Flow shop Problem with transportation considerations

Nacira Chikhi and Mourad Boudhar

Faculty of Mathematics, USTHB University,
BP 32 Bab-Ezzouar, El-Alia 16111, Algiers , Algeria
nacira_chikhi@hotmail.fr
mboudhar@yahoo.fr
http://www.usthb.dz

Abstract. Motivated by applications in manufacturing systems. This paper deals with a scheduling problem of independent tasks with additional constraints (transportation times), where the objective is to minimize the total completion time. This problem arises in automated cells and is a complex flow shop problem with a transportation robot or conveyor. Since the problem is NP-hard, heuristics are developed to give near optimal solutions. Two new programming algorithms are also proposed for solving some special cases of this problem. Finally, we evaluate the proposed heuristics, giving experimental results on randomly generated test problems.

Key words: Scheduling, flow shop, tranport, makespan, heuristics.

1 Introduction

In most manufacturing systems, semi-finished jobs are transferred from one facility to another for further processing through material handling systems such as automated guided vehicles (AGVs) and conveyors. In the last four decades, many books and numerous papers have been published in the area of machine scheduling. However, most of the published literature explicitly or implicitly assumes that either there is an infinite number of transporters or that jobs are transported instantaneously from one location to another without transportation time involved.

These displacements were not therefore taken in account at the time of the construction of the scheduling. However this assumption is often not justified in practice, there are many situations in which it must not be abandoned as being unrealistic. For example, in computer systems the output of a job on one processor may require a communication time so as to become the input to a succeeding job on another processor and in manufacturing systems , there may be a transportation time from one production facility to another. This model can be illustrated in the case of robotic cells that are found in manufacturing systems of semiconductors or textiles, in which an automated guided vehicle is charged to displace jobs. It can also be illustrated by the example of a workshop for

electroplating whose process consists of coating a part by a thin layer of metal on pieces. The displacement of pieces is done mainly by a transporter(hoist) moving horizontally on a rail as it is shown in Figure 1.

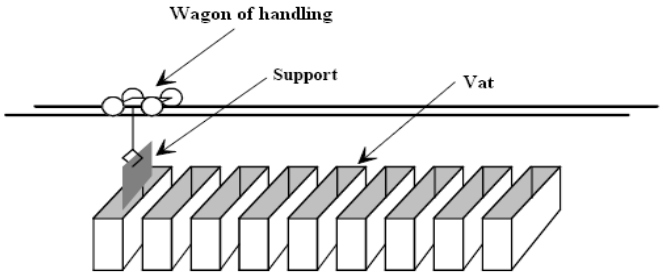


Fig. 1. Industrial application

In this paper, our problem can be defined as follows. We are given a set J of n independent jobs $J = \{J_1, ..., J_n\}$ to be processed on 2 machines M_1 and M_2 in a flow shop with unlimited buffer spaces on both machines. Each job must first be processed on machine M_1, then machine M_2.

The processing time of a job J_i on machine M_k is p_{ik}. We assume that all of the jobs start at machine M_1. Once a job is processed on machine M_1, it is transported to machine M_2 by a transporter. The transporter is initially located at machine M_1. It has a capacity of c, i.e. it can carry up to c jobs in one shipment.

The transportation time from machine M_1 to machine M_2 is denoted by t (the round-trip requires 2t). We assume that the loading and the unloading times are included in the processing times of jobs and are not considered separately. Our goal is to schedule the jobs so as to minimize the makespan .

We follow the commonly used three-field notation $\alpha/\beta/\gamma$ for machine scheduling problems. In the α field, we use notation "TF" to denote a flow-shop problem with transportation between machines. Hence, $TF2/\upsilon = x, c = y/C_{max}$, represents the 2-machine flow shop problem with x transporters, each with capacity y. So our problem thus defined is denoted $TF2/\upsilon = 1, c \geq 1/C_{max}$

Chen and Lee [1] studied a two-machine flow-shop problem with several conveyors of any capacity noted $TF2/\upsilon \geq 1, c \geq 1/C_{max}$. They gave a dynamic algorithm that solves in polynomial time the problem $TF2/p_{i1} = p, \upsilon \geq 1, c \geq 1/C_{max}$. In our work, we limit to only one conveyor and we give two polynomial algorithms for the solution of two particular cases of the general problem $TF2/\upsilon = 1, c \geq 1/C_{max}$. We also propose heuristics to solve the general problem which is NP-hard[1].

This article is organized as follows: the second section is devoted to the mathematical formulation, a mixed integer linear programming model is proposed to determine the schedule with minimum makespan. This model has been tested using CPLEX solver. Section 3 is dedicated to the calculation of lower bounds and the presentation of some subproblems that can be solved polynomially. As for the fourth section some heuristics are presented for the solution of the general problem and some numerical tests are carried out to show the performance and the efficiency of the different heuristics in the last section. Finally, we provide a conclusion at the end of this article.

2 Problem transformation

The following notation is used for the mathematical representation of the general problem.
d_{i1}: is the starting time of the execution of the first operation of the job i on the machine M_1. d_{is}: is the starting time of the transport of the job i and d_{i2}: is the starting time of the execution of the second operation of the job i on the machine M_2
For every couple (i, j) of jobs, we introduce the following binary variable: a_{ij} equal to 1 if $d_{i1} < d_{j1}$, and 0 otherwise. b_{ij} equal to 1 if $d_{i2} < d_{j2}$, and 0 otherwise. α_{ij} equal to 1 if $d_{is} < d_{js}$, and 0 otherwise.
The objective function is to minimize C_{max};
subject to:

$$
\begin{array}{lll}
a_{ij} + a_{ji} = 1 & \forall\ i,j = \overline{1,n}\ ;\ i<j\ \ and\ \ i \neq j & (1)\\
d_{i1} + p_{i1} - d_{j1} \leq (1 - a_{ij}).M & \forall\ i,j = \overline{1,n}\ \ and\ \ i \neq j & (2)\\
d_{is} \geq d_{i1} + p_{i1} & \forall\ i = \overline{1,n} & (3)\\
\alpha_{ij} + \alpha_{ji} \leq 1 & \forall\ i,j = \overline{1,n}\ ;\ i<j\ \ and\ \ i \neq j & (4)\\
d_{js} - d_{is} \geq 2t * \alpha_{ij} - \alpha_{ji} * M & \forall\ i,j = \overline{1,n}\ \ and\ \ i \neq j & (5)\\
d_{is} - d_{js} \geq 2t * \alpha_{ji} - \alpha_{ij} * M & \forall\ i,j = \overline{1,n}\ \ and\ \ i \neq j & (6)\\
\sum_{j=1,i\neq j}^{n} (1 - \alpha_{ij} - \alpha_{ji}) \leq c - 1 & \forall\ i = \overline{1,n} & (7)\\
d_{i2} \geq d_{is} + t & \forall\ i = \overline{1,n} & (8)\\
b_{ij} + b_{ji} = 1 & \forall\ i,j = \overline{1,n}\ ;\ i<j\ \ and\ \ i \neq j & (9)\\
d_{i2} + p_{i2} - d_{j2} \leq (1 - b_{ij}).M & \forall\ i,j = \overline{1,n}\ \ and\ \ i \neq j & (10)\\
d_{i2} + p_{i2} \ \leq \ C_{max} & \forall\ i = \overline{1,n} & (11)\\
a_{ij},\ \ b_{ij},\ \ \alpha_{ij} \in \{0,1\} & \forall\ i,j = \overline{1,n} & (12)\\
d_{i1},\ \ d_{is},\ \ d_{i2} \in \mathbf{N} & \forall\ i = \overline{1,n} & (13)
\end{array}
$$

Where M is a very large value

Constraints (1), (2) and (3) concern the first machine: Constraints (1) mean that for any two jobs J_i and J_j, either J_i precedes J_j on the first machine, or J_j precedes J_i. Constraints (2) require that the first machine executes only one job at a time and (3) assure that a job can not be transported from the first machine to the second machine, once the first operation of this job is finished.

Constraints (4), (5), (6) and (7) are constraints on the conveyor (vehicle) and on the jobs to transport: Constraints (4) express that all jobs must be

transported between the two machines. Constraints (5) and (6) indicate that any job J_i is transported from the first machine to the second machine either before or after another job J_j, or at the same time and show that the transport time of a round-trip of the vehicle requires 2t. The constraints (7) express that the number of transported jobs at any time must be smaller than the vehicle capacity.

Constraints (8), (9) and (10) concern the second machine. Constraints (8) induce that the execution of the second operation of a job can only begin once the job has arrived to the second machine. Constraints (9) express that all jobs must be executed by the second machine. Constraints (10) assure that the second machine executes only one job at a time. The constraints (11) imply that the end of execution of any job is lower or equal to the makespan. Constraints (12) and (13) indicate the type of variables.

The number of variables and the number of constraints of a mathematical model are indications that measure its dimension and the efficiency of the modeling. The number of variables of our model is $3n^2$ and the number of constraints is $n(17n-1)/2$.

From this formulation, we can derive a lower bound by relaxing the constraints (12) and (13). The relaxed problem can be solved using a linear programming solver (CPLEX for example). The inconvenience of this technique consists in a large number of constraints ($n(23n-1)/2$ constraints).

3 Testing of the model with CPLEX

Table 1. Results obtained by the Cplex solver.

n	c	t	Pbms	avr-time
5	2	1	20	0.183
5	2	5	20	0.4725
5	3	1	20	0.715
5	3	5	20	0.6665
6	2	1	20	0.185
6	2	5	20	103.02
6	3	1	20	0.1815
6	3	5	20	0.1807
7	2	1	20	1.03
7	2	5	20	106.755
7	3	1	20	2.064
7	3	5	20	5.459
8	2	1	20	18.489
8	2	5	20	105.65
8	3	1	20	15.363
8	3	5	20	170.12
9	2	1	13	189.32
9	2	5	8	200.56
9	3	1	10	1030.245
9	3	5	6	186.342
10	2	1	5	1045.32
10	2	5	4	1230.458
10	3	1	4	7456.23
10	3	5	3	5131.47
50	20	1	1	6597,8
50	20	5	1	12588,78
70	30	1	1	14265.236
70	30	5	1	24698.24

The linear models with integer and binary variables can be solved by efficient solvers such as LINGO, CPLEX, etc. Our mathematical model has been tested

on a Pentium IV 3.06 GHz Personal Computer with 512 Mo RAM using Cplex Solver. The processing times p_{i1} and p_{i2} are generated by a uniform law in $[1, 100]$. We fixed the number of jobs and let the vehicle capacity and the transportation time vary. For every case, 20 problems are solved and the average execution time (for which the optimal solution is obtained) is computed in seconds. The results are given in the Table 1.

Computational experiments show that the largest problem that can be solved within at least 18 minutes is a two-machine and nine-jobs problem and the largest problem that can be solved within at least 7 hours is a two-machine and seventy-jobs problem.

4 Study of bounds and some subproblems

We proposed two lower bounds LB_1 and LB_2 for the objective function :

- $LB_1 = \max\{(\lceil \frac{n}{c} \rceil - 1) * 2t + t + \min_{1 \leq i \leq n} \{p_{i1}\} + \min_{1 \leq i \leq n} \{p_{i2}\}, \max_{1 \leq i \leq n} \{p_{i1} + p_{i2}\} + t\}$.
- $LB_2 = \max\{\sum_{1 \leq i \leq n} p_{i,1} + \min_{1 \leq i \leq n} \{p_{i2}\} + t, \sum_{1 \leq i \leq n} p_{i2} + \min_{1 \leq i \leq n} \{p_{i1}\} + t\}$.

We studied some subproblems of the general problem $TF2/\upsilon = 1, c \geq 1/C_{max}$. We mention especially the following cases:

Case 1: $p_{i1} \geq 2t, p_{i2} \geq \max_{1 \leq i \leq n} \{p_{i1}\}$

Algorithm 1

Begin

1: Arrange and process jobs in the increasing order (SPT rule) in relation to p_{i1}
2: (In every batch, we have only one job).

End

Theorem 1. *The algorithm 1 resolve the two problems TF2/*$p_{i1} \geq 2t, p_{i2} \geq \max_{1 \leq i \leq n} \{p_{i1}\}, \upsilon = 1, c \geq 1/C_{max}$ *and* $TF2/t \leq p_{i1} \leq \frac{3}{2}t, p_{i2} \geq 2t, \upsilon = 1, c \geq 1/C_{max}$ *in* $O(n \log n)$.

Case 2: $p_{i1} \leq \frac{2t}{c}, p_{i2} \geq 2t$

Algorithm 2

Begin

1: Find a job J_j of J having the minimum execution time on the first machine M_1.
2: Process the job J_j in first position and transport it alone in a first batch.
3: $J := J \setminus \{J_j\}$.
4: Arrange the jobs of J in the decreasing order (LPT rule) in relation to p_{i2} and process them in this order after the first job .

End

Theorem 2. *The algorithm 2 gives an optimal solution for the two problems* $TF2/p_{i1} \leq \frac{2t}{c}, p_{i2} \geq 2t, \upsilon = 1, c \geq 1/C_{max}$ *and* $TF2/p_{i1} \leq \frac{2t}{c}, p_{i2} \geq \frac{2t}{c}, p_{min2} \geq 2t, \upsilon = 1, c \geq 1/C_{max}$ *in* $O(n \log n)$.

Case 3: The problem $TF2/p_{i1} = p, \upsilon = 1, c \geq 1/C_{max}$:
When execution times on the first machine are identical and execution times on the second machine are any, the problem $TF2/p_{i1} = p, \upsilon = 1, c \geq 1/C_{max}$ is polynomial and can be solved by the dynamic algorithm of Chen and Lee [1]. The problem $TF2/p_{i2} = p, \upsilon = 1, c \geq 1/C_{max}$ is also polynomially solvable by the dynamic algorithm of Chen and Lee

5 Heuristics

Recall that in general the problem $TF2/\upsilon = 1, c \geq 1/C_{max}$ is NP-hard, so we propose some heuristics for its solution. We have used several rules of priority, based on the notion of priority between jobs to process. Their main advantages are, in general, their simplicity and especially their speed. Five rules have been applied therefore for the scheduling of jobs on the two machines. The two rules R_1 and R_2 are based on the coupling of jobs. The third rule of investment, is based on the algorithm of Jonhson [**?**]. We improved it in order to take into account the transportation times. Finally, we use the two rules SPT (Shortest Processing Time) and LPT (Longest Processing Time) that are based on the arranging of jobs.

These heuristics are also based on the following procedure that allows the construction of different batches. The principle of this procedure is to choose a maximal set of jobs that follows the job J_i (according to the initial order) so as the sum of the these execution times in this set, on the first machine is lower or equal to the time of the round-trip of the vehicle 2t plus a small amount of time (τ). Once this set of jobs is found, these jobs will be transported with the job J_i in one batch. On the other hand, if such a set doesn't exist, the job J_i will be transported alone in a batch.

Procedure of contruction of batches
Begin

1: $i := 1$, $\ell := 1$;
2: **while** $i < n$ **do**
3: **if** $p_{i+1,1} \geq 2t$ **then**
4: $B_\ell := \{J_i\}$, $\ell := \ell + 1$, $i := i + 1$.
5: (B_l represents the batch Number l);
6: **if** i=n **then**
7: $B_\ell := \{J_n\}$, $\ell := \ell + 1$, $i := i + 1$;
8: **end if**
9: **else**
10: Find $J_{i+1}, ..., J_k$ in J as:
11: $\sum_{j=i+1}^{k} p_{j,1} \leq 2t + \tau$ and $k - i + 1 \leq c$
12: $B_\ell := \{J_i, J_{i+1}, ..., J_k\}$, $\ell \leftarrow \ell + 1$, $i := k + 1$;
13: **if** i=n **then**
14: $B_\ell := \{J_n\}$, $\ell := \ell + 1$, $i := i + 1$;
15: **end if**
16: **end if**

17: **end while**
18: $L := \ell - 1$, $d_1 := \sum_{j \in Bch_1} p_{j,1}$, $r_1 := \sum_{j \in Bch_1} p_{j,1}$
19: $s := \sum_{j \in Bch_1} p_{j,1}$, $c_1 := \sum_{jinBch_1} p_{j,1} + \sum_{j \in Bch_1} p_{j,2} + t$
20: **for** $k = 2$to L **do**
21: $r_k := s + \sum_{j \in Bch_k} p_{j,1}$
22: $d_k := max\{r_k, d_{k-1} + 2t\}$
23: $s := r_k$
24: $c_k := max\{d_k + t, c_{k-1}\} + \sum_{j \in Bch_k} p_{j,2}$,
25: **end for**
26: $C_{max} := c_L$.

End

The first heuristic named H_1 is based on a new rule R_1. This rule forms pairs (J_k, J_j) such that the job J_k have the shortest execution times on M_1 and the job J_j have the longest execution times on M_2. We build a sequence of jobs reassembling all the couples. Finally, we construct the set of batches.

Algorithm H_1

Begin

1: **while** The list of jobs J is not empty **do**
2: Find a job J_k having the shortest execution time on the first machine.
3: $J := J \setminus \{J_k\}$.
4: Find a job J_j having the longest execution time on the second machine
5: $J := J \setminus \{J_j\}$
6: **end while**
7: Apply the previous procedure of batch construction according to the order of jobs determined by the previous loop.

End

Another version of the heuristic H_1 is denoted H_2. It has the same principle as H_1 except that it is based on another new rule named R_2 which forms couples (J_k, J_j) in which the jobs J_k have the longest time of execution on the second machine M_2 and the jobs J_j have the shortest execution times on first machine M_1. Once the pairs are created, we arrange them in the same order and we form a sequence of jobs. Finally, we apply the procedure of the construction of batches on the sequence of jobs obtained.

Algorithm H_2

Begin

1: **while** The list of jobs J is no empty **do**
2: Find a job J_k having the longest execution time on the second machine;
3: $J := J \setminus \{J_k\}$;
4: Find a job J_j having the shortest execution time on the first machine;
5: $J := J \setminus \{J_j\}$;
6: **end while**
7: Apply the procedure of construction of batches according to the order of jobs determined by the previous loop.

End

We propose another heuristic H_3 based on the LPT rule.

Algorithm H_3
Begin

1: Find a job J_k having the shortest execution time on the first machine;
2: Process the job J_k in first position and transport it alone in a first batch.
3: $J := J \setminus \{J_k\}$;
4: Arrange the remaining jobs of T in the decreasing order relative to the execution times on the second machine and process them after the first job.
5: Apply the procedure of construction of batches on the jobs in the order as in J.

End

The fourth heuristic that we propose named H_4 is based on the LPT rule, that consists to arrange jobs in the decreasing order relative to the execution times. In our case, we apply this rule for the job execution times on the machine M_2.

Algorithm H_4
Begin

1: Arrange jobs according to the decreasing order relative to the execution times on the second machine.
2: Apply the procedure of construction of batches.

End

Algorithm H_5
Begin

1: Build a two machines pseudo problem with execution times on the first machine $p'_{i1} = \max(p_{i1}, 2t)$ and $p'_{i2} = p_{i2}$ on the second machine (p_{i1} and p_{i2} are the execution times of the initial flow-shop).
2: Apply the algorithm of Johnson to this pseudo problem to get an ordering of jobs
3: Apply the procedure of construction of batches according to this order.

End

6 Tests according to the uniform law

Until now, there is no method in the literature which treat precisely the problem $TF2/\upsilon = 1, c \geq 1/C_{max}$. So we can not make a comparison with the heuristics that we propose.

However, we have tested the developed methods by using several instances generated randomly according to the uniform law. We have coded our algorithms in Delphi 7 and have run them on a Pentium IV 3.06 GHz Personal Computer with 512 Mo RAM.

We generated 100 instances for each number of jobs and we applied the heuristics cited above on these instances. Some results obtained for the uniform law are summarized in table 2 which follows, where we give the percentage with the best completion time where the solution found by the heuristic is better as compared to the other solutions, the percentage where the makespan is equal to

the lower bound and the average of performance ratio of the heuristics and the average execution time of each heuristic (in milliseconds).

For the first case (a), we suppose that the job execution times on the two machines as well as the vehicle capacity follow a uniform distribution in $[1, 10]$ and the transportation times follow a uniform law in the interval $[1, 100]$.

In the second case (b), we suppose that the job execution times on the two machines follow a uniform law in the interval $[1, 50]$ and the transportation time as well as the vehicle capacity have a uniform distribution in $[1, 10]$. The obtained results are represented in the Table 2.

Table 2. Summary of the tests.

		(a) $pi1, pi2, c \in \overline{1,10}$, $t \in \overline{1,100}$					**(b)** $pi1, pi2 \in \overline{1,50}$, $t, c \in \overline{1,10}$				
	$\tau = 0$	H_1	H_2	H_3	H_4	H_5	H_1	H_2	H_3	H_4	H_5
n=10:	pC_{max}	28%	24%	59%	15%	5%	36%	19%	76%	39%	15%
	Opt	7%	1%	18%	3%	2%	30%	16%	62%	36%	7%
	av-tim	4,71	3,44	6,37	3,71	5.28	6,46	4,35	6,12	3,74	8.01
	av-Rat	1,27	1,25	1,39	1,35	1,29	1,021	1,064	1,016	1,041	1,058
	Mx-Rat	1,89	1,96	2,32	2,15	2.44	1,133	1,215	1,155	1,206	1,188
n=50:	pC_{max}	21%	9%	74%	4%	0%	35%	17%	74%	39%	7%
	Opt	7%	1%	10%	4%	0%	30%	10%	66%	37%	4%
	av-tim	10,02	9,39	12,51	8,43	12.72	11,24	8,6	14,01	8,6	13,45
	av-Rat	1,070	1,077	1,074	1,078	1.099	1,006	1,009	1,004	1,012	1,014
	Mx-Rat	1,208	1,227	1,189	1,309	1.309	1,037	1,047	1,032	1,037	1,034
n=100:	pC_{max}	37%	13%	44%	2%	1%	31%	19%	72%	37%	1%
	Opt	9%	0%	10%	0%	0%	20%	12%	59%	33%	1%
	av-tim	19,04	15,79	21,2	15,81	21.09	18.09	15.8	21.55	15.58	21.78
	av-Rat	1,026	1,023	1,035	1,026	1.033	1.002	1.006	1.002	1.006	1.007
	Mx-Rat	1,094	1,107	1,086	1,099	1.158	1,017	1,019	1,014	1,021	1.024
n=1000:	pC_{max}	54%	30%	32%	9%	0%	36%	31%	62%	40%	5%
	Opt	6%	1%	5%	0%	0%	32%	27%	55%	39%	1%
	av-tim	152,73	138,91	174,2	140,17	169.03	158,9	146,2	180,3	146,1	179.9
	av-Rat	1,003	1,004	1,004	1,002	1.004	1,000	1,000	1,000	1,000	1.001
	Mx-Rat	1,052	1,076	1,026	1,006	1.033	1,001	1,001	1,001	1,001	1.001

We define the calculated parameters:

pCmax: is the percentage for which the heuristic H provides a better solution than the other heuristics.

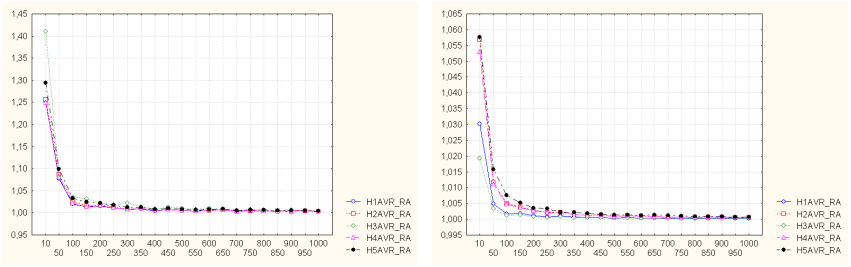
opt: is the percentage for which the solution obtained by the heuristic H coincides with the lower bound LB.

Ratio(H): is the performance ratio of the heuristic H, $Ratio(H) = \frac{Sol(H)}{LB}$.

av-Rat: is the average performance ratio, $average - Ratio(H) = \frac{\sum_{k=1}^{100} Ratio_k(H)}{100}$, k is the number of the instance.

mx-Rat: the maximum of the performance ratio.
av-tim: the average of the execution time.
Dev(H): is the deviation of the heuristic H, Dev(H)=$\frac{Sol(H)-LB}{LB}$.
avr-Dev(H): is the average deviation of the heuristic H, $avr - Dev(H) = \frac{\sum_{k=1}^{100} Dev_k(H)}{100}$.

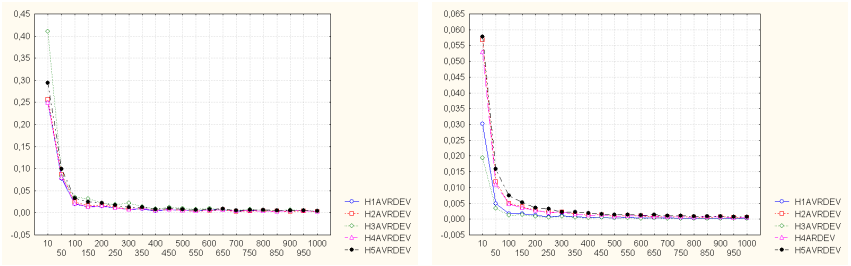
For the different heuristics, we established the average of the performance ratio by applying the heuristic on instances randomly built according to a uniform law. The results are illustrated in Figures 2 and 3.



(a) $pi1, pi2, c \in \overline{1,10}$, $t \in \overline{1,100}$ **(b)** $pi1, pi2 \in \overline{1,50}$, $t, c \in \overline{1,10}$

Fig. 2. Average performance ratio of the heuristics according to "n"

With regard to the average deviations, the obtained graphs are shown in Figure 9.



(a) $pi1, pi2, c \in \overline{1,10}$, $t \in \overline{1,100}$ **(b)** $pi1, pi2 \in \overline{1,50}$, $t, c \in \overline{1,10}$

Fig. 3. Average deviations of the heuristics according to "n"

We generated 100 instances for each number of jobs and we applied the heuristics on these instances ($pi1, pi2 \in [1,50]$, $t, c \in [1,10]$) in the cases $\tau = 0$ and $\tau = 1$. The results are given in Table 3.

Table 3. Comparison between the cases $\tau = 0$ and $\tau = 0$.

(a) $\tau = 0$

	$\tau = 0$	H_1	H_2	H_3	H_4	H_5
n=10:	av-Rat	1,021	1,056	1,016	1,041	1,055
	av-tim	1,09	0,47	1.26	0.92	1.72
n=1000:	av-Rat	1,000	1,000	1,000	1,000	1,001
	av-tim	61,04	60,89	87,47	43,33	86,23

(b) $\tau = 1$

$\tau = 1$	H_1	H_2	H_3	H_4	H_5
n=10	1.030	1,059	1,019	1,055	1,059
	1.25	1,1	1,53	0,16	1.39
n=1000	1,000	1,001	1,000	1,000	1,001
	64,71	59,91	85,16	42,85	83,94

We note that when $\tau = 0$, the results are better than in the case $\tau > 0$.

Table 4. Comparison with optimal solutions

n	c	t	Opt	time(s)	H_3C_{max}	time(m.s)	bC_{max}	$R(H_3)$
5	3	5	29	2,14	33	16	32	0.137
			38	0,59	44	0	39	0.157
			31	0,41	39	0	31	0.258
			31	0,03	32	16	32	0.032
			42	0,05	44	16	44	0.047
			38	0,53	39	15	39	0.026
			30	2,19	34	16	30	0.133
			32	0,11	42	16	33	0.312
			34	0,05	40	0	35	0.176
			39	1,94	46	15	41	.0179
10	3	1	55	0,05	57	0	55	0.036
			58	0,03	58	0	58	0
			53	0,02	55	0	53	0.037
			64	0,03	64	0	64	0
			52	0,03	57	16	53	0.09
			59	0,05	59	0	59	0
			49	0,03	49	0	49	0
			59	0,05	59	15	59	0
			65	0,05	65	31	65	0
			54	0,03	54	62	54	0
			64	0,06	64	16	64	0

For the different methods developed, there are not any precise conditions so that a method is better than another one. It depends on the number of jobs and the transportation time. However, we have compared the best solutions generated by all the heuristics denoted bC_{max} and the solutions generated by the heuristic H_3 denoted H_3C_{max}, which we claim to be better than the others according to the results of the preceding tests, with the exact solutions found by the Cplex software. For this, we have randomly generated some instances of reduced sizes $n \in \{5, 10\}$. Indeed, the processing times p_{i1} and p_{i2} are generated

by a uniform law in $[1, 10]$. To measure the efficiency of the heuristic H_3, we calculate the relative distance between the solution given by the heuristic H_3 and the optimal solution as follows: $R(H) = \frac{Cmax(H)-Opt}{opt}$. Some results of this experimentation are given in the Table 4 with the CPU time in seconds (Time) of the optimal solution. The average CPU time for the heuristic H_3 is smaller than 0.002 seconds. Table 4 clearly shows the efficiency of the heuristic H_3.

In general, results obtained for the different tests reveal that the heuristics based on the LPT rule generally give very good solutions that are optimal in most cases.

7 Conclusion

We studied the flow-shop problem with two machines connected by a conveyor. The performance criteria chosen is the total execution time (makespan). We introduced a transport system of jobs: robot or vehicle. We presented and modeled our problem as a linear program in integer and binary variables. We also proposed lower bounds that are going to serve like reference to appraise the quality of solutions obtained by the developed methods. Some subproblems of the general problem are analyzed and solved in polynomial time. Having shown that the general problem is NP-Hard, we developed the heuristics. Tests have been carried on several instances randomly generated in order to study the performance of the different proposed heuristics.

Research in this field remains open. The introduction of conveyors to the flow-shop problem brings us to conceive other models that are related to the characteristic of storage spaces and the number of conveyors. We may also consider to develop the meta-heuristic or exact methods.

References

1. ZL. Chen and CY. Lee. Machine scheduling with transportation considerations. Journal of scheduling, 4 :3-24, (2001).
2. N. Chikhi. Two machine Flow-shop with transportation time. Thesis of magister. Faculty of Mathematics, USTHB University,Algiers, (2008).
3. J. Hurink and S. Kunst. Flow-shop problems with transportation times and a single robot. Universit Osnabrack, (1998).
4. S. S. Panwalker. Scheduling of a two machine flow shop with travel time between machines. J.Opl.Res.Soc, 42, No 7: 609-613, (1991).
5. H. Kise. On an automated two-machine flowshop scheduling problem with infinite buffer. Journal of the Operations Research Society of Japan, 34 : 354-361, (1991).
6. M. Pinedo. Scheduling: theory, Algorithms, and Systems. Prentice-Hall: Englewoods Cliffs, NJ, (1995).
7. JW. Stevens and DD. Gemmill. Scheduling a two-machine flowshop with travel times to minimize maximum lateness. International Journal of Production Research; 35:1-15,(1997).
8. W. Yu. The two-machine flow shop problem with delays and the one-machine total tardiness problem. Ph.D. Dissertation, Eindhoven University of Technology, (1996).