THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 22, Number 3/2021, pp. 227–234

AN EFFICIENT ALGORITHM FOR LOT PERMUTATION FLOW SHOP SCHEDULING PROBLEM

Cristina Elena DODU, Mircea ANCĂU

Technical University of Cluj-Napoca, Department of Manufacturing Engineering,
B-dul Muncii 103-105 400641 Cluj Napoca
Corresponding author: Cristina Elena DODU, E-mail: cristina.dodu@tcm.utcluj.ro

Abstract. This work is concerned about the minimization of the makespan in a generalization of the classical permutation flow shop dealing with the production lots. The flow shop scheduling problem is one of the most popular machine scheduling problems and this paper proposes an original way to apply PFSP on scheduling a bunch of lots. The jobs constituting a production lot have identical processing-times. The article proposes to find the optimal sequence in which the lots will be scheduled to flow in the machines using an improved version of the tabu search meta-heuristic.

Key words: lot permutation flow shop scheduling, permutation flow shop scheduling, PFSP, makespan, the completion time, tabu search.

1. INTRODUCTION

The permutation flow shop problem denoted as *PFSP* is a classic scheduling problem where n jobs $\{j_1, j_2, ..., j_n\}$ must be processed on a set of m machines $\{i_1, i_2, ..., i_m\}$. The problem definition implies each job must visit all machines in the same order. Each job contains exactly m operations. The processing time of a job j on machine i is denoted by t_{ij}. No machine can run more than one operation at the same time. For the sequence $\pi = \{\pi_1, \pi_2, ..., \pi_n\}$ constituting a possible jobs permutation when processed by the machines, the completion time denoted by c_{ij} is calculated based on the following set of equations:

$$c_{ij} = \max\left\{c_{i-1j}, c_{ij-1}\right\} + t_{ij},\quad i = 1\ldots m,\ j = 1\ldots n \tag{1}$$

$$c_{0,j} = 0,\ \ j = 1\ldots n \tag{2}$$

$$c_{i,0} = 0,\ \ i = 1\ldots m\ . \tag{3}$$

The maximum completion time or makespan $C_{\max}$ refers to the last job n on the last machine m:

$$C_{\max}(\pi) = c_{mn}\ . \tag{4}$$

The objective is to determine the optimal jobs arrangement with the shortest possible total jobs execution or makespan $C_{\max}$, when all n jobs are processed on the m machines, such as

$$C_{\max} \le C_{\max}(\pi)\ . \tag{5}$$

The waiting time of a job j on machine i is given by the formula:

$$W_{ij} = \begin{cases} 0 & , \quad c_{i-1j} \le c_{ij-1} \\ c_{i-1j} - c_{ij-1}, & \quad c_{ij-1} < c_{i-1j} \end{cases} ,\quad i = 1\ldots m,\ j = 1\ldots n\ . \tag{6}$$

In the production, the set of jobs that is consecutively processed with the same operation on the same machine is called lot. The problems which involve the analysis of lots can be solved using one of the scheduling theories: lot streaming and job batching. Lot streaming refers to the process of dividing jobs to speed up production through several stages as quickly as possible rather than batch scheduling which invokes the process of grouping jobs to improve the use of resources and customer satisfaction.

The adequate use of resources during manufacturing is one of the main concerns in scheduling jobs. Grouping the identical jobs in a lot is mainly done to minimize set-up times and costs on each machine. Changing the number of the identical jobs in a lot is always given by the customer needs.

In [6], Pots and Van proposed a general model which combines batching and lot-sizing decisions with scheduling and presented a review of research on this type of model. They referred to batching as the decision to schedule similar jobs from the same family contiguously and extended the model to be processed concurrently on different machines. Lot streaming (lot-sizing), which involves dividing production lots or jobs into sub-lots, and then processing the overlapped sub-lots on different machines, has been considered an efficient strategy for minimizing makespan [7].

In a traditional flow shop scheduling, each job is unique, indivisible and it cannot be transferred to the next machine before its processing is finished [6, 8]. This paper proposes an original way to apply *PFSP* on scheduling a set of lots (a bunch of identical jobs) that is called *LPFSP*. The number of lots and the number of jobs for each lot are predetermined and the focus is on finding the optimal sequence of lots with the shortest possible total lots execution makespan $C_{\max}$. Supposing a lot L_k is divided into p equal jobs $\left\{ j_1, j_2, ..., j_p \right\}$, as per the traditional scheduling problem, a job is indivisible and it cannot be transferred to the next machine before its processing is finished on the current machine. The operations, in a lot, have identical processing-times. Considering each lot as sub-problem of *PFSP*, for the first lot L_1 the completion time of each job on each machine is given by the formulas (1), (2), (3). Starting with the lot L_2, the completion time of each job on each machine takes in consideration the completion time of the execution of the last job on each machine from the previous lot. Let be L_{k-1} which is divided into q equal jobs $\left\{ j_1, j_2, ..., j_q \right\}$ with the same processing time and L_k which is divided into p equal jobs $\left\{ j_1, j_2, ..., j_p \right\}$ where $t_i = t_{i1} = t_{i2} = ... = t_{ip}$. The completion time of each job on each machine is given by the formulas:

$$c_{ij}(L_k) = \max\left\{ c_{i-1q}(L_{k-1}) + c_{i-1j}, c_{iq}(L_{k-1}) + c_{ij-1} \right\} + t_i, \quad i = 1\ldots m, \; j = 1\ldots p \tag{7}$$

$$c_{0,j} = 0, \quad j = 1\ldots p \tag{8}$$

$$c_{i,0} = 0, \quad i = 1\ldots m \,. \tag{9}$$

For the sequence $\pi_L = \left\{ \pi_{L_1}, \pi_{L_2} \ldots \pi_{L_n} \right\}$ constituting a possible lots permutation when processed by the machines, the completion time is calculated based on the relation:

$$C_{\max}(\pi_L) = c_{mn}(L_n)\,. \tag{10}$$

In order to complete the processing of the lots with minimum makespan, an optimized lots arrangement has to be determined:

$$C_{\max} \le C_{\max}(\pi_L)\,, \text{ for each possible sequence } \pi_L \,. \tag{11}$$

Similarly, for the lot L_k, the waiting time of a job j on machine i is given by the formula:

$$W_{ij}(L_k) = \begin{cases} 0, & c_{i-1j}(L_k) \le c_{ij-1}(L_k) \\ c_{i-1j}(L_k) - c_{ij-1}(L_k), & c_{ij-1}(L_k) < c_{i-1j}(L_k) \end{cases}, \quad i = 1...m, j = 1...p \tag{12}$$

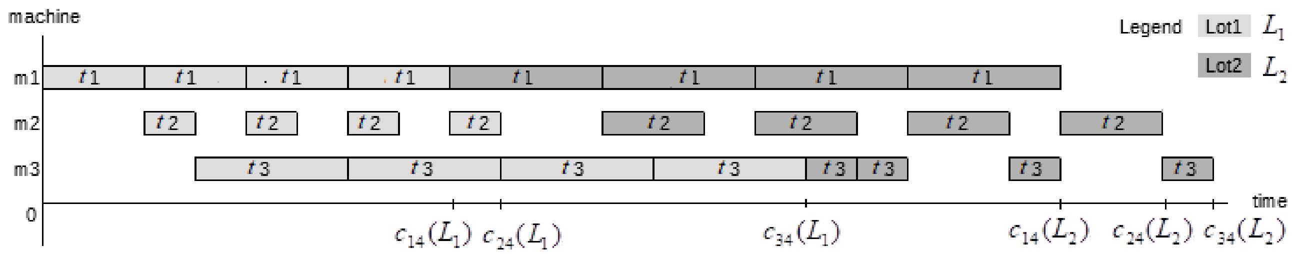

Fig. 1 – Gantt diagram example for the processing of two lots with four identical jobs on three machines.

PFSP is well known as NP-hard problems. The purpose of this research is to provide a meta-heuristic for obtaining an optimal solution of *LPFSP* as a generalization of *PFSP*. In the next section, the lot permutation flow shop scheduling problem (*LPFSP*) is formulated and the algorithm is proposed. The last two sections analyze the results and provide the conclusions.

2. THE PROPOSED APPROACH

Let consider *PFSP* with n jobs and m machines when the processing times for each job on each machine is known and denominated as t_{ij} and let formulate *LPFSP* by multiplying each job j_p from *PFSP* on each lot L_p, where the size of the lot is less or equals with n:

$$L_p = \left\{ j_1, j_2, ..., j_{n_p} \right\}, \; p = 1 \ldots n . \tag{13}$$

For *LPFSP* it is required the optimal sequence in which the minimum makespan is obtained. *LPFSP*'s performance is measured comparing the global solution with the lower bound's value denoted *LB*, given by the formula:

$$Dist = \frac{C_{\max} - LB}{LB} 100\%. \tag{14}$$

The lower bound's value is calculated by Taillard's formula [1]:

$$LB = \max \left\{ S_i, i = 1 \ldots m \right\} < C_{\max} . \tag{15}$$

where S_i is obtained summing up the processing time of all the lots on the machine i, B_i is the minimum amount of time before machine i starts to work and A_i is the minimum amount of time that the machine i remains inactive after its work up to the end of the operations .

$$S_i = B_i + \sum_{p=1}^{n} T_{p_i} + A_i . \tag{16}$$

T_{p_i}, the processing time of the L_p with n_p jobs on the machine i, is given by

$$T_{p_i} = \sum_{j=1}^{n_p} t_{ij}, \; p = 1 \ldots n . \tag{17}$$

The minimum amount of time before the machine i starts to work is calculated as minimum amount of time for each lot before the machine i starts to work, B_{p_i}:

$$B_i = \min \left\{ B_{p_i}, \; p = 1 \ldots n \right\} . \tag{18}$$

Because each lot L_p contains n_p identic jobs, B_{p_i} is given by

$$B_{p_i} = \sum_{k=1}^{i-1} t_{k0}, \;\; p = 1 \ldots n . \tag{19}$$

Similarly, A_i, the minimum amount of time that the machine i remains inactive after its work up to the end of the operations, is reduced to the minimum amount of time for the inactivity of the machine i after its jobs completion for each lot, A_{p_i}:

$$A_i = \min\left\{A_{p_i},\; p = 1 \ldots n\right\} \tag{20}$$

$$A_{p_i} = \sum_{k=i+1}^{m} t_{k0}, \;\; p = 1 \ldots n . \tag{21}$$

The proposed approach obtains the optimal sequence of *LPFSP* with a Tabu Search algorithm.

Tabu Search denoted as *TS*, as a single-point meta-heuristic emissary, localizes the best candidate from $NH(sol)$-the neighbourhood of a proposed solution sol, as it is described by the Tabu Search methodology (Glover [2, 3]). The idea of tabu search is to bypass the search becoming grounded in local minima by preventing "backwards" moves. This is usually achieved by constructing a list of the last n variable-value assignments (*TL*). When picking the next variable-value assignment, those on the list are forbidden, or Tabu. Taillard [11] tested various types of neighbourhoods resulting from changing the position of one job proved to be the best, Ben-Daya and Al-Fawzan [9] proposed a tabu search with the intensification and diversification schemes which provides better moves and the results obtained are similar as Taillard [11], Nowicki and Smutnicki [10] focused on the intensification strategy using a long term memory for recording and recovering elite solutions found during the search in order to resume the search from attractive neighbours of these solution not previously visited (called Back Jump Tracking). Dodu and Ancău [5] proposed a *TS* with the intensive concentric exploration that overconducted to the study of the *PFSP* using the production lots. The proposed approach of *TS* for the classical *PFSP* starts with *NEH* algorithm [4] – the champion among the constructive heuristics used in [5, 9, 10, 11] and uses a simple but effective technique for generating the neighbourhoods (the random shifting of two jobs indexes operation as one of the Taillard's tested options [11]) and it differs from [9, 10, 11] by using a global tabu list. The algorithm has two parameters: the number of the iterations and the size of the neighbourhood. The values of the parameters (2 000 iterations and 100 neighbourhoods for each current solution) were chosen experimentally in order to ensure the solution's quality over the running time for Taillard's benchmark [1]. The first usage of the proposed approach of *TS* is for solving *PFSP* with n jobs and m machines which provides the initial solution for *LPFSP*. The second time, it is re-used for solving *LPFSP* with n lots, m machines, each lot having a different number of jobs. The approach proposes a random generated number of jobs, less or equal with n, for each job:

Table 1

How the jobs (sub-lots) are chosen for each lot in *LPFSP* with 4 lots from *PFSP* with 4 jobs

PFSP with 4 jobs	j_1	j_2	j_3	j_4
Random number between 1 and 4:	2	4	1	3
LPFSP with 4 lots:	L_1	L_2	L_3	L_4
Jobs for each lot:	$\{j_1, j_1\}$	$\{j_2, j_2, j_2, j_2\}$	$\{j_3\}$	$\{j_4, j_4, j_4\}$

In the production's environment, the numbers of the identical jobs in all the lots are defined by the customer's needs before starting the scheduling routine. The procedure of randomly generating the number of jobs will be replaced by the desired sequence of jobs number for all the lots.

2.1. *LPFSP* algorithm

Step 1: Generate randomly the number for each lot: $n_p, p=1\ldots n$ (sequence of jobs number for all lots)

Step 2: Run *TS* algorithm on *PFSP* starting with the initial solution obtains from *NEH* [4] algorithm and obtained the solution for *PFSP*, denoted $pfsp\_sol=\{j_1, j_2,.., j_n\}$

Step 3: Build the initial solution for *LPFSP* from $pfsp\_sol$:

$$lpfsp\_initial=\{L_1, L_2,.., L_n\},\ L_p=\left\{j_1, j_2,...j_{n_p}\right\},\ p=1\ldots n,\ j_1=j_2=...=j_{n_p}=j_p\in pfsp\_sol$$

Step 4: Run *TS* algorithm on *LPFSP* starting with the initial solution $lpfsp\_initial$ and obtained the solution for *LPFSP* denoted $lpfsp\_sol$

Step 5: Evaluate $lpfsp\_sol$ and $lpfsp\_initial$ by (11)

2.2. *TS* algorithm

Step 1: Build $initialSolution$ with *NEH* [4]

$$sol=initialSolution, sol^*=initialSolution, iterationsNumber=2000$$

Step 2: Adds sol to *TL*

Step 3: While the neighbors from $NH(sol)$ don't meet the maximum size of the neighborhood $=100$, it generates a candidate solution by interchanging two jobs indexes and if the candidate is not in *TL* then add the candidate to $NH(sol)$

Step 4: Orders ascending $NH(sol)$ by $C_{\max}$ values and sets sol with first solution from $NH(sol)$

Step 5: If $C_{\max}(sol)<C_{\max}(sol^*)$ then $sol^* = sol$

Step 6: If $iterationsNumber$ doesn't meet maximum value then goes to ***Step 2***

3. THE ANALYZES OF THE RESULTS

The benchmark for *LPFSP* is generated from Taillard's benchmark [1] for *PFSP* with 20 jobs and 5 machines, 20 jobs and 10 machines, 20 jobs and 20 machines, 50 jobs and 5 machines, 50 jobs and 10 machines and 50 jobs and 20 machines, with the following rules:

– the number of the lots is equal with the number of the jobs;

– the job j_p from *PFSP* belongs to L_p and the processing time of j_p is the same for all jobs in L_p;

– the number of the jobs from a lot n_p is randomly generated from 1 to n.

For all instances from a benchmark set for *LPFSP* are calculated:

- the Average

$$Mean=\frac{1}{Iterations\ Number}\sum_{i=1}^{iterationsNumber} C_{\max i} \tag{22}$$

- Standard Deviation

$$SD=\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(C_{\max_i}-Mean)^2} \tag{23}$$

- Standard Score (S) - how many Standard Deviation from the $Mean$ of $C_{\max}$

$$S=\frac{C_{\max}-Mean}{SD} \tag{24}$$

- Confidence Interval (CI) of the $Mean$ with a 95% Level of Confidence.

LPFSP algorithm performs well for all the sets for 20 jobs and 50 jobs. For the problem 4 from 50 jobs and 5 machines set (Table 2), the initial solution is equal with the lower bound. For all the problems, the gap between the initial solution and the lower bound is very small. In order to see if these results are statistically significant it is provided the 95% confidence interval. The width of the confidence interval depends on the large sample size of the $C_{\max}$ set. For 20 jobs sets, the large sample size of $C_{\max}$ and the small standard deviation have combined to give small confidence interval. Also for those sets, the Score is mostly 3 or 4. Deviation is a measure of central tendency of $C_{\max}$ set. A large standard deviation value means that the $C_{\max}$ values are farther away from the *Mean*, *LPFSP*'s exploration has conducted far distant from the initial solution (for 50 jobs sets).

LPFSP algorithm, coded in Java, run on a PC INTEL.Core-i5 CPU @ 2.30 GHz processor 16 GB and the interval of the execution time is between 1 minute and 43 minutes.

Table 2

Results of *LPFSP* running over each problem from 20 jobs and 5 machines Taillard benchmark sets

Problem	**Generated sequence of jobs number for all lots**					**[CI 95%]**	
LB	**Cmax(InitSol)**	**Distance(InitSol)**	**Cmax(Sol)**	**Distance(Sol)**	**Mean**	**SD**	**Score**
1.	11 4 13 8 4 3 2 4 10 7 6 16 11 4 12 17 18 5 13 10					10467.88 10485.31	
9823.0	10326.0	0.0005121	9888.0	0.0000662	10476.59	198.72	3.0
2.	14 20 14 12 20 8 20 17 7 6 15 3 11 7 5 12 9 5 6 16					14387.59 14414.51	
13480.0	14343.0	0.0006402	13589.0	0.0000809	14401.05	306.98	3.0
3.	8 15 15 9 18 15 5 19 5 20 11 13 11 4 11 12 2 19 5 17					12976.90 13008.34	
11649.0	12723.0	0.0009220	11926.0	0.0002378	12992.62	358.46	3.0
4.	15 20 9 7 14 7 11 19 7 3 2 18 2 16 6 16 15 3 20 11					15330.47 15356.91	
14599.0	14656	0.0000390	14642.0	0.0000295	15343.69	301.52	3.0
5.	6 20 12 13 13 19 2 14 10 10 15 12 6 19 6 15 12 14 15 5					14508.01 14532.24	
13850.0	14796.0	0.0006830	13926	0,0.0000549	14520.12	276.29	3.0
6.	20 5 5 18 3 17 11 10 13 10 12 5 2 14 13 12 9 6 12 8					13024.26 13048.48	
12121.0	12376.0	0.0002104	12268.0	0.0001213	13036.37	276.22	3.0
7.	18 19 5 8 18 11 3 14 13 16 16 20 19 14 8 10 14 10 8 10					15308.94 15333.56	
14557.0	14754.0	0.0001353	14582.0	0.0000172	15321.25	280.64	3.0
8.	11 5 18 18 3 8 10 20 7 14 7 20 17 7 11 20 17 19 4 4					14802.01 14830.52	
13824.0	**13875.0**	**0.0000369**	**13875.0**	**0.0000369**	**14816.27**	**325.08**	**3.0**
9.	17 6 13 4 15 16 18 13 17 7 8 4 11 7 2 19 3 9 8 16					13376.69 13403.67	
11926.0	12501.0	0.0004821	12089.0	0.0001367	13390.18	307.68	5.0
10.	17 11 2 10 15 10 11 13 18 7 16 14 20 4 9 9 5 5 12 10					12265.17 12291.59	
10781.0	11347.0	0.0005250	11066.0	0.0002644	12278.38	301.25	5.0

Table 3

Results of *LPFSP* running over each problem from 20 jobs and 10 machines Taillard benchmark sets

Problem	**Generated sequence of jobs numbers for all lots**					**[CI 95%]**	
LB	**Cmax(InitSol)**	**Distance(InitSol)**	**Cmax(Sol)**	**Distance(Sol)**	**Mean**	**SD**	**Score**
1.	8 13 4 13 11 4 3 19 7 14 6 10 16 18 17 13 5 14 4 19					15734.58 15761.52	
13650.0	14899.0	0.0009150	14716.0	0.0007810	15748.05	307.18	4.0
2.	8 2 10 4 9 3 5 20 6 9 10 2 20 9 20 19 4 6 10 18					14814.08 14835.82	
12661.0	14201.0	0.0012163	14009.0	0.0010647	14824.95	247.89	4.0
3.	5 11 3 7 18 6 18 5 11 3 9 4 16 3 5 9 13 10 8 12					12345.76 12366.38	
10229.0	11725.0	0.0014625	11667.0	0.0014058	12356.07	235.06	3.0
4.	13 2 15 17 2 20 19 6 10 20 7 4 2 6 5 10 8 11 20 19					13970.43 13996.67	
12427.0	13346.0	0.0007395	12860.0	0.0003484	13983.55	299.23	4.0
5.	11 7 16 4 6 12 8 10 15 9 20 20 20 14 12 9 13 19 9 6					15781.79 15812.54	
13685.0	14804.0	0.0008177	14431.0	0.0005451	15797.17	350.58	4.0
6.	14 11 20 7 18 7 10 16 5 7 15 12 4 17 20 19 6 9 20 15					16168.55 16199.49	
13826.0	15038.0	0.0008766	14959.0	0.0008195	16184.02	352.81	4.0
7.	8 3 7 8 3 18 15 18 20 11 19 3 8 8 12 4 12 8 7 15					14060.32 14086.16	
13146.0	14113.0	0.0007356	13172.0	0.0000198	14073.24	294.63	4.0
8.	8 10 6 19 17 12 11 6 16 10 20 9 5 8 8 12 10 17 5 2					15103.20 15130.06	
12320.0	14507.0	0.0017752	13985.0	0.0013515	15116.63	306.28	4.0
9.	18 19 18 5 19 18 4 9 5 2 18 15 4 12 20 20 12 13 8 9					18263.83 18289.16	
16832.0	18374.0	0.0009161	17313.0	0.0002858	18276.50	288.73	4.0
10.	3 6 10 2 13 13 17 10 7 15 15 6 13 20 13 8 13 2 6 7					14269.13 14292.65	
11997.0	14193.0	0.0018305	13373.0	0.0011470	14280.89	268.12	4.0

Table 4

Results of *LPFSP* running over each problem from 50 jobs and 5 machines Taillard benchmark sets

Problem	**Generated sequence of jobs numbers for all lots**					**[CI 95%]**	
LB	**Cmax(InitSol)**	**Distance(InitSol)**	**Cmax(Sol)**	**Distance(Sol)**	**Mean**	**SD**	**Score**
1.	3 3 44 19 49 29 7 50 43 17 21 34 34 23 21 15 45 50 38 16 9 23 22 9 38 4 36 48 13 15 41 15 12 9 34 45 27 9 49 13 10 49 15 38 10 27 18 10 10 42					71145.89 71276.51	
68435.0	**68447.0**	**0.0000018**	**68447.0**	**0.0000018**	**71211.20**	**1489.33**	**2.0**
2.	32 28 24 41 46 30 35 31 40 9 28 30 2 21 49 16 24 44 5 23 7 3 14 7 26 7 17 23 49 8 36 50 26 3 35 28 31 10 22 29 12 42 42 19 9 19 49 26 8 46					73025.96 73154.72	
67965.0	**68592.0**	**0.0000923**	**68592.0**	**0.0000923**	**73090.34**	**1468.04**	**4.0**
3.	36 2 8 6 35 42 20 7 50 23 44 43 7 50 15 47 13 39 15 5 6 20 30 2 6 36 29 5 42 47 45 28 2 24 17 24 39 34 5 5 32 25 22 28 8 48 19 3 10 45					66815.96 66952.12	
60817.0	63395.0	0.0004239	62567.0	0.0002877	66884.04	1552.47	3.0
4.	14 10 16 29 3 30 2 26 16 46 42 7 46 40 26 4 31 12 4 19 47 10 21 25 19 26 28 33 29 5 5 17 23 43 5 46 22 9 13 2 40 14 45 30 41 31 47 22 37 22					71758.43 71866.20	
69677.0	**69797.0**	**0.0000172**	**69677.0**	**0.00**	**71812.32**	**1228.80**	**2.0**
5.	14 37 26 45 9 36 12 35 21 45 28 41 31 49 32 9 35 39 35 41 32 18 50 32 28 27 44 23 19 11 48 18 32 10 10 34 30 35 17 19 27 38 24 43 29 16 26 40 37 3					86508.74 86661.26	
81924.0	**82819.0**	**0.0001092**	**81924.0**	**0.00**	**86585.00**	**1739.02**	**3.0**
6.	25 41 15 35 3 47 8 10 38 24 32 17 41 25 11 36 13 23 2 41 39 11 50 48 35 38 31 5 25 16 11 15 15 19 24 43 42 36 4 32 31 3 47 13 49 22 22 35 43 18					78303.21 78418.18	
74230.0	74454.0	0.0000302	74265.0	0.0000047	78360.69	1310.84	4.0
7.	46 4 7 5 39 25 47 9 45 28 7 2 27 12 36 33 47 44 7 38 30 5 9 20 29 38 20 9 23 6 32 33 16 11 5 43 34 22 38 12 3 2 4 37 27 28 41 29 27 48					70249.44 70372.25	
64968.0	**66140.0**	**0.0001804**	**66140.0**	**0.0001804**	**70310.84**	**1400.29**	**3.0**
8.	35 41 8 35 29 46 15 31 27 28 43 34 39 23 39 24 49 18 38 15 18 32 46 15 11 3 4 28 38 17 28 46 3 26 9 38 11 9 43 36 47 12 49 22 28 10 18 9 2 5					74372.44 74516.84	
68244.0	**69685.0**	**0.0002112**	**69685.0**	**0.0002112**	**74444.64**	**1646.46**	**3.0**
9.	36 46 42 27 26 35 4 43 12 27 35 42 10 12 8 25 3 3 10 22 28 37 4 32 33 16 18 39 13 28 5 45 28 21 48 32 43 46 48 12 11 48 23 42 14 18 44 20 17 50					73293.25 73424.67	
67495.0	71406.0	0.0005795	68927.0	0.0002122	73358.96	1498.40	3.0
10.	40 47 10 44 37 15 40 36 5 31 26 23 19 42 27 14 36 48 37 14 7 34 32 42 37 17 48 3 4 33 25 6 3 32 26 37 14 39 9 47 36 38 18 31 6 9 33 47 50 23					82964.72 83105.94	
79198.0	80260.0	0.0001341	79291.0	0.0000117	83035.33	1610.26	3.0

Table 5

Results of *LPFSP* running over each problem from 50 jobs and 10 machines Taillard benchmark sets

Problem	**Generated sequence of jobs numbers for all lots**					**[CI 95%]**	
LB	**Cmax(InitSol)**	**Distance(InitSol)**	**Cmax(Sol)**	**Distance(Sol)**	**Mean**	**SD**	**Score**
1.	8 21 2 39 25 34 39 27 50 48 41 46 11 19 5 32 29 23 34 19 45 20 31 8 26 14 32 7 15 24 31 44 44 15 50 32 42 43 35 9 15 36 26 17 37 48 24 21 31 5					88609.88 88753.83	
76196.0	83751.0	0.0009915	83082.0	0.0009037	88681.85	1641.30	4.0
2.	35 38 18 12 12 30 45 4 23 45 31 34 20 20 4 7 28 10 42 20 14 4 6 50 7 2 38 34 15 34 11 25 31 28 7 46 6 32 45 24 42 22 27 40 44 10 6 35 12 18					74569.32 74688.34	
61860.0	70861.0	0.0014551	70234.0	0.0013537	74628.83	1356.96	4.0
3.	5 31 34 5 26 47 4 19 14 41 28 36 34 39 16 14 50 7 32 38 32 35 2 22 10 21 12 29 15 17 49 26 6 37 17 16 46 14 10 29 38 11 37 13 35 46 13 39 34 44					80509.37 80654.21	
68156.0	72890.0	0.0006946	72663.0	0.0006613	80581.79	1651.34	5.0
4.	19 27 50 38 41 5 35 33 18 50 11 49 14 5 36 36 31 14 43 18 26 48 41 6 48 16 27 46 44 28 36 49 9 40 37 29 47 5 27 29 28 15 12 45 14 39 47 42 30 50					97727.45 97889.53	
85168.0	93526.0	0.0009814	90007.0	0.0005682	97808.49	1847.96	5.0
5.	19 7 39 45 29 26 21 29 16 31 31 30 11 42 6 2 50 45 28 15 9 5 3 8 10 50 16 31 40 15 45 5 3 41 29 38 16 49 33 13 14 31 39 19 47 21 22 20 12 28					80144.37 80270.98	
67812.0	73257.0	0.0008030	71957.0	0.0006112	80207.67	1443.53	6.0
6.	7 42 41 12 21 32 46 16 10 32 45 31 29 5 14 20 35 20 23 50 44 38 32 33 44 9 14 42 7 41 33 4 50 11 50 43 17 45 19 23 19 11 21 44 28 30 8 16 35 47					88933.36 89078.21	
75084.0	81763.0	0.0008895	81763.0	0.0008895	89005.78	1651.49	5.0
7.	7 38 9 45 30 40 23 22 37 4 27 46 44 25 24 42 25 31 44 21 44 31 12 20 48 20 49 30 9 29 41 10 9 36 34 2 49 3 23 22 42 24 27 6 43 28 41 42 24 34					91185.20 91337.38	
77848.0	85836.0	0.0010261	84592.0	0.0008663	91261.29,	1735.18	4.0
8.	35 30 49 7 39 19 31 41 38 33 2 34 21 39 5 32 16 13 12 28 5 35 7 17 41 44 31 24 2 4 24 45 7 25 33 40 40 47 25 25 4 49 28 3 25 16 28 49 22 33					84825.91 84966.27	
74531.0	**77472.0**	**0.0003946**	**77472.0**	**0.0003946**	**84896.09**	**1600.41**	**5.0**
9.	44 17 18 49 31 21 36 19 27 48 43 42 36 21 28 26 26 36 21 4 46 12 33 21 40 33 34 24 22 21 8 21 28 6 10 17 10 10 6 31 24 13 30 21 5 22 9 9 50 8					77059.73 77193.99	
64991.0	**70819.0**	**0.0008967**	**70819.0**	**0.0008967**	**77126.86**	**1530.77**	**5.0**
10.	41 35 23 24 19 26 19 36 34 47 8 49 34 47 9 49 44 10 41 35 26 34 11 16 12 47 25 29 20 33 47 21 47 24 49 21 24 8 47 30 18 18 27 2 41 20 36 34 18 12					91879.81 92016.57	
80440.0	84878.0	0.0005517	84758.0	0.0005368	91948.19	1559.37	5.0

Table 6

Results of *LPFSP* running over each problem from 20 jobs and 20 machines Taillard benchmark sets

Problem	Generated sequence of jobs numbers for all lots					[CI 95%]	
LB	**Cmax(InitSol)**	**Distance(InitSol)**	**Cmax(Sol)**	**Distance(Sol)**	**Mean**	**SD**	**Score**
1.	9 5 5 19 11 9 5 9 18 15 8 17 18 11 19 14 5 6 18 5					18409.76 18432.51	
14729.0	18241.0	0.0023844	17611.0	0.0019567	18421.13	259.36	4.0
2.	8 4 20 6 20 12 3 10 5 13 5 17 8 10 9 11 18 2 9 15					15966.24 15987.64	
2366.0	15968.0	0.0029128	15148.0	0.0022497	15976.94	244.01	4.0
3.	2 10 9 12 15 4 11 11 15 7 13 20 12 10 15 10 19 7 17 20					19674.71 19697.36	
15009.0	19432.0	0.0029469	18875.0	0.0025758	19686.04	258.26	4.0
4.	4 17 11 8 4 18 4 10 8 9 18 15 4 18 15 7 19 3 8 5					16062.90 16085.99	
12190.0	16261.0	0.0033396	15122.0	0.0024053	16074.45	263.29	4.0
5.	16 13 10 19 7 6 5 5 2 5 18 9 7 16 12 9 3 16 16 6					16551.29 16572.38	
12965.0	16363.0	0.0026209	15766.0	0.0021604	16561.83	240.38	4.0
6.	14 16 9 17 16 15 13 9 12 13 14 18 17 16 18 16 6 19 8 18					22310.41 22341.57	
17740.0	21273.0	0.0019915	21088.0	0.0018873	22325.99	355.24	4.0
7.	13 19 11 7 9 18 10 8 11 15 17 16 7 18 4 7 8 14 15 18					19575.84 19599.50	
15351.0	19196.0	0.0025047	18536.0	0.0020748	19587.67	269.69	4.0
8.	7 7 12 7 9 9 17 15 20 16 12 15 16 20 6 8 17 20 8 18					20449.71 20472.55	
17309.0	20154.0	0.0016437	19564.0	0.0013028	20461.13	260.4	4.0
9.	9 17 4 18 20 16 4 8 7 6 2 15 15 6 11 14 14 14 9 14					18448.44 18478.03	
14142.0	17927.0	0.0026764	17414.0	0.0023137	18463.23	337.36	4.0
10.	9 8 10 18 10 14 12 16 11 17 11 20 9 8 12 3 4 2 17 13					18254.79 18278.78	
5508.0	18183.0	0.0017249	17272.0	0.0011375	18266.78	1273.60	4.0

4. CONCLUSION

Grouping the identical jobs in a lot is mainly done to improve the scheduling of the jobs and the number of identical jobs in a lot is always given by the customer needs. *TS* is used two times for *LPFSP*: for getting the initial solution and when it provides the global solution. For the initial solution *TS* starts with the initial solution provided by *NEH* [4] and solves *PFSP*. The sequence of jobs from the global solution of *PFSP* imposes the order of lots for the initial solution of *LPFSP*. The initial solution obtained applying this rule is a local optimum of *LPFSP* and can be considered successfully a global optimum of *LPFSP*.

REFERENCES

1. E. TAILLARD, *Benchmarks for basic scheduling problems*, European Journal of Operational Research, **64**, *2*, pp. 278–285, 1993.
2. F. GLOVER, *Tabu search–Part 1*, ORSA Journal on Computing, **1**, *3*, pp. 190–206, 1989.
3. F. GLOVER, *Tabu search–Part 2*, ORSA Journal on Computing, **2**, *1*, pp. 4–32, 1990.
4. M. NAWAZ, E. ENSCORE, I. HAM, *A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem*, Omega-International Journal of Management Science, **11**, pp. 91–95, 1983.
5. C. DODU, M. ANCĂU, *A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem*, Studia Universitatis Babes-Bolyai, Informatica, **65**, *1*, pp. 104–115, 2020.
6. C.N. POTTS, W.L.N. VAN, *Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity*, Journal of the Operational Research Society, **43**, pp. 395–406, 1992.
7. D.A. ROSSIT, F. TOHMÉ, M. FRUTOS, J. BARD, D. BROZ, *A non-permutation flowshop scheduling problem with lot streaming: A mathematical model*, International Journal of Industrial Engineering Computations, **7**, pp. 507–516, 2016.
8. D.A. ROSSIT, F. TOHMÉ, M. FRUTOS, *The Non-Permutation Flow-Shop scheduling problem: A literature review*, Omega, **77**, pp. 143–153, 2018.
9. M. BEN-DAYA, M. AL-FAWZAN, *A tabu search approach for the flow shop scheduling problem*, European Journal of Operational Research, **109**, *1*, pp. 88–95, 1998.
10. E. NOWICKI, C. SMUTNICKI, *A fast tabu search algorithm for the permutation flow-shop problem*, European Journal of Operational Research, **91**, pp.160–175, 1996.
11. E. TAILLARD, *Some efficient heuristic methods for the flowshop sequencing problem*, European Journal of Operational Research, **47**, pp. 67–74, 1990.

Received December 29, 2020