New complexity results on scheduling problem in a robotic cell

Nacira Chikhi, Moncef Abbas, Rachid Benmansour, Said Hanafi

RAIRO-Oper. Res. 51 (2017) 749–762
DOI: 10.1051/ro/2016053

RAIRO Operations Research
www.rairo-ro.org

NEW COMPLEXITY RESULTS ON SCHEDULING PROBLEM IN A ROBOTIC CELL

Nacira Chikhi[1,2], Moncef Abbas[1], Rachid Benmansour[2] and Saïd Hanafi[2]

Abstract. This paper explores the coordinated scheduling problem between production and transportation in a two stage flow shop with dedicated machines. There are two dedicated machines at the first stage and one common machine at the second stage. Each job has to be processed on a specified machine at the stage 1 depending on job type. A transporter with limited capacity is available to transport the semi-finished jobs from stage 1 to stage 2 for further processing. The objective is to minimize the makespan, *i.e.* the maximum completion time of all the jobs. The main focus is on the case where the transporter capacity is equal to two. New complexity results related to this case are established. Due to the NP-hardness of the general problem, we develop approximative approach to tackle the problem. Computational results indicate that the obtained solutions within moderate CPU time are of high quality.

Mathematics Subject Classification. 90B35, 90C59, 90C11.

Received December 29, 2015. Accepted August 28, 2016.

1. Introduction

Scheduling is an important process widely used in manufacturing and production. Appropriate scheduling can reduce material handling time and provides solutions for job sequencing. In a flow shop manufacturing system, semi-finished jobs are transferred between machines or stages by transports. Typically in most classical shops scheduling models, transportation times are neglected or the availability of transporters are ignored. In this study, however, not only transportation times is considered but also transporter's capacity.

The problem addressed in this paper is a special case of classical two-stage flow shop. We consider two dedicated machines at the first stage and a common machine at the second one. The jobs are transported between stages by a single robot and the objective is to minimize the makespan, *i.e.* the maximum completion time of all the jobs. This problem is different from machine environment studied in a simple two-stage flow shop problem because of the presence of dedicated machines at the first stage. Furthermore, transportation capacity and times are taken into account. Many papers focus on two-stage productions scheduling where transportation is performed by a single robot. Panwalkar [11] studied a two-machine flow shop problem with travel times between machines. A polynomial algorithm is proposed to tackle the problem. Kise [7] considered a

Keywords. Flowshop – complexity – makespan – dynamic algorithm – tranportation.

[1] Université des Sciences et de la Technologie USTHB, Laboratoire AMCD&RO, BP32, Bab-Ezzouar, 16111, Alger, Algérie. nacira.chikhi@gmail.com; mabbas@usthb.dz

[2] Université de Valenciennes et du Hainaut Cambrésis, LAMIH – UMR CNRS 8201, Le Mont-Houy 59313 Valenciennes cedex 9, France. rachid.benmansour, said.hanafi@univ-valenciennes.fr

two-machine flow shop problem makespan with one transporter and proved the NP-Hardness of the problem in the case of identical transportation times. Hurink and Knust [6] established complexity results for some special cases of a flow-shop problem on m machines with transportation times. Lee and Chen [8] studied 2 types of transportation in the flow shop problem, the first one consists of the transfer of semi-finished jobs and the second concerns the delivery of finished jobs. Both transporters's capacity and transportation times were taken into account. Tang and Liu [12] considered a two-machine flow shop where a single machine is followed by a batching machine. There is a transporter to carry the jobs between machines. The objective of the problem is to minimize the makespan, they formulate it as a mixed integer programming model and then prove that it is strongly NP-hard. A heuristic algorithm is proposed for solving this problem and its worst case performance is analyzed. The scheduling of a robotic cell in which jobs are processed on two tandem machines is studied by Levner *et al.* [9]. The job transportation between the machines is done by a transportation robot. The robotic cell has limitations on the intermediate space between the machines for storing the work-in-process. A polynomial algorithm is proposed with the proof of optimality to tackle the problem. The authors in [14] studied a problem that arises in a flow shop environment where there are two processing stages and a single transporter that is available to deliver the finished jobs from the first stage to the second. There is a single machine in the first stage and two parallel machines in the second stage. The transporter can carry only one job in each shipment. They proposed a fast heuristic with its performance analysis. Ahmadizar and Shahmaleki [1] studied a group-shop scheduling problem with sequence-dependent set-up times and transportation times. The objective is to minimize the makespan. The problem was formulated as a disjunctive programming problem. The problem considered in [13] is a two-stage hybrid flow shop where a discrete machine is followed by a batching machine. The authors analyzed the computational complexity of a class of two-machine problems with dynamic job arrivals. For the NP-complete problems, they proposed heuristics and provided their performance ratios. The robotic scheduling problem in blocking hybrid flow shop cells is studied in [4]. Multiple part types, unrelated parallel machines, multiple robots and machine eligibility constraints are considered.

The organization of the rest of this paper is as follows. Section 2, is devoted to describe the problem. In Section 3, we focus our study on a particular problem where the capacity is equal to two and the processing times of jobs are identical on the first stage. In Section 4, we prove that the problem is NP-Hard in the case where the capacity is equal to two and the processing times of jobs are identical on the second stage. Section 5, presents an approximative approach for solving the general problem. Computational experiments are carried out in Section 6 to show the performance of the proposed method. Finally, the last section contains a conclusion and some prospects for future research.

2. Statement of the problem

The problem $TF3|\sigma = 2, \upsilon = 1, c \geq 1|C_{\max}(N)$ is defined as follows. The machine environment consists of two stages in series. There exists two dedicated machines M_1 and M_2 in stage one and a single common machine M_3 in stage two with unlimited buffer spaces. There is a set $N = \{1, 2, \ldots, n\}$ of n independent jobs available from time zero to be processed on this two-stage flow shop. These jobs belong to 2 job types: type 1, $N_1 = \{1, 2, \ldots, n_1\}$ and type 2, $N_2 = \{n_1 + 1, n_1 + 2, \ldots, n_1 + n_2\}$ with $n_1 + n_2 = n$. Each job consists of two operations, of which the first on the dedicated machine M_k, $k \in \{1, 2\}$ if it is of part type k, and the second is performed on the common machine in stage two. Each job $i \in N$ has processing time p_i^1 on its dedicated machine in stage one and p_i^2 on the common machine in stage two. Preemption is not allowed and the machines can process only one job at a time. All jobs are transported by a single robot from stage one to stage two for further processing. The robot can carry up to c jobs in one shipment. The transportation time from the first stage to the second stage is denoted by t (a round-trip requires $2t$). The loading and unloading times are insignificant. The objective is to minimize the makespan, *i.e.*, the maximum job completion time in the second stage denoted by $C_{\max}$. Our problem is denoted $TF3|\sigma = 2, \upsilon = 1, c \geq 1|C_{\max}(N)$ according to the commonly used three-field notation of Graham *et al.* [5] for machine scheduling problems, ($\sigma = 2$ means that every job is comprised of two operations and $\upsilon = 1$ means that there is only one conveyor with a capacity c). The problem

TABLE 1. Complexity results on the general problem.

Problem characteristics	Problem complexity
$p_i^1 = p_1\ \forall i \in N,\ c = 1$	Polynomial [2]
$p_i^2 = p_2\ \forall i \in N,\ c = 1$	Polynomial [2]
$p_i^2 = p_2\ \forall i \in N,\ c \geq 3$	Strongly NP-Hard [2]

$TF3|\sigma = 2, \upsilon = 1, c \geq 1|C_{\max}(N)$ is strongly NP-hard since a particular problem without transportation times, denoted $F3|\sigma = 2|C_{\max}(N)$ was shown to be strongly NP-hard in [10]. A formulation for our problem is proposed in [3]. We summarize previous complexity results on the problem $TF3|\sigma = 2, \upsilon = 1, c \geq 1|C_{\max}(N)$ in Table 1. We denote P the problem "$TF3|\sigma = 2, \upsilon = 1, c = 2|C_{\max}(N)$" and we focus our study on the two problems with capacity $c = 2$, $P1$("$TF3|\sigma = 2, p_j^1 = p_1 \forall j \in N, \upsilon = 1, c = 2|C_{\max}$") and $P2$ ("$TF3|\sigma = 2, p_j^2 = p_2 \forall j \in N, \upsilon = 1, c = 2|C_{\max}$") respectively, which are particular cases of the problem $TF3|\sigma = 2, \upsilon = 1, c = 2|C_{\max}(N)$.

- Problem $P1$: In this problem, the processing times of jobs are identical on the first stage and the robot capacity is equal to two.
- Problem $P2$: In this problem, the processing times of jobs are identical on the second stage and the robot capacity is equal to two.

3. Identical processing times on the first stage ($P1$)

Although the general problem $TF3|\sigma = 2, \upsilon = 1, c = 2|C_{\max}(N)$ is NP-Hard, we show that there are particular cases which can be solved in polynomial time under the following assumption:

Assumption 3.1. The processing times on the first stage are identical *i.e.* $\forall j \in N : p_j^1 = p_1$.

We distinguish two cases for the problem $P1$; the case where $p_1 \geq 2t$ and the case where $p_1 \leq 2t$.

3.1. Case $p_1 \geq 2t$

We establish some properties to obtain an optimal schedule for the problem $P1$ in the case where $p_1 \geq 2t$ based on LPT (Longest Processing Time) rule.

Proposition 3.2. *There exists an optimal solution for the problem $P1$ such that jobs of each type are sequenced in the non-increasing order of p_j^2,* i.e. *LPT(N_1) and LPT(N_2). Moreover, if $p_1 \geq 2t$, where p_1 is the identical processing time of jobs on the first stage and $2t$ is a round trip of the robot, then there exits an optimal solution such that each batch contains exactly two jobs of different type and this batch is transported from the first stage to the second stage immediately after the two jobs are completed on the first stage.*

Proof. Under Assumption 3.1, if the jobs of each type are sequenced in the non-increasing order of p_j^2, we reduce the idleness on the common machine. In this case and since the transporter's capacity is equal to two, the first transported batch at time $d_1 = p_1$ contains the two first jobs completed at the same time. Once the transporter returns to the first stage at time $p_1 + 2t$, there is no completed job at this time because $p_1 \geq 2t$. The transporter will wait until $d_2 = d_1 + p_1$ and then transports the two completed jobs at this time. In the same manner, each batch "k" contains one job of type 1 and another of type 2 and both of them are transported at time $d_k = kp_1$. Finally, all the jobs are processed on the common machine in order of their arrival to stage 2. This schedule gives the optimal makespan by minimizing the idleness on the common machine. A simple pairwise argument can prove that it is optimal to process jobs in N_1 and N_2 respectively in LPT(N_1) and LPT(N_2) order according to their processing times p_i^2 at the second stage as follows.

Assume another type of schedule is optimal. In this optimal schedule, there must be a pair of consecutive batches of jobs, each batch contains exactly two jobs of different types. Consider the first batch of jobs contains

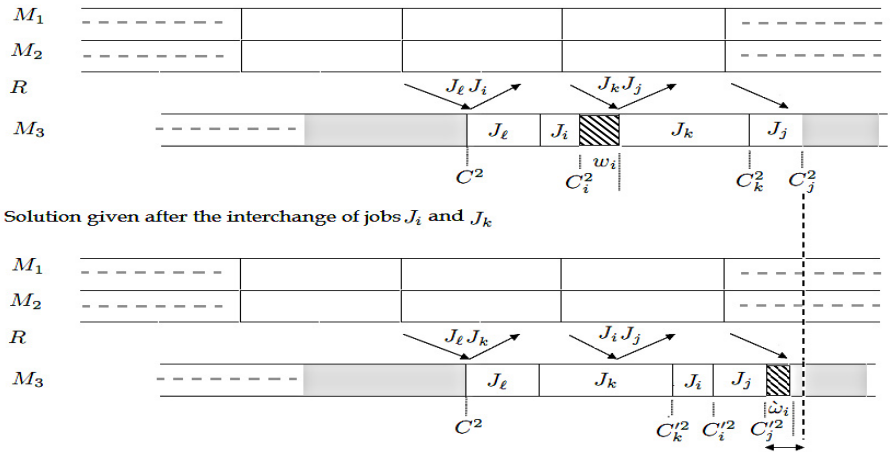

FIGURE 1. Interchanging of the two jobs J_i and J_k.

J_ℓ of type 1 and J_i of type 2 followed by the second batch of jobs J_j of type 1 and J_k of type 2, such that the following condition $p_k^2 > p_i^2$ is satisfied. It is sufficient to show that under this condition and after a pairwise interchange of jobs J_i and J_k, the makespan is reduced.

Suppose that in this original schedule job J_i precedes job J_k and job J_j follows job J_k as seen in Figure 1.

Let C_i^2 and C_k^2 denote the completion time of jobs J_i and J_k respectively on machine M_3 under the original schedule and let $C_i'^2$ and $C_k'^2$ denote the completion time of jobs J_i and J_k respectively on machine M_3 after the pairwise interchange. Let w_i is the created idle time after the processing of job J_i on machine M_3 in the original schedule and let C^2 is the makespan of a partial schedule on machine M_3 before the processing of jobs of the batch containing the jobs J_ℓ and J_i on the same machine.

It is sufficient to show that $C_i'^2 \leq C_k^2$ under the condition described above.

The completion time of job J_k on M_3 under the original schedule is $C_k^2 = C^2 + p_\ell^2 + p_i^2 + w_i + p_k^2$, whereas the completion time of job J_i on M_3 after the pairwise interchange is $C_i'^2 = C^2 + p_\ell^2 + p_k^2 + p_i^2$. It is easy to see that $C_i'^2 \leq C_k^2$.

On the other hand, it is also interesting to know at what time machine M_3 becomes available for processing job J_j. Interchanging jobs J_i and J_k clearly reduce the starting time of job J_j on the machine M_3 as shown in Figure 1. Therefore, it is optimal to process jobs in LPT(N_1) and LPT(N_2) order according to their processing times p_i^2 on the common machine at the second stage. □

3.2. Case $p_1 \leq 2t$

We propose a dynamic algorithm to solve optimally the problem $P1$ in the case where $p_1 \leq 2t$ and where the number of jobs of each type are identical (*i.e.* $n_1 = n_2$). This dynamic programming algorithm is valid under Assumption 3.1 and where the jobs of each type are sequenced in the non-increasing order of p_j^2. We show that for $p_1 \leq t$, the problem becomes polynomially solvable. We denote η the number of the jobs of each type. The parameters of the dynamic programming algorithm as shown in Figure 2 are as follows.

Let η is the the number of jobs of each type, *i.e.* $\eta = n_1 = n_2$.

Let us assume that the sets N_1 and N_2 are sorted in the non-increasing order according to the processing times p_j^2.

Let $F(k_1, k_2, d(\frac{k_1+k_2}{2}))$ be the minimum completion time of a *partial* schedule after delivering the first $\frac{k_1+k_2}{2}$ batches containing the first k_1 jobs of type 1 and the first k_2 jobs of type 2 such that the last batch is transported at time $d(\frac{k_1+k_2}{2})$. In this case, k_1 (resp. k_2) is the index of the last job of the partial schedule transported in N_1 (resp. N_2).

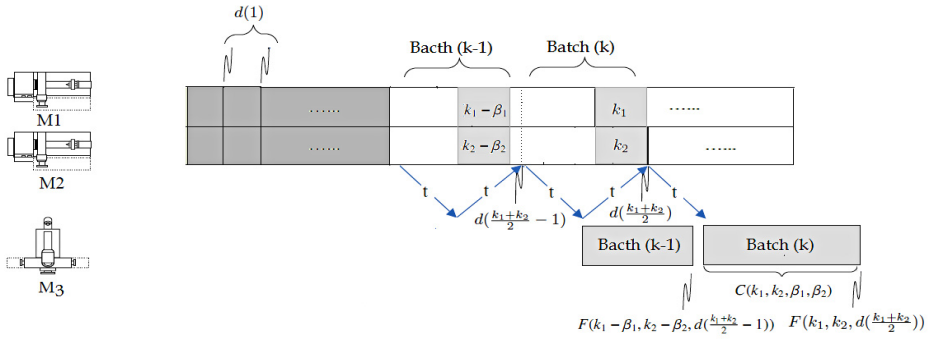

FIGURE 2. Parameters of the dynamic algorithm.

Let $C(k_1, k_2, \beta_1, \beta_2)$ be the minimum increase of makespan due to the delivering at time $d(\frac{k_1+k_2}{2})$ of the last batch containing β_1 (resp. β_2) of jobs of type 1 (resp. type 2).

Initialization:

- $F(1, 1, d(1)) = p_1 + t + p_1^2 + p_{\eta+1}^2$; $F(2, 0, d(1)) = 2p_1 + t + p_1^2 + p_2^2$; $F(0, 2, d(1)) = 2p_1 + t + p_{\eta+1}^2 + p_{\eta+2}^2$.
- $C(k_1, k_2, 1, 1) = p_{k_1}^2 + p_{\eta+k_2}^2$; $C(k_1, k_2, 2, 0) = p_{k_1}^2 + p_{k_1-1}^2$; $C(k_1, k_2, 0, 2) = p_{\eta+k_2}^2 + p_{\eta+k_2-1}^2$
- $d(\frac{k_1+k_2}{2}) = \begin{cases} p_1, & \text{if} \quad k_1 = k_2 = 1; \\ 2p_1, & \text{if} \quad k_1 + k_2 = 2 \text{ and } k_1 k_2 = 0. \end{cases}$

Recurrence relations:

For $k = 2, \ldots, \eta$:

If $p_1 \leq t$ then we have $d(k+1) = d(k) + 2t$

If $t < p_1 \leq 2t$ then we have 2 cases: $d(k+1) = \begin{cases} d(k) + 2t, \text{ or} \\ p_1 \lfloor \frac{d(k)}{p_1} \rfloor + 2p_1. \end{cases}$

For $k_1, k_2 \leq \eta$ we have:

$$F(k_1, k_2, d(\tfrac{k_1+k_2}{2})) = \min \begin{cases} \max\left\{F\left(k_1 - 1, k_2 - 1, d\left(\dfrac{k_1+k_2}{2} - 1\right)\right);\ d\left(\dfrac{k_1+k_2}{2}\right) + t\right\} + C\left(k_1, k_2, 1, 1\right) \\ \max\left\{F\left(k_1 - 2, k_2, d\left(\dfrac{k_1+k_2}{2} - 1\right)\right);\ d\left(\dfrac{k_1+k_2}{2}\right) + t\right\} + C\left(k_1, k_2, 2, 0\right) \\ \max\left\{F\left(k_1, k_2 - 2, d\left(\dfrac{k_1+k_2}{2} - 1\right)\right);\ d\left(\dfrac{k_1+k_2}{2}\right) + t\right\} + C\left(k_1, k_2, 0, 2\right) \end{cases}$$

Optimal value:

$$\min\left\{\max\left\{F(\eta - \beta_1, \eta - \beta_2, d(\eta - 1)); d(\eta) + t\right\} + C(\eta, \eta, \beta_1, \beta_2) : \beta_1 + \beta_2 = 2\right\}.$$

Theorem 3.3. *The dynamic algorithm solves optimally the problem* $TF3/v = 1, t \leq p_j^1 = p_1 \leq 2t\ \forall j \in N, c = 2/C_{\max}$ *in complexity* $O(2^\eta)$.

Proof. If $t < p_1 \leq 2t$ then we have 2 cases: $d(k+1) = \begin{cases} d(k) + 2t, \text{ or} \\ p_1 \lfloor \frac{d(k)}{p_1} \rfloor + 2p_1. \end{cases}$ Recall that $1 \leq k \leq \eta$, then our dynamic algorithm solves optimally the problem $TF3/v = 1, t \leq p_j^1 = p_1 \leq 2t\ \forall j \in N, c = 2/C_{\max}$ in complexity $O(2^\eta)$. □

Theorem 3.4. *The dynamic algorithm solves optimally the problem* $TF3/v = 1, p_j^1 = p_1 \leq t\ \forall j \in N, c = 2/C_{\max}$ *in complexity* $O(\eta^2)$.

TABLE 2. Processing times.

Jobs	1	2	3	4	5	6
p_j^1	2	2	2	2	2	2
p_j^2	6	5	4	3	2	1

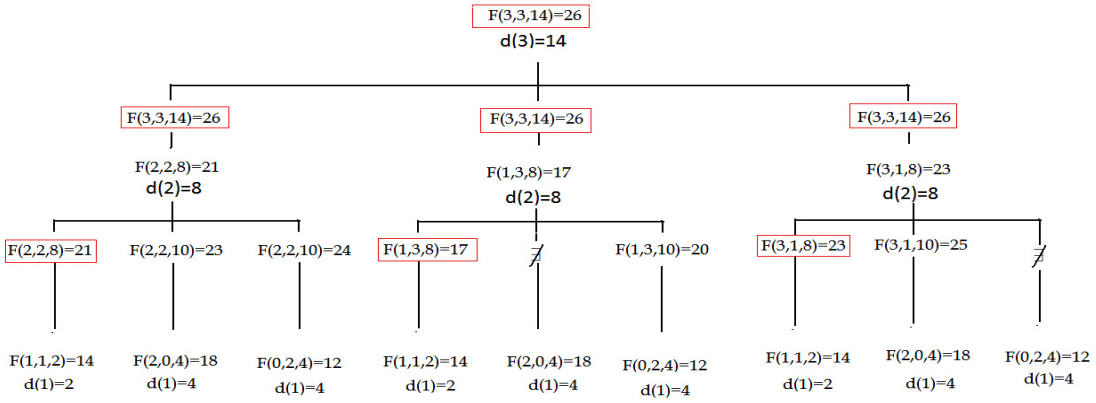

FIGURE 3. Application of the dynamic algorithm.

Proof. To obtain the complexity of this algorithm for the case $p_1 \leq t$, there are a total of η possibilities for the immediate departure points. But for each departure $k = \frac{k_1+k_2}{2}$, we have $k \leq \eta$. In addition, in the recurrence relation, for each possible combination of (k_1, k_2), there exists one possible value of $F(k_1, k_2, d(\frac{k_1+k_2}{2}))$ because $d(\frac{k_1+k_2}{2})$ has only one value for the case $p_1 \leq t$. Hence the overall complexity can be calculated as follows:

Consider $\Omega = \{(k_1, k_2) : k_1 + k_2 = 2k,\ k \leq \eta\}$ is the set of possibilities of combinations (k_1, k_2).

For $0 \leq i \leq \eta$, we denote $u_i = \{(i, k_2) : i + k_2 = 2k\ k \leq \eta\}$ and $\mu_i = cardinal(u_i)$. Thus $Cardinal(\Omega) = \sum_{0 \leq i \leq \eta} \mu_i$.

It is easy to check that

(1) If η and i are even numbers then $\mu_i = \frac{\eta}{2} + 1$;
(2) If η is even and i is odd then $\mu_i = \frac{\eta}{2}$;
(3 If η is odd and i is even then $\mu_i = \frac{\eta+1}{2}$;
(4) If η and i are odd numbers then $\mu_i = \frac{\eta+1}{2}$.

In the same manner, we deduce that:

$$\begin{cases} \text{if } \eta \text{ is even number then, } \text{Cardinal}(\Omega) = \frac{\eta^2}{2} + \eta + 1; \\ \text{if } \eta \text{ is odd number then, } \text{Cardinal}(\Omega) = \frac{\eta^2+1}{2} + \eta. \end{cases}$$

Therefore, the overall complexity of the dynamic algorithm for the case $p_1 \leq t$ is $O(\eta^2)$ which is polynomial. □

Example

We illustrate the running of the dynamic algorithm on the following example. Let us consider 6 independent jobs with the transportation time is $t = 3$ and where the sets of jobs are $N_1 = \{1, 2, 3\}$ and $N_2 = \{4, 5, 6\}$. The processing times of the jobs on the two stages are given in Table 2.

The dynamic programming algorithm generates the optimal value for the makespan as shown in Figure 3. From Figure 3, there are three optimal solutions given by the dynamic programming algorithm with $C^*_{\max} = 26$.

The jobs of the type 1 are processed on machine M_1 according to the order $\{1, 2, 3\}$ and the jobs of type 2 are processed on machine M_2 in this order $\{4, 5, 6\}$.

$\sigma_1^* = \{1, 4, 2, 5, 3, 6\}$ The different batches associated to this optimal schedule are $B_1 = \{1, 4\}$, $B_2 = \{2, 5\}$ and $B_3 = \{3, 6\}$ with the departure times 2, 8 and 14 respectively.

$\sigma_2^* = \{1, 4, 5, 6, 2, 3\}$ The different batches associated to this optimal schedule are $B_1 = \{1, 4\}$, $B_2 = \{5, 6\}$ and $B_3 = \{2, 3\}$ with the departure times 2, 8 and 14 respectively.

$\sigma_3^* = \{1, 4, 2, 3, 5, 6\}$ The different batches associated to this optimal schedule are $B_1 = \{1, 4\}$, $B_2 = \{2, 3\}$ and $B_3 = \{5, 6\}$ with the departure times 2, 8 and 14 respectively.

Recall that in the case of instances of large size, the dynamic programming algorithm can find optimal solutions for the problem $P1$: case $p_1 \leq 2t$ and $n_1 = n_2 = \eta$ but it can take too long time. To tackle this, we have developed the following heuristic H_p for solving the same problem.

Algorithm 1. Heuristic H_p.

Depending on the job type, decompose the set N of n jobs into two mutually exclusive families N_1, N_2.
Sort jobs in N_1 in non-increasing order such that $p_i^2 \geq p_{i+1}^2$ and denote the result list $\pi^1 = \{\pi_1^1, \ldots, \pi_{n_1}^1\}$;
Sort jobs in N_2 in non-increasing order such that $p_i^2 \geq p_{i+1}^2$ and denote the result list $\pi^2 = \{\pi_1^2, \ldots, \pi_{n_2}^2\}$;
Process the jobs in π^1 on the machine M_1 and process the jobs in π^2 on the machine M_2.
$B_1 = \{\pi_1^1, \pi_1^2\}$ is the first batch transported at time $d_1 = p_1$.
Let $\pi^1 := \pi^1 - \{\pi_1^1\}$ and $\pi^2 := \pi^2 - \{\pi_1^2\}$,
$i := 2$; $j := 2$; $k := 2$; $S := \emptyset$;
while $\pi^1 \neq \emptyset$ or $\pi^2 \neq \emptyset$ **do**
  Find the set $\{\pi_i^1, \ldots, \pi_{k_1}^1\}$ in $\pi^1/k_1 = \lfloor \frac{d(k)+2t}{p_1} \rfloor$.
  Find $\{\pi_j^2, \ldots, \pi_{k_2}^2\}$ in π^2 such that $k_2 = k_1$.
  $S := S \bigcup \{\pi_i^1, \ldots, \pi_{k_1}^1, \pi_j^2, \ldots, \pi_{k_2}^2\}$.
  Sort S in LPT rule according to the processing times on the second stage.
  Select the first two jobs in the ordered set S and transport them in a the k^{th} batch B_k at time $d_k = d_1 + 2t(k-1)$.
  $i := k_1 + 1$; $j := k_2 + 1$; $S := S/B_k$; k:=k+1;
end while
$B = \bigcup_{k=1}^{\eta} B_k$ (B is the set of batches).
Transport the batches in B from the first stage to the second stage to be processed on the common machine.
Process the jobs at the second stage according to the FCFS rule.
Calculate the corresponding makespan for this schedule.

4. Identical processing times on the second stage $P2$

In this section, we shall prove the NP-Hardness of the problem $P2$ ($TF3|\sigma = 2, p_j^2 = p_2 \forall j \in N, \upsilon = 1, c = 2|C_{\max}$) by reduction from 3-partition problem, which is a known strong NP-complete problem. First, we define the 3-partition problem.

The 3-partition problem.
Given a set $H = \{1, \ldots, 3h\}$ of $3h$ items, for each item $j \in H$, it is associated a positive integer size a_j satisfying $a/4 < a_j < a/2$, and $\sum_{j=1}^{3h} a_j = ha$, for some integer a, do there exist h disjoint subsets $H_1, H_2, \ldots, H_h$ of H such that $|H_i| = 3$ and $\sum_{j \in H_i} a_j = a$ for $i = 1, 2, \ldots, h$?.

We shall construct an instance of $P2$ that can be transformed polynomially to a given instance of the 3-partition problem. To prove that the problem $P2$ is strongly NP-Hard, we construct an instance of the problem and prove that 3-Partition problem has a solution if and only if there is a feasible solution to the constructed instance

Theorem 4.1. *The problem P2 is NP-Hard in the strong sense.*

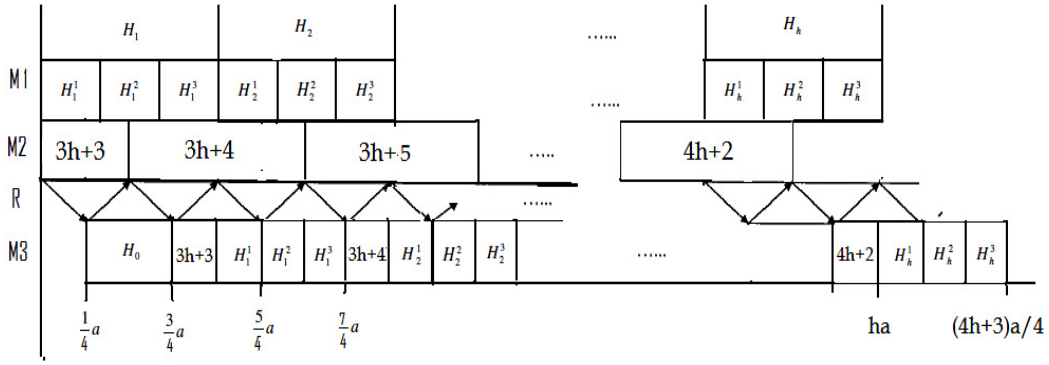

FIGURE 4. A schedule for instance of problem $TF3|\sigma = 2, p_j^2 = p_2, \forall j \in N, \upsilon = 1, c = 2|C_{\max}$.

Proof. We establish the theorem by a reduction from the 3-Partition problem. Given an arbitrary instance of 3-Partition problem, our scheduling problem $P2$ can be constructed as follows.

- Number of jobs $n = 4h + 2$ with $n_1 = 3h + 2$ and $n_2 = h$;
- Capacity of the robot $c = 2$;
- Transportation time $t = a/4$;
- Processing times of the jobs on the first and the second stage:
  - $p_j^1 = a_j$ for $j = 1, \ldots, 3h$, $p_{3h+1}^1 = p_{3h+2}^1 = 0$, $H_0 = \{3h + 1, 3h + 2\}$,
  - $p_j^1 = a$ for $j = 3h + 4, \ldots, 4h + 2$, and $p_{3h+3}^1 = a/2$,
  - $p_j^2 = a/4$ for $j = 1, \ldots, 4h + 2$.
- Threshold value $y = (4h + 3)a/4$.

We will show that there is a solution to the instance of 3-Partition problem if and only if a schedule for the constructed scheduling instance with a makespan value no greater than y exists.

$\longrightarrow$ If there is a solution to 3-Partition problem, we show that there is a schedule to the problem $P2$ with a makespan of no more than y. Given a solution to 3-Partition problem, $H_1, H_2, \ldots, H_h$, we construct a schedule for the problem $P2$ as shown in Figure 4. In this schedule, the first delivery trip carries two jobs $H_0 = \{3h+1, 3h+2\}$ in a first batch and departs at time 0. The second batch contains the first job of H_1 which is of type 1 and the job $3h + 3$ of type 2. However the third batch contains the last two jobs of H_1. In the same manner, each batch among the rest of batches contains exactly two jobs. The transporter starts transporting the $2h + 1$ batches, one by one at time $(\ell - 1)\frac{a}{2}$ such that ℓ is the index of the batch B_ℓ. It is clear that the above schedule is feasible and the makespan is $y = (4h + 3)a/4$.

$\longleftarrow$ Conversely, assume that there exists a schedule for the created instance of our problem with a makespan of no greater than y. We can obtain that there is precisely $2h + 1$ batches. Indeed, the minimal possible number of batches is $2h+1$. This is due to the limited capacity of the transporter ($c = 2$). If there exist more than (2h+1) batches in the schedule, we suppose that there are $2h + 2$ batches, thus the corresponding total transportation time of the transporter is greater than $\frac{a}{4} + (2h + 2)\frac{a}{2} = (4h + 3)\frac{a}{4}$. This is a contradiction because the jobs of the last batch should be processed on the common machine at the second stage and the makespan in this case will be more than y. From this, it follows that there exists exactly $2h + 1$ batches to be transported from the first stage to the second. Consider that B_1, B_2,...,B_{2h+1} the different batches and each batch B_ℓ for all $1 \leq \ell \leq 2h + 1$ contains precisely two jobs. Therefore, the two jobs $3h + 1$ and $3h + 2$ are transported in a first batch B_1 at time 0. Moreover, the earliest possible date for starting processing jobs on the common machine is

TABLE 3. Complexity results for the problem (P).

Problem characteristics	Problem complexity
$P1$: $p_1 \geq 2t$	Polynomial (Proposition. 3.2)
$P1$: $n_1 = n_2$ and $p_1 \leq t$	Polynomial (Theorem. 3.4)
$P2$	Strongly NP-Hard (Theorem. 4.1)

$\frac{a}{4} = y - (4h+2)\frac{a}{4}$. The corresponding processing time of all the jobs on the common machine at the second stage is $(4h+2)\frac{a}{4}$. This implies that there is no idle time on the common machine after the start of processing the first batch B_1 at time $\frac{a}{4}$.

Let the starting transporting times of the $2h+1$ batches by the transporter from the first stage to the second stage be denoted by d_1, d_2,...,d_{2h+1}, respectively. Denote the corresponding arrival times of these batches on the second stage by S_1, S_2,...,S_{2h+1}, respectively. Hence, $S_\ell = d_\ell + \frac{a}{4}$ for all $1 \leq \ell \leq 2h+1$ which implies that $S_1 = d_1 + \frac{a}{4} = \frac{a}{4}$. The transporter returns to the first stage at time $\frac{a}{2}$. Therefore, the second batch B_2 must be transported at time no later than $\frac{a}{2}$ and since $t = \frac{a}{4}$, then $S_2 = d_2 + \frac{a}{4} \leq \frac{3a}{4}$ and recall that $S_2 \geq S_1 + \frac{a}{2} = \frac{3a}{4}$, this means that $d_2 = \frac{a}{2}$ and in the same manner $d_3 = a$. Now, we prove that $\sum_{j\in H_k} p_j^1 = a$. First, we show that $\sum_{j\in H_1} p_j^1 = a$. If $\sum_{j\in H_1} p_j^1 > a$, then the instant of transporting the batch B_3 is greater than a ($d_3 > a$). Recall that the returned time of the transporter to the first stage is at time a, then as a result, there is an idle time after the batch B_2 on the transporter, which is a contradiction. By the same discussion, this indicates that the batch B_ℓ must be transported at time $(\ell-1)\frac{a}{2}$. If $\sum_{j\in H_1} p_j^1 < a$, then an idle time will be created on the first dedicated machine which is a contradiction. We obtain that $\sum_{j\in H_k} p_j^1 = a$. This implies the existence of a solution to 3-Partition problem. Combining the "if" part and the "only if" part, we have proved the theorem. □

From the previous theorem and in the case in which the transportation time is a variable t_j, we deduce the following corollary.

Corollary 4.2. *$TF3|\sigma = 2, p_j^2 = p_2, t_j \geq 0\ \forall j \in\ N, \upsilon = 1, c = 2|C_{\max}$ is strongly NP-hard.*

Complexity results on the problem $TF3|\sigma = 2, \upsilon = 1, c = 2|C_{\max}$ are summarized in Table 3.

5. Lower and upper bounds

In this section, we develop a heuristic approach GH to find approximate solutions for the $TF3|\sigma = 2, \upsilon = 1, c \geq 1|C_{\max}$ problem. We incorporate the transportation times in the processing times by using a procedure which can be regarded as an extension of the Panwalkar Algorithm [11]. The different steps of our heuristic are as follows. First, we modify the processing times of each job such that $q_i^1 = \max\{p_i^1, 2t\}$ and $q_i^2 = p_i^2$, $\forall i \in N$, then we create two sets U and V of jobs using the SPT (Shortest Processing Time) and LPT rules. We obtain the permutation sequence π by combining the two ordered sets of jobs U and V. For each job type, we built a batch with a maximum number of jobs such that the sum of their processing times is less than or equal to the time of around-trip $2t$ and their number is less than or equal to $\frac{c}{2}$. All these jobs will be transported in one batch. The rest of batches are constructed in the same manner. Finally, we use the FCFS (First Come First Served) rule.

We give three lower bounds for the problem $TF3|\sigma = 2, \upsilon = 1, c \geq 1|C_{\max}(N)$, which are proposed in [2]. The workload of a machine M_k corresponds to the sum of job processing times that the machine must carry and it is denoted by W_k.

$$W_1 = \sum_{i\in N_1} p_i^1, \quad W_2 = \sum_{i\in N_2} p_i^1, \quad W_3 = \sum_{i\in N} p_i^2.$$

Proposition 5.1. *The three following bounds are valid lower bounds:*

- $LB^1 = W_3 + \min\limits_{1\leq i\leq n}\{p_i^1\} + t.$

Algorithm 2. Heuristic GH.

Let $q_i^1 = \max\{p_i^1, 2t\}$ and $q_i^2 = p_i^2$, $\forall i \in N$.
Sort jobs in $U = \{i : q_i^1 \leq q_i^2\}$ in non-decreasing order such that $q_i^1 \leq q_{i+1}^1$;
Sort jobs in $V = \{i : q_i^1 > q_i^2\}$ in non-increasing order such that $q_i^2 \geq q_{i+1}^2$;
Let π be the sequence composed by the ordered set U followed by the ordered set V.
Process the jobs in $\pi^1 = \pi \cap N_1 = \{\pi_1^1, \ldots, \pi_{n_1}^1\}$ on the machine M_1.
Process the jobs in $\pi^2 = \pi \cap N_2 = \{\pi_1^2, \ldots, \pi_{n_2}^2\}$ on the machine M_2.
$B_1 = \{\pi_1^1, \pi_1^2\}$ is the first batch.
$i := 2$; $j := 2$; $\ell := 2$; $\pi^1 := \pi^1 - \{\pi_1^1\}$; $\pi^2 := \pi^2 - \{\pi_1^2\}$;
while $\pi^1 \neq \emptyset$ or $\pi^2 \neq \emptyset$ **do**
  $k_1 = \max\{\alpha \mid \alpha - i + 1 \leq \lfloor c/2 \rfloor$ *and* $\sum_{k=i}^{\alpha} p_{\pi_k^1}^1 \leq 2t\}$;
  $k_2 = \max\{\beta \mid \beta - j + 1 \leq \lfloor c/2 \rfloor$ *and* $\sum_{h=j}^{\beta} p_{\pi_h^2}^2 \leq 2t\}$;
  Set $B_\ell := \{\pi_i^1, \ldots, \pi_{k_1}^1\} \cup \{\pi_j^2, \ldots, \pi_{k_2}^2\}$ and $\ell := \ell + 1$;
  Set $\pi^1 := \pi^1 - \{\pi_i^1, \ldots, \pi_{k_1}^1\}$ and $i := k_1 + 1$;
  Set $\pi^2 := \pi^2 - \{\pi_j^2, \ldots, \pi_{k_2}^2\}$ and $j := k_2 + 1$;
end while
$B = \bigcup_{k=1}^{\ell-1} B_k$ (B is the set of batches).
Transport the batches in B from the first stage to the second stage to be processed on the common machine.
Process the jobs at the second stage according to the FCFS rule.
Calculate the corresponding makespan for this schedule.

- $LB^2 = \max\{W_1, W_2\} + \min\limits_{1 \leq i \leq n} \{p_i^2\} + t$.
- $LB^3 = 2t(\lceil \frac{n}{c} \rceil - 1) + t + \min\limits_{1 \leq i \leq n} \{p_i^2\} + \min\limits_{1 \leq i \leq n} \{p_i^1\}$.

Corollary 5.2. *$LB = \max\{LB^1, LB^2, LB^3\}$ is also a lower bound for the makespan.*

6. Computational experimentation

In this section, we present the results of our experiment designed to evaluate empirically the proposed heuristics. In the literature, to the best of our knowledge, there is no benchmarks for such problem. Therefore, we adopt generation schemes given in [12]. We have carried out computational experiments on a set of randomly generated instances from a discrete uniform distribution. These computational experiments have two goals. The first one is the evaluation of heuristic GH which is dedicated to solve the general problem, whereas the second one is the study of performance of heuristic H_p in term of solution quality and CPU time.

6.1. Computational experiments on the general problem

The MIP model proposed in [3] was solved using CPLEX 12.2, running on an Intel Core i7 PC, 2.9 GHz Processor and 6 GB of RAM within a time limit of 1 hour. For instances of reduced size, the processing times p_i^s ($\forall i \in N, s \in \{1, 2\}$) are generated from a discrete uniform distribution on $[1, 20]$. The values of number of jobs for each type are $n_1 = n_2 = 0.5 \times n$. For each combination of parameters (n, c, t), we randomly generated 10 instances. The results are given in Table 4, where:

- $TimeMIP$: mean CPU time of the MIP in seconds (s).
- $TimeGH$: mean CPU time of the heuristic GH in milliseconds (m.s).
- $\overline{Dev}\%$: mean percentage deviation for the 10 instances of the solution obtained by the heuristic with respect to the best-found solution BFS obtained by the MIP, $Dev\% = 100 \times \frac{C_{\max}(GH) - BFS}{BFS}$.

In the case of instances of largest size, four classes of instances are considered with different values of number of jobs $n \in \{30, 100, 500, 1000\}$ such that $n_1 = n_2 = 0.5 \times n$. For each value of n, 100 instances are generated using a uniform distribution as described in Table 5. For every problem instance, the makespan $C_{\max}(GH)$

TABLE 4. Computational experiments on instances of small size.

n	c	t	TimeMIP(s)	$\overline{\text{Dev}}$%	TimeGH(m.s)
10	1	1	38, 25	0, 31	3, 1
		5	32, 96	1, 52	3, 44
		10	2541, 31	3, 70	3, 1
	5	1	833, 42	2, 96	3, 1
		5	793, 06	3, 53	3, 1
		10	1394, 37	5, 65	3, 2
	10	1	1040, 36	2, 96	4, 6
		5	715, 39	3, 45	3, 1
		10	1269, 34	6, 55	4, 7
20	1	1	3600	0, 28	6.3
		5	3600	2, 35	7.8
		10	3600	1, 91	6.3
	5	1	3600	0, 63	7.8
		5	3600	2, 73	6.3
		10	3600	4, 99	6.4
	10	1	3600	0, 58	7.8
		5	3600	2, 73	7.7
		10	3600	4, 98	7.9

TABLE 5. Classes of instances.

Class	c	t	p_i^s
Cl_1	1	$DU[1, 10]$	$DU[1, 30]$
Cl_2	1	$DU[1, 10]$	$DU[1, 50]$
Cl_3	1	$DU[1, 10]$	$DU[1, 100]$
Cl_4	$DU[1, 10]$	$DU[1, 10]$	$DU[1, 100]$

obtained by the heuristic GH and the lower bound LB (see. Sect. 5), are computed. The relative error ratio can be defined as follows: error Ratio (ER)=$100 \times \frac{C_{\max}(GH)-LB}{LB}$.

From Table 4, in most cases the CPU time by the MIP model to solve any instance increases with the number of jobs. We see also from Table 4, that the proposed heuristic GH provides a good performance for solving small-size instances. Indeed, this heuristic gives, in most cases, good solutions in a reasonably short CPU time and has small percentage deviation. The average deviation of the heuristic GH over the 180 instances is 2.87%. In addition, this heuristic provides optimal solutions in most of the cases.

In Figures 5 and 6, we compare the lower bound LB and the value of $C_{\max}(H)$ obtained by the heuristic GH.

Table 6 shows the results of computational experiments for solving large-scale instances. Different parameters are used to evaluate the performance of the heuristic GH as follows. The average and maximum values of error ratios Avg.ER and Max.ER respectively The average and maximum required times for running the proposed heuristic denoted by Avg.Tim and Max.Tim respectively are calculated in (m.s). All the computational results of error ratio in Table 6 indicate that small difference exists between lower bound and heuristic makespan. Therefore, the heuristic can find near-optimal solution for small and large size instances in a very short CPU time.

6.2. Computational experiments on the problem ($P1$)

To asses the performance of the heuristic H_p. The following parameters are considered:

- *Number of jobs* n:$\{50, 100, 200, 500\}$, $(\eta = 0.5n)$
- *Transportation time* t: $U[1, 10]$ and $U[1, 100]$.
- *Processing times* p_1: $U[t, 2t]$.

TABLE 6. Computational results of error ratios.

		Heuristic *GH*			
Class	n	Avg.ER	Max.ER	Avg.Tim	Max.Tim
Cl_1	30	0.84	3.19	8, 43	16
	100	0.26	0.88	15, 94	32
	500	0.05	0.19	59, 22	63
	1000	0.04	0.09	118, 09	141
Cl_2	30	0.44	1.98	9, 35	16
	100	0.20	0.70	16, 73	32
	500	0.03	0.14	60, 16	78
	1000	0.02	0.07	125, 83	313
Cl_3	30	0.19	1.05	9, 07	16
	100	0.08	0.33	16, 87	31
	500	0.02	0.07	59, 68	63
	1000	0.01	0.03	115, 27	125
Cl_4	30	0.59	2.78	8, 4	16
	100	0.12	0.33	17, 34	31
	500	0.02	0.07	62, 17	78
	1000	0.01	0.03	124, 08	141

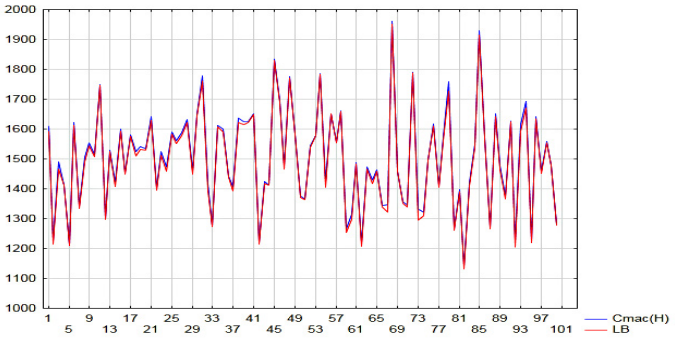

FIGURE 5. Comparison between the makespan of the heuristic *GH* and the lower bound LB (*Classs*: $Cl_4,\ n = 30$).

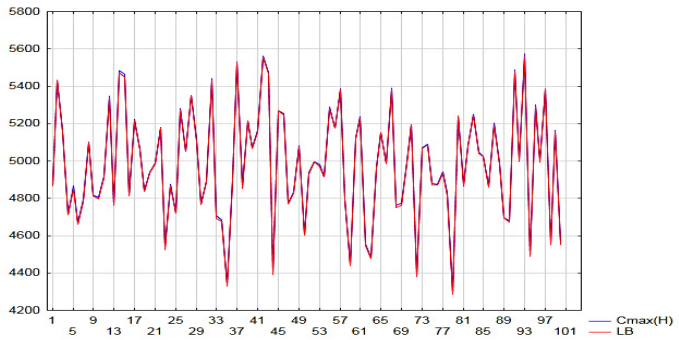

FIGURE 6. Comparison between the makespan of the heuristic *GH* and the lower bound LB (*Class*: $Cl_4,\ n = 100$).

TABLE 7. Computational results of Heuristic H_p (Case 1).

	t	$U[1,10]$			$U[1,100]$		
n	p_i^2	$[1,30]$	$[1,50]$	$[1,100]$	$[1,30]$	$[1,50]$	$[1,100]$
50	Avg%dev	0	0	0	0.061	0.063	0.057
	Max%dev	0	0	0	0.232	0.306	0.397
	Avg-CPU	0.002	0.003	0.002	0.002	0.003	0.003
100	Avg%dev	0	0	0	0.022	0.019	0.015
	Max%dev	0	0	0	0.077	0.071	0.087
	Avg-CPU	0.002	0.002	0.002	0.003	0.003	0.003
200	Avg%dev	0	0	0	0.01	0.009	0.004
	Max%dev	0	0	0	0.029	0.043	0.025
	Avg-CPU	0.002	0.003	0.003	0.003	0.003	0.003
500	Avg%dev	0.001	0.002	0	0.004	0.002	0.001
	Max%dev	0.002	0.002	0	0.012	0.007	0.004
	Avg-CPU	0.002	0.002	0.002	0.003	0.003	0.004

TABLE 8. Computational results of Heuristic H_p (Case 2).

	t	$U[1,\ 10]$			$U[1,\ 100]$		
	p_i^2 Type1	$[1,\ 30]$	$[1,\ 50]$	$[1,\ 100]$	$[1,\ 30]$	$[1,\ 50]$	$[1,\ 100]$
n	p_i^2 Type2	$[50,\ 100]$	$[50,\ 100]$	$[50,\ 100]$	$[50,\ 100]$	$[50,\ 100]$	$[50,\ 100]$
50	Avg%dev	0	0	0	0,087	0,0931	0,0843
	Max%dev	0	0	0	1,700	1,6993	1,3673
	Avg-CPU	0,313	0,188	0,188	1,016	0,328	0,344
100	Avg%dev	0	0	0	0,05	0,05	0,04
	Max%dev	0	0	0	1,05	0,84	0,70
	Avg-CPU	0,266	0,281	0,406	0,266	0,266	0,406
200	Avg%dev	0	0	0	0,01	0,01	0,01
	Max%dev	0	0	0	0,42	0,42	0,27
	Avg-CPU	0,438	0,531	0,438	0,438	0,531	0,531
500	Avg%dev	0.001	0.002	0	0,001	0,001	0,001
	Max%dev	0.002	0.002	0	0,004	0,006	0,006
	Avg-CPU	1,141	1,266	1,891	1,125	1,250	1,234

- *Processing times* p_i^2: 2 Categories "Case 1" and "Case 2":
  → **"Case 1"** Both types:$U[1,30]$, $U[1,50]$ and $U[1,100]$.
  → **"Case 2"** $\begin{cases} Type1 : \text{U[1, 30], U[1, 50] } and \text{ U[1, 100].} \\ Type2 : \text{U[50, 100].} \end{cases}$

For each combination of number of jobs, we randomly generate 100 problem instances and for every problem instance, the heuristic makespan, denoted by $C_{\max}(H_p)$, and the lower bound of makespan, denoted by LB, are computed. The relative error ratio can be defined as: $\%\text{dev} = 100 \times \frac{C_{\max}(H_p) - LB}{LB}$. The average and maximum values of error ratios for each combination denoted by Avg%dev and Max%dev, respectively are used to evaluate the performance of the proposed heuristic. The average and maximum required CPU time to run the heuristic H_p are computed (in seconds).

From the computational results of error ratio in Tables 7 and 8, we can observe that in the case of small transportation time $t \in [1,10]$, the heuristic H_p reached the optimal solution for the most of problems. In the case of transportation times $t \in [1,100]$, very small difference exists between lower bound and makespan obtained

by the heuristic H_p which shows the performance of heuristic H_p especially in the case of small transportation time of the robot.

7. Conclusion

In this paper, we investigated a scheduling problem in a robotic cell of type flow shop. The objective is to find a joint production and transportation schedule to minimize the makespan. We focus on the case where the transporter's capacity is equal to two and we establish new complexity results related to this problem. Indeed, we prove the NP-hardness of the problem in the case of identical processing time on second stage. A fast heuristic is developed to tackle the general problem which is NP-Hard. The extensive experiments indicate the performance of the proposed heuristic in term of solution quality and CPU time.

Two directions are suggested for further extensions of our research. First, it would be interesting to consider the reverse model that has a common machine in stage one and two dedicated machines in stage two. Second, to study more general cases of the problem, such as the inclusion of different objective functions.

References

[1] F. Ahmadizar and P. Shahmaleki, Group shop scheduling with sequence-dependent setup and transportation times. *Appl. Math. Model.* **38** (2014) 5080–5091.

[2] N. Chikhi, M. Abbas, A. Bekrar, R. Benmansour and S. Hanafi, On the complexity of robotic flow shop with transportation constraints. In *ROADEF – 15ème congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision*, Bordeaux, France (2014).

[3] N. Chikhi, R. Benmansour, A. Bekrar, S. Hanafi and M. Abbas, A case study of a two-stage flow shop with dedicated machines and a single robot. In *Control, Decision and Information Technologies (CoDIT)* (2014).

[4] A. Elmi and S. Topaloglu, A scheduling problem in blocking hybrid flow shop robotic cells with multiple robots. *Comput. Oper. Res.* **40** (2013) 2543–2555.

[5] R.L. Graham, E.L. Lawler, J.K. Lenstra and A. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.* **5** (1979) 287–326.

[6] J. Hurink and S. Knust, Makespan minimization for flow-shop problems with transportation times and a single robot. *Discrete Appl. Math.* **112** (2001) 199–216.

[7] H. Kise, On an automated two-machine flowshop scheduling problem with infinite buffer. *J. Oper. Res. Soc. Jpn.* **34** (1991) 354–361.

[8] C.Y. Lee and Z.L. Chen, Machine scheduling with transportation considerations. *J. Sched.* **4** (2001) 24.

[9] E. Levner, K. Kogan and I. Levin, Scheduling a two-machine robotic cell: A solvable case. *Ann. Oper. Res.* **57** (1995) 217–232.

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

[11] S. Panwalkar, Scheduling of a two-machine flowshop with travel time between machines. *J. Oper. Res. Soc.* (1991) 609–613.

[12] L. Tang and P. Liu, Two-machine flowshop scheduling problems involving a batching machine with transportation or deterioration consideration. *Appl. Math. Modelling* **33** (2009) 1187–1199.

[13] F.S. Yao, M. Zhao and H. Zhang, Two-stage hybrid flow shop scheduling with dynamic job arrivals. *Comput. Oper. Res.* **39** (2012) 1701–1712 .

[14] W.-Y. Zhong and L.-H. Lv, Hybrid flowshop scheduling with interstage job transportation. *J. Oper. Res. Soc. China* **2** (2014) 109–121.