APEM journal
Advances in Production Engineering & Management
Volume 13 | Number 2 | June 2018 | pp 136–146
https://doi.org/10.14743/apem2018.2.279
ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Multi-criteria blocking flow shop scheduling problems: Formulation and performance analysis

Lebbar, G.[a,*], **El Abbassi, I.**[b], **Jabri, A.**[c], **El Barkany, A.**[d], **Darcherif, M.**[e]

[a,c,d]Mechanical Engineering Laboratory, Faculty of Sciences and Techniques, Fez, Morocco
[a,b,e]ECAM-EPMI, Research Laboratory in Industrial Eco-innovation and Energetics Quartz-Lab, Cergy-Pontoise, France

ABSTRACT

Most of the prior investigations related to production scheduling problems have solely focused on the optimization of an individual criterion under a single constraint; nevertheless, this is most of the time out of true in a real situation. This paper deals with multi machines permutation flow shop scheduling with limited buffers capacity and different release dates of jobs, where the performance is measured by the minimization of the weighted sum of maximum tardiness and makespan. To tackle this NP-hard problem, we present a mixed-integer linear programming model (MILP). Thereafter, using CPLEX software, we generate a set of tests in an endeavor to examine formulation for dissimilar size problems in terms of optimality solution and computational CPU time complexity. Experiment results show that overall the proposed model is computationally avaricious to solve the considering problem.

ARTICLE INFO

Keywords:
Permutation flow shop scheduling;
Tardiness;
Makespan;
Limited buffer;
Release date;
Mixed-integer linear programming model (MILP);
CPLEX software

**Corresponding author:*
ghitalebbar.gl@gmail.com
(Lebbar, G.)

Article history:
Received 8 July 2017
Revised 6 April 2018
Accepted 9 May 2018

1. Introduction

Scheduling is a technique of decision-making that greatly affects the ability to reach the success of the organizations, by drawing up a detailed operational plan, company can establish a mandate and adopt a process that will allow it to maintain the desired level of customer service, make the best use of resources, and subsequently, minimize costs and maximize profit. The core motor of this operational plan is to find the best schedule that will achieve an adequacy between a job to be performed and the resources available for its realization, in the most effective way. In fact, owing to the unpredictability of the production systems that are organized in workshops, jobs are often executed according to temporal and technological constraints.

On that account, the resolution of permutation flow shop scheduling problems (PFSP) is a longstanding challenge that has been widely studied over the last decades; this basic scheduling environment is portrayed by a set of *n* jobs, which are to be performed sequentially, by a set of *m* serial machines system in the same permutation. Henceforth, the processing steps are identical for all the manufactured products and the storage capacity buffers is limited. The case of limited storage capacity is of greatest interest as the majority of research allows at least some storage. However, limited capacity buffers give subsequently rise to situations that are known in the literature as blocking flow shop scheduling problems (BFSP). In this circumstance, a job cannot leave a machine unless the next downstream machine is available for processing. This blocking

restriction appointed as a release when starting blocking (RSB) describes one of the precedence relations, which can exist between the different structures.

Following the standard three-field notation of [1], the bicriteria *m*-machines BFSP with different release dates of jobs taking into account the maximum tardiness and makespan criteria is denoted as $F_m|block, r_k|C_{max}, T_{max}$. If works carried out till now have proven that BFSP with more than two machines is strongly NP-hard, when the makespan $C_{max}$is a measure of performance, it can be obviously inferred that the considered problem cannot be solved optimally in O(n log n).

However, a great deal of studies have been developed touching the approaches of resolution of the scheduling problems, ranging from the branch-and-bound algorithm that appear to be effective to erect exact methods in view of the considered problem complexity [2], into the heuristics aiming at obtaining approximate solutions in a reasonable calculation time [3]. In this context, concerning the RSB constraint, Trabelsi [4] proposed exact methods, approached methods and lower bound in order to determinate the best schedule reducing makespan under mixed blocking constraint for the BFSP and its extension. Wang *et al*. [5] developed a hybrid genetic algorithm for blocking flow shop scheduling problem. Tasgetiren *et al*. [6] presented iterated greedy algorithms for solving the BFSP with the makespan criterion. Sadaqua and Moraga [7] suggested meta-heuristic for randomized priority search (Meta-RaPS) to minimize makespan. Another survey is described in [8], in such study, authors proposed a branch-and-bound algorithm to deal with the same problem with total completion time criterion. Eddaly *et al*. [9] suggested a hybrid combinatorial particle swarm optimization algorithm (HCPSO) as a technique of resolution to minimize makespan of BFSP. Thus, regarding the release dates restriction, Ardakan *et al*. [10] studied two machines flow shop scheduling problem with release dates to minimize the number of tardy jobs in a permutation. Moreover, Kalcznski and Kanburowski [11] presented a new heuristic algorithm to solve the identical problem with makespan criterion. Huang *et al*. [12] proposed an improved genetic algorithm for solving JSP based on process sequence flexibility. Xu *et al*. [13] established a mathematical model based on process sequence flexibility and machine selection flexibility, and proposed an improved bat algorithm to solve it.

In an industrial environment, a decision maker often needs to take into account concurrently two or more conflicting criteria. Hence, the multi-objective flow shop scheduling problems have been the subject of extensive studies. Unfortunately, the majority of bicriteria flow shop investigations consider the combination of makespan only with total flow time or with total completion time [14-16]. Very few studies have been related to the minimization of maximum tardiness and makespan as two criteria in flow shop environment. The first being the investigation of [17], wherein the authors presented a branch-and-bound procedure for permutation flow shop with bicriteria of the both considered measure of performance and searched thereafter the set of Pareto optimal. Chakravarthy and Rajendran [18] formulated the same study with the simulated annealing approach. Allahverdi [19] proposed a new heuristic for *m*-machines flow shop scheduling problems with the objective of minimizing a weighted sum of makespan and maximum tardiness. Recently, Fernandez-Viagas and Framinan [20] have suggested two new non-population-based algorithms for the permutation flow shop scheduling problem to minimize makespan subject to a maximum tardiness. In contrast, research works, in this field, are more than welcome.

As well as, mixed-integer linear programming (MILP) represents one of detailed mathematical statement mostly used for proving the optimality status of the current master problem solution. It is based on the fact that only some of the variables are constrained to be integers, while other variables are allowed to be non-integers. Using some specific optimization software, e.g., (CPLEX, M-XPRESS, LINGO, and Lp-solve), efficient solution relative to the aforementioned mathematical formulation can be obtained for different size problems. Great effort has been devoted to the study of MILP flow shop models. The most comprehensive and interesting is that of [21], wherein, the authors discussed the computational performance of two famous families of mixed-integer linear programming models for solving the regular permutation flow shop problems with makespan criterion. Similarly, Ranconi and Bergin [22] have evaluated mixed-integer linear programming formulations for a flow shop environment with finite and infinite storage

capacity, with the aim to minimize the total earliness and tardiness. In [23], authors have provided a detailed mixed-integer model for a flow shop with mixed blocking. Moreover, Moslehi and Khorasanian [8] have presented two mixed binary integer programming models, one of which is based on the departure times of jobs from machines, and the other on the idle and blocking times of jobs

As can be seen from the above, the problem of $F_m|block, r_k|C_{max}, T_{max}$ has not been discussed with any exact or meta-heuristic method in the literature thus far. In fact, our contribution aims to propose and assess the computational performance of MILP model for multi-machines BFSP with release dates of jobs, where the objective is to minimize the weighted sum of maximum tardiness and makespan.

The remainder of the paper is organized as follows. In Section 2, the definition of the problem under study, the formulation of the proposed mixed integer linear programming and the adopted method for identifying the efficient frontier of the bicriteria model are presented. Section 3 is devoted to test problems, experiments and the computational results. Finally, conclusions and suggestions for future research areas are provided in Section 4.

2. Problem formulation and used methods

2.1 The multi-objective blocking flow shop scheduling problems under r_k constraint

The problem $F_m|block, r_k|C_{max}, T_{max}$ consists of a finite set $(J_1, \dots, J_n)$ of jobs which are to be performed sequentially on a set $(M_1, \dots, M_m)$ of machines in the same routing of process, i.e., sequence of job in all the machines system is identical, thus characterizing a permutation flow shop scheduling problem. Further, each job is composed of $(O_{k1}, \dots, O_{km})$ constructive operations O_{ki}, where, each operation has a positive integer running time equal to P_{ki} , each job has a due date d_k, and an integer release dates r_k for which it becomes available and ready for processing. It is assumed that there is no capacity buffers between machines and the job of each machine remains blocked until the next downstream machine is free. At each time, each job can be performed solely on one machine and each machine can process only one job at the time. Eventually, the setup times are included in the routing times of operations and the process cannot be interrupted at any time, to be resumed later on another machine. It is presumed that there is no idle time on machine 1, and there is no blocking time on the last machine.

In order to illustrate the BFSP with different release dates of jobs, an example is provided in Fig. 1, it exemplifies a simple case of the afore-mentioned snag, where three machines and four jobs are considered. Further, given a processing sequence $P = (J_1, J_2, J_3, J_4)$, the running times of jobs on machines and other required data are summarized in Table 1.

From the above example, it can be sighted that there is an RSB constraint between machine 1 and machine 2, therefore, machine 1 rests blocked by a job 2 until job 1 finishes processing in the second machine. The example shows also, that there is an intermediate idle time between jobs 2 and 3 processed by machines 2 and 3, respectively. Besides, even if the first machine becomes free after blocking, it can be observed that, there in a non- productive times during which jobs are not ready for processing due to their release dates that must be respected. Hence, taking into account the date of availability of each job in the first machine must be considered foremost.

Table 1 Example data

Jobs / Machines	1	2	3	4
1	1	3	6	2
2	4	2	2	3
3	2	1	3	2
Release dates	2	3	8	4

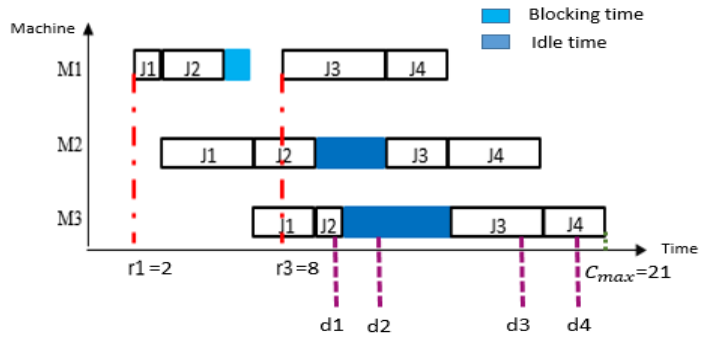

Fig. 1 Gantt chart of BFSP example with different release dates

The target of this study is to obtain the topper schedule, which minimizes the weighted sum of maximum tardiness, and makespan, respectively. According to [17], those scheduling criteria of interest can be formulated as follows:

$$C_{max} = \max_{j\in[1....n]} C_{jm} \tag{1}$$

$$T_{max} = \max_{j\in[1....n]} T_{jm} \tag{2}$$

where, $j = 1, \dots, n$, m is the last machine in the production system. C_{jm} is a completion time of job j on the machine m and $T_{jm} = \max(C_{jm} - d_k, 0)$ represents the tardiness of job j.

The weighted objective function method allows identifying the efficient frontier for the multicriteria problem. This intuitive technique is most broadly used given its relevance proficiency regarding the convex problems, for which, it guarantees to find solutions on the entire Pareto-optimal set. Thereby, the aim is to return to a mono-objective problem of optimization, for which there exist several methods of resolution. The simplest way is to assign to each objective function $f_i(x), i = 1, \dots, p$, an appropriate user-supplied weight factor w_i according to the relative importance of each objective, so as to, obtain an aggregated scalar objective function. The product that provides the scalar objective $F(x, w)$ can be posed as follows:

$$F(x, w) = \sum_{i=1}^{p} w_i f_i(x) \tag{3}$$

That is, $\sum_{i=1}^{p} w_i = 1$ and $w_i > 0$, these values have to be normalized if the scales and units of the objectives are different as reported by [22], also they cannot be equal to 0, in order that the set of solutions does not occur narrow Pareto optimality.

Since the scales and the units of our considered criteria scheduling are similar. The weights have no to be normalized. Consequently, the objective function of the problem under study may be computed as:

$$Z = \alpha\, C_{max} + \beta\, T_{max} \tag{4}$$

Where, α and β are the weights factors of makespan and maximum tardiness. As long as, $\alpha, \beta > 0$ and α+β=1, the objective function can then be expressed in the following way:

$$Z = \alpha(\max_{j\in[1....n]} C_{jm}) + (1-\alpha)\max(\max_{j\in[1....n]}(C_{jm} - d_k, 0)) \tag{5}$$

2.2 The proposed mathematical formulation

In this section, a mixed-integer linear programming model for the $F_m|block, r_k|C_{max}, T_{max}$ problem is formulated. With the addition of the release dates of jobs constraint, this model is inspired from the models presented in [23]. We have adapted our model to consider the different con-

straints and the measures of performance afore-mentioned at the same time. Indices, parameters and variables being using in this formulation are given below.

Indices

k Job index
i Machine index
j Index on the position of job k in sequence P

Parameters

n Total number of jobs, $k = 1, 2,\ldots, n$
m Total number of machines, $i = 1,2,\ldots,m$
t_{ki} Processing time of job k on the *i-th* machine, $k = 1, 2,\ldots,n$, $i = 1, 2,\ldots, m$
D_k Due date of job k, $k = 1, 2,\ldots, n$
r_k Release date of job k, $k = 1, 2,\ldots, n$
α Weight associated with makespan
β Weight associated with total tardiness
$Z(P)$ Objective value of schedule P
ε_k $\begin{cases} 1 & \text{If there is a blocking constraint between machine } M_i \text{ and machine } M_{i+1} \\ 0 & \text{Otherwise} \end{cases}$
μ_k $\begin{cases} 1 & \text{If there is an idle time between machine } M_i \text{ and machine } M_{i+1} \\ 0 & \text{Otherwise} \end{cases}$
θ_k $\begin{cases} 1 & \text{If there is no blocking constraint and no idle time} \\ 0 & \text{Otherwise} \end{cases}$

Variables

H_{ji} Starting time of job at position j in sequence P on the ith machine
C_{ji} Completion time of job at position j in sequence P on the ith machine
x_{kj} $\begin{cases} 1 & \text{If job } k \text{ is at position } j \text{ in the sequence } P \\ 0 & \text{Otherwise} \end{cases}$

The MILP model

For the modelization of the $F_m|block, r_k|C_{max}, T_{max}$ problem, the $\mu_k, \varepsilon_k, \theta_k$ and x_{kj} are binary parameters and variables, the others are given as integer values. The mixed-integer linear programming MILP is provided as follows:

$$\text{Min } Z(p) = \alpha\ Cmax + (1-\alpha)\ Tmax \qquad \alpha > 0 \tag{6}$$

subject to

$$\sum_{k=1}^{n} x_{kj} = 1 \quad \forall j \in \{1 \ldots n\} \tag{7}$$

$$\sum_{j=1}^{m} x_{kj} = 1 \quad \forall k \in \{1 \ldots n\} \tag{8}$$

$$Cmax \geq C_{jm} \quad \forall j \in \{1 \ldots n\} \tag{9}$$

$$Tmax \geq C_{jm} - \sum_{k=1}^{n} x_{kj}\ D_k \quad \forall j \in \{1 \ldots n\} \tag{10}$$

$$H_{j1} \geq \sum_{k=1}^{n} r_k\ x_{kj} \quad \forall j \in \{1 \ldots n\} \tag{11}$$

$$H_{ji} \geq H_{j-1,i} + \sum_{k=1}^{n} t_{ki}\, x_{kj} \quad \forall\, j \in \{2 \ldots n\} \quad \forall\, i \in \{1 \ldots m\} \tag{12}$$

$$H_{ji} \geq H_{j,i-1} + \sum_{k=1}^{n} t_{ki-1}\, x_{kj} \quad \forall\, j \in \{1 \ldots n\} \quad \forall\, i \in \{2 \ldots m\} \tag{13}$$

$$H_{ji} \geq H_{j-1,i+1} \quad \forall\, j \in \{2 \ldots n\} \quad \forall\, i \in \{2 \ldots m\} \tag{14}$$

$$H_{ji} \geq C_{j-1,i}\theta_k + H_{j-1,i+1}\,\varepsilon_k + C_{j,i-1}\,\mu_k \quad \forall\, j \in \{2 \ldots n\} \quad \forall\, i \in \{2 \ldots m\} \tag{15}$$

$$C_{j1} \geq \sum_{k=1}^{n} r_k\, x_{kj} + \sum_{k=1}^{n} t_{k1}\, x_{kj} \quad \forall\, j \in \{1 \ldots n\} \tag{16}$$

$$C_{ji} \geq H_{ji} + \sum_{k=1}^{n} t_{ki}\, x_{kj} \quad \forall\, j \in \{1 \ldots n\} \tag{17}$$

$$C_{jm} \geq H_{jm} + \sum_{k=1}^{n} t_{ki}\, x_{kj} \quad \forall\, j \in \{1 \ldots n\} \tag{18}$$

$$x_{kj} \epsilon \{0,1\} \quad \forall\, k \in \{1 \ldots . n\}\, \forall\, j \in \{1 \ldots n\} \tag{19}$$

$$H_{ji} \geq 0 \quad \forall\, j \in \{1 \ldots . n\} \quad \forall\, i \in \{1 \ldots m\} \tag{20}$$

$$C_{ji} \geq 0 \quad \forall\, \mathrm{j} \in \{1 \ldots . n\} \quad \forall\, \mathrm{i} \in \{1 \ldots m\} \tag{21}$$

The objective function (Eq. 6) considers the minimization of the weighted sum of makespan and maximum tardiness. Constraints set (Eq. 7) ensures that every position of the sequence has to incorporate exactly one job. Constraint set (Eq. 8) shows that each job has to be set in the sequence only one time. Constraint set (Eq. 9) indicates that the makespan must be greater than or equal the maximum completion time on the last machine. The maximum tardiness is calculated in constraint set (Eq. 10). Constraint set (Eq. 11) forces jobs to start the processing after their release times. Constraint set (Eq. 12) states the precedence constraint between two operations that have to be performed by the same machine, i.e., in order that a machine starts the forthcoming operation, the operation in a process must be completed. Constraint set (Eq. 13) gives the precedence constraint between two successive operations of the same job, that is, the job cannot start its operation in the downriver machine until it finishes its operation in the upriver machine. RSB constraint between machine M_i and machine M_{i+1} is modeled in constraint set (Eq. 14). Further, constraint set (Eq. 15) designates that, for each starting time H_{ji}, only one constraint will be active. The constraint set (Eq. 15) calculates the completion time of the job in the first machine taking into account the date of availability of each job. Constraints set (Eq. 17) and (Eq. 18) corresponds to the computation of the completion time of jobs in the machine M_i and the machine M_m, respectively. Moreover, (Eq. 19) mentions that x_{kj} is a boolean variable, it is equivalent to 1 if job j_k is assigned to the *j-th* position of sequence and 0 otherwise. Finally, constraints set (Eq. 20) and (Eq. 21) are the logical constraints that state the non-negativity constraints for the starting and the completion times of job.

2.3 The branch-and-bound algorithm

The branch-and-bound algorithm consists of a systematic enumeration of candidate solutions, each node of the tree corresponds to a subproblem designated to as the search tree created by candidate solution, In each iteration of a B&B algorithm, a node is picked for exploration from the pool of live nodes corresponding to unexplored feasible subproblems using some selection procedures. Before calculating the candidate solution, the branch is investigated against upper and lower bounds, if it cannot produce a fitter solution than the best one found so far by the al-

gorithm, the branch is weeded out. Generally, the algorithm goes through two main steps. In the first, it is a question of separating a set of solutions into subsets; whereas in the second phase, it consists of evaluating the solutions of a subset by increasing the value of the best solution of this subset. The search stops when there is no unexplored parts of the solution space left, and the optimal solution is then the one recorded as current best [24]. The procedure is considered as branching downward as represented in Fig. 2.

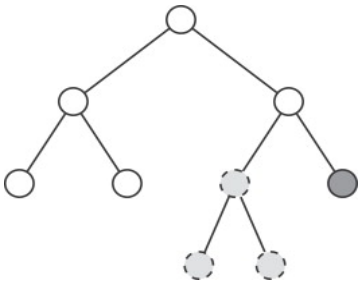

Fig. 2 Branch and bound procedure

3. Results and discussion

3.1 Test problems and experiments

In order to assess the computational behavior of the proposed MILP model, difficult problem instances should be considered. As a matter of fact, problem size was varied with the instances mentioned bellow.

$$(n, m) = \{(10,7), (10,10), (15,3), (15,7), (20,3), (20,7)\}$$

In the experiment design adopted for the considered problem, processing times were randomly generated as uniformly distributed integers, with range (1, 100). As reported by [25], there exist two reasons for generating routing times between 1 and 100. The first is related to the historical evenness. i.e., the majority of research works dealing with the computational tests including scheduling problems, sample processing times from a uniform distribution on (1, 100) or a normal distribution with a mean of 100 and a standard deviation as large as 25. The second reason concerns the fact that it is better to use data that are representative of real scheduling problems. Generating processing times from the uniform distribution of small interval such as (1, 10) for example, are much easier because the dominance rules of [26] create many more ties. As result, optimal solutions become easier to find, which would not necessarily be realistic due to the unsuitable conclusions that may result.

According to several research works that have been done on the test problems for the maximum tardiness criterion, it can be spotted that problem "hardness" depends on two parameters *TF* and *DR* [27]. Where the first represent the tardiness factor that affects the average number of jobs tardy, and the second parameter corresponds practically to the dispersion range of due dates that controls their variance, respectively. The aforementioned parameters constitute the range from which the due dates of jobs have to be generated. In this context, while *P* represents the sum of processing times of all jobs, the due dates of jobs d_k were generated randomly using a uniform distribution on the interval:

$$d_k \, \epsilon \left[P\left(1 - TF - \frac{DR}{2}\right), P\left(1 - TF + \frac{DR}{2}\right) \right] \tag{22}$$

The values of *TF* and *DR* have been set following the different scenarios shown in Table 2, Those scenarios represent the different configuration adopted by [22] to fix *TF* and *DR* parameters.

Table 2 Due date range and tardiness factor scenarios

Scenarios	Tardiness factor	Due date range
Scenario 1	0.2	0.6
Scenario 2	0.2	1.2
Scenario 3	0.4	0.6
Scenario 4	0.4	1.2

The release date of jobs r_k is another integer parameter generated from a uniform distribution (0, QP), as pointed out by [28], Q is a factor that determines the range of distribution, it has been set to 0.4, and P is the sum of processing times of all jobs. Further, for every problem size, the weight factor α took three different values of 0.25, 0.5 and 0.75, respectively.

As a result, these choices beget 12 different configuration combinations. Thus, as 6 test problems for each combination were generated, 72 experimental conditions were tested.

The proposed MILP were coded in IBM ILOG AMPL 12.6.0 and then solved using CPLEX 12.6.0 software. The digital works were executed in a computer microprocessor core i3, with 4.0 GB of RAM memory. We imposed 3600 seconds as a time limit, i.e., if the optimal solution had not been found, and verified in that amount of time, the process will be stopped.

3.2 Computational results

The computational results are presented in Table 3, this table summarizes the computational CPU time in seconds used by CPLEX to solve test problems and the total number of nodes generated and explored during the branch and bound procedure.

Table 3 Summary of computational results

n	m	α	TF	DR	CPU time (s)	B & B nodes
10	7	0.25	0.2	0.6	1.78	4284
				1.2	1.88	3723
			0.4	0.6	0.89	4197
				1.2	0.85	1143
		0.5	0.2	0.6	1.09	3860
				1.2	0.67	2794
			0.4	0.6	1.04	1617
				1.2	0.95	1302
		0.75	0.2	0.6	1.94	2374
				1.2	1.93	2119
			0.4	0.6	1.25	1229
				1.2	1.00	1068
10	10	0.25	0.2	0.6	3.56	5784
				1.2	1.72	4604
			0.4	0.6	2. 46	3666
				1.2	2.01	3537
		0.5	0.2	0.6	5.58	8684
				1.2	3.65	4244
			0.4	0.6	2.20	5505
				1.2	1.56	5185
		0.75	0.2	0.6	4.60	5343
				1.2	3.58	4044
			0.4	0.6	3.67	4957
				1.2	2.01	3205

Table 2 (Continuation)

n	*m*	α	*TF*	*DR*	*CPU Time(s)*	B & B *Nodes*
15	3	0.25	0.2	0.6	15.94	17636
				1.2	13.91	14763
			0.4	0.6	13.03	13417
				1.2	12.84	11639
		0.5	0.2	0.6	16.87	20997
				1.2	12.08	20642
			0.4	0.6	14.98	15364
				1.2	11.88	15014
		0.75	0.2	0.6	15.08	19100
				1.2	13.14	18445
			0.4	0.6	14.03	15885
				1.2	6.05	14167
15	7			1.2	79.25	40577
		0.5	0.2	0.6	125.08	65007
				1.2	98.11	31942
			0.4	0.6	74.08	54228
				1.2	30.11	36061
		0.75	0.2	0.6	95.11	63159
				1.2	83.13	54137
			0.4	0.6	89.18	51775
				1.2	45.09	49062
20	3	0.25	0.2	0.6	150.87	170900
				1.2	101.03	100749
			0.4	0.6	135.73	149889
				1.2	109.76	99500
		0.5	0.2	0.6	268.69	198901
				1.2	198.21	104379
			0.4	0.6	233.67	130612
				1.2	100.85	99794
		0.75	0.2	0.6	210.8	164491
				1.2	199.61	110355
			0.4	0.6	254.03	146881
				1.2	216.00	122471
20	7	0.25	0.2	0.6	399.82	233430
				1.2	345.63	230743
			0.4	0.6	397.29	221390
				1.2	296.36	212214
		0.5	0.2	0.6	419.40	323640
				1.2	401.20	283696
			0.4	0.6	399.70	303735
				1.2	321.56	232126
		0.75	0.2	0.6	419. 94	383687
				1.2	212.36	155537
			0.4	0.6	385.88	218405
				1.2	369.55	205890

The first experiments used test problem containing ten jobs and three machines, At this problem size appointed as small, the proposed MILP is able to find the optimal solution within a time limit and more precisely, within reasonable computation times. Nevertheless, for wide size problem, MILP requires relatively excessive amounts of time and the number of nodes explored rises quickly for some problems.

As can be seen from Table 2, several factors related to the experimental design may clarify the sizable dissimilarity in the problem hardness. The most significant revolve around the tardi-

ness factor TF and the due date range factor DR. By fixing α and DR and focusing on the increase of the value of TF, it can be perceptible that the CPU time and the number of nodes explored become smaller. This is due to the trend of manifold jobs to be attributed untimely due date when the tardiness factor increase, which subsequently generate a considerable number of tardy jobs. On the other side, as the value of DR increases, little time variation occurs in solving problem, and search tree becomes small when α and TF are fixed. This outcome can be explained by the feebleness of the bounds resulting from the feasible schedule. When it comes to a wide number of schedules that must be assessed, the computation times increase.

Furthermore, for problem size with more than 15 jobs, the computation times and the number of nodes are comparatively large, this is due to the data taken into account in the experimental design that is tough to evaluate. For instance, when the weight factor, the tardiness factor, and the due date range factor are 0.25, 0.2 and 0.6, respectively, the computation times and the number of nodes for problem with job size equal to 20 represent 99.55 % and 98.16 % of the CPU time and the number of nodes for problem with job size equal to 10. Wholesale, most of the problems can be solved within a bearable time. As the experiment shows, all the 72 test problems were solved to completion in less than seven minutes of computing times.

4. Conclusion

This paper examined the computational properties of an MILP developed to solve optimally the blocking flow shop scheduling problem with different release dates of jobs, where the objective is to minimize the weighted sum of makespan and maximum tardiness. It highlighted that the aspect of the developed MILP is on a large scale greatly in quality of the computation times and the total number of nodes explored in the branch and bound tree. On top of that, the suggested MILP can optimally solve the problem within a reasonable time for a small size problem, but it involves an immoderate amount of time and search tree becomes wide in larger size problems. Notwithstanding, results are considered tolerable since the proposed formulation requires less than seven minutes of computing times.

Based on the research described in this paper, multiples fruitful research areas involving the considered problem comes to mind. Correspondingly, it will be interesting to develop heuristic algorithms for large-sized problems guaranteeing the resolution in a reasonable calculation time. Besides, the development of a powerful exact method for the BFSP under release dates constraint and with maximum tardiness and makespan criteria is also an interesting issue for study. Finally, it will be useful to prolong the bicriteria approach considered in our study to other complex scheduling environments, in particular into the hybrid flow shop scheduling problems.

References

[1] Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics,* Vol. 5, 287-326, doi: 10.1016/S0167-5060(08)70356-X.

[2] Chu, C. (1992). A branch-and-bound algorithm to minimize total tardiness with different release dates, *Naval Research Logistics (NRL),* Vol. 39, No. 2, 265-283, doi: 10.1002/1520-6750(199203)39:2<265::AID-NAV3220390209>3.0.CO;2-L.

[3] Lebbar, G., El Barkany, A., Jabri, A. (2016). Scheduling problems of flexible manufacturing systems: Review, classification and opportunities, *International Journal of Engineering Research in Africa*, Vol. 26, 142-160, doi: 10.4028/www.scientific.net/JERA.26.142.

[4] Trabelsi, W. (2012). Ordonnancement des systèmes de production flexibles soumis à différents types de contraintes de blocage, Recherche opérationnelle [cs.RO], Universite de Lorraine, France.

[5] Wang, L., Zhang, L., Zheng, D.-Z., (2006). An effective hybrid genetic algorithm for flow shop scheduling with limited buffers, *Computers & Operations Research*, Vol. 33, No. 10, 2960–2971, doi: 10.1016/j.cor.2005.02.028.

[6] Tasgetiren, M.F., Kizilay, D., Pan, Q.-K., Suganthan, P.N. (2017). Iterated greedy algorithms for the blocking flow shop scheduling problem with makespan criterion, *Computers & Operations Research*, Vol. 77, 111-126, doi: 10.1016/j.cor.2016.07.002.

[7] Sadaqa, M., Moraga, R.J. (2015). Scheduling blocking flow shops using Meta-RaPS, *Procedia Computer Science*, Vol. 61, 533-538, doi: 10.1016/j.procs.2015.09.211.

[8] Moslehi, G., Khorasanian, D. (2013). Optimizing blocking flow shop scheduling problem with total completion time criterion, *Computers & Operations Research*, Vol. 40, No. 7, 1874-1883, doi: 10.1016/j.cor.2013.02.003.
[9] Eddaly, M., Jarboui, B., Siarry, P. (2016). Combinatorial particle swarm optimization for solving blocking flow shop scheduling problem, *Journal of Computational Design and Engineering*, Vol. 3, No. 4, 295-311, doi: 10.1016/j.jcde.2016.05.001.
[10] Abouei Ardakan, M., Hakimian, A., Rezvan, M.T. (2014). A branch-and-bound algorithm for minimizing the number of tardy jobs in a two-machine flow-shop problem with release dates, *International Journal of Computer Integrated Manufacturing*, Vol. 27, No. 6, 519-528, doi: 10.1080/0951192X.2013.820349.
[11] Kalczynski, P.J., Kamburowski, J. (2012). An empirical analysis of heuristics for solving the two-machine flow shop problem with job release times, *Computers & Operations Research,* Vol. 39, No. 11, 2659-2665, doi: 10.1016/j.cor.2012.01.011.
[12] Huang, X.W., Zhao, X.Y., Ma, X.L. (2014). An improved genetic algorithm for job-shop scheduling problem with process sequence flexibility, *International Journal of Simulation Modelling*, Vol. 13, No. 4, 510-522, doi: 10.2507/IJSIMM13(4)CO20.
[13] Xu, H., Bao, Z.R., Zhang, T. (2017). Solving dual flexible job-shop scheduling problem using a Bat Algorithm, *Advances in Production Engineering & Management*, Vol. 12, No. 1, 5-16, doi: 10.14743/apem2017.1.235.
[14] Yeh, W.-C. (1999). A new branch-and-bound approach for the $n/2/$flow shop$/\alpha F+\beta C_{\max}$ flow shop scheduling problem, *Computers & Operations Research*, Vol. 26, No. 13, 1293-1310, doi: 10.1016/S0305-0548(98)00106-3.
[15] Yandra, Tamura, H. (2007). A new multiobjective genetic algorithm with heterogeneous population for solving flow shop scheduling problems, *International Journal of Computer Integrated Manufacturing*, Vol. 20, No. 5, 465-477, doi: 10.1080/09511920601160288.
[16] Sayin, S., Karabati, S. (1999). Theory and methodology a bicriteria approach to the two-machine flow shop scheduling problem, *European Journal of Operational Research,* Vol. 113, No. 2, doi: 10.1016/S0377-2217(98)00009-5.
[17] Daniels, R.L., Chambers, R.J. (1990). Multiobjective flow-shop scheduling, *Naval Research Logistics (NRL),* Vol. 37, No. 6, 981-995, doi: 10.1002/1520-6750(199012)37:6<981::AID-NAV3220370617>3.0.CO;2-H.
[18] Chakravarthy, K., Rajendran, C. (1999). A heuristic for scheduling in a flow shop with the bicriteria of makespan and maximum tardiness minimization, *Production Planning & Control*, Vol. 10, No. 7, 707-714, doi: 10.1080/095372899232777.
[19] Allahverdi, A. (2004). A new heuristic for *m*-machine flow shop scheduling problem with bicriteria of makespan and maximum tardiness, *Computers & Operations Research*, Vol. 31, No. 2, 157-180, doi: 10.1016/S0305-0548(02)00143-0.
[20] Fernandez-Viagas, V., Framinan, J.M. (2015). Efficient non-population-based algorithms for the permutation flow shop scheduling problem with makespan minimization subject to a maximum tardiness, *Computers & Operations Research*, Vol. 64, 86-96, doi: 10.1016/j.cor.2015.05.006.
[21] Stafford Jr, E.F., Tseng, F.T., Gupta, J.N.D. (2005). Comparative evaluation of MILP flow shop models, *Journal of the Operational Research Society*, Vol. 56, No. 1, 88-101, doi: 10.1057/palgrave.jors.2601805.
[22] Ronconi, D.P., Birgin, E.G. (2012). Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness, In: Ríos-Mercado, R., Ríos-Solís, Y. (eds.), *Just-in-Time Systems, Springer Optimization and Its Applications,* Vol. 60, Springer, New York, 91-105, doi: 10.1007/978-1-4614-1123-9_5.
[23] Trabelsi, W., Sauvey, C., Sauer, N. (2012). Heuristics and metaheuristics for mixed blocking constraints flow shop scheduling problems, *Computers & Operations Research*, Vol. 39, No. 11, 2520-2527, doi: 10.1016/j.cor.2011.12.022.
[24] Clausen, J. (1999). *Branch and bound algorithms – Principles and examples,* Department of Computer Science, University of Copenhagen, 1-30.
[25] Baker, K.R., Keller, B. (2010). Solving the single-machine sequencing problem using integer programming, *Computers & Industrial Engineering*, Vol. 59, No. 4, 730-735, doi: 10.1016/j.cie.2010.07.028.
[26] Emmons, H. (1969). One-machine sequencing to minimize certain functions of job tardiness, *Operations Research,* Vol. 17, No. 4, 701-715, doi: 10.1287/opre.17.4.701.
[27] Potts, C.N., Van Wassenhove, L.N. (1982). A decomposition algorithm for the single machine total tardiness problem, *Operations Research Letters*, Vol. 1, No. 5, 177-181, doi: 10.1016/0167-6377(82)90035-9.
[28] Parsamanesh, A.H., Sahraeian, R. (2015). Solving single machine sequencing to minimize maximum lateness problem using mixed integer programming, *Journal of Quality Engineering and Production Optimization*, Vol. 1, No. 1, 33-42, doi: 10.22070/JQEPO.2015.187.