RAIRO-Oper. Res. 50 (2016) 223–232
DOI: 10.1051/ro/2015015

RAIRO Operations Research
www.rairo-ro.org

TWO MACHINES FLOW SHOP WITH REENTRANCE AND EXACT TIME LAG

Karim Amrouche[1,2] and Mourad Boudhar[2]

Abstract. This paper considers a reentrant flow shop with two machines and exact time lag L, in which each task may be processed in this order M_1, M_2, M_1 and there is an identical time lag between the completion time of the first operation and the start time of the second operation on the first machine. The objective is to minimize the total completion time. We prove the NP-hardness of a special case and we give some special subproblems that can be solved in polynomial time.

Mathematics Subject Classification. 90B35.

Received September 6, 2013. Accepted June 1, 2015.

1. Introduction

In the classical flow shop scheduling problems, it is assumed that each task visits each machine only once. But this is often violated in practice. For example, in semiconductor manufacturing, each wafer revisits the same machine for multiple processing steps [17]. This reentrant characteristic can also be found in signal processing, printing circuit boards [18]. However, there are some real world situations where two operations are connected by a temporal gap constraint. This means that there is a waiting time between the end of the operation on the upstream machine and the beginning of the operation on the downstream machine, this constraint is called time lag. Note that there exist three types of time lag: minimal time lag, maximal time lag and exact time lag. The third case obeys to strong constraints, where the minimal time lag is equal to the maximal time lag. In the literature this situation is known as the coupled-task scheduling problem. The latter has been studied by Shapiro [16], Orman and Potts [15] and is motivated by radar scheduling applications where tasks corresponding to transmitting radio waves and listening to potential echoes are coupled.

We propose a study of two machines flow shop with reentrance and exact time lag L denoted $F2|chain-reentrant, l_j = L|C_{\max}$. Each task may be processed in this order M_1, M_2, M_1 and there is an exact time lag L between the completion time of the first operation and the start time of the second operation on the first machine (called also primary machine). The objective is to minimize the makespan. Note that the problem $F2|chain-reentrant|C_{\max}$ is equivalent to the problem $V2||C_{\max}$ [13].

Keywords. Flow shop, reentrance, time lag, makespan, complexity.

[1] University of Algiers 3, Faculty of Economics and Management sciences, 2 street Ahmed Waked, Dely Brahim, Algiers, Algeria. amrouche-karim@hotmail.com

[2] RECITS laboratory, Faculty of Mathematics, USTHB University, BP 32 Bab-Ezzouar, 16111 El-Alia, Algiers, Algeria. mboudhar@usthb.dz

For the two machines flow shop without reentrance, the makespan criterion is the most studied in the literature. The simplest case occurs when all jobs have the same time lag L *i.e.* the task j cannot be started on the second machine before $l_j = L$ units of time of its completion time on the first machine. One can easily show that this problem is equivalent to the case without time lags by shifting all operations of the second machine with L units of time. Therefore Johnson's algorithm [11] gives the optimal solution. Now consider the case where all time lags are different. The problem $F2|l_j|C_{\max}$ was studied by Mitten in [14]. He proved that if we restrict to permutation schedules, the optimal solution is given by Johnson's algorithm with modifications of processing times: for each task the new processing time is obtained by adding the time lag to the processing time on both machines. But in the general case (without restriction to permutation schedules), minimization of the makespan is NP-Hard, Dell'Amico [7], Yu *et al.* [20] and Lawler *et al.* [12] proved that this problem is NP-Hard even with unit processing times jobs, or if the processing times depend only on the tasks but not machines, or in case the time-lag cannot take only two values. Yu [19] proposed a condition for which the permutation scheduling becomes dominant, *i.e.*: $\forall i, j \; l_i \leq l_j + max\{p_{1j}, p_{2j}\}$. Other researchers are interested in coupled-tasks scheduling problems denoted (CT). A coupled task is composed of two distinct operations, separated by an exact time lag. In [16], Shapiro discussed practical cases of (CT) and gave heuristics with numerical experiments. Orman and Potts studied the same problem in one machine [15], whose objective is to minimise the makespan. They identified several problems and hierarchies according to their complexity. Ahr *et al.* [3] proposed an exact algorithm using dynamic programming to solve the problem of (CT) in a machine for small instances; when L is fixed, this problem is denoted $1|coupled-task, a_i = a, l_i = L, b_i = b|C_{\max}$. This algorithm has been adapted by Brauner *et al.* in [6] to solve a (CT) problem motivated by time management problems of cyclic production with robots. Other researchers were concerned with the approximation of these problems. Thus Ageev and Baburin [1] proposed a 7/4 and 3/2-approximation to solve problem $1|coupled-task, a_i = b_i = 1, l_i = L|C_{\max}$ and $F2|a_i = b_i = 1, l_i = L|C_{\max}$, Ageev and Kononov [2] gave several approximation results and non-approximability terminals according to the values of a_i and b_i. Few works were performed by adding constraints to coupled tasks. Blazewicz et al. [4] showed that the polynomial problem $1|coupled-task, a_i = b_i = 1, l_i|C_{\max}$ is NP-Hard by adding a precedence constraint between coupled tasks. In [5], Boudhar and Meziani studied the two stages hybrid flow shop with recirculation of tasks at the second stage.

Fondrevelle *et al.* [9] minimized a non-classical criterion based on the weighted sum of machine completion times, They showed that this criterion generalizes the makespan and they derived several complexity results for two and three machines problems. Zhang *et al.* [21] considered the on-line two-machine scheduling problem with time lags. They proved, for the two machines flow shop problem with time lags that no on-line delay algorithm has a competitive ratio better than $(\sqrt{5+1})/2 \approx 1.618$, and that a greedy algorithm is still the best on-line non-delay algorithm. Emna *et al.* [8] studied the permutation flowshop scheduling problem with sequence dependent set up times and time lags. Their aim was to minimize the number of tardy jobs. They proposed two mathematical programming formulations and developed a simulated annealing algorithm.

The remainder of this paper is as follows: in Section 2, we examine the problem and define the notations used. In Section 3, a study of complexity is presented. The problem is NP-hard. Polynomial cases are presented in Section 4. The conclusion ends the paper.

2. Problem notations and definitions

Let $T = \{T_1, T_2, \ldots, T_n\}$ be the set of n independent tasks to schedule on a set of two machines $M = \{M_1, M_2\}$. The workshop is of type flow shop. Each task must be executed according to this order $M_1 \rightarrow M_2 \rightarrow M_1$ with a time lag $l_i = L$ between the completion time of the first operation on the first machine and the start time of its second operation on the same machine (see Fig. 1).

For scheduling tasks denote:

- $a_{[i]}$: The first operation of the task T_i on the first machine.
- $b_{[i]}$: The operation of the task T_i on the second machine.
- $c_{[i]}$: The second operation of the task T_i on the first machine.

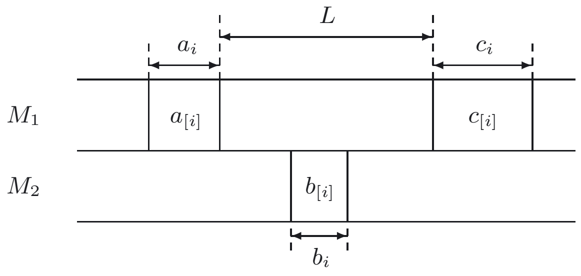

FIGURE 1. The tasks processing pattern.

TABLE 1. Processing times of the 5 tasks.

T_i	T_1	T_2	T_3	T_4	T_5
a_i	2	3	2	1	1
b_i	2	1	3	1	3
c_i	1	2	1	2	3

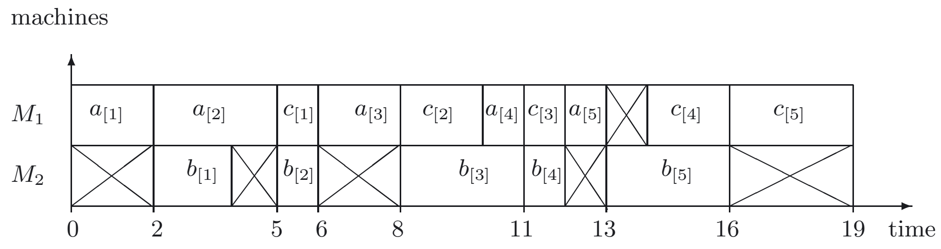

FIGURE 2. 5 tasks schedule instance on the two machines.

- a_i: The processing time of the first operation of the task T_i on the first machine.
- b_i: The processing time of the task T_i on the second machine, $b_i \leq L$.
- c_i: The processing time of the second operation of the task T_i on the first machine.

Illustrative example
To illustrate the problem, consider the following instance: we have to schedule 5 independent tasks T_1, T_2, T_3, T_4 and T_5. Processing times of the tasks on both machines are given in Table 1 and the time lag $L = 3$.

A schedule is given in Figure 2.

Definition 2.1. We say that we have formed a batch of n iterlaced tasks or a batch of n tasks, if for any task of this batch, there are exactly $n-1$ other operations between their two operations of the first machine (see Fig. 3). The processing time of a batch is its length.

3. NP-HARDNESS

In this section, we focus our study on the complexity of the problem in case $a_j = c_j = \text{const}$. The problem is NP-hard in the strong sense by a reduction from the 3-partition problem.

Theorem 3.1. *The problem $F2|chain - reentrant, l_j = L|C_{\max}$ is NP-Hard in the strong sense.*

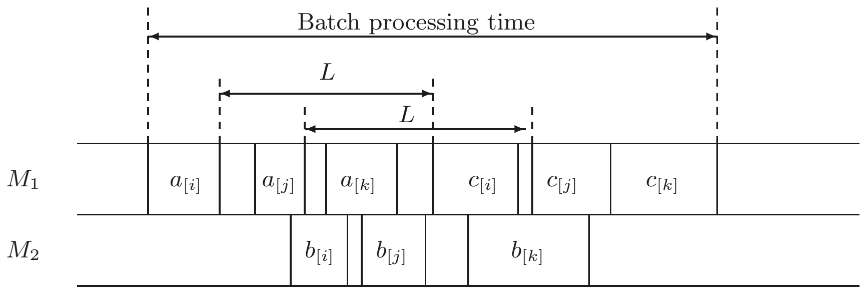

FIGURE 3. Three interlaced tasks.

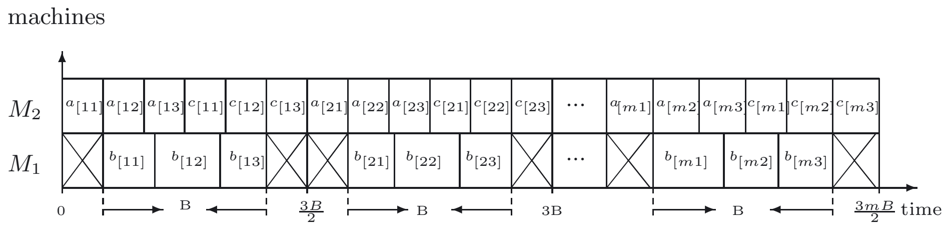
machines
M_2
M_1
$a_{[11]}$ $a_{[12]}$ $a_{[13]}$ $c_{[11]}$ $c_{[12]}$ $c_{[13]}$ $a_{[21]}$ $a_{[22]}$ $a_{[23]}$ $c_{[21]}$ $c_{[22]}$ $c_{[23]}$ ⋯ $a_{[m1]}$ $a_{[m2]}$ $a_{[m3]}$ $c_{[m1]}$ $c_{[m2]}$ $c_{[m3]}$
$b_{[11]}$ $b_{[12]}$ $b_{[13]}$ $b_{[21]}$ $b_{[22]}$ $b_{[23]}$ ⋯ $b_{[m1]}$ $b_{[m2]}$ $b_{[m3]}$
0
B
$\frac{3B}{2}$
B
3B
B
$\frac{3mB}{2}$
time

FIGURE 4. m batches composed by 3 interlaced tasks.

Proof. Consider the following decision problem known as 3-partition: given a set $\bar{A} = \{\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_n\}$ of positive integers strictly between $B/4$ and $B/2$ such that $\sum_{\bar{a}_i \in \bar{A}} \bar{a}_i = mB$, are there m disjoint subsets $S_1, S_2, \ldots, S_m$ of weight B? The 3-partition problem is NP-hard in the strong sense [10]. We show that this problem is polynomially reduced to the following decision problem: given $3m$ independent tasks $\{T_1, T_2, \ldots, T_n\}$ $(n = 3m)$ with the followings processing times: $a_i = B/4$, $b_i = \bar{a}_i, c_i = B/4$ and a time lag $L = B/2$, is there a schedule of the $3m$ tasks on the two machines M_1 and M_2 with a completion time less than or equal to $3mB/2$?

Our problem belongs to the NP class because we can verify in polynomial time if a permutation of tasks satisfies all the constraints of the problem.

We prove that the scheduling problem has a solution if and only if the 3-partition problem has a solution.

If the 3-partition problem has a solution, then there exists a partition of $\bar{A}$ into m disjoint subsets of cardinality 3 and weight equal to B to which we associate a set of tasks $T = \{T_{[11]}, T_{[12]}, T_{[13]}\} \cup \ldots \cup \{T_{[m1]}, T_{[m2]}, T_{[m3]}\}$; ($T_{[i,j]}$ is the task numbered j, $j = \overline{1,3}$ of the batch i, $i = \overline{1,m}$)

We construct a solution to the scheduling problem as shown in Figure 4 with $C_{\max} = m(3B/2) = 3mB/2$.

this solution is feasible because:

- $b_{[i1]}$: Finishes before the start of $c_{[i1]}$ because its processing time is strictly less than $B/2 = L$.
- $b_{[i2]}$: Finishes before the start of $c_{[i2]}$ because the processing time of $b_{[i3]}$ is greater than $B/4$.
- $b_{[i2]}$: Starts after the end of $a_{[i2]}$ because the processing time of $b_{[i1]}$ is greater than $B/4$.
- $b_{[i3]}$: Starts after the end of $a_{[i3]}$ because the processing time of $b_{[i1]}$ $+b_{[i2]}$ is greater than $B/2 = L$.

If the scheduling problem has a solution less or equal to $3mB/2$ then the solution is as shown in Figure 4. We have m batches of tasks where each batch is composed by 3 interlaced tasks, the processing time of each batch is equal to $3B/2$. No task is processed alone because its processing time is equal to B. Two tasks cannot be processed together because their processing times are equal to $5B/4$. In this case, we have $B + 5B/4 = 9B/4 > 3B/2$. Also, if three tasks are processed alone, the total processing time is equal to $3B > 3B/2$ (see Fig. 5).

Also, we have $b_{[i1]}+b_{[i2]}+b_{[i3]} = B$ $(\forall i = \overline{1,m})$ because we cannot find a batch i for which $b_{[i1]}+b_{[i2]}+b_{[i3]} > B$, the solution in this case is not feasible (the time lag is not respected for this batch). In addition, we cannot find

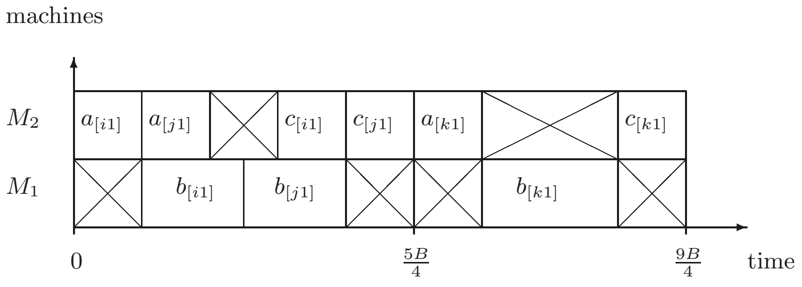

Figure 5. Batches composed by two interlaced tasks and one task.

a batch i for which $b_{[i1]} + b_{[i2]} + b_{[i3]} < B$, due to the reason that if this batch exists, there is another batch j for which $b_{[j1]} + b_{[j2]} + b_{[j3]} > B$ (because $b_{[ij]} = \bar{a}_i$ and $\sum_{\bar{a}_i \in \bar{A}} \bar{a}_i = mB$); the solution in this case is not feasible (the time lag is not respected for the batch j). So, the formed batch gives a solution for the problem of 3-partition. □

The problem $F2|chain-reentrant, l_j = L|C_{\max}$ is thus NP-hard, we give in the next section some polynomial subproblems that solve using a transformation of our problem into a maximum weight matching problem.

4. Polynomial subproblems

We show in this section that when processing times a_j and c_j are greater or equal to $L/2$, the problem is solved by a polynomial algorithm based on the maximum weight matching.

Remark 4.1. If all processing times on the first machine (*i.e.* a_j and c_j) are greater than $L/2$, then the maximum batch size is equal to two and the formed batches can not be interlaced.

Proof. As we have $a_j, c_j > L/2$ for all tasks, then the sum of processing times of any two operations on the first machine is greater than L, since we have $l_j = L \ \forall j$ then we can not interlace more than two tasks in the same batch. □

Theorem 4.2. *The problem* $F2|chain - reentrant, a_j > L/2, c_j > L/2, l_j = L|C_{\max}$ *reduces to the maximum weight matching.*

Proof. We construct a graph $G = (V, E)$ such that:

V: the Set of vertices is the set of tasks.
E: Set of edges such that$(T_l, T_{l'}) \in E$ if and only if $a_{l'} \leq L$ and $c_l \leq L$.

Let λ be the vertex-edge incidence matrix of the graph $G = (V, E)$ where

$$\lambda_{lk} = \begin{cases} 1, \text{ if the edge k is incident to the vertex l;} \\ 0, \text{ if not.} \end{cases}$$

For $k = \overline{1, n}$, $l = \overline{1, q}$ (q the number of edges).

Let ρ_k= $\min\{d_{s_k t_k}, d_{t_k s_k}\}$ be the cost of edge k, if edge k is incident to the vertices s_k and t_k.
where:

$d_{s_k t_k}$: The processing time of the two interlaced tasks s_k and t_k in this order.
$d_{t_k s_k}$: The processing time of the two interlaced tasks t_k and s_k in this order.

We have $\begin{cases} d_{s_k t_k} = a_{s_k} + c_{t_k} + (2 * L - \bar{y}_1) \\ d_{t_k s_k} = a_{t_k} + c_{s_k} + (2 * L - \bar{y}_2) \end{cases}$
where $\bar{y}_1 = L - \max\{a_{t_k}, c_{s_k}, b_{s_k} + b_{t_k} - L\}$ and $\bar{y}_2 = L - \max\{a_{s_k}, c_{t_k}, b_{s_k} + b_{t_k} - L\}$ are idle times.
Then $\begin{cases} d_{s_k t_k} = a_{s_k} + c_{t_k} + L + \max\{a_{t_k}, c_{s_k}, b_{s_k} + b_{t_k} - L\} \\ d_{t_k s_k} = a_{t_k} + c_{s_k} + L + \max\{a_{s_k}, c_{t_k}, b_{s_k} + b_{t_k} - L\} \end{cases}$

The linear model of the problem is

$$\min C_{\max} = \left(\sum_{k=1}^{q} \rho_k x_k\right) + \sum_{l=1}^{n}(1 - \sum_{k=1}^{q} \lambda_{lk} x_k)\delta_l.$$

$$\text{s.c.} \begin{cases} \sum_{k=1}^{q} \lambda_{lk} x_k \leq 1 \text{ for } l = \overline{1,n} \\ x_k \in \{0,1\} \text{ for } k = \overline{1,q} \end{cases}$$

where:

- $x_k = \begin{cases} 1, \text{ if the two tasks joined by edge k are interlaced in the same batch;} \\ 0, \text{ if not.} \end{cases}$
- $\sum_{k=1}^{q} \rho_k x_k$: the sum of processing times of batches that contain interlaced tasks.
- $\sum_{l=1}^{n}(1 - \sum_{k=1}^{q} \lambda_{lk} x_k)\delta_l$: the sum of processing times of batches that contain one task.
- $\delta_l = a_l + c_l + L$: processing time of batch which contain, the task l.
- $\sum_{k=1}^{q} \lambda_{lk} x_k \leq 1$ indicates that each task, is at most, in one batch.

We have:

$$\left(\sum_{k=1}^{q} \rho_k x_k\right) + \sum_{l=1}^{n}\left(1 - \sum_{k=1}^{q} \lambda_{lk} x_k\right)\delta_l = \left(\sum_{l=1}^{n} \delta_l\right) - \left(\sum_{l=1}^{n}\sum_{k=1}^{q} \lambda_{lk} x_k \delta_l - \sum_{k=1}^{q} \rho_k x_k\right)$$

$$= \left(\sum_{l=1}^{n} \delta_l\right) - \left(\sum_{k=1}^{q}\sum_{l=1}^{n} \lambda_{lk} \delta_l x_k - \sum_{k=1}^{q} \rho_k x_k\right)$$

$$= \left(\sum_{l=1}^{n} \delta_l\right) - \sum_{k=1}^{q}\left(\sum_{l=1}^{n} \lambda_{lk} \delta_l - \rho_k\right) x_k.$$

$$= \left(\sum_{l=1}^{n} a_l + c_l + L\right) - \sum_{k=1}^{q}(\sum_{l=1}^{n} \lambda_{lk}(a_l + c_l + L) - \min\{d_{s_k t_k}, d_{t_k s_k}\}) x_k.$$

$$= \left(\sum_{l=1}^{n} a_l + c_l + L\right) - \sum_{k=1}^{q}\left(\sum_{l=1}^{n} \lambda_{lk}(a_l + c_l + L) - \min\left\{\begin{array}{l} a_{s_k} + c_{t_k} + L + \max\{a_{t_k}, c_{s_k}, b_{s_k} + b_{t_k} - L\}, \\ a_{t_k} + c_{s_k} + L + \max\{a_{s_k}, c_{t_k}, b_{s_k} + b_{t_k} - L\} \end{array}\right\}\right) x_k.$$

$$= nL + \sum_{l=1}^{n} a_l + \sum_{l=1}^{n} c_l - \sum_{k=1}^{q}(a_{s_k} + c_{s_k} + L + a_{t_k} + c_{t_k} + L - \min\left\{\begin{array}{l} a_{s_k} + c_{t_k} + L + \max\{a_{t_k}, c_{s_k}, b_{s_k} + b_{t_k} - L\}, \\ a_{t_k} + c_{s_k} + L + \max\{a_{s_k}, c_{t_k}, b_{s_k} + b_{t_k} - L\} \end{array}\right\}) x_k.$$

$$= nL + \sum_{l=1}^{n} a_l + \sum_{l=1}^{n} c_l - \sum_{k=1}^{q} \max\left\{\begin{array}{l} 2L + a_{s_k} + c_{s_k} + a_{t_k} + c_{t_k} - a_{s_k} - c_{t_k} - L - \max\{a_{t_k}, c_{s_k}, b_{s_k} + b_{t_k} - L\}, \\ 2L + a_{s_k} + c_{s_k} + a_{t_k} + c_{t_k} - a_{t_k} - c_{s_k} - L - \max\{a_{s_k}, c_{t_k}, b_{s_k} + b_{t_k} - L\} \end{array}\right\} x_k.$$

$$= nL + \sum_{l=1}^{n} a_l + \sum_{l=1}^{n} c_l - \sum_{k=1}^{q} \max \left\{ \begin{array}{l} L + c_{s_k} + a_{t_k} - \max\{a_{t_k}, c_{s_k}, b_{s_k} + b_{t_k} - L\}, \\ L + a_{s_k} + c_{t_k} - \max\{a_{s_k}, c_{t_k}, b_{s_k} + b_{t_k} - L\} \end{array} \right\} x_k.$$

$$= nL + \sum_{l=1}^{n} a_l + \sum_{l=1}^{n} c_l - \sum_{k=1}^{q} \max \left\{ \begin{array}{l} \min\{L + c_{s_k}, L + a_{t_k}, 2L + c_{s_k} + a_{t_k} - b_{s_k} - b_{t_k}\}, \\ \min\{L + c_{t_k}, L + a_{s_k}, 2L + a_{s_k} + c_{t_k} - b_{s_k} - b_{t_k}\} \end{array} \right\} x_k.$$

$nL + \sum_{l=1}^{n} a_l + \sum_{l=1}^{n} c_l$is a constant greater than

$$\sum_{k=1}^{q} \max \left\{ \begin{array}{l} \min\{L + c_{s_k}, L + a_{t_k}, 2L + c_{s_k} + a_{t_k} - b_{s_k} - b_{t_k}\}, \\ \min\{L + c_{t_k}, L + a_{s_k}, 2L + a_{s_k} + c_{t_k} - b_{s_k} - b_{t_k}\} \end{array} \right\} x_k.$$

Then, minimizing $C_{\max}$ is equivalent to maximizing

$$\sum_{k=1}^{q} \max \left\{ \begin{array}{l} \min\{L + c_{s_k}, L + a_{t_k}, 2L + c_{s_k} + a_{t_k} - b_{s_k} - b_{t_k}\}, \\ \min\{L + c_{t_k}, L + a_{s_k}, 2L + a_{s_k} + c_{t_k} - b_{s_k} - b_{t_k}\} \end{array} \right\} x_k.$$

Hence, the linear model reduces to the maximum weight matching problem. □

The following algorithm solves the problem $F2|\mathit{chain\text{-}reentrant}, a_j > L/2, c_j > L/2, l_j = L|C_{\max}$.

Algorithm $\mathcal{A}1$

```
begin
(1) From the graph $G = (V, E)$ construct a new valued graph $H = (V, E)$ where each edge in $E$ is valued by
        $\max \left\{ \begin{array}{l} \min\{L + c_{s_k}, L + a_{t_k}, 2L + c_{s_k} + a_{t_k} - b_{s_k} - b_{t_k}\}, \\ \min\{L + c_{t_k}, L + a_{s_k}, 2L + a_{s_k} + c_{t_k} - b_{s_k} - b_{t_k}\} \end{array} \right\}$
    if the edge is incident to the vertices $s_k$ and $t_k$.
(2) Find the maximum weight matching $M$ in the graph $H$.
(3) Construct the solution as follows:
      • For each pair of $M$, interlace the two corresponding tasks in the same batch.
      • Put each of the remaining tasks in one batch.
(4) Schedule the batches in arbitrary order (without idle time).
end.
```

The maximum number of possible edges in G is $\frac{n(n-1)}{2}$ and the best known algorithm for the maximum weight matching is in O($n^{2.5}$). Hence, also the algorithm A1 runs in $O(n^{2.5})$.

Example 4.3. Let us process 5 tasks $T_1, \ldots T_5$. The processing times on the two machines are given in Table 2 and the time lag $L = 4$.

TABLE 2. Processing times of the 5 tasks.

T_i	T_1	T_2	T_3	T_4	T_5
a_i	2	3	5	2	5
b_i	2	4	3	4	3
c_i	5	2	2	5	3

1- The pairwise tasks $(1,2)$, $(1,3)$, $(1,4)$, $(1,5)$, $(2,3)$, $(2,5)$, $(3,5)$, $(4,1)$, $(4,2)$, $(4,3)$, $(4,5)$, $(5,3)$, can not be interlaced in the same batch. So, the corresponding edges are not formed (the conditions are not satisfied). Hence, the graph $G=(V,E)$ contains 5 vertices (number of tasks) and 8 edges: $\{(2,1),(2,4),(3,1),(3,2),$ $(3,4),(5,1),(5,2),(5,4)\}$.
2- Construction of the valued graph H (see Fig. 6):
$d(2,1)=\min\{4+2,4+2,8+4-6\}=6$.
$d(2,4)=\min\{4+2,4+2,8+4-8\}=4$.
$d(3,1)=\min\{4+2,4+2,8+4-6\}=6$.
$d(3,2)=\min\{4+2,4+3,8+5-7\}=6$.
$d(3,4)=\min\{4+2,4+2,8+4-7\}=5$.
$d(5,1)=\min\{4+3,4+2,8+5-5\}=6$.
$d(5,2)=\min\{4+3,4+3,8+6-7\}=7$.
$d(5,4)=\min\{4+3,4+2,8+5-7\}=6$.
3- The maximum weight matching $M=\{(5,2),(3,1)\}$.
4- The optimal sequence is given by: $(5,2),(3,1),4$ and the completion time is equal to 41 (see Fig. 7).

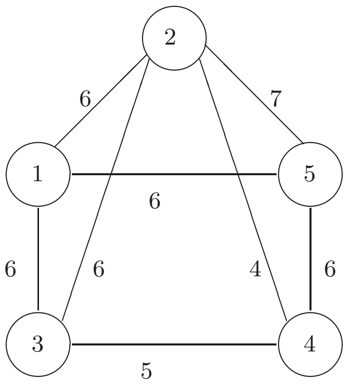

Figure 6. The valued graph H.

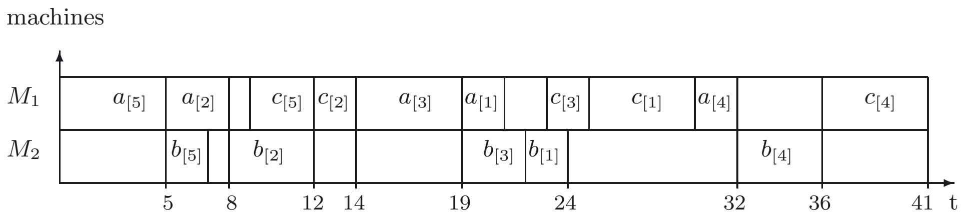

Figure 7. The solution obtained using Algorithm $\mathcal{A}1$.

Theorem 4.4. *The problem $F2|chain-reentrant, b_j=l_j=L, a_i+c_j\leq L|C_{\max}$ is polynomially solved by Algorithm $\mathcal{A}2$.*

The following algorithm solves the problem $F2|chain-reentrant, b_j=l_j=L, a_i+c_j\leq L|C_{\max}$.

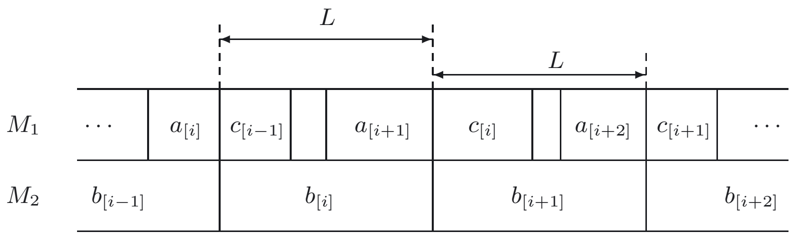

FIGURE 8. Two interlaced tasks.

Algorithm $\mathcal{A}2$

```
begin
  Construct the solution as follows:
  • Find two different tasks $k$ and $l$ such as: $a_k + c_l = min_{i \neq j}\{a_i + c_j\}$.
  • Form a list $T_k, \ldots, T_i, T_{i+1}, \ldots, T_l$ where the task $T_k$ is scheduled at the beginning and $T_l$ at the end.
  • Schedule any two tasks $T_i, T_{i+1}$ as is shown in Figure 8.
end.
```

Proof. As we have $b_j = L$, then the maximum cardinality of the formed batches is equal to two, but the formed batches in this case can be interlaced as it's shown in Figure 8 (the batches are $\langle T_{i-1}, T_i\rangle$ and $\langle T_{i+1}, T_{i+2}\rangle$). As we have also the condition that the sum of any two operations on the first machine is less or equal to L, then all batches can be interlaced. So, to minimize makespan, it is sufficient to put the two tasks that have the smallest sum $\min_{i\neq j}\{a_i + c_j\}$ at the beginning and the end of schedule and put the other tasks in the middle. The value of makespan is equal to $C_{\max} = \min_{i\neq j}\{a_i + c_j\} + nL$. □

Proposition 4.5. *The problem $F2|chain - reentrant, a_j = b_j = c_j = l_j = L|C_{\max}$ is polynomially solved by choosing any order of tasks.*

Proof. the proof is obvious and the $C_{\max}$ is equal to $2nL$ if n is even and it is equal to $(2n + 1)L$ if not. □

Proposition 4.6. *The problem $F2|chain - reentrant, a_j = b_j = c_j = c, l_j = L|C_{\max}$ is polynomially solved.*

Proof. $C_{\max} = 2nc +$ the idle times in the first machine. For minimizing the idle times, we form batches of cardinality $\lfloor \frac{L}{c} \rfloor + 1$, each batch processing time is equal to $c(2 + \lfloor \frac{L}{c} \rfloor) + L$ (the idle time in each bath is lower than c) except the last batch which is formed by the $(n - \lfloor \frac{n}{\{\lfloor \frac{L}{c} \rfloor + 1\}} \rfloor)$ remaining tasks with processing time equal to $c(2 + (n - \lfloor \frac{n}{\{\lfloor \frac{L}{c} \rfloor + 1\}} \rfloor)) + L$. Therefore, the makespan value is equal to the sum of all batch processing times, that is minimized. □

Proposition 4.7. *The problem $F2|chain - reentrant, a_j = a > L, l_j = L|C_{\max}$ is polynomially solved by choosing any order of tasks.*

Proof. The maximum batch cardinality is equal to one and the solution is obtained by scheduling all tasks one after one in arbitrary order without idle time. The value of makespan is $C_{\max} = \sum_{j=1}^{n} c_j + n * (L + a)$. □

Note that the symetric case (*i.e.* $c_j = c > L$) is also polynomial and $C_{\max} = \sum_{j=1}^{n} a_j + n * (L + c)$.

Remark 4.8. If $b_j = 0$, our problem is equivalent to the problem $1|coupled - task, l_i = L|C_{\max}$ (see introduction).

5. Conclusion

Research in scheduling has yielded in-depth results in the last few years. Constraints taken into account in recent work are becoming increasingly complex.

We studied the problem $F2|chain - reentrant, l_j = L|C_{\max}$ with the goal to minimize the total completion time. This problem is flow shop on two machines with duplication of tasks on the first machine and an exact time lag $l_i = L$ between the completion time of the first operation of task T_i on the first machine and the start time of its second operation on the same machine.

The problem in its general form is NP-hard in the strong sense because the case without time lags is already NP-hard in the strong sense. We showed that one of the particular problems is also NP-hard in the strong sense by a reduction from the 3-partitions problem. We have also proposed some polynomial cases for which we have developed algorithms for their resolution based on the maximum weight matching.

As a future prospect for our work, methods of resolution may be used, such as some exact methods (dynamic programming, branch and bound procedure, etc.) or using approch methods like heuristics or meta-heuristics.

Acknowledgements. The authors gratefully wish to thank the anonymous reviewers for their careful reading of the paper and for their valuable and useful comments. Their contributions greatly helped to improve this paper.

References

[1] A.A. Ageev and A.E. Barburin, Approximation algorithms for UET scheduling problems with exact delays. *Oper. Res. Lett.* **35** (2007) 533–540.

[2] A.A. Ageev and A.V. Kononov, Approximation Algorithms for Scheduling Problems with Exact Delays. In WAOA, vol. 4368 of *Lect. Notes Comput. Sci.* (2006) 1–14.

[3] D. Ahr, J. Békési, G. Galambos, M. Oswald and G. Reinelt, An exact algorithm for scheduling identical coupled tasks. *Math. Methods Oper. Res.* **59** (2004) 193–203.

[4] J. Blazewicz, K. Ecker, T. Kis, CN. Potts, M. Tanas and J. Whitehead, Scheduling of coupled tasks with unit processing times. *J. Sched.* **13** (2010) 453–461.

[5] M. Boudhar and N. Meziani, Two-stage hybrid flow shop with recirculation. *Int. Trans. Oper. Res.* **17** (2010) 239–255.

[6] N. Brauner, G. Finke, V. Lehoux-Lebacque, C. Potts and J. Whithead, Scheduling of coupled tasks and one-machine no-wait robotic cells. *Comput. Oper. Res.* **36** (2009) 301–307.

[7] M. Dell'Amico, Shop problems with two machines and time lags. *Oper. Res.* **44** (1996) 777–787.

[8] E. Dhouib, J. Teghem and T. Loukir, Minimizing the Number of Tardy Jobs in a Permutation Flowshop Scheduling Problem with Setup Times and Time Lags Constraints. *J. Math. Model. Algorithms* **12** (2013) 85–99.

[9] J. Fondrevelle, A. Oulamara, M.C. Portmann, Permutation flowshop scheduling problems with time lags to minimize the weighted sum of machine completion times. *Int. J. Prod. Econ.* **33** (2007) 168–176.

[10] M.R. Garey and D.S. Johnson, Computers and Intractability. W.H. Freeman and Company, New York (1979).

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

[12] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, Sequencing and scheduling theory: algorithms and complexity. In *Handb. Oper. Res. Manag. Sci.* Edited by S.C. Graves, P.H. Zipkin and A.H.G. Rinnooy Kan. North Holland, Amesterdam (1993).

[13] V. Lev and I. Adiri, V-shop scheduling. *Eur. J. Oper. Res.* **18** (1984) 51–56.

[14] L.G. Mitten, Sequencing n jobs on two machines with arbitrary time lags. *Manage. Sci.* **5** (1959) 293–298.

[15] A.J. Orman and C.N. Potts, On the Complexity of Coupled-task Scheduling. *Discrete Appl. Math.* **72** (1997) 141–54.

[16] R.D. Shapiro, Scheduling coupled tasks. *Nav. Res. Logist. Quarterly* **27** (1980) 489–97.

[17] F.D. Vargas-Villamil and D.E. Rivera, A model predictive control approach for real-time optimization of reentrant manufacturing lines. *Comput. Ind.* **45** (2001) 45–57.

[18] M.Y. Wang, S.P. Sethi and S.L. Van De Velde, Minimizing makespan in a class of reentrant shops. *Oper. Res.* **45** (1997) 702–712.

[19] W. Yu, The two-machine flowshop problem with delays and the one-machine total tardiness problem. Ph. D. thesis, Technische Universiteit Eindhoven (1996).

[20] W. Yu, H. Hoogeveen and J.K. Lenstra, Minimizing makespan in a two-machine flow shop with delays and unit-time operations is np-hard. *J. Sched.* **7** (2004) *const*3–348.

[21] X. Zhang and S. Van de Velde, On-line two-machine open shop scheduling with time lags. *Eur. J. Oper. Res.* **204** (2010) 14–19.