Scientific Research Publishing

American Journal of Operations Research, 2021, 11, 15-34
https://www.scirp.org/journal/ajor
ISSN Online: 2160-8849
ISSN Print: 2160-8830

Identical Machine Scheduling Problem with Sequence-Dependent Setup Times: MILP Formulations Computational Study

Farouk Yalaoui, Nhan Quy Nguyen

Chaire Connected Innovation, ICD-LOSI, Université de Technologie de Troyes, Troyes, France
Email: farouk.yalaoui@utt.fr, nhan_quy.nguyen@utt.fr

How to cite this paper: Yalaoi, F. and Nguyen, N.Q. (2021) Identical Machine Scheduling Problem with Sequence-Dependent Setup Times: MILP Formulations Computational Study. *American Journal of Operations Research*, **11**, 15-34.
https://doi.org/10.4236/ajor.2021.111002

Received: September 8, 2020
Accepted: January 18, 2021
Published: January 21, 2021

Abstract

This work aims to give a systematic construction of the two families of mixed-integer-linear-programming (MILP) formulations, which are graph-based and sequence-based, of the well-known scheduling problem $P_m \mid r_j, s_{ij} \mid \sum C_j$. Two upper bounds of job completion times are introduced. A numerical test result analysis is conducted with a two-fold objective 1) testing the performance of each solving methods, and 2) identifying and analyzing the tractability of an instance according to the instance structure in terms of the number of machines, of the jobs setup time lengths and of the jobs release date distribution over the scheduling horizon.

Keywords

Identical Machine Scheduling Problem, Release Date, Sequence Dependent Setup Time

1. Introduction

Many machine scheduling problems practically require setup times prior to the processing of jobs. It is common that the setup times of one job are dependent on the last job that has been previously processed on the same machine. For instance, in the package printing industry, the time taken to prepare the ink colors for a task is dependent on the colors that have been used on the last printing task. In this paper, we consider an identical parallel machine scheduling problem with jobs' sequence-dependent setup times and release dates. The problem's objective is to minimize the total completion time. This problem is denoted by the three fields notation [1] as $P_m \mid r_j, s_{ij} \mid \sum C_j$.

Parallel machine scheduling with job setup times problem is widely studied in the literature. Readers can refer to the survey of Allahverdi [2] for a broader family of similar problems. For that reason, we limit our literature review to the problem $P_m | r_j, s_{ij} |.$ with time criteria objectives. To the best of our knowledge, the first study of this problem dates back to 1993 by Guinet [3]. Nessah *et al.* [4] proved that the problem $P_m | r_j, s_{ij} | \sum C_j$ is NP-Hard in the strong sense. They solved the problem by an efficient Branch-and-Bound algorithm with a strong dominance property. To solve the problem with the total weighted completion time minimization objective, Fowler *et al.* [5] developed a hybridized genetic algorithm with dispatching rules. To solve the problem with the makespan minimization objective, Montoya-Torres *et al.* [6] established a randomized search heuristic. Lin *et al.* [7] proposed a genetic algorithm to solve the minimization of the maximum lateness. However, those mentioned works have not introduced any mathematical formulation of the problem with the min-sum criterion.

As far as we know, Kurz and Askin [8] were the first to construct an integer programming for the $P_m | r_j, s_{ij} | C_{\max}$ problem. However, their formulation, which is graph-based, contains one non-linear constraint to ensure the completion times of two consecutive jobs. Anderson *et al.* [9] developed a network-based mixed-integer-linear programming (MILP) to the earliness/tardiness minimization problem. Nonetheless, the formulation included the big-M whose value is known to be very impacting on the model solving time. One can find an extensive analysis of the formulation of a similar scheduling problem on a single machine environment in the paper of Nogueira *et al.* [10]. This work aimed to structure many families of formulations of the single-machine environment. The paper proposed new upper bounds for all formulations and compared their performances by thorough numerical tests. According to our knowledge, a similar analysis is still a void in the literature for identical parallel machine scheduling.

For this reason, we extend the works of Nessah *et al.* [4] and Nogueira *et al.* [10] to analyze the performances of different exact approaches/MILP formulations on the problem. The current paper is organized as follows:

- In Section 2, we introduce two MILP formulations: graph-based formulation and sequence-based formulation. We establish tight upper bounds for the completion times of jobs for each formulation.
- In Section 3, we apply the test protocol introduced by Nessah *et al.* [4] to 1) quantify how three factors of the data structure (the arrival density, the setup time relative length and the number of machines) would impact the tractability of resolutions methods and 2) compare the performance of resolution methods for each instance structure.
- In Section 4, we draw conclusions from the numerical tests and present four perspectives of future researches.

2. Mathematical Formulation

We consider a problem where one has to schedule n jobs on m identical machines. m should be strictly inferior to n unless the problem becomes trivial. Each job j has a release date r_j when it is ready to process and a sequence-dependent setup time s_{ij} if it is preceded immediately by job *i* on the same machine. We consider three following assumptions:

- The setting up of a job can be done before the release date of this job.
- The first job to be processed in each machine does not need to be setup because it is preceded by no other job.
- All the values of the setup time must satisfy the triangle inequality $s_{ij} \le s_{ik} + s_{kj}$ for all job *i*, *j*, *k*. This assumption is important to avoid the insertion of a job *k* with $p_k = 0$ and $s_{ij} < s_{ik} + s_{kj}$ for some jobs *i* and *j*.

Sets and parameters

- $\mathcal{J} = 1, \cdots, n$: set of jobs.
- $\mathcal{M} = 1, \cdots, m$: set of machines.
- r_j : release date of job *j*.
- p_j : processing time of job *j*.
- s_{ij} : Setup time if job *j* follows job *i* to be processed on the same machine.

2.1. A Graph-Based Formulation

We introduce a dummy job, Job 0, which has processing time equal to zero ($p_0 = 0$) and it is available at the beginning of the scheduling horizon ($r_0 = 0$). The setup times required all jobs preceding task 0 are also zero. To limit the number of identical machines we must have a constraint that assures at most m jobs can connect to job 0.

Set

- $\mathcal{J}' = 0, \cdots, n$: set of jobs including imaginary job 0.

Decision variables

- C_j : the completion time of job *j*.
- x_{ij} : the sequence decision variable. $x_{ij} = 1$ if job *j* processes right after job *i* on the same machine (denoted by $i \to j$); otherwise $x_{ij} = 0$.

Figure 1 illustrates a solution to an instance that has 10 jobs and 3 machines. The graph on the above depicts the sequencing of jobs shown in the table shown on Figure 1.

Formulation GF

$$\text{Minimize} \quad \sum_{j \in \mathcal{J}} C_j \tag{1}$$

subject to

$$C_j \ge r_j + p_j, \quad \forall j \in \mathcal{J} \tag{2}$$

$$C_j \ge C_i + s_{ij} + p_j - \left(1 - x_{ij}\right)\left(\bar{C}_i + s_{ij} - r_j\right), \quad \forall i, j \in \mathcal{J} \tag{3}$$

$$\sum_{j \in \mathcal{J}} x_{0j} \le m \tag{4}$$

$$\sum_{i \in \mathcal{J}'} x_{ij} = 1, \quad \forall j \in \mathcal{J} \tag{5}$$

$$\sum_{i \in \mathcal{J}} x_{ji} \le 1, \quad \forall j \in \mathcal{J} \tag{6}$$

$$x_{jj} = 0, \quad \forall j \in \mathcal{J} \tag{7}$$

$$\begin{tikzcd}
 & 5 \arrow[r] & 4 \arrow[r] & 7 \arrow[r] & 9 \\
0 \arrow[ur] \arrow[r] \arrow[dr] & 8 \arrow[r] & 6 & & \\
 & 2 \arrow[r] & 1 \arrow[r] & 10 \arrow[r] & 3
\end{tikzcd}$$

x	0	1	**2**	3	4	**5**	6	7	**8**	9	10
0			**1**			**1**			**1**		
1											1
2		1									
3											
4								1			
5					1						
6											
7										1	
8							1				
9											
10				1							

Figure 1. An example of the graph-based formulation with 10 jobs and 3 machines.

$$C_j \geq 0, \ \forall i, j \in \mathcal{J} \tag{8}$$

$$x_{ij} \in \{0,1\}, \ \forall i, j \in \mathcal{J} \tag{9}$$

with $\bar{C}_i$ is an upper bound of the completion time of task *i*.

Constraint (2) makes sure that a job *j* must start after its release date (r_j). Constraint (3) ensures the equation $C_j \geq C_i + S_{ij} + p_j$ to be valid only if job *i* precede job *j*, *i.e.* $x_{ij} = 1$. Otherwise, $C_i - \bar{C}_i + p_j + r_j \leq p_j + r_j \leq C_j$ because $C_i < \bar{C}_i$. By constraint (4), at most *m* jobs can follow job 0 because of *m* available machines. Constraint (5) ensures that each job, except job 0, can have only one precedence. Constraint (6) limits each job, except job 0, to have at most one follower. Constraint (7) forbids a job to follow itself.

We develop an upper bound for the completion time of job *j* for the identical machine scheduling environment, based on the work of Nogueira *et al.* [10] for the single machine environment.

Theorem 1. There is an optimal schedule such that the number of jobs scheduled on any machine is less than or equal to $n - m + 1$ (Yalaoui and Chu [11]).

Proposition 2. $\bar{C}_j^M = \max\left\{r_j, \max_{i \in \mathcal{J}, i \neq j}(r_i) + \sum_{i=m}^{n} \rho_{[i]} - \rho_j\right\} + p_j$ is an upper bound of job *j*'s completion time C_j according to theorem 1, where:

- $\rho_j = p_j + s_{j\bullet}$.
- $\rho_{[j]}$ is the *j*th sorted element of ρ in a non-decreasing order.
- $s_{j\bullet} = \max_{k \in \mathcal{J}, k \neq j} s_{jk}$.

Proof. We consider firstly a schedule with one machine where each task has the maximum setup time for any following job: $s_{j\bullet} = \max_{k \in \mathcal{J}, k \neq j} s_{jk}$.

Let *j* be the last job to be scheduled on this machine. Let us assume that job *k* precedes job *j*. Hence, $C_j = \max\{r_j, C_k + s_{kj}\} + p_j \leq \max\{r_j, C_k + s_{k\bullet}\} + p_j$.

Furthermore, $C_k + s_{k\cdot} \le \max_{i\in\mathcal{J},i\ne j} r_i + \sum_{i\in\mathcal{J},i\ne j}\rho_i$ then $C_j \le \max\left\{r_j, \max_{i\in\mathcal{J},i\ne j} r_i + \sum_{i\in\mathcal{J},i\ne j}\rho_i\right\} + p_j$.

We consider secondly a schedule with two machines, according to theorem 1, there is at least one task, denoted *l*, to be scheduled on the second machine. The improved job *j*'s completion time bound is then:

$$\begin{aligned} C_j &\le \max\left\{r_j, \max_{i\in\mathcal{J},i\ne j} r_i + \sum_{i\in\mathcal{J},i\ne j}\rho_i - \rho_l\right\} + p_j \\ &\le \max\left\{r_j, \max_{i\in\mathcal{J},i\ne j} r_i + \sum_{i\in\mathcal{J},i\ne j}\rho_i - \rho_{[1]}\right\} + p_j \end{aligned} \tag{10}$$

We consider a general case of *m* machine. By applying the same process, one can have a strict bound of the completion time of job *j*:

$$C_j \le \max\left\{r_j, \max_{i\in\mathcal{J},i\ne j}(r_i) + \sum_{i\in\mathcal{J},i\ne j}\rho_i - \sum_{i=1}^{m-1}\rho_{[i]}\right\} + p_j \tag{11}$$

In addition, $\sum_{i\in\mathcal{J},i\ne j}\rho_i - \sum_{i=1}^{m-1}\rho_{[i]} = \sum_{i=m}^{n}\rho_{[i]} - \rho_j$, we can rewrite the upper bound as:

$$C_j \le \max\left\{r_j, \max_{i\in\mathcal{J},i\ne j}(r_i) + \sum_{i=m}^{n}\rho_{[i]} - \rho_j\right\} + p_j = \bar{C}_j^M \tag{12}$$

□

2.2. Sequence-Based Formulations

The intuition behind the sequence formulation is to have a global job sequence cut into job sequences of machines. First, we use the variables v_j^k to make a jobs sequence: if $v_j^k = 1$ then job *j* occupies the *k*th position on the scheduling sequence, otherwise $v_j^k = 0$. Second, we cut the sequence into parallel machines by using variable θ. If $\theta_k = 1$ then the job at the position *k* will start to process on a new machine.

We take the same example introduced previously with 10 jobs and 3 machines to illustrate the assignment of variables for the sequence-based formulation. The value assigned to the variables θ is shown in Table 1.

The sequencing of jobs on machines is as follow:

- Machine 1: $5 \to 4 \to 7 \to 9$.
- Machine 2: $8 \to 6$.
- Machine 3: $2 \to 1 \to 10 \to 3$.

Auxiliary variable β_{ij}^k denotes the sequencing of two jobs *i* and *j*. If $\beta_{ij}^k = 1$ then job *j* follows job *i* and takes the *k*th position of the scheduling sequence. The completion time of the job assigned to position *k* is denoted by y^k.

Table 1. The value of variables v and θ.

k	1	2	3	4	5	6	7	8	9	10
Job	**<u>5</u>**	4	7	9	**<u>8</u>**	6	**<u>2</u>**	1	10	3
θ_k	**1**	0	0	0	**1**	0	**1**	0	0	0

Formulation SF

Set

- $\mathcal{K} = \{1, \cdots, n\}$: jobs' position.

Variables

- y^k : the completion time of job at position *k*.
- v_j^k : is equal to 1 if job *j* occupies position *k*.
- θ_k : is equal to 1 if job at the position *k* will start to process on a new machine.
- β_{ij}^k : an auxiliary variable, is equal to 1 if job *j* follows job *i* at *k*th position.

$$\text{Minimize} \quad \sum_{k \in \mathcal{K}} y^k \tag{13}$$

subject to

$$\sum_{k \in \mathcal{K}} v_j^k = 1 \quad \forall j \in \mathcal{J} \tag{14}$$

$$\sum_{j \in \mathcal{J}} v_j^k = 1 \quad \forall k \in \mathcal{K} \tag{15}$$

$$y^k \geq \sum_{j \in \mathcal{J}} v_j^k \left(r_j + p_j \right) \quad \forall k \in \mathcal{K} \tag{16}$$

$$\beta_{ij}^k \geq 1 - \left(2 - v_i^{k-1} - v_j^k \right) \quad \forall i, j \in \mathcal{J} : i \neq j, \forall k \in \mathcal{K} \tag{17}$$

$$y^k \geq y^{k-1} + \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}} \beta_{ij}^k s_{ij} + \sum_{i \in \mathcal{J}} v_j^k p_j - \theta_k \left(\overline{y}^k + s^{\max} - r_{\min} \right) \quad \forall k \in \mathcal{K} \tag{18}$$

$$\sum_{k \in \mathcal{K}} \theta_k \leq m \tag{19}$$

$$y_0 = 0; \theta_1 = 1 \tag{20}$$

$$v_j^k \in \{0, 1\} \quad \forall j \in \mathcal{J}, k \in \mathcal{K} \tag{21}$$

$$\beta_{ij}^k \in \{0, 1\} \quad \forall i, j \in \mathcal{J}, k \in \mathcal{K} \tag{22}$$

$$y^k \geq 0 \quad \forall k \in \mathcal{K} \tag{23}$$

where $\overline{y}^k$ is an upper bound of the completion time of the job that takes the *k*th position, $s^{\max} = \max_{i, j \in \mathcal{J}} s_{ij}$ and $r_{\min} = \min_{j \in \mathcal{J}} r_j$.

Constraints (14) and (15) limit one job at one position and vice-versa. Constraint (16) ensures a job *j* to start after its release date when it is assigned to *k*th position. Constraint (17) triggers the variable β_{ij}^k to be greater or equal to 1 if job *i* is in the position $k-1$ and job *j* is in the position *k*, otherwise β_{ij}^k is unconstrained. Constraint (18) computes the completion time of the job taking the *k*th slot. Constraint (19) limits the number of machines to *m*. The last constraints initialize and bound the decision variables.

It is important to note that the integral constraint of variable β_{ij}^k can be relaxed, *i.e.* $\beta_{ij}^k \in [0, 1]$, without violating the integrality of the solution [10].

We introduce in the following proposition an upper bound for y^k. This is an adaptation from the completion time upper bound for a single machine scheduling, which is introduced by Nogueira *et al.* [10], to identical parallel machines scheduling.

Proposition 3. $\bar{y}^k = \max_{j\in\mathcal{J}}\left(p_j + r_j\right) + \sum_{j=\max\{n-k+1,m\}}^{n} \rho_{[j]}$ is a valid upper bound of the completion time of job at the kth position, y^k, for any schedule with respect to theorem 1 where:

- $\rho_j = p_j + s_j^{\max}$.
- $\rho_{[j]}$ is the jth sorted element of ρ in a non-decreasing order.
- $s_{j\bullet} = \max_{k\in\mathcal{J},k\neq j} s_{jk}$.

Proof. Completion time of the job at the first position cannot exceed $\max_{j\in\mathcal{J}}\left(p_j + r_j\right)$. The job in the second position cannot complete after $\max_{j\in\mathcal{J}}\left(p_j + r_j\right) + \rho_{[n]}$, and so on, until we reach the job at the $(n-m+2)^{\text{th}}$ position. Completion time of this job is bounded the same way as the job at $(n-m+1)^{\text{th}}$ position. Because from the position 1 to $(n-m+2)^{\text{th}}$ there would be at least one job $k \in \{1,\cdots,n-m+2\}$ that has $\theta_k = 1$ (by theorem 1). Since $p_k + r_k \le \max_{j\in\mathcal{J}}\left(p_j + r_j\right)$ and $p_k + s_k < \sum_{j=m}^{n} \rho_{[j]}$ then $y^{n-m+2} \le \bar{y}^{n-m+2} = \bar{y}^{n-m+1}$. In the same way, from the position 1 to $(n-m+\tau)$ there would be at least $\tau > 0$ jobs $1 \le k \le n_m + \tau$ that have $\theta_k = 1$.

$$\text{Consequently, } \bar{y}^{n-m+1} = \bar{y}^{n-m+2} = \cdots = \bar{y}^{n}. \quad \square$$

Table 2 resumes the number of constraints and variables of the two formulations.

3. Numerical Tests

To conduct the numerical test, we use the same benchmark introduced by [4]. As an attempt to parameterize the instances, we use three factors: jobs' arrival density (*JAD*), jobs' setup time relative length (*SRL*) and the number of machines (*m*).

The objective of the numerical tests is two-fold. First, we try to observe the sensibility of each solving method to the corresponding data structure. Second, we tend to examine the impacts of the data structure to the tractability of the problem.

The following subsection will mention the random generators of the instances.

3.1. Numerical Tests Protocol

The number of jobs generated is $n \in \{5,10,15,20,25,30\}$ and the number of machines generated is $m \in \{2,3,5\}$.

Job's release date is generated randomly according to the uniform distribution which takes the value between $\left[0, 50.5\times\alpha\times\frac{n}{m}\right]$. Jobs arrival density is scaled by

Table 2. The number of constraints and variables.

Formulations	GF	SF
No. variables	$O(n^2)$	$O(n^3)$
No. Constraints	$O(n^2)$	$O(n^3)$

the value of α which takes value from the set $\{0.6, 0.8, 1.5, 2.0, 3.0\}$. When $\alpha = 0.6$, jobs arrival would be densest and earliest. In the contrary, when $\alpha = 3.0$, jobs arrival would distribute more evenly along the scheduling horizon.

The processing times of jobs are uniformly generated in $[1,100]$. The setup time s_{ij} is equal to $a \times \min\{p_i, p_j\}$, with *a* randomly generated in $[A,B] \in \{[0.01,0.1],[0.05,0.1],[0.1,0.2],[0.2,0.5],[0.1,0.5]\}$. **Table 3** resumes the parameters of the test generation.

For each data structure $\{n, m, \alpha, A\}$, we generate 10 instances, hence, each problem size $\{n, m\}$ has 250 instances generated. The total number of instances tested is 6000 instances.

We compare the CPLEX solving MILP formulations with an exact method: the Branch-and-Bound algorithm developed by Nessah *et al.* [4].

The MILP formulations are solved by *IBM ILOG CPLEX* 12.8. The Branch-and-Bound algorithm is coded in C++. The computer runs on Window 7 professional 64bits (Dell Optiplex 9020, CPU Intel Core i5-4690 3.5GHz and 8192MB RAM). We report in this section the following KPIs (Key Performance Indicators):

- *Time* (in second): solving time, which is limited to 3600 seconds.
- *LPGap* (in percentage): the LP relaxation gap. For the solution obtained by solving MILP formulations, *LPGap* is the gap between the best objective of the integral solution and LP relaxation one, *i.e.* when all the integrity constraint are removed. When Branch-and-Bound algorithm solves the instance, *LPGap* is the gap between the best solution objective and the best lower bound.
- *MLBGap* (the maximal lower bound, in percentage): the gap between the objective value found by each method to the best lower bound found by all the method. The *MLBGap* value can be considered as the worst possible gap to the optimal solution of each instance tested. This KPI is used to compare the quality of the actual solution found by a method to others.
- *Opt*: percentage of instances solved to optimality.

Figure 2 illustrates how to calculate the *LPGap* and *MLBGap*.

Table 3. Random instances generation's parameters.

Param.	Description	Generation
n	Number of jobs	$\{5,10,15,20,25,30,35,40\}$
m	Number of machines	$\{2,3,5\}$
α	*JAD* scale parameter	$\{0.6,0.8,1.5,2.0,3.0\}$
r_j	Release date	$U\left(0, 50.5 \times \alpha \times \frac{n}{m}\right)$
p_j	Processing time	$U(1,100)$
$[A,B]$	Scale factor intervals	$\{[0.01,0.1],[0.05,0.1],[0.1,0.2],[0.2,0.5],[0.1,0.5]\}$
a	*SRL* scale parameter	$U(A,B)$
s_{ij}	Sequence dependent setup time	$a \times \min\{p_i, p_j\}$

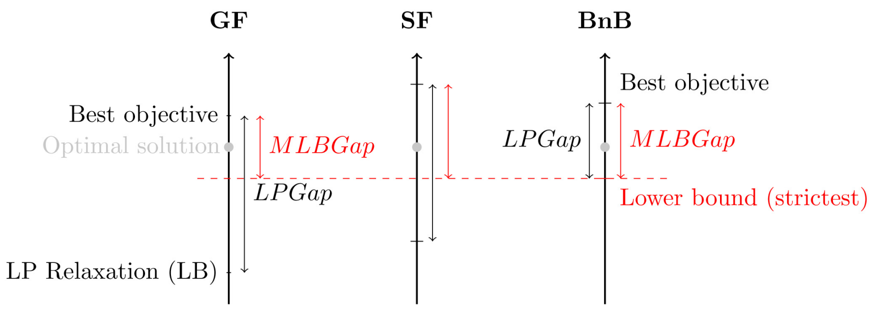


Figure 2. An example of *MLBGap* and *LPGap*.

3.2. Results Analysis

3.2.1. Computational Times (*CPUTimes*)

Impacting factor: Number of machines

Table 4 shows the results of average computational times of each method according to the variation of machines numbers *m*. Adding the more machine takes the Branch-and-Bound (*BnB*) largely more time to solve: when the number of machines goes from 2 to 5 the average *CPUTimes* increase nearly seven times. The solving time of the SF formulation increases also with the number of machines, but more slowly in comparison to the Branch-and-Bound algorithm. In the contrary, adding more machines reduces the solving times of the graph-based MILP by 32%.

Figure 3 plots the average *CPUTimes* of each value of *m* in function of *n* numbers of jobs. The solving time three solving methods discriminate when the number of jobs is greater than or equal to 15. While the *CPUTimes* of CPLEX for *GF* formulation increase in a stable way, the solving time of the other two methods increases in an exponential fashion.

Impacting factor: Jobs' arrival density

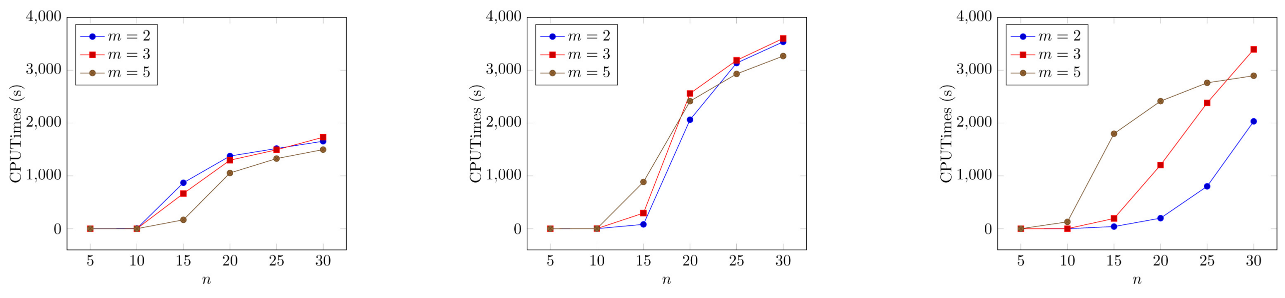
From the data of Table 4, one can see the non-negligent impact of the distribution of the release times of jobs to the solving times. For the two sparsest arrival rate ($\alpha = 3.0$ and $\alpha = 2.0$), *GF* has outstanding solving time. Figure 4 plots the average *CPUTimes* of each scale factor α in function of *n* numbers of jobs. Data from this figure show us that the solving time of *GF* formulation seems to be independent of the problem size when the release dates are sparse. On the contrary, *GF* formulation is very sensible to the dense release time when the resolution time of CPLEX for this formulation worsens exponentially. *BnB* algorithm is also sensitive to this factor: the computational times are reduced by half when α goes from 0.6 to 3.0. The computational time increase with α in $[0.6, 1.5]$ and decreases with α in $[1.5, 3.0]$. The global trend for all solving methods is that it takes a shorter time to solve an instance with a sparse release date.

Impacting factor: Jobs' setup times relative length

Table 4 presents the average computational times and the setup time scale factor ranges. Globally, the solving time tends to increase when the relative length of jobs' setup time increase. The least sensitive method to this factor is *GF*

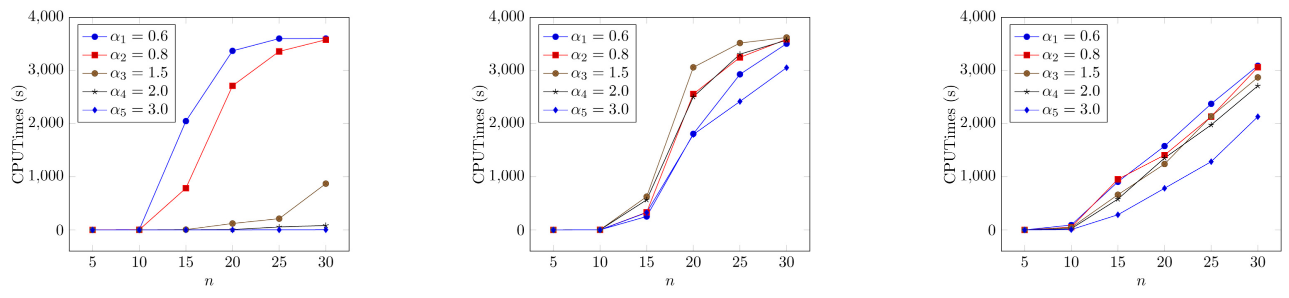
Table 4. The average computational times (in second) of instances grouped by the number of machines m, the arrival density scale α and setup time scale a. The best value found by a method is in bold.

Factor	Value	*GF*	*SF*	*BnB*
m	2	753.4	1056.2	**208.4**
	3	**691.3**	1208.9	756.1
	5	**510.0**	1246.1	1420.5
α	0.6	1804.9	998.4	**989.8**
	0.8	1372.3	1227.9	**909.7**
	1.5	**67.8**	1441.3	816.7
	2.0	**12.5**	1276.2	786.9
	3.0	**0.3**	908.3	472.1
a	[0.01, 0.1]	612.7	911.9	**540.1**
	[0.05, 0.1]	657.7	966.3	**619.2**
	[0.1, 0.2]	**634.2**	1220.9	757.0
	[0.2, 0.5]	**682.2**	1383.6	1076.0
	[0.1, 0.5]	**671.1**	1369.3	982.8



(a) Formulation *GF* (b) Formulation *SF* (c) Branch-and-Bound

Figure 3. The average value of *CPUTime* according to the number of machines m and number of jobs n.



(a) Formulation *GF* (b) Formulation *SF* (c) Branch-and-Bound

Figure 4. The average value of *CPUTime* according to arrival density scale parameter α and number of jobs n.

MILP and the most sensitive method to this factor is *BnB*. Concerning the computational time, the job setup times relative lengths are less impacting than the other two factors. Figure 5 plots the average *CPUTimes* of each scale factor a in function of n numbers of jobs.

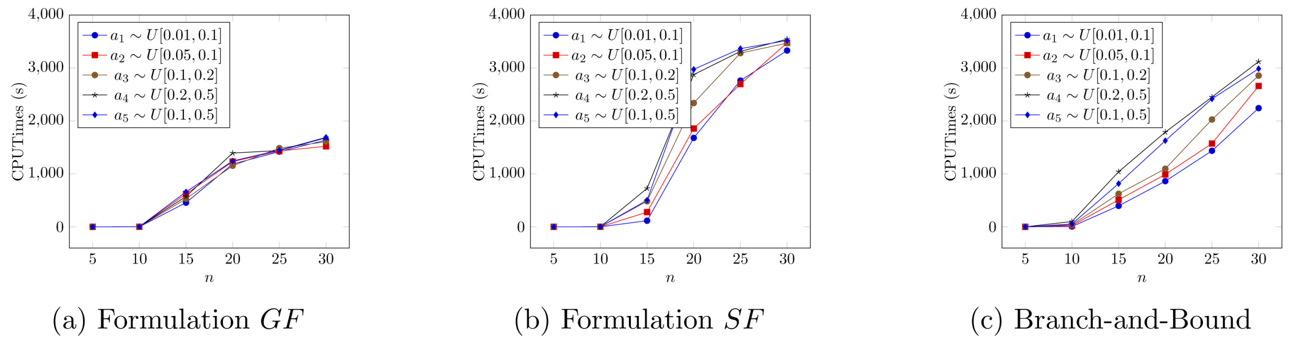


Figure 5. The average value of *CPUTime* according to setup time scale factor *a* and number of jobs *n*.

Table 5 shows the correlations between the computational times and the number of machines, the release date density, and the setup time scale.

3.2.2. Linear Relaxation Gap (*LPGaps*)

Impacting factor: Number of machines

Table 6 show the result on the average *LPGap* of each method to different numbers of machines. When the number of machines increases, the *LPGap* of *GF* is reduced three times while the *LPGap* of *SF* and *BnB* increase slightly.

Figure 6 details the impact of the number of machines to the average *LPGap* of each method in the function of *n* jobs. The *LPGap* curve of the Branch-and-Bound increases in a very stable way when the number of jobs rises up when the number of machines is small ($m = 2$) the *LPGap* of the *BnB* algorithm nearly zero regardless of the number of jobs.

Impacting factor: Jobs' arrival density

Table 6 presents the average *LPGaps* of each solving methods in function of the arrival density scale parameter α . One can observe an important impact of α to the quality of the solution found all three methods, especially the solving of formulation *GF*. The critical value of α is 1.5, the *LPGaps* deteriorates when $\alpha > 1.5$.

Figure 7 adds a dimension which is the problem size, to the observation. The quality of the solution found by CPLEX solving *GF* differs greatly since *n* reaches 15, while for the two other methods, the threshold of *n* which differs the solution quality is 25.

Impacting factor: Jobs' setup relative length

Table 6 presents the average *LPGap* of each solving methods in function of the jobs setup time scale parameter *a*. It seems that this parameter controls very weakly the KPI *LPGap* of the three methods. One can notice only a very slight increment of the *LPGap* in all methods to when the setup times increase.

In Figure 8, with more details on the instance size, one can observe a slight impact of the setup time length to the solving of *SF* formulation when *n* reaches 25. For the two other methods, the impact of the setup times length to the *LPGap* is not clear.

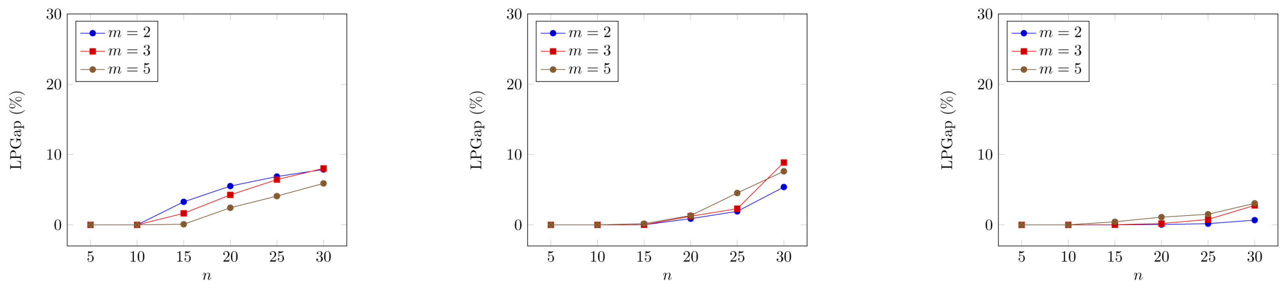
Altogether, Branch-and-Bound has tightest *LPGap*. The all-average gap is equal to 0.6% and the standard deviation is 1.7%. *GF* has the largest *LPGap*

Table 5. Correlation between the computational times and m, α and a.

Factor	*GF*	*SF*	*BnB*
Number of machine (m)	−1.00	0.86	0.99
Arrival density scale parameter (α)	−0.84	−0.27	−0.97
Average setup times scale parameter ($\bar{a}$)	0.82	0.95	1.00

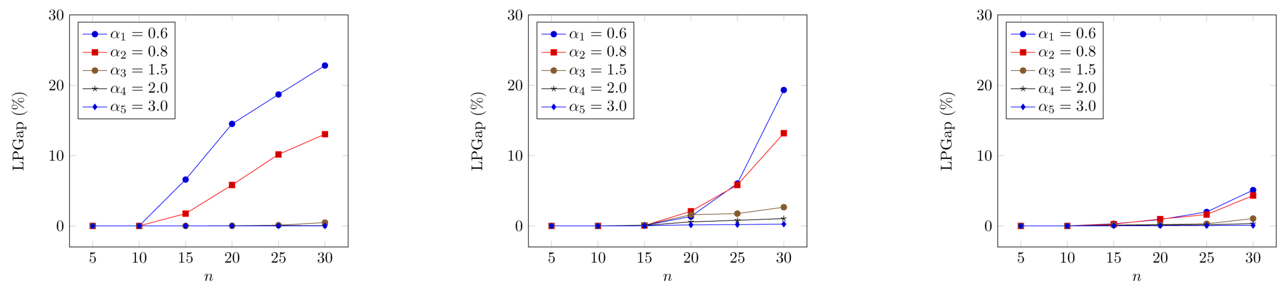
Table 6. The *LPGap* (in percentage) corresponding to the impacting factors (number of machines, arrival density scale and setup time scale) and their respective values.

Factor	Value	*GF*	*SF*	*BnB*
m	2	3.13%	0.56%	**0.04%**
	3	2.47%	0.71%	**0.20%**
	5	1.32%	1.20%	**0.61%**
α	0.6	7.96%	1.49%	**0.64%**
	0.8	3.55%	1.59%	**0.56%**
	1.5	**0.02%**	0.69%	0.12%
	2.0	**0.01%**	0.29%	0.07%
	3.0	**0.00%**	0.07%	0.01%
a	[0.01, 0.1]	2.04%	0.39%	**0.14%**
	[0.05, 0.1]	2.19%	0.53%	**0.20%**
	[0.1, 0.2]	2.16%	0.88%	**0.23%**
	[0.2, 0.5]	2.67%	1.25%	**0.47%**
	[0.1, 0.5]	2.47%	1.07%	**0.37%**



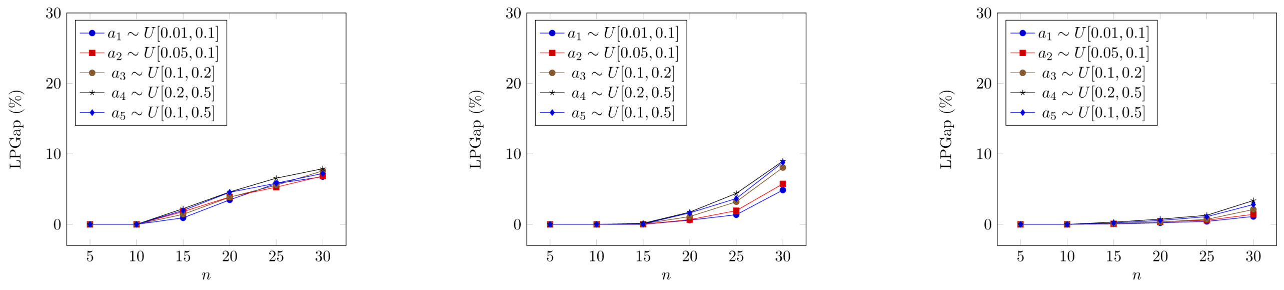
(a) Formulation GF (b) Formulation SF (c) Branch-and-Bound

Figure 6. The average value of *LPGap* according to number of machines m and number of jobs n.



(a) Formulation GF (b) Formulation SF (c) Branch-and-Bound

Figure 7. The average value of *LPGap* according to arrival density scale parameter α and number of jobs n.



(a) Formulation *GF* (b) Formulation *SF* (c) Branch-and-Bound

Figure 8. The average value of *LPGap* according to setup time scale factor *a* and number of jobs *n*.

when the release dates are dense where $\alpha \in \{0.6, 0.8\}$ while yielding excellent results at other value of α. This fact underlines our previous observation about the sensitivity of *GF* to the job availability distribution. Table 7 states the correlation between the *LPGap* and the impacting factors.

3.2.3. Gap to the Best-Known Lower Bound

The quality of the solution found by each method/formulation can be evaluated by the gap of the best-known lower bound (*MLBGap*) to the actual solution. This gap can be considered as the maximum possible gap to the optimal solution.

Figures 9-11 shows the impact of the scale factors on three paradigms over the *MLBGap*. When having the worst *LPGap*, *GF* has the best *MLBGap* in all instance size and in any scale factors. Inversely, the Branch-and-Bound yields the worst *MLBGap* while having the best *LPGap*. This observation underlines again the fact that the lower-bound of the Branch-and-Bound calculated by the *mSPRT* heuristic [4] is tight. Since the calculation of this lower bound is dependent to the number of machines, the *MLBGap* is also impacted by the number of machines. The empirical data shows that the release date density is very impacting to the value of *MLBGap*. Scale factor α diversifies the performance of all paradigms when $n \geq 20$, especially the Branch-and-Bound.

The global results on *MLBGap* of the three methods according to the setup times length scale factors are shown in Table 8. The correlation between the *MLBGap* and the impacting factors is shown in Table 9.

3.2.4. Percentage of Instances Being Solved to Optimality

For this KPI, we tend to examine the tractability of an instance according to its structure, *i.e.* quantify how easy an instance can be solved according to a specific structure. First, we introduce the graphs of the evolution of the percentages of the instances being solved to optimality in function of the problem size, and, corresponding to the value of *m*, α and *a*. As for the other KPI which are previously introduced, those graphs give us an idea on how well each solving method cope with the instance in an specific structure. Second, we introduce an cross-table analysis, with combined two factors. This analysis would help us to have an insight on how easy an instance can be solved with a specific pair of configurations.

Table 7. Correlation between the gap to lower bound and *m*, *a* and $\bar{a}$.

Factor	*GF*	*SF*	*BnB*
Number of machine (*m*)	−1.00	0.99	1.00
Arrival density scale parameter (*a*)	−0.78	−0.95	−0.90
Average setup times scale parameter ($\bar{a}$)	0.96	0.97	0.98

Table 8. The average *MLBGap* (in percentage) corresponding to the impacting factors (number of machines, arrival density scale and setup time scale) and their respective values.

Factor	Value	*GF*	*SF*	*BnB*
m	2	**0.11%**	0.14%	0.23%
	3	**0.33%**	0.44%	0.95%
	5	**0.39%**	0.53%	1.49%
α	0.6	**0.48%**	**0.48%**	1.26%
	0.8	**0.33%**	0.44%	1.10%
	1.5	**0.02%**	0.12%	0.21%
	2.0	**0.00%**	0.06%	0.08%
	3.0	**0.00%**	0.01%	0.02%
a	[0.01, 0.1]	**0.05%**	0.07%	0.28%
	[0.05, 0.1]	**0.08%**	0.11%	0.38%
	[0.1, 0.2]	**0.15%**	0.20%	0.49%
	[0.2, 0.5]	**0.31%**	0.40%	0.83%
	[0.1, 0.5]	**0.24%**	0.33%	0.69%

Table 9. Correlation between the *MLBGap* and the impacting factors.

Factor	*GF*	*SF*	*BnB*
Number of machine (*m*)	0.87	0.89	0.96
Arrival density scale parameter (*a*)	−0.84	−0.91	−0.88
Average setup times scale parameter ($\bar{a}$)	0.99	1.00	0.99

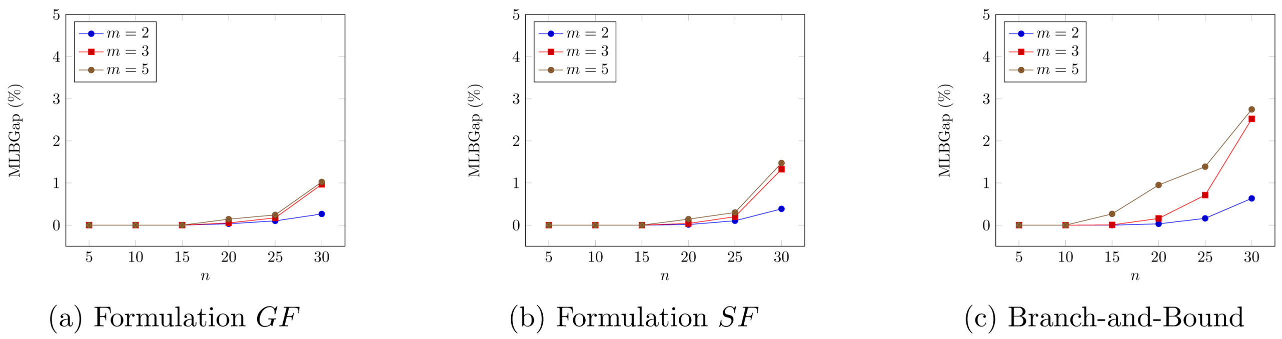
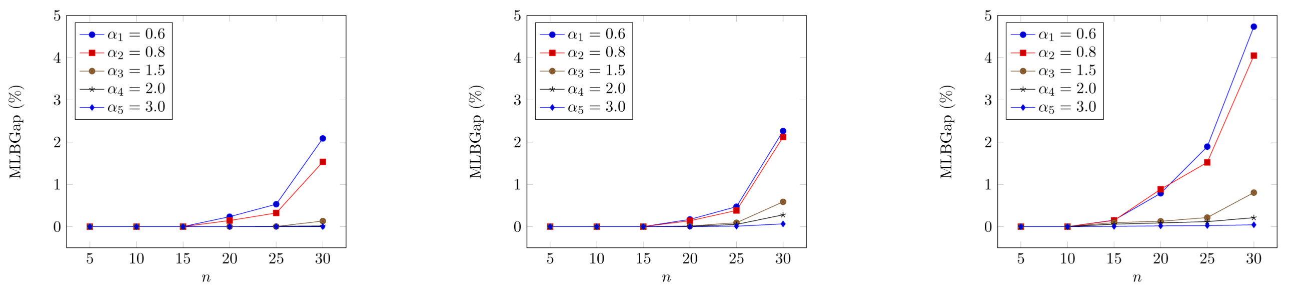
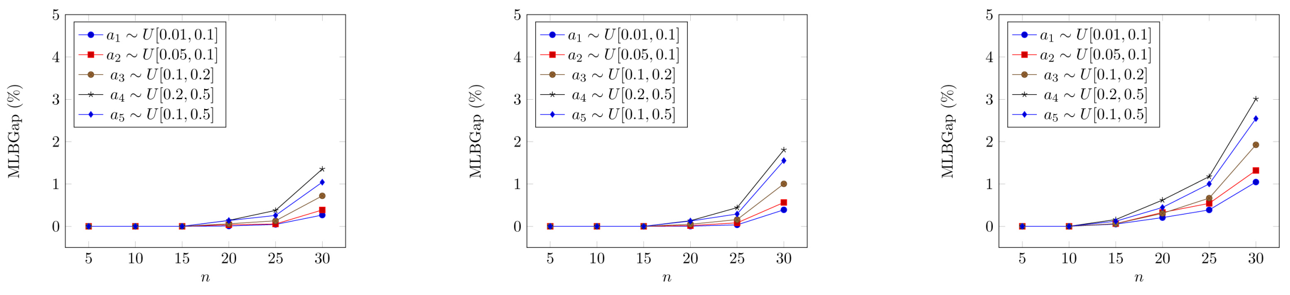


Figure 9. The average value of *MLBGap* according to number of machines *m* and number of jobs *n*.



(a) Formulation *GF* (b) Formulation *SF* (c) Branch-and-Bound

Figure 10. The average value of *MLBGap* according to arrival density scale parameter α and number of jobs n.



(a) Formulation *GF* (b) Formulation *SF* (c) Branch-and-Bound

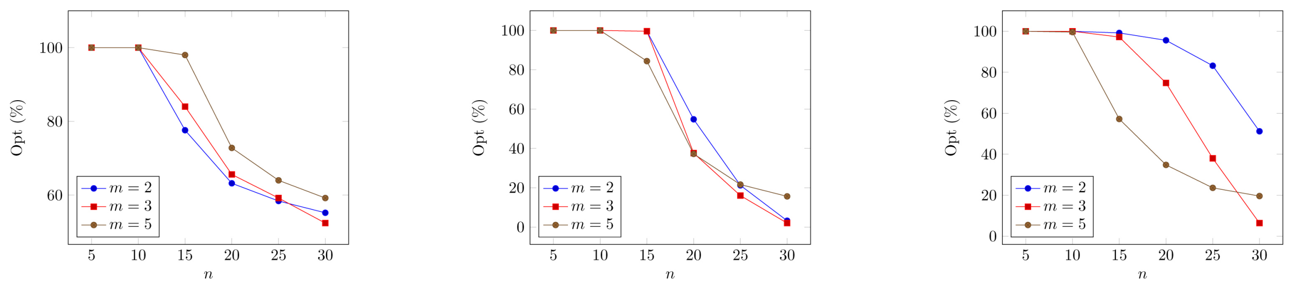
Figure 11. The average value of *MLBGap* according to setup time scale factor a and number of jobs n.

Figures 12-14 show the percentages of instances that are solved optimally according to each scale factor.

Cross-Table 10 presents the percentage of instances being solved to optimality corresponding to each pair of the release date density and the setup time length scale. CPLEX solves the instance easily to optimal by *GF* formulation when α is greater than or equal to 1.5, regardless of the value of setup-times. For the Branch-and-Bound algorithm, both values of α and a harm the number of instances being solved to optimality. The relation between the number of instance solved and the structure α and a is more random when CPLEX solves instances by *SF* formulation. When we combined all methods, the number of optimal solutions increase with α and decrease with a.

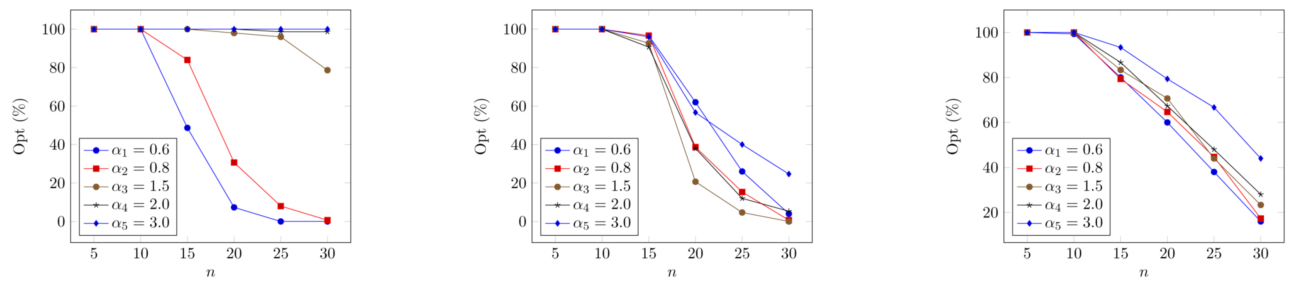
Cross-Table 11 presents the percentage of instances being solved to optimality corresponding to each pair of the setup time length scale and the number of machines. From the data, we can remark a clear frontier between easy-to-solve and hard-to-solve instances classified by the number of machine for the Branch-and-Bound algorithm. Combining all methods, an instance is easier to solve with less number of machines and smaller setup times.

Cross-Table 12 presents the percentage of instances being solved to optimality corresponding to each pair of the jobs release dates density and the number of machines. From the data, one can observe that both the number of machines and the distribution of jobs arrival date have the discriminating power to separate easy-to-solve instance to hard-to-solve instances. However, since CPLEX solves instances with evenly distributed arrival dates outstandingly good with *GF*,



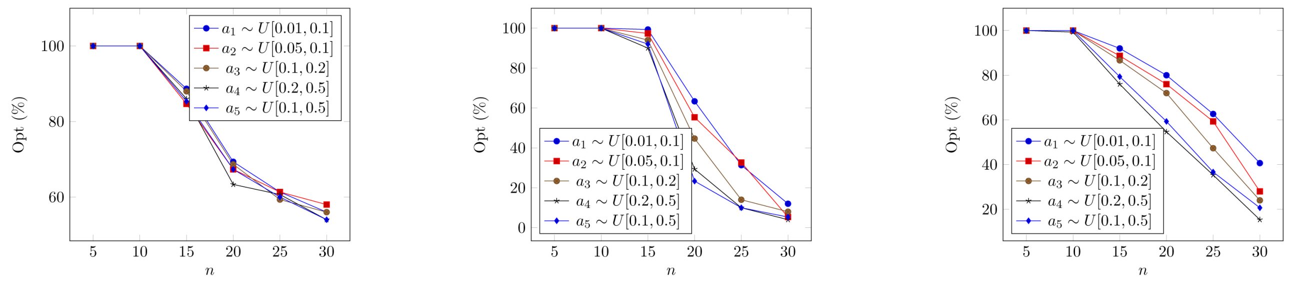
(a) Formulation GF (b) Formulation SF (c) Branch-and-Bound

Figure 12. The percentage of instances solved to optimality according to number of machines m and number of jobs n.



(a) Formulation GF (b) Formulation SF (c) Branch-and-Bound

Figure 13. The percentage of instances solved to optimality according to arrival density scale parameter α and number of jobs n.



(a) Formulation GF (b) Formulation SF (c) Branch-and-Bound

Figure 14. The percentage of instances solved to optimality according to setup time scale factor a and number of jobs n.

Table 10. The percentage of instances solved to optimality according to [A, B] and α.

		α				
	[A,B]	0.6	0.8	1.5	2.0	3.0
GF	[0.01, 0.1]	52%	67%	100%	100%	100%
	[0.05, 0.1]	49%	65%	99%	100%	100%
	[0.1, 0.2]	53%	65%	98%	100%	100%
	[0.2, 0.5]	49%	65%	97%	99%	100%
	[0.1, 0.5]	53%	61%	99%	99%	100%
SF	[0.01, 0.1]	88%	80%	69%	77%	80%
	[0.05, 0.1]	83%	80%	67%	73%	83%
	[0.1, 0.2]	77%	67%	64%	65%	81%

Continued

	[0.2, 0.5]	66%	861%	59%	65%	78%
	[0.1, 0.5]	71%	63%	59%	61%	71%
BnB	[0.01, 0.1]	87%	86%	87%	87%	87%
	[0.05, 0.1]	81%	83%	82%	85%	94%
	[0.1, 0.2]	76%	79%	81%	81%	89%
	[0.2, 0.5]	63%	68%	74%	75%	85%
	[0.1, 0.5]	71%	73%	74%	75%	83%
All	[0.01, 0.1]	93%	93%	100%	100%	100%
	[0.05, 0.1]	88%	93%	99%	100%	100%
	[0.1, 0.2]	86%	86%	99%	100%	100%
	[0.2, 0.5]	77%	79%	98%	99%	100%
	[0.1, 0.5]	79%	83%	99%	100%	100%
Scale		0%	25%	50%	75%	100%

Table 11. The percentage of instances solved to optimality according to [A, B] and m.

		m		
	[A, B]	2	3	5
GF	[0.01, 0.1]	77%	77%	83%
	[0.05, 0.1]	76%	77%	83%
	[0.1, 0.2]	76%	79%	81%
	[0.2, 0.5]	76%	75%	82%
	[0.1, 0.5]	74%	76%	83%
SF	[0.01, 0.1]	70%	66%	67%
	[0.05, 0.1]	68%	65%	63%
	[0.1, 0.2]	61%	58%	61%
	[0.2, 0.5]	59%	53%	54%
	[0.1, 0.5]	57%	54%	54%
BnB	[0.01, 0.1]	96%	76%	66%
	[0.05, 0.1]	92%	77%	57%
	[0.1, 0.2]	88%	70%	57%
	[0.2, 0.5]	80%	61%	49%
	[0.1, 0.5]	86%	63%	49%
All	[0.01, 0.1]	99%	91%	88%
	[0.05, 0.1]	98%	91%	86%
	[0.1, 0.2]	93%	89%	84%
	[0.2, 0.5]	88%	83%	82%
	[0.1, 0.5]	92%	84%	84%
Scale		20%	950%	100%

Table 12. The percentage of instances solved to optimality according to α and m.

		m		
	α	2	3	5
GF	0.6	36%	40%	52%
	0.8	48%	52%	61%
	1.5	94%	93%	99%
	2.0	100%	99%	100%
	3.0	100%	100%	100%
SF	0.6	73%	64%	57%
	0.8	64%	60%	52%
	1.5	57%	52%	50%
	2.0	60%	54%	58%
	3.0	61%	66%	82%
BnB	0.6	85%	69%	43%
	0.8	87%	73%	43%
	1.5	91%	70%	50%
	2.0	91%	66%	58%
	3.0	88%	69%	85%
All	0.6	87%	71%	61%
	0.8	87%	74%	64%
	1.5	96%	94%	99%
	2.0	100%	99%	100%
	3.0	100%	100%	100%
Scale		20%	50%	100%

the release date density has more discriminating power. When combined all solving methods, more than 94% of the instances are solved to optimality with sparse jobs availability ($\alpha \geq 1.5$). When α is less than 1.5, then, instances with less number of machines are easier to be solved to optimality.

4. Conclusions and Perspective

First, we present in this paper two MILP formulations to solve the parallel machine scheduling with task release dates and sequence-dependent setup times to minimize total completion time. We provide also two upper bounds for the job's completion times corresponding to each formulation.

Second, by a thorough analysis of the numerical tests, we observe a considerable impact of the job arrival rate, and a non-negligible impact of the number of machines to the quality of the solution found and the time-performance of the solving algorithm. The cross-table analysis at the end of the numerical tests section could establish a base for further classification methods.

We developed four perspectives on future research of this problem. First, we may notice the difference between the *LPGap* and the actual gap of each instance. Thus, a better estimation of the lower bound can be very helpful to improve the computational effort. Second, we tend to test the performances of GF, SF, and Branch-and-Bound with other objective functions such as makespan minimization and total weighted completion time minimization. Third, we notice that the MILP formulation is very sensitive to the upper bound of the completion time so a tighter upper bound could be helpful. Finally, one can observe that the performance of the different methods is input data characteristics (jobs' arrival density scale parameter and setup time scale factors) dependent. Further work is to develop a new method corresponding to our previous conclusion.

Acknowledgements

This research is supported by Chaire Connected Innovation tenured by Prof. Farouk Yalaoui.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Graham, R.L., Lawler, E.L., Lenstra, J.K. and Kan, A.R. (1979) Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics*, **5**, 287-326. https://doi.org/10.1016/S0167-5060(08)70356-X

[2] Allahverdi, A. (2015) The Third Comprehensive Survey on Scheduling Problems with Setup Times/Costs. *European Journal of Operational Research*, **246**, 345-378. https://doi.org/10.1016/j.ejor.2015.04.004

[3] Guinet, A. (1993) Scheduling Sequence-Dependent Jobs on Identical Parallel Machines to Minimize Completion Time Criteria. *The International Journal of Production Research*, **31**, 1579-1594. https://doi.org/10.1080/00207549308956810

[4] Nessah, R., Chu, C. and Yalaoui, F. (2007) An Exact Method for $P_m \mid sds, s_{ij} \mid \sum_{i=1}^{n} C_i$ Problem. *Computers & Operations Research*, **34**, 2840-2848. https://doi.org/10.1016/j.cor.2005.10.017

[5] Fowler, J.W., Horng, S.M. and Cochran, J.K. (2003) A Hybridized Genetic Algorithm to Solve Parallel Machine Scheduling Problems with Sequence Dependent Setups. *International Journal of Industrial Engineering: Theory Applications and Practice*, **10**, 232-243.

[6] Montoya-Torres, J.R., Soto-Ferrari, M., Gonzalez-Solano, F. and Alfonso-Lizarazo, E.H. (2009) Machine Scheduling with Sequence-Dependent Setup Times Using a Randomized Search Heuristic. 2009 *International Conference on Computers & Industrial Engineering*, Troyes, 6-9 July 2009, 28-33. https://doi.org/10.1109/ICCIE.2009.5223742

[7] Lin, S.-W., Lee, Z.-J., Ying, K.-C. and Lu, C.-C. (2011) Minimization of Maximum Lateness on Parallel Machines with Sequence-Dependent Setup Times and Job Release Dates. *Computers & Operations Research*, **38**, 809-815. https://doi.org/10.1016/j.cor.2010.09.020

[8] Kurz, M. and Askin, R. (2001) Heuristic Scheduling of Parallel Machines with Sequence-Dependent Set-Up Times. *International Journal of Production Research*, **39**, 3747-3769. https://doi.org/10.1080/00207540110064938

[9] Anderson, B.E., Blocher, J.D., Bretthauer, K.M. and Venkataramanan, M.A. (2013) An Efficient Network-Based Formulation for Sequence Dependent Setup Scheduling on Parallel Identical Machines. *Mathematical and Computer Modelling*, **57**, 483-493. https://doi.org/10.1016/j.mcm.2012.06.029

[10] Nogueira, T.H., De Carvalho, C. and Ravetti, M.G. (2014) Analysis of Mixed Integer Programming Formulations for Single Machine Scheduling Problems with Sequence Dependent Setup Times and Release Dates. *Optimization Online*, **39**.

[11] Yalaoui, F. and Chu, C. (2002) Parallel Machine Scheduling to Minimize Total Tardiness. *International Journal of Production Economics*, **76**, 265-279. https://doi.org/10.1016/S0925-5273(01)00175-X