polynomial case. We then consider a particular case discussed in [12] for which we show that the presented proof is incorrect, and then we reestablish the result.

Corollary 3.1. *For $F2(PD^m, 1)||C_{\max}$, if there is at most one job of each type T_k, $k \in \{1,\ldots,m\}$ (*i.e. $n = m$*), then an optimal solution is obtained by scheduling the jobs in a non-decreasing order of their processing processing times on the dedicated machines.*

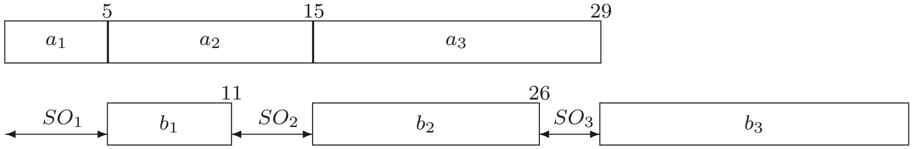

FIGURE 1. Counterexample.

Proof. As there is one job per type, then the completion times of the jobs on the first stage correspond to their processing times on the dedicated machine. Using Theorem 2.2 it should be easy to derive the result. □

Given a permutation π, Yang [12] associates for each job $\pi_u \in T_k$, $k \in \{1,\ldots,m\}$, the term $SO_{\pi_u} = C_{k,\pi_u} - C_{0,\pi_{u-1}}$ with $C_{0,\pi_0} = 0$. Based on this concept, the author proposes the following algorithm:

Algorithm *SO*

(1) Find job $J_j \in J$ with the smallest SO value, and schedule it first. Break ties arbitrarily.
(2) Remove J_j from J and repeat Step 1 until $J = \emptyset$.
(3) Calculate and output $C_{\max}$ and stop.

Based on this algorithm, Yang [12] proposes the following polynomial case.

Theorem 3.2 [12]**.** *For problem $F2(PD^m, 1)||C_{\max}$, if $a_i \leq b_i$ for all $J_i \in J$, then algorithm SO generates an optimal schedule.*

We are going to show that the proof proposed in [12] is inaccurate, and then reestablish a new one. In his proof, the author claims the following:

> "Suppose that there exists an optimal schedule which is generated by a rule different from Algorithm SO. Then, there exists job $J_j \in J$ which would have the smaller SO than the immediately preceding job on M_0 does."

This implies that Algorithm SO is supposed to generate a permutation in which SO values appear in a non-decreasing order which is not always true as we can see from the following example. Consider the problem instance with $n = 3$ and $m = 1$. Let $a_1 = 5$, $b_1 = 6$, $a_2 = 10$, $b_2 = 11$, $a_3 = 14$ and $b_3 = 15$ (See Fig. 1). It should be clear that Algorithm SO generates the optimal schedule $\langle J_1, J_2, J_3\rangle$ with $SO_1 = 5$, $SO_2 = 4$, $SO_3 = 3$. As it can be seen, SO values are not in a non-decreasing order. We now propose a new proof.

Proof. The considered case is such that $\underline{T_k} = \emptyset\ \forall k \in \{1,\ldots,m\}$. Using Corollary 2.4, we conclude that there exists an optimal solution where the jobs of T_k are scheduled on M_k in a non-decreasing order of a_i. This determines the order of the jobs on the dedicated machines. Theorem 2.2 ensures that an optimal solution is obtained by scheduling the jobs on M_0 in a non-decreasing order of their completion times on the first stage.

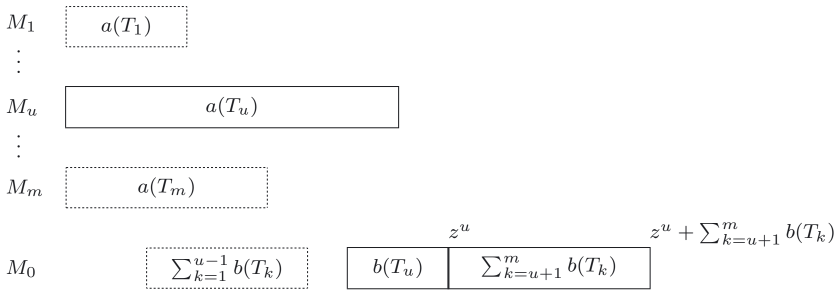

FIGURE 2. Makespan expression of π_{HHD}.

Note that at each step of Algorithm SO, selecting the job with least SO value, amounts to the selection of the job with the minimum completion time on the first stage. By proceeding in that way, Algorithm SO ensures that:

- The jobs of the same type are selected in a non-decreasing order of a_i.
- The jobs on M_0 are selected a non-decreasing order of their completion times on the first stage.

Which corresponds to the optimal solution described earlier. □

4. Worst case analysis

In this section we first introduce two $(2 - 1/m)$-approximation algorithms. We then show that the worst case analysis proposed in [12] for a heuristic is false and reestablish the result. We also discuss the worst case behavior of H_{OLC}.

We introduce now a new heuristic for $F2(PD^m, 1)||C_{\max}$ and calculate its worst-case error bound.

Heuristic H_{HHD}

(1) Apply JR to each type T_k and calculate $\delta_k = z^k - b(T_k)$ for $k \in \{1, \ldots, m\}$.
(2) Re-index the types such that $\delta_1 \leq \delta_2 \leq \ldots \leq \delta_m$.
(3) Retain schedule $\pi_{HHD} = \langle T_1, T_2, \ldots, T_m \rangle$ where each set T_k, $k \in \{1, \ldots, m\}$, is scheduled according to JR.

Theorem 4.1. *The relative worst-case error bound of* π_{HHD} *is given by* $C_{\max}(\pi_{HHD})/C^{\star}_{\max} \leq 2 - 1/m$, *and the bound is tight.*

Proof. Let $T_u \in \{T_1, \ldots, T_m\}$ be the type of jobs leading to the makespan in π_{HHD} (see Fig. 2). By definition of T_u, the completion time of its jobs on M_0 is exactly z^u. Given the position of T_u on M_0, we get $C_{\max}(\pi_{HHD}) = z^u + \sum_{k=u+1}^{m} b(T_k) = \delta_u + \sum_{k=u}^{m} b(T_k)$.

If $u = m$ then $C_{\max}(\pi_{HHD}) = z^m \leq C^{\star}_{\max}$, otherwise two cases have to be considered.

If $\delta_u \leq (1 - 1/m)C^\star_{\max}$ then $C_{\max}(\pi_{HHD}) = \delta_u + \sum_{k=u}^m b(T_k) \leq \delta_u + b(J) \leq (2 - 1/m)C^\star_{\max}$.

Otherwise we have $\delta_k \geq \delta_u > (1 - 1/m)C^\star_{\max}$ for $u + 1 \leq k \leq m$. As $z^k = \delta_k + b(T_k) \leq C^\star_{\max}$ then $b(T_k) \leq C^\star_{\max}/m$ for $u+1 \leq k \leq m$. Hence, we derive that

$$\begin{aligned} C_{\max}(\pi_{HHD}) &= z^u + \sum_{k=u+1}^{m} b(T_k) \\ &\leq z^u + \frac{m-u}{m} C^\star_{\max} \\ &\leq C^\star_{\max} + \frac{m-1}{m} C^\star_{\max} \\ &\leq \left(2 - \frac{1}{m}\right) C^\star_{\max}. \end{aligned}$$

To show that the bound is tight, we consider the following problem instance with $n = mw$, $T_k = \{J_{(k-1)w+i}, 1 \leq i \leq w\}$, $k \in \{1, \ldots, m\}$, where w is an integer such that $w > m \geq 2$. Let $a_{(k-1)w+1} = m - 1$ for $k \in \{1, \ldots, m\}$, $a_{(k-1)w+i} = 1/w$ for $k \in \{1, \ldots, m\}$ and $2 \leq i \leq w$, and $b_i = 1/w$, $\forall J_i \in J$.

When considering JR for a given set T_k, $k \in \{1, \ldots, m\}$, it is possible to schedule job $J_{(k-1)w+1}$ first. Hence H_{HHD} can generate the order $\pi_{HHD} = \langle J_1, \ldots, J_{mw}\rangle$ with $C_{\max}(\pi_{HHD}) = 2m-1$. However job $J_{(k-1)w+1}$ must be scheduled in the last position of set T_k, $k \in \{1, \ldots, m\}$. Hence an optimal solution is given by schedule $\pi^\star = \langle J_2, \ldots, J_w, J_{w+2}, \ldots, J_{2w}, J_{2w+2}, \ldots, J_{mw}, J_1, J_{w+1}, \ldots, J_{(m-1)w+1}\rangle$ with $C^\star_{\max} = m - 1/w + m/w$. We see that $C_{\max}(\pi_{HHD})/C^\star_{\max}$ goes to $2 - 1/m$ as w tends to infinity. □

We add here that the problem instance used to show the bound tightness in the proof of Theorem 4.1 was used in [12]. However, Yang [12] affirms that $C^\star_{\max} = m+1/w$, while the correct value is $C^\star_{\max} = m-1/w+m/w$. Indeed the completion time on M_0 of the sub-sequence $\pi^\star = \langle J_2, \ldots, J_w, \ldots, J_{2w+2}, \ldots, J_{mw}\rangle$ is less than the completion time of J_1 on M_1. We introduce now a new polynomial case.

Theorem 4.2. *If for the $F2(PD^m, 1)||C_{\max}$ problem, we constraint the jobs of each type to be directly scheduled after each other on M_0, then H_{HHD} is optimal.*

Proof. As the jobs of each type are constrained to be directly scheduled after each other on M_0 and given Theorem 2.1, we can see that it is dominant to schedule jobs of each type according to JR. Hence the problem reduces to the scheduling of m fictive jobs J'_k (one per type) where $a'_k = \delta_k = z^k - b(T_k)$, and $b'_k = b(T_k)$ for $1 \leq k \leq m$. Using Corollary 3.1 we can conclude that H_{HHD} is optimal. □

We now consider heuristic H_{PSSWZ} introduced in [10] for the $A_m||C_{\max}$ problem which guarantees a tight error bound of $2 - 1/m$. Heuristic H_{PSSWZ} consists solving an $F_2||C_{\max}$ problem where for each job, the processing time on the first machine corresponds to the mean of the m processing times on the first stage of the $A_m||C_{\max}$ problem, and the processing time on the second machine corresponds to the processing time on the assembly machine. When applied to $F2(PD^m, 1)||C_{\max}$,

H_{PSSWZ} reduces to constructing a sequence π_{PSSWZ} obtained by applying JR for all the jobs with the processing times a_i/m and b_i.

Theorem 4.3. *The relative worst-case error bound of π_{PSSWZ} is given by $C_{\max}(\pi_{PSSWZ})/C^{\star}_{\max} \leq 2 - 1/m$, and the bound is tight.*

Proof. From [10] we derive that $C_{\max}(\pi_{PSSWZ})/C^{\star}_{\max} \leq 2 - 1/m$.

To show that the bound is tight, we consider the following problem instance with $n = wm - w + 1$ where w is an integer such that $w > m \geq 2$. Each type T_k for $k \in \{1, \ldots, m-1\}$ contains w identical jobs with $a_i = m$ and $b_i = 1$. Type T_m contains a single job with $a_i = wm$ and $b_i = 2$. An optimal solution is obtained by scheduling w times a combination of $m-1$ jobs (with one job of each type T_k for $k \in \{1, \ldots, m-1\}$), then we schedule the single job of T_m in the last position. This gives us $C^{\star}_{\max} = wm + m + 1$. However H_{PSSWZ} could generate a solution in which the single job of T_m is scheduled first then the rest of the jobs appear in an arbitrary order with $C_{\max}(\pi_{PSSWZ}) = 2wm - w + 2$. We see that $C_{\max}(\pi_{PSSWZ})/C^{\star}_{\max}$ goes to $2 - 1/m$ as w tends to infinity. □

In [12], Yang proposes the following heuristic for $F2(PD^m, 1)||C_{\max}$.

Heuristic H_Y[12]

(1) Apply JR to the jobs in each T_k, $k \in \{1, \ldots, m\}$ by assuming $T_u = \emptyset$ for all $u \neq k$ for $u \in \{1, \ldots, m\}$.
Set K_{ki} be the completion time for job $J_i \in T_k$ on M_0 for $k \in \{1, \ldots, m\}$.
(2) Schedule jobs in non-decreasing order of $K_{0,i}$ for $J_i \in J$.
(3) Calculate and output $C_{\max}$ and stop.

Yang [12] claims that H_Y is a generalization of H_{OLC} introduced in [9] for $F2(PD^2, 1)||C_{\max}$. In fact this is not true. Indeed in step 2 of H_Y jobs are scheduled in non-decreasing order of $K_{0,i}$ while in H_{OLC}, the jobs are scheduled in non-decreasing order of their completion times on the dedicated machines. Hence, Yang [12] is considering a different heuristic. We add here that for computational results concerning the actual generalization of H_{OLC}, the reader is referred to [1].

Yang [12] claims that $C_{\max}(\pi_{H_Y})/C^{\star}_{\max} \leq 2 - 1/m$. The following Theorem proves that it is false and establishes the actual bound.

Theorem 4.4. *The relative worst-case error bound of H_Y is given by $C_{\max}(\pi_{H_Y})/C^{\star}_{\max} \leq 2$, and the bound is tight.*

Proof. We have $C_{\max}(\pi_{H_Y}) \leq \max_{1 \leq k \leq m} a(T_k) + b(J) \leq 2C^{\star}_{\max}$. To demonstrate that the bound is tight, we consider the following problem instance with $n = 2$ and $m = 2$, $T_1 = \{J_1\}$ and $T_2 = \{J_2\}$. Let $a_1 = 2$, $b_1 = w$, $a_2 = w$ and $b_2 = 1$ where $w > 2$.

We have $K_{0,1} = w + 2$ and $K_{0,2} = w + 1$. Hence H_Y will generate schedule $\pi_{H_Y} = \langle J_2, J_1 \rangle$ with $C_{\max}(\pi_{H_Y}) = 2w + 1$. However the optimal solution is $\pi^{\star} = \langle J_1, J_2 \rangle$ with $C^{\star}_{\max} = w + 3$. We see that $C_{\max}(\pi_{H_Y})/C^{\star}_{\max}$ goes to 2 as w tends to infinity. □

Regarding the analysis given by Yang [12] for the error bound of H_Y the proof is wrong as we are about to show. We fist recall the following Lemma.

Lemma 4.5 [12]**.** *An upper bound for the* $F2(PD^m, 1)||C_{\max}$ *problem is given by*

$$z^{UB} = \min \left\{ \begin{array}{c} \max\limits_{1\leq k\leq m} a(T_k) + b(J); \\ \min\limits_{1\leq k\leq m} \{\max\{z^k, \max\limits_{\substack{1\leq i\leq m \\ i\neq k}} a(T_i)\} + \sum\limits_{\substack{1\leq i\leq m \\ i\neq k}} b(T_i)\} \end{array} \right\}. \quad (4.1)$$

As to the worst case analysis proposed in [12] for H_Y, the presented proof is based on the assumption that $C_{\max}(\pi_{H_Y}) \leq z^{UB}$ which is false as showed by the following example.

Example 4.6. Consider the problem instance with $n = 3$ and $m = 2$, $T_1 = \{J_1\}$ and $T_2 = \{J_2, J_3\}$. Let $a_1 = 4$, $b_1 = 12$, $a_2 = 6$, $b_2 = 4$, $a_3 = 6$ and $b_3 = 3$. For T_1 we have $a(T_1) = 4$, $b(T_1) = 12$ and $z^1 = 16$. For T_2 we have $a(T_2) = 12$, $b(T_2) = 7$ and $z^2 = 15$ with the order $\langle J_2, J_3\rangle$. H_Y will generate the order $\pi_{H_Y} = \langle J_2, J_3, J_1\rangle$ on M_0 with $C_{\max}(\pi_{H_Y}) = 27$. However $z^{UB} = \min\{\max\{a(T_1), a(T_2)\} + b(J); \max\{z^1, a(T_2)\} + b(T_2); \max\{z^2, a(T_1)\} + b(T_1)\} = 23$. We finally remark that z^{UB} still constitutes a valid upper bound on the optimal makespan ($C^\star_{\max}$).

We now establish the worst case ratio of H_{OLC}.

Theorem 4.7. *The relative worst-case error bound of* H_{OLC} *is given by* $C_{\max}(\pi_{H_{OLC}})/C^\star_{\max} \leq 2 - 1/m$*, and the bound is tight.*

Proof. As for both H_{OLC} and H_{HHD}, we can start from the same sequences on the dedicated machines, we can derive from Theorem 2.2 that $C_{\max}(\pi_{H_{OLC}}) \leq C_{\max}(\pi_{H_{HHD}})$. Using Theorem 4.1 we get $C_{\max}(\pi_{H_{OLC}})/C^\star_{\max} \leq 2 - 1/m$. To show the bound tightness it is possible to utilize the same problem instance used in the proof of Theorem 4.1. □

It is worth noting that the proof presented in [12] for the worst case error bound on H_Y, holds for any heuristic H as long as it verifies

$$C_{\max}(\pi_H) \leq z^{UB}. \quad (4.2)$$

In fact the proof presented in [12] is similar to the one presented in [9], and in both works assumption (4.2) has been made without sufficient justifications. Indeed in [9] inequality (4.2) was supposed to be evident. While in [12] it was justified by the following argument

> "H_{OLC} applies Johnson's rule to jobs in each T_k separately and merge m schedules together".

It can be seen that this argument is very weak. Indeed it may be applied to any heuristic as long as it starts from Johnson's order for each type. This is clearly wrong as proven in Example 4.6. Never the less (4.2) is true for H_{OLC} as we shall prove.

Lemma 4.8. $C_{\max}(\pi_{H_{OLC}}) \le z^{UB}$.

Proof. Note that for a given $1 \le k \le m$, the value $\max\{z^k, \max_{\substack{1\le i\le m\\ i\neq k}} a(T_i)\} + \sum_{\substack{1\le i\le m\\ i\neq k}} b(T_i)$ is an upper bound for any solution in which the jobs of each type are constrained to be directly scheduled after each other on M_0 starting by the jobs of T_k. Theorem 4.2 specifies the best way to schedule those blocks of jobs and hence we derive that $C_{\max}(\pi_{H_{OLC}}) \le C_{\max}(\pi_{H_{HHD}}) \le z^{UB}$. □

5. Conclusion

In this paper we introduced new elimination rules and polynomial cases for the $F2(PD^m, 1)||C_{\max}$ problem. We also proposed a new approximation algorithm with a tight worst case error of $2 - 1/m$. We also discussed the work presented in [12] where we corrected several errors and reestablished the worst case analysis for a heuristic. An interesting issue that deserves future investigation is to consider generalizing some of the presented results for the configuration with several dedicated machines on both stages. It is also interesting to investigate more sophisticated approximation algorithms.

References

[1] N. Dridi, H. Hadda and S. Hajri-Gabouj, Méthode heuristique pour le problème de flow shop hybride avec machines dédiées. *RAIRO – Oper. Res.* **43** (2009) 421–36.

[2] H. Hadda, A note on “A heuristic method for two-stage hybrid flow shop with dedicated machines”. *Comput. Oper. Res.* **40** (2013) 2283.

[3] H. Hadda, N. Dridi and S. Hajri-Gabouj, Etude du flow shop hybride à deux étages avec machines dédiées sous contrainte dindisponibilité. In *Proc. 5th International Conference.* Rabat, Morocco (2007).

[4] H. Hadda, N. Dridi and S. Hajri-Gabouj, A note on the two-stage hybrid flow shop problem with dedicated machines. *Optim. Lett.* **6** (2012) 1731–1736.

[5] H. Hadda, N. Dridi and S. Hajri-Gabouj, The two-stage assembly flow shop scheduling with an availability constraint: worst case analysis. *J. Math. Modell. Algorithms Oper. Res.* **13** (2014) 233–245.

[6] H. Hadda, N. Dridi and S. Hajri-Gabouj, Exact resolution of the two-stage hybrid flow shop with dedicated machines. *Optim. Lett.* **8** (2014) 2329-2339.

[7] S.M. Johnson, Optimal two- and three-stage production schedules with setup times included. *Naval Res. Logist. Quart.* **1** (1954) 61–68.

[8] B.M.T. Lin, The strong NP-hardness of two-stage flowshop scheduling with a common second-stage machine. *Comput. Oper. Res.* **26** (1999) 695–698.

[9] C. Oguz, B.M.T. Lin and T.C.E. Cheng, Two-stage flowshop with a common second-stage machine. *Comput. Oper. Res.* **24** (1997) 1169–1174.

[10] C.N. Potts, S.V. Sevast’janov, V.A. Strusevich, L.N. Van Wassenhove and C.M. Zwaneveld, The two-stage assembly scheduling problem: Complexity and approximation. *Oper. Res.* **43** (1995) 346–355.

[11] S. Wang and M. Liu, A heuristic for two-stage hybrid flow shop with dedicated machines. *Comput. Oper. Res.* **40** (2013) 438–450.

[12] J. Yang, A two-stage hybrid flow shop with dedicated machines at the first stage. *Comput. Oper. Res.* **40** (2013) 2836–2843.