Received January 27, 2018, accepted March 4, 2018, date of publication March 13, 2018, date of current version April 18, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2815598

A Computational Study of the Two-Machine No-Wait Flow Shop Scheduling Problem Subject to Unequal Release Dates and Non-Availability Constraints

MOHAMED LABIDI[1], ANIS KOOLI[2], TALEL LADHARI[2,3], ANIS GHARBI[1], AND UMAR S. SURYAHATMAJA[1]
[1]Department of Industrial Engineering, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia
[2]Ecole Supérieure des Sciences Economiques et Commerciales de Tunis, Université de Tunis, Tunis 1089, Tunisia
[3]Business Analytics and Decision Making Laboratory, Tunis Business School, Université de Tunis, Bir El Kassaa 2059, Tunisia

Corresponding author: Mohamed Labidi (mlabidi@ksu.edu.sa)

This work is funded by the National Plan for Science, Technology and Innovation (MAARIFAH), King Abdulaziz City for Science and Technology, Kingdom of Saudi Arabia, Award Number (12-MAT3123-02).

ABSTRACT We consider the problem of scheduling a set of jobs subject to unequal release dates on a two-machine flow shop where the no-wait and non-availability constraints are considered so as to minimize the makespan. The contribution of this paper is two-fold. First, we propose a new mathematical formulation for the problem and derive valid inequalities. Second, we propose new lower bounds that are based on single and two-machine relaxations. These lower bounds are embedded onto a branch-and-bound algorithm enhanced by elimination and dominance rules. Computational results show that our exact methods consistently outperform the state-of-the-art branch-and-bound procedure.

INDEX TERMS Scheduling, flow shop, no-wait, non-availability, lower bounds, branch-and-bound algorithm.

I. INTRODUCTION

In this paper, we address the two-machine no-wait flow shop scheduling problem subject to unequal release dates and non-availability constraints. The problem can be defined as follows. A set $J = \{1, \ldots, n\}$ of jobs have to be scheduled on a two-machine flow shop environment without preemption. Each job $j \in J$ is available at its release date r_j and have to be processed during p_{1j} unit times on the first machine M_1 and p_{2j} unit times on the second machine M_2. The jobs are subject to the no-wait requirement, i.e., while a job terminates its execution on the first machine it must immediately begins the execution on the second machine (no idle time between the two parts is permitted). The two machines are subject to a non-availability constraint, i.e., the machines are unavailable during an interval of time. We assume that the two intervals overlap and the unavailability on machine M_1 (denoted by $[s_1, t_1]$) starts and ends before the starting and ending of the unavailability on machine M_2 (denoted by $[s_2, t_2]$), respectively. The jobs are non-resumable since they must completely restart if they cannot finish before the unavailable intervals. The problem is to find a feasible schedule, such that the time C_{max} at which all the jobs are completed (makespan) is minimized (see Example 1). Using the notation specified in Pinedo [29] and the notation of the availability constraints specified by Lee [27], this problem can be denoted $F2|nr - a, nwt, r_j|C_{max}$. The problem is *NP*-hard in the strong sense since it is a generalization of the well known flow shop problem subject to release dates $F2|r_j|C_{max}$. Besides it's theoretical challenge, the no-wait flow shop problem with unavailability periods has many real life applications [2]. Indeed, in many industrial environments, a machine might be unavailable for processing due to many reasons, such as sudden breakdowns (stochastic unavailability) or due to a planned preventive maintenance operation where the period of unavailability is already known (deterministic unavailability). In addition, usual no-wait requirement exists in many production technology processes, for example, temperature of the material impose that each operation follow the previous one immediately. Likewise, in the pharmaceutical or chemical manufacturing, the production of certain component ask for the no-wait of some produced batches.

In the literature, there is a large amount of works dealing with related problems. The $F2|r_j|C_{max}$ problem have been widely studied [8]–[13]. The most effective procedure is probably the one proposed by Cheng *et al.* [8] who proposed a branch-and-bound algorithm able to solve instances up to 500 jobs in few seconds. Other variant of the two-machine flow shop have been studied in the literature (in the presence of time lags, ready times, and delivery times). The reader is referred to Haouari and Ladhari [19] for more details.

The two-machine flow shop problem under non-availability constraints was proven to be $\mathcal{NP}$-hard [33]. Lee [27] was one of the first researchers to address this type of problems. He proposed several heuristic algorithms for many configurations: semi resumable, nonresumable and resumable jobs, non-availability period on the first, second machine and both machines. Later, many researchers followed the same path. Recently, Hnaien *et al.* [20] considered the two-machine flow shop problem with an availability constraint on the first machine. The authors proposed two mixed integer programming models and a branch-and-bound procedure which solves instances of size up to 100 jobs. For a survey on scheduling problems under non-availability constraints, we refer the reader to the article of Ma *et al.* [28].

Moreover, some work deals with the two-machine no-wait flow shop with availability constraints. Cheng and Liu [5] presented a polynomial time approximation scheme (PTAS) for the nonresumable case when an unavailable interval is imposed on only one machine, or the unavailable intervals on the two machines overlap. Their approximation scheme seems to be interesting only in theory since its complexity contains a huge coefficient whose value depends on the desired accuracy. Later, they improved the existing results by proposing a 3/2-approximation algorithm which is more efficient in practice for the same problems [6]. Kubzin and Strusevich [25] have considered all possible scenarios of handling the job affected by the non-availability (resumable, non-resumable or semi-resumable). For all these scenarios, they offered approximation algorithms with a worst-case ratio of 3/2. For the resumable scenario, they presented a 4/3-approximation algorithm.

In our best knowledge, only one paper dealt with the problem under consideration. Indeed, Ben Chihaoui *et al.* [3] introduced the problem and proposed exact method and heuristic procedures. Firstly, they proposed a set of upper bounds: two constructive heuristics, a genetic algorithm and a greedy search procedure. After that, the authors proposed several lower bounds based on single machine relaxations. The different components were embedded within a branch-and-bound algorithm to solve exactly the problem. Computational results showed that the algorithm could only solve instances with up to 15 jobs. These results show the intractability of the problem and the necessity to propose efficient procedures to deal with its difficulty. The objective of this paper is to propose effective exact methods for the problem under consideration. More precisely, we make the following contributions:

1) We propose a mathematical model for the problem and enhance it with valid inequalities.
2) We derive several lower bounds based on single and two-machine relaxations.
3) We introduce a preprocessing procedure which reduces the size of the instances.
4) We implement branch-and-bound algorithms based on the proposed procedures.
5) We conduct a comprehensive computational study that make evidence that our exact methods outperform already existing procedures.

The remainder of this paper is organized as follows. Section II presents the mathematical formulation and the valid inequalities. Section III recalls the state-of-the art lower bounds and presents the new derived ones. The components of the branch-and-bound algorithm are described in Section IV. Computational experiments on a set of randomly generated instances are reported in Section V. Finally, concluding remarks are given in Section VI.

TABLE 1. Data of example 1.

	1	**2**	**3**	**4**	**5**
$\mathbf{r}_j$	1	46	5	94	84
$\mathbf{p}_{1j}$	14	54	68	39	4
$\mathbf{p}_{2j}$	76	22	68	52	6

Example 1: Consider the 5-job instance defined by Table1, where the first and the second machines are unavailable during time periods [100,180] and [120,220], respectively.

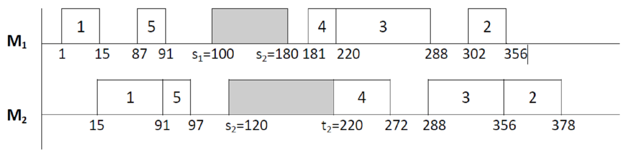


FIGURE 1. Optimal schedule of example 1.

Figure 1 depicts the optimal schedule which yields a makespan $C_{max} = 378$.

II. MATHEMATICAL FORMULATION

In this section, we propose a mathematical formulation for the problem. To this end, we used the assignment-based formulation originally introduced by Lasserre and Queyranne [26] for single machine problems and then adapted it to this problem. Before presenting the model, we introduce a useful result allowing to eliminate some decision variables.

A. PRELIMINARY RESULT

In order to derive the formulation, we first replaced the unavailability intervals by two fictious jobs, denoted hereafter by α and β defined as follows:

- $r_\alpha = s_1, p_{1\alpha} = s_2 - s_1, p_{2\alpha} = t_1 - s_2$,
- $r_\beta = s_2, p_{1\beta} = t_1 - s_2, p_{2\beta} = t_2 - t_1$.

In the sequel, we denote by $J_f = J \cup \{\alpha, \beta\}$, the set of jobs to schedule plus the two dummy jobs. The key idea of the formulation is to consider $n + 2$ positions (one position for each job in the set J_f) and to determine for each position what is the job to be scheduled. It is clear that the two dummy jobs α and β must be scheduled at positions that will match the unavailability intervals. Thus, after a first look, we must consider the $n+2$ positions for these two latter jobs. In order to reduce the number of positions to consider, we will search for the maximum number of jobs that can be scheduled before the unavailability intervals. To this end, we compute the schedule based on the Shortest Remaining Processing Time (SRPT) rule on both machines as follows:

- on the first machine, the release dates are r_j and the processing times are equal to p_{1j}.
- on the second machine, the release dates are equal to $r_j + p_{1j}$ and the processing times are p_{2j}.

It is worth recalling that the SRPT rule consists in scheduling at each time period the job with the minimum remaining processing time. Chu [14] proved the following lemma:

Lemma 1: The completion time of position k given by the SRPT rule is a lower bound on the completion time of the job processed at position k in any sequence.

Let k^i_{max}, $i = 1, 2$, be the maximum index verifying $S^i(k^i_{max}) \leq s_i$ where $S^i(.)$ is the completion time given by the SRPT rule of the job scheduled at the position indicated between brackets on machine M_i. Thus, we have the following result:

Fact 1: $k_{max} = \min(k^1_{max}, k^2_{max})$ represent an upper bound on the number of jobs that can be scheduled before the unavailability period.

Proof 1: On one hand, considering only the processing time on the first machine represents a relaxation to the problem. Thus, based on the result of Lemma 1, it is clear that k^1_{max} constitutes an upper bound on the maximum number of jobs that can be scheduled before the starting time s_1 of the unavailability interval on the first machine.

On the other hand, since job j can not be scheduled before $r_j + p_{1j}$ then this latter value constitutes a lower bound on the starting time of job j on the second machine. Applying the same reasoning as in M_1 leads to k^2_{max} to be an upper bound on the maximum number of jobs that can be scheduled before the starting time s_2 of the unavailability interval on M_2.

As a result $k_{max} = \min(k^1_{max}, k^2_{max})$ is an upper bound on the number of jobs that can be scheduled before the unavailability period.

Corollary 1: The set $\mathcal{P}_j$ of possible positions for job j is:

$$\mathcal{P}_j = \begin{cases} \{1, \ldots, n+2\} & \text{if } \; j \in J \\ \{1, \ldots, k_{max}+1\} & \text{if } \; j = \alpha \\ \{2, \ldots, k_{max}+2\} & \text{if } \; j = \beta \end{cases}$$

B. THE MODEL

The formulation is based on binary variables x_{jk} which determines whether the job j is scheduled at position k, i.e., x_{jk} is equal to 1 if job j is scheduled at position k and 0 otherwise. Continuous variables C^1_k and C^2_k are also used to compute the completion times of the job scheduled at position k on machines M_1 and M_2, respectively. A continuous variable $\mathcal{C}_\alpha$ is necessary to retrieve the completion time of the first dummy job α.

The mathematical model, hereafter denoted by (*MIP*), is given as follows.

$$\min C^2_n \tag{1}$$

$$\text{s.t.} \sum_{k \in \mathcal{P}_j} x_{jk} = 1, \quad \forall j \in J_f \tag{2}$$

$$\sum_{j \in J_f} x_{jk} = 1, \quad \forall k \in \mathcal{P}_j \tag{3}$$

$$C^1_k \geq \sum_{j \in J_f} (r_j + p_{1j}) x_{jk}, \quad \forall k = 1, \ldots, n+2 \tag{4}$$

$$C^1_k \geq C^1_{k-1} + \sum_{j \in J_f / k \in \mathcal{P}_j} p_{1j} x_{jk}, \quad \forall k = 2, \ldots, n+2 \tag{5}$$

$$C^1_k \geq C^2_{k-1}, \quad \forall k = 2, \ldots, n+2 \tag{6}$$

$$C^2_k \geq C^1_k + \sum_{j \in J_f / k \in \mathcal{P}_j} p_{2j} x_{jk}, \quad \forall k = 1, \ldots, n+2 \tag{7}$$

$$\mathcal{C}_\alpha \geq C^1_k - L(1 - x_{\alpha k}), \quad \forall k = 1, \ldots, k_{max}+1 \tag{8}$$

$$\mathcal{C}_\alpha = s_2, \tag{9}$$

$$x_{\alpha k} \leq x_{\beta k+1}, \quad \forall k = 1, \ldots, k_{max}+1 \tag{10}$$

$$C^1_k, C^2_k \geq 0, \quad \forall k = 1, \ldots, n+2 \tag{11}$$

$$x_{jk} \in \{0, 1\}, \quad \forall j \in J_f, \; k \in \mathcal{P}_j \tag{12}$$

The objective function (1) minimizes the makespan on the second machine. Constraints (2) and (3) state that each job is scheduled at only one position and at each position only one job is processed, respectively. Constraints (4) ensure that the job scheduled at position k can not start before its release date while constraints (5) ensure that the same job can not start before the completion time of the job scheduled at position $k - 1$. Constraints (6) force the no-wait requirement. Constraints (7) are used to compute the completion times on the second machine. At this point, it is worth noting that variables C^2_k can be eliminated from the formulation. We kept them to simplify the presentation of the model. Constraints (8) allow to retrieve the completion time of the dummy job α on the first machine where L is a large value. Constraint (9) force this latter completion time to be equal to s_2. Constraints (10) force the second dummy job β to be scheduled immediately after job α. Thus, constraints (9) and (10) ensure that the fictious jobs are scheduled on the unavailability period. Finally, Constraints (11) and (12) are the continuous and integrality constraints, respectively.

C. VALID INEQUALITIES

We also added valid inequalities to strengthen the model. These constraints are inspired from those of Haouari and Kharbeche [18] and Kooli and Serairi [23] who proposed a set of valid inequalities for the assignment-based formulation. The key idea is based on the computation of lower bounds on the completion times of the positions. Let λ^i_{jk}, $i = 1, 2$, be a lower bound on the completion time on machine i if job j if scheduled at position k. It is clear that $\sum_{j=1}^{n} \lambda^i_{jk} x_{jk}$ represents a lower bound on the completion time on machine i of the job processed at position k. Thus, the following Proposition holds.

Proposition 1: Inequalities

$$C^i_k \geq \sum_{j=1}^{n} \lambda^i_{jk} x_{jk}, \quad \forall i = 1, 2, \ \forall k = 1, \ldots, n \tag{13}$$

are valid for (MIP).

Kooli and Serairi [23] proposed lower bounds on the completion times of positions in the case of the single machine problem with release dates. The authors used the Shortest Remaining Processing Time rule to derive a set of schedules by omitting each time a job from the initial set of jobs. They proved that $max(C_{k-1}(J\backslash\{j\}), r_j)+p_j$ constitutes a valid lower bound on the completion time of job j if it is scheduled at position k, where $C_{k-1}(J\backslash\{j\})$ is the completion time of position $k - 1$ computed by the SRPT rule on the set of jobs $J\backslash\{j\}$. In our case, in order to compute the values λ^i_{jk}, we used the same approach on both machines M_1 and M_2 separately, i.e., on the first machine, the release dates are r_j and the processing times are equal to p_{1j}. On the second machine, since job j can not be scheduled before $r_j + p_{1j}$ then the release dates are equal to $r_j + p_{1j}$ and the processing times are p_{2j}.

III. LOWER BOUNDS

In this section, we describe the proposed lower bounds in the literature. Then, we present a new single machine-based lower bound. Finally, we propose several lower bounds based on two machine relaxations.

A. LOWER BOUNDS OF THE LITERATURE

In order to derive a set of lower bounds, Ben Chihaoui *et al.* [3] replaced the unavailability intervals by the two dummy jobs (see Section II-A) and used a classical and widely used flow shop relaxation scheme which consists in relaxing the capacity of all the machines but one, say M_k $(k = 1, 2)$. The obtained problem is the well-known single machine problem with release dates (or heads) and delivery times (or tails) denoted by $1|r_j, q_j|C_{max}$.

A first simple and easy way to compute a lower bound based on this scheme consists in removing the first machine and considering only the second one. Thus, we get a set of jobs $J \cup \{\alpha, \beta\}$ with $r_j = r_j + p_{1j}$, $p_j = p_{2j}$ and $q_j = 0$, $\forall j \in J \cup \{\alpha, \beta\}$. The resulting problem is optimally solved by scheduling the jobs in the non-decreasing order of their release dates. This bound is referred by LB_1 and is computed in $O(n \log n)$-time.

A second lower bound is based on the relaxation of the second machine and considering the processing times of the jobs on M_2 (p_{2j}) as the delivery times. The preemptive version of the resulting problem $1|r_j, q_j|C_{max}$, is optimally solved using the so-called Jackson's Preemptive Schedule [21]. It's worth noting that the two dummy jobs are constrained to start at their release dates. The obtained preemptive lower bound is denoted by LB_2 and is calculated in $O(n \log n)$-time. Similar to the previous relaxation but without considering the preemption, a valid lower bound (hereafter referred as LB_3) is derived by solving the latter $\mathcal{NP}$-hard problem [4] optimally by the branch-and-bound algorithm developed by Gharbi and Labidi [15].

Another interesting lower bound is computed by considering the same relaxation scheme for LB_1 but without adding the two fictious jobs. Instead, the unavailability period on the second machine $[s_2, t_2]$ is taken into account. Following the notation of Schmidt [32], the resulting problem is denoted by $1, NC_{win}|r_j, q_j|C_{max}$ where the tails (q_j) are equal to zero. This latter problem is then solved optimally by the dynamic programming algorithm developed by Kacem and Haouari [22]. The obtained lower bound is denoted by LB_4 and clearly dominates LB_1.

After this brief description of the lower bounds of the literature, we move to the presentation of the new proposed bounds.

B. NEW SINGLE MACHINE-BASED LOWER BOUND

The computation of this lower bound is similar in spirit to LB_4 but instead of relaxing the first machine, the second one is relaxed and the machine is not available during $[s_1, t_1]$. The resulting problem $1, NC_{win}|r_j, q_j|C_{max}$, where the heads, the processing times and the tails are, respectively, equal to r_j, p_{1j} and p_{2j}, $\forall j \in J$, is solved using the Integer Linear Program developed by Gharbi *et al.* [16]. It is clear that this bound (denoted by LB_5) dominates LB_2 and LB_3.

C. TWO-MACHINE BASED LOWER BOUNDS

In the sequel, we propose two lower bounds by relaxing some constraints from the original problem in order to get polynomial solvable problems.

1) TRAVELING SALESMAN-BASED LOWER BOUND

The idea of this lower bound consists on relaxing the release dates and the non-availability constraints which are replaced by the two dummy jobs α and β. The obtained problem is $F2|nwt|C_{max}$. Gilmore and Gomory [17] formulated this latter problem as a Traveling Salesman Problem and proved that finding an optimal tour is similar to finding the optimal permutation. Based on this result, we can derive a lower bound by $\min_{j \in J} r_j + Opt(J)$, where $Opt(.)$ is the optimal solution of the $F2|nwt|C_{max}$ problem calculated on the set between brackets. Repeating this result for every release date and considering each time the subset of jobs having their release greater than the actual considered release date, we can derive a lower bound LB_6 which can be formally presented as follows:

$$LB_6 = \max_{j=1,\dots,n} \left\{ r_j^* + Opt(j \in J/r_j \geq r_j^*) \right\}$$

where $r_1^* \leq r_2^* \leq \dots \leq r_n^*$ are the heads r_j $(j \in J)$ sorted in the non-decreasing order.

Since Gilmore and Gomory [17] showed that the two-machine no-wait flow shop problem can be obtained in $O(n \log n)$-time, then it is clear that LB_6 is computed in $O(n^2 \log n)$.

2) PREEMPTION-BASED LOWER BOUND

An interesting lower bound, denoted hereafter by LB_7, is computed by relaxing the no-wait requirement and the non-availability constraints which are replaced by the two dummy jobs α and β. In addition, the preemption is allowed on the first machine. In this context, Cheng *et al.* [7] proved that the two-machine 1-minimal flow shop problem with preemption (denoted by $F2|r_j, 1-min, pmtn|C_{max}$) is polynomially solvable in $O(n \log n)$-time. Note that the two-machine flow shop is called 1-minimal if the processing times satisfy the constraints $p_{1j} \leq p_{2j}$, $\forall j \in J$ [1], [24]. In order to satisfy the 1-minimal condition, the instance is modified by setting $p_{1j} = \min(p_{1j}, p_{2j})$, $\forall j \in J$.

For the sake of completeness, we briefly sketch the basic idea of the algorithm developed by Cheng *et al.* [7].

1) Apply the SRPT rule on the first machine.
2) Schedule the jobs on the second machine according to the non-decreasing completion times on the first machine.

IV. BRANCH-AND-BOUND ALGORITHM

In this section, we provide a detailed description of our branch-and-bound algorithm.

A. PREPROCESSING PROCEDURE

We present two procedures that are executed before starting the branch-and-bound procedure. The first one consists in adjusting the release dates and thus can tighten the value of the lower bounds. The second one, consists in eliminating some jobs from the instance in order to reduce the search space which affects the computational burden.

1) ADJUSTMENT RULE

A simple approach, that was presented by Chihaoui *et al.* [3], for adjusting the release dates is based on the fact that the two machines are constrained to handle any job in the unavailability periods. Clearly, if for any job j $(j = 1, \dots, n)$ with $r_j \leq t_2$ satisfying one of the two following constraints $(r_j + p_{1j} > s_1)$ or $(r_j + p_{1j} + p_{2j} > s_2)$ then the release date of job j should be adjusted as follows: $r_j = max(t_1, t_2 - p_{1j})$.

2) ELIMINATION RULE

We generalize the elimination rule that was presented by Tadei *et al.* [13] for the $F2|r_j|C_{max}$ problem. This rule can be formulated as follows.

Proposition 2: Let $\sigma = (\sigma(1), \sigma(2), \dots, \sigma(n))$ be a complete schedule. If there exists $1 < k < n$ such that the following inequalities hold:

$$\min_{k \leq l \leq n} \left(r_{\sigma(l)} + \sum_{h=1}^{l} p_{h\sigma(l)} \right) \geq C_{k-1}^i, \quad \forall i = 1, 2$$

then the subsequence $(\sigma(1), \sigma(2), \dots, \sigma(k-1))$ can be eliminated from the instance.

Proof 2 (Obvious): Because of the non-availability constraints, we will use this proposition in a more adequate and intelligent manner. To this end, we divide the set of jobs J into two subsets as follows:

- $J_1 = \{j \in J$ such that $r_j + p_{1j} \leq s_1$ and $r_j + p_{1j} + p_{2j} \leq s_2\}$ the subset of jobs that potentially can be scheduled before the non-availability constraints,
- $J_2 = J \backslash J_1$ the subset of jobs that must be scheduled after the non-availability constraints.

The key idea is to derive a schedule on subset J_1 and to test if the generated schedule can be eliminated from the solution. For a more comprehensive presentation, we present the elimination rule by Algorithm 1 in which $C^i(.)$ represents the completion time of the schedule between brackets on machine M_i.

Algorithm 1 Elimination Rule

```
Construct a schedule on the subset $J_1$
**if** $C^1(J_1) \leq s_1$ and $C^2(J_1) \leq s_2$ **then**
    Remove the jobs of $J_1$ from the instance
    Construct a schedule on the subset $J_2$
**end if**
Apply Proposition 2 on the derived schedule
```

For the construction of the sequences, we used the fuzzy scheduling method introduced by Cheng *et al.* [7] for $F2|r_j|C_{max}$ problem. Let $D^l(\sigma ij) = C^l(\sigma ij) - C^l(\sigma ji)$; $l = 1, 2$. A fuzzy function is introduced:

$$\mu_\sigma(i, j) = 0.5 - \frac{D(\sigma ij)}{2\, D_{max}(\sigma)}$$

where $D(\sigma ij) = \sum_{l=1}^{2} \alpha_l D^l(\sigma ij)$ and $D_{max}(\sigma) = \max_{i,j} |D(\sigma ij)|$. The real numbers α_1 and α_2 must verify the equality $\alpha_1 + \alpha_2 = 1$. In our tests, we have set $\alpha_1 = \alpha_2 = 0.5$. The fuzzy function is used to estimate the probability that job i precedes job j for the completion of schedule σ. Thus, the next job i chosen to terminate schedule σ is identified by $\max_{i \in \mathcal{NS}} \min_{j \in \mathcal{NS}} \mu_\sigma(i, j)$ where $\mathcal{NS}$ is the set of unscheduled jobs. The procedure is repeated iteratively for the set of unscheduled jobs J_1 or J_2 to derive a schedule. The derived schedule is also used as the starting solution in the branch-and-bound algorithm.

B. UPPER BOUND

In order to get quickly a strong upper bound, a genetic local search algorithm was implemented in the root node of the tree. We have modified the code of the genetic procedure of Rakrouki *et al.* [31]. The procedure is based on an initial population consisting in a set of solutions generated randomly. Each solution is evaluated by means of its fitness (the value of its makespan). At each iteration, crossover and mutation operations are applied on the current population. After that, a local search step is applied on the generated solutions using two local search heuristics, each one being selected under a probability.

C. BRANCHING SCHEME AND SEARCH STRATEGY

In order to solve the $F2|nr-a, nwt, r_j|C_{max}$ problem, two types of search strategy have been implemented within our branch and bound algorithm:

- Depth first strategy: it consists in branching on the node in a depth-first order.
- Best active new node strategy: it consists in branching on the node which has the smallest lower bound among the most recently created nodes.

D. DOMINANCE RULE

For the aim of improving the performance of our branch-and-bound algorithm, a dominance rule reducing the size of the search space is developed. This dominance rule is based on the set of unscheduled jobs. It's an adaptation of the dynamic programming dominance rule firstly proposed by Potts and Van Wassenhove [30]. The rule can be stated as follows.

Proposition 3: Given two initial partial sequences σ *and* σ' constituting different permutations of the same job set such that $C^i(\sigma') \geq C^i(\sigma), \forall i = 1, 2$ (break ties arbitrarily), then σ dominates σ' and the node corresponding to partial sequence σ' can be pruned.

In order to efficiently use this dominance rule, we used a Hash Table to store the informations. Each entry of the table corresponds to several list of jobs. Each list is sorted in Lexicographic order. Moreover, each list has a list of couples of completion times C^1_{max} and C^2_{max}. In a node of the branch-and-bound algorithm, we have four situations:

- There is no entry in the hash table corresponding to the set of scheduled jobs. Thus, an entry must be created.
- The two completion times are lower than a previously stored couple of completion times then the completion times are updated.
- The two completion times are greater than a previously stored couple of completion times then the node is pruned.
- There is no relation between the completions times of the partial sequence and a previously stored couple of completion times then a couple of completion times is added to the list of jobs.

The rule runs in $O(n \log n)$-time. Indeed, we must sort the set of scheduled jobs in Lexicographic order which is obtained in $O(n \log n)$ and the comparison of two set of jobs is done in $O(n)$.

V. COMPUTATIONAL RESULTS

The performance of the proposed procedures has been assessed on a set of a randomly generated instances. All the algorithms were coded in C++ and compiled with Visual Studio 2010. The *MIP* instances have been solved using Cplex 12.6. All the computational experiments were carried out on a Quad Core 3.40 GHz Personal Computer with 16 GB RAM under Windows 7 Ultimate environment. A time limit equal to 3600 seconds has been set for each variant of our exact procedure.

A. DATA SET

The scheme of the generation of the instances is the same as the one used by Chihaoui *et al.* [3], i.e.,

- The number of jobs n is equal to 10, 15, 20, 25, 30, 35, 40, 45 and 50 jobs.
- The processing times on machine M_1 and M_2 are drawn from the discrete uniform distribution [1,100].
- The release dates are drawn from the discrete uniform distribution $[1,\ 100 \cdot R]$, where $R \in \{1, 2, n, 2n\}$.
- The starting time s_1 of the unavailability period on machine M_1 was fixed at the beginning ($s_1 = 0.25 \cdot T$) or in the middle ($s_1 = 0.5 \cdot T$) where T represents an horizon of the schedule computed by $T = \sum_{j \in J} p_{1j} + \sum_{j \in J} p_{2j}$. The starting time s_2 of the unavailability period on machine M_2 is set to $s_1 + 0.25 \cdot \frac{T}{n}$
- The ending times t_1 and t_2 are set to $s_1 + \frac{T}{n}$ and $s_2 + 1.25 \cdot \frac{T}{n}$, respectively.
- 10 instances were generated for each combination of n, R and s_1 for a total of 400 instances.
- The instances were grouped into 8 sets $(S1, S2, \ldots, S8)$ where the odd sets (S_1, S_3, S_5 and S_7) corresponds to $s_1 = 0.25 \cdot T$ and the even sets to $s_1 = 0.5 \cdot T$.

B. COMPARISON OF THE LOWER BOUNDS

The results of a comparison of our lower bounds, according to the variation of the set $(S1, S2, \ldots, S8)$ and n $(10, 15, 20, 25, 30, 35, 40, 45, 50)$, are depicted in Table 2. For each lower bound, we provide the following:

- Gap: the mean percentage deviation with respect to the Genetic-based upper bound.
- Time: the mean CPU time (in seconds). The computational times less than 0.01 second are denoted by "-".
- Max: the number of times where the lower bound is maximal, i.e., gives the best value among all the lower bounds $LB_i (i = 1, \ldots, 9)$.

In addition to the lower bounds described in Section III, we added two other lower bounds: LB_8 is based on the linear relaxation of (*MIP*) and LB_9 is based on the linear relaxation of (*MIP*) along with constraints (13).

Table 2 depicts:

TABLE 2. Detailed performance of the lower bounds.

		LB1			LB2			LB3			LB4			LB5			LB6			LB7			LB8			LB9		
Set	n	Gap (%)	Time (s)	Max	Gap (%)	Time (s)	Max	Gap (%)	Time (s)	Max	Gap (%)	Time (s)	Max	Gap (%)	Time (s)	Max	Gap (%)	Time (s)	Max	Gap (%)	Time (s)	Max	Gap (%)	Time (s)	Max	Gap (%)	Time (s)	Max
	10	12.43	-	2	7.72	-	0	7.62	-	1	12.22	-	2	7.45	0.02	1	7.45	-	7	3.66	-	1	12.65	0.08	3	5.50	0.08	3
	15	9.26	-	1	10.58	-	1	10.58	-	1	9.26	-	1	10.58	0.01	1	10.58	-	7	2.39	-	1	9.34	0.24	2	3.97	0.24	4
	20	4.46	-	3	13.80	-	0	13.80	-	0	4.46	-	3	13.80	0.01	0	13.80	-	7	1.43	-	3	4.57	0.43	2	2.82	0.43	3
	25	4.47	-	0	12.00	-	0	11.99	-	0	4.47	-	0	11.99	0.01	0	11.99	-	9	1.33	-	1	4.67	0.73	0	2.84	0.73	0
S1	30	6.79	-	2	8.64	-	1	8.64	-	1	6.79	-	2	8.64	0.02	1	8.64	-	8	0.90	-	0	7.04	1.30	3	1.63	1.30	3
	35	3.44	-	1	11.64	-	0	11.64	-	0	3.44	-	1	11.64	0.02	0	11.64	-	8	0.68	-	1	3.57	1.87	2	1.39	1.87	2
	40	3.14	-	3	8.37	-	0	8.37	-	0	3.14	-	3	8.37	0.02	0	8.37	-	7	1.24	-	1	3.32	2.75	3	2.14	2.74	3
	45	6.02	-	0	6.59	-	0	6.59	-	0	6.02	-	0	6.59	0.02	0	6.59	-	10	0.95	-	0	6.11	3.17	0	2.36	3.15	0
	50	4.64	-	2	8.01	-	0	7.99	-	0	4.64	-	2	7.99	0.03	0	7.99	-	8	0.73	-	0	4.75	4.70	1	1.92	4.64	2
	10	12.76	-	1	23.58	-	0	22.83	-	0	12.28	-	4	22.04	0.03	4	3.75	-	2	5.28	-	1	3.97	0.09	1	3.65	0.09	1
	15	17.78	-	2	14.20	-	0	13.79	-	1	17.75	-	3	12.33	0.05	2	2.60	-	5	5.93	-	2	4.17	0.20	2	4.01	0.20	2
	20	17.51	-	1	8.51	-	0	8.34	-	0	17.49	-	2	8.22	0.04	0	1.22	-	8	5.60	-	0	2.92	0.52	0	2.57	0.51	1
	25	5.85	-	2	19.45	-	0	19.45	-	0	5.83	-	3	18.05	0.01	0	1.71	-	7	3.78	-	1	3.86	1.01	2	3.68	1.00	2
S2	30	8.42	-	0	13.59	-	0	13.56	-	0	8.42	-	0	11.69	0.04	0	1.43	-	10	3.63	-	0	3.53	1.40	0	3.17	1.40	0
	35	5.96	-	3	15.77	-	0	15.73	-	0	5.96	-	3	15.18	0.03	0	1.71	-	7	3.07	-	2	2.43	2.30	3	2.32	2.28	3
	40	8.12	-	2	9.51	-	0	9.46	-	0	8.08	-	2	8.88	0.09	0	1.58	-	8	4.22	-	2	3.30	2.57	1	3.16	2.56	2
	45	7.92	-	1	9.55	-	0	9.53	-	0	7.92	-	1	9.04	0.09	0	1.39	-	9	3.83	-	0	2.99	4.02	1	2.90	3.99	1
	50	7.48	-	2	9.51	-	0	9.51	-	0	7.48	-	2	9.15	0.03	0	1.14	-	8	3.91	-	1	2.59	6.30	2	2.55	6.21	2
	10	5.16	-	2	15.72	-	0	15.72	-	0	4.94	-	5	15.54	0.01	0	6.24	-	5	5.40	-	1	5.47	0.08	2	4.38	0.08	2
	15	5.91	-	3	11.68	-	0	11.68	-	0	5.91	-	3	11.68	0.01	0	2.81	-	6	6.21	-	2	3.61	0.20	2	3.21	0.21	4
	20	10.89	-	2	7.43	-	0	7.43	-	0	10.89	-	2	7.43	0.01	0	1.84	-	7	11.34	-	1	2.39	0.44	3	2.39	0.45	3
	25	3.93	-	3	14.17	-	0	14.11	-	0	3.93	-	3	14.11	0.01	0	2.01	-	5	4.27	-	4	2.34	0.80	3	2.25	0.80	4
S3	30	4.52	-	4	10.42	-	0	10.42	-	0	4.52	-	4	10.42	0.01	0	1.26	-	6	4.69	-	4	1.81	1.27	4	1.77	1.28	4
	35	7.33	-	1	6.74	-	0	6.74	-	0	7.33	-	1	6.74	0.02	0	1.15	-	9	7.46	-	1	2.85	2.12	1	2.80	2.11	1
	40	6.01	-	1	8.28	-	1	8.28	-	1	6.01	-	1	8.28	0.03	1	1.07	-	8	6.22	-	1	2.44	2.36	2	2.38	2.35	3
	45	3.68	-	2	8.45	-	0	8.45	-	0	3.68	-	2	8.45	0.01	0	0.95	-	8	3.71	-	1	2.37	3.48	2	2.32	3.43	2
	50	5.58	-	1	6.12	-	1	6.12	-	1	5.58	-	1	6.12	0.04	1	0.87	-	9	5.62	-	1	1.90	5.19	2	1.90	5.13	2
	10	8.83	-	3	26.34	-	0	25.23	-	0	8.59	-	6	22.98	0.05	2	5.57	-	2	5.16	-	1	4.24	0.08	3	3.69	0.08	3
	15	4.87	-	4	24.26	-	0	24.19	-	0	4.86	-	4	20.23	0.04	0	2.34	-	5	3.59	-	4	3.73	0.22	2	3.41	0.22	4
	20	15.76	-	1	11.05	-	2	10.93	-	2	15.71	-	1	10.81	0.04	2	1.67	-	9	6.10	-	1	3.78	0.46	3	3.48	0.46	3
	25	7.22	-	3	15.08	-	0	15.03	-	0	7.14	-	3	14.27	0.02	0	2.54	-	6	3.40	-	2	2.69	0.86	3	2.64	0.86	3
S4	30	5.76	-	1	15.99	-	0	15.92	-	0	5.74	-	1	15.25	0.04	0	1.84	-	8	3.65	-	1	3.29	1.52	2	3.19	1.52	2
	35	7.88	-	0	11.14	-	0	11.12	-	0	7.87	-	0	10.07	0.03	0	1.60	-	9	4.04	-	0	3.02	2.21	0	2.84	2.21	1
	40	6.81	-	0	11.61	-	0	11.57	-	0	6.80	-	0	10.55	0.06	0	1.44	-	10	4.06	-	0	3.41	3.28	0	3.22	3.26	0
	45	11.96	-	1	6.28	-	0	6.27	-	1	11.96	-	1	5.87	0.05	1	1.51	-	8	4.90	-	0	2.28	4.20	2	2.21	4.16	2
	50	8.83	-	2	8.96	-	0	8.96	-	0	8.83	-	2	8.55	0.05	0	1.44	-	8	4.12	-	1	2.28	5.77	0	2.17	5.70	2
	10	0.88	-	7	2.39	-	5	2.01	-	6	0.88	-	7	2.01	2.01	6	1.02	-	7	2.44	-	2	18.84	0.07	0	0.54	0.07	8
	15	0.52	-	8	0.46	-	7	0.46	-	7	0.52	-	8	0.46	0.46	7	0.43	-	8	1.32	-	4	25.31	0.15	0	0.31	0.15	9
	20	0.81	-	6	0.81	-	4	0.76	-	5	0.81	-	6	0.76	0.76	5	0.43	-	5	1.36	-	4	24.32	0.30	0	0.29	0.30	10
	25	0.90	-	5	0.38	-	6	0.23	-	9	0.90	-	5	0.23	0.23	9	0.35	-	6	1.65	-	2	30.03	0.44	0	0.31	0.43	7
S5	30	0.39	-	5	0.24	-	5	0.18	-	7	0.39	-	5	0.18	0.18	7	0.22	-	6	0.87	-	2	27.30	0.63	0	0.20	0.62	8
	35	0.29	-	8	0.59	-	4	0.42	-	6	0.29	-	8	0.42	0.42	6	0.48	-	4	0.37	-	5	27.43	0.92	0	0.24	0.87	8
	40	0.29	-	7	0.29	-	7	0.25	-	7	0.29	-	7	0.25	0.25	7	0.24	-	8	0.47	-	4	28.80	1.22	0	0.20	1.13	9
	45	0.47	-	8	1.17	-	3	1.17	-	3	0.47	-	8	1.17	1.17	3	0.50	-	3	0.73	-	5	24.03	1.76	0	0.19	1.63	10
	50	0.34	-	6	0.43	-	3	0.38	-	4	0.34	-	6	0.38	0.38	4	0.29	-	6	0.48	-	3	28.78	2.15	0	0.30	1.98	6
	10	1.93	-	5	5.73	-	2	5.64	-	3	1.16	-	7	5.64	0.00	3	1.96	-	5	2.57	-	4	12.07	0.08	0	1.72	0.08	5
	15	0.76	-	7	1.03	-	5	0.83	-	5	0.76	-	7	0.83	0.02	5	0.87	-	7	1.57	-	3	18.04	0.16	0	0.62	0.16	9
	20	1.08	-	4	1.74	-	4	1.57	-	5	1.08	-	4	1.57	0.02	5	0.71	-	5	1.81	-	2	21.81	0.27	0	0.37	0.27	9
	25	0.88	-	5	0.81	-	6	0.74	-	8	0.83	-	6	0.74	0.01	8	0.46	-	7	1.90	-	1	23.69	0.44	0	0.34	0.43	7
S6	30	0.25	-	7	1.37	-	4	1.29	-	4	0.25	-	7	1.29	0.02	4	0.68	-	6	0.97	-	4	26.67	0.62	0	0.23	0.60	9
	35	0.27	-	7	0.58	-	3	0.58	-	3	0.27	-	7	0.58	0.00	3	0.27	-	6	0.42	-	6	26.13	0.89	0	0.15	0.84	8
	40	0.18	-	7	0.50	-	5	0.41	-	6	0.18	-	7	0.41	0.05	6	0.23	-	6	0.72	-	4	26.53	1.28	0	0.16	1.20	7
	45	0.16	-	8	0.39	-	4	0.32	-	6	0.16	-	8	0.32	0.00	6	0.23	-	4	0.42	-	4	27.39	1.60	0	0.14	1.49	8
	50	0.36	-	6	0.54	-	4	0.46	-	6	0.36	-	6	0.46	0.10	6	0.34	-	5	0.51	-	4	26.42	2.12	0	0.23	1.97	7
	10	0.33	-	6	0.22	-	6	0.06	-	8	0.33	-	6	0.06	-	8	0.22	-	6	1.73	-	2	35.38	0.07	0	0.06	0.07	9
	15	0.10	-	7	0.11	-	7	0.09	-	8	0.10	-	7	0.09	-	8	0.06	-	8	0.66	-	5	39.32	0.15	0	0.05	0.15	9
	20	0.05	-	9	0.20	-	8	0.20	-	8	0.05	-	9	0.20	-	8	0.05	-	9	0.44	-	5	40.93	0.25	0	0.04	0.25	10
	25	0.00	-	10	0.10	-	7	0.10	-	7	0.00	-	10	0.10	-	7	0.03	-	7	0.42	-	4	38.64	0.40	0	0.00	0.38	10
S7	30	0.23	-	6	0.13	-	7	0.03	-	9	0.23	-	6	0.03	-	9	0.13	-	7	0.45	-	5	40.08	0.60	0	0.02	0.57	9
	35	0.00	-	10	0.10	-	9	0.05	-	9	0.00	-	10	0.05	-	9	0.08	-	9	0.43	-	4	39.83	0.83	0	0.00	0.79	10
	40	0.00	-	10	0.04	-	9	0.04	-	9	0.00	-	10	0.04	-	9	0.04	-	9	0.17	-	4	40.01	1.10	0	0.00	1.02	10
	45	0.02	-	9	0.22	-	7	0.22	-	8	0.02	-	9	0.22	-	8	0.11	-	7	0.12	-	4	40.28	1.45	0	0.02	1.35	10
	50	0.01	-	8	0.05	-	7	0.04	-	8	0.01	-	8	0.04	-	8	0.02	-	9	0.03	-	8	40.61	1.92	0	0.01	1.74	8
	10	0.43	-	6	0.28	-	7	0.07	-	9	0.43	-	6	0.07	-	9	0.19	-	8	0.86	-	4	33.57	0.07	0	0.37	0.07	6
	15	0.01	-	9	0.26	-	8	0.21	-	8	0.01	-	9	0.21	-	8	0.18	-	8	0.21	-	6	33.83	0.15	0	0.00	0.15	10
	20	0.05	-	9	0.08	-	7	0.07	-	8	0.05	-	9	0.07	-	8	0.04	-	8	0.43	-	3	40.28	0.25	0	0.04	0.24	9
	25	0.06	-	9	0.09	-	8	0.07	-	9	0.06	-	9	0.07	-	9	0.02	-	9	0.11	-	6	40.20	0.39	0	0.05	0.38	9
S8	30	0.01	-	9	0.01	-	9	0.01	-	9	0.01	-	9	0.01	-	9	0.01	-	9	0.33	-	4	38.32	0.57	0	0.01	0.54	10
	35	0.15	-	7	0.17	-	4	0.15	-	5	0.15	-	7	0.15	-	5	0.11	-	8	0.22	-	6	40.70	0.84	0	0.09	0.79	7
	40	0.00	-	10	0.05	-	9	0.05	-	9	0.00	-	10	0.05	-	9	0.00	-	10	0.16	-	5	40.37	1.09	0	0.00	1.02	10
	45	0.03	-	9	0.13	-	7	0.10	-	8	0.03	-	9	0.10	-	8	0.07	-	7	0.20	-	3	40.55	1.40	0	0.03	1.31	9
	50	0.01	-	8	0.32	-	5	0.29	-	6	0.01	-	8	0.29	-	6	0.05	-	6	0.08	-	6	38.66	1.88	0	0.00	1.73	8

- LB_6 and LB_9 exhibit the best performance among all the developed lower bounds. Indeed, LB_9 outperforms LB_6 for instances with $n \leq 30$ and the performance is inverted on instances with $n > 30$. Moreover, LB_6 yields to the maximum lower bound in almost 72% of the instances with a relatively large difference to LB_9 (54% of the instances). Furthermore, LB_9 needs more computational effort (more than 1 second for $n \geq 30$).

TABLE 3. Performance of the branch-and-bound algorithm.

	B1		B2		B3		D1		D2		D3	
n	SI (%)	Time (s)	SI (%)	Time (s)	SI (%)	Time (s)	SI (%)	Time (s)	SI (%)	Time (s)	SI (%)	Time (s)
10	100	0.19	100	0.18	100	0.18	100	0.19	100	0.18	100	0.18
15	100	6.16	100	0.74	100	0.71	100	6.46	100	0.86	100	0.79
20	86	724.13	100	49.25	100	42.73	85	754.6	100	68.18	100	60.75
25	60	1544.35	69	1267.21	69	1229.4	60	1557.37	70	1296.82	70	1230.47
30	55	1622.99	51	1847.13	59	1582.47	55	1625.59	50	1897.59	58	1612.26

- Interestingly, we see that the valid inequalities have a strong impact on the performance of the mathematical formulation.
- Clearly, LB_5 outperforms the state-of-the-art lower bounds (LB_2 and LB_3) but its performance remains however quite weak compared with the other lower bounds.
- We remark that LB_7 provides better performance than LB_4 for the sets $S1$, $S2$, $S3$ and $S4$ but this performance seems dramatically sensitive to the other sets ($S5$, $S6$, $S7$ and $S8$) where LB_4 dominates LB_7.

At this point, the results suggest that computing LB_5 or LB_9 at each node of the tree is not worthwhile. Consequently, in each node N, we compute the lower bound $LB(\mathcal{NS})$ on the set of unscheduled jobs as follows: $LB(\mathcal{NS}) = max(LB_6, LB_7)$. However, in order to speed up our exact procedure, LB_6 was computed only for the minimal release date, i.e., $LB_6 = \min_{j \in J} r_j + Opt(J)$ where $Opt(J)$ is the optimal solution of the $F2|nwt|C_{max}$ problem calculated on the job set J. Moreover, it should be made clear that the lower bound at the root node of the search tree is defined as follows: $LB(root\ node) = \max(LB_4, LB_5, LB_6, LB_7, LB_9)$.

C. PERFORMANCE OF THE BRANCH-AND-BOUND ALGORITHM

Actually, we implemented six variants of our exact procedure denoted by $B1$, $B2$ and $B3$ for the variants using the best active new node strategy and $D1$, $D2$ and $D3$ for those using the depth first strategy. The utilization of the elimination and dominance rules is as follows:

- $B1$ and $D1$: only the elimination rule.
- $B2$ and $D2$: only the dominance rule.
- $B3$ and $D3$: both the elimination and dominance rules.

In Table 3, we evaluate the performance of the different branch-and-bound procedures. We provide the number of solved instance (denoted by *SI*) together with the average CPU time (denoted by *Time*). All the exact procedures are compared for n equal to 10, 15, 20, 25, 30. According to this Table, we notice that all the branch-and-bound variants are able to solve 100% of the instances for n equal to 10 and 15 with a computational time less than 7 second. Moreover, $B2$, $B3$, $D2$ and $D3$ enable to solve all of the generated instances for n equal 20 within an average CPU time of 42, 73 seconds for $B3$. Clearly, $B3$ and $D3$ exhibit the best performance due to the importance of the dominance and elimination rules. Following those two variants, only few instances still unsolved (12, 5% and 26, 25% for n equal to 25 and 30, respectively). However, $D3$ is outperformed by $B3$ in term of computational effort.

D. COMPARISON OF THE EXACT METHODS

The performance of the the best variant of our branch-and-bound algorithm, namely $B3$, according to the variation of the set and the number of jobs n is analyzed in Table 4. This procedure is compared to the exact resolution of the mathematical model (*MIP*) along with constraints (13). To have an exact idea about the performance of our approaches, we coded the branch-and-bound proposed by Chihaoui *et al.* [3] as described by the authors with only a minor modification on the upper bound. For clarity, we recall the different components of this algorithm:

- The depth first strategy is considered,
- The lower bounds used are $LB1$ and $LB4$. If $LB1$ fails to eliminate the node then $LB4$ is computed,
- The root node is the maximum between $LB2$, $LB3$ and $LB4$,
- The upper bound developed is our genetic algorithm described in Section IV-B.

The depicted results show the high efficiency of our exact solution approaches. Indeed, we observe that, on one hand, for the sets $S5$, $S6$, $S7$ and $S8$, our branch-and-bound algorithm $B3$ enables to solve almost all the generated instances within a very short CPU time (less than 1 second in about all the instances). Moreover, our formulation do the same but it requires more CPU times in few cases. By contrast, we see that the state-of-the-art algorithm of Chihaoui *et al.* [3] is not able to solve all the cases for the same set of instances while it requires much more CPU time. On the other hand, $B3$ is able to solve all the generated instances for the set $S1$, $S2$, $S3$ and $S4$ for $n \leq 20$ with a maximum average CPU time of 133.69 seconds. Nevertheless, the performance of our procedures ($B3$ and Formulation) remain acceptable for instances with $n \geq 25$ where only few cases still not solved within the limit. Interestingly, we remark that the formulation is able to solve more instances than $B3$ for this type of instances on $S2$ and $S4$. For the same set of instances, the B&B of Chihaoui *et al.* [3] can solve only few cases for $n = 20$ and fail to solve all but one instance with $n \geq 25$.

TABLE 4. Comparison of the exact methods.

		B3		Formulation		B&B Chihaoui et al. (2011)	
Set	**n**	**SI (%)**	**Time (s)**	**SI (%)**	**Time (s)**	**SI (%)**	**Time (s)**
	10	100	0.19	100	0.41	100	0.09
	15	100	0.64	100	15.69	100	329.56
S1	20	100	58.09	90	932.95	20	2907.94
	25	80	1172.20	10	3250.40	0	3600.7
	30	50	1810.23	20	3105.46	0	3601.25
	10	100	0.22	100	0.40	100	0.12
	15	100	2.04	100	9.13	90	538.71
S2	20	100	133.69	70	1380.81	10	3507.81
	25	0	3603.20	20	2940.47	0	3600.97
	30	0	3605.43	0	3628.93	0	3601.36
	10	100	0.19	100	0.32	100	0.08
	15	100	0.47	100	7.90	100	8.61
S3	20	100	15.91	80	1757.47	20	3276.11
	25	60	1750.75	50	2019.85	10	3240.77
	30	20	3095.74	10	3280.67	0	3601.23
	10	100	0.23	100	0.39	100	0.09
	15	100	1.30	100	7.48	100	34.93
S4	20	100	131.46	70	1418.75	30	3073.93
	25	10	3305.81	20	2981.30	0	3600.82
	30	10	3453.52	20	2908.85	0	3601.47
	10	100	0.16	100	0.34	100	0.07
	15	100	0.29	100	2.84	100	0.14
S5	20	100	1.20	90	362.26	70	1118.48
	25	100	0.93	100	6.64	70	1080.4
	30	90	691.50	40	2173.52	70	1080.57
	10	100	0.17	100	0.28	100	0.08
	15	100	0.33	100	0.63	100	1.97
S6	20	100	0.53	100	1.95	70	1131.67
	25	100	0.85	100	1.83	80	722.05
	30	100	1.18	100	4.76	70	1080.56
	10	100	0.13	100	0.32	100	0.07
	15	100	0.30	100	0.41	100	86.8
S7	20	100	0.48	100	0.61	90	360.23
	25	100	0.74	100	2.16	100	0.36
	30	100	1.13	100	2.10	80	720.53
	10	100	0.16	100	0.34	100	0.07
	15	100	0.28	100	0.55	100	37.59
S8	20	100	0.46	100	0.60	90	360.22
	25	100	0.73	100	2.15	100	0.35
	30	100	1.04	100	1.35	90	360.5

VI. CONCLUSION

In this paper, we have presented a computational study for the two-machine no-wait flow shop scheduling problem with a single unavailability period on each machine and job release dates. This problem arises in the context of many real life industrial problems (i.e., scheduling with planned preventive maintenance operations). We have presented a mathematical formulation and valid inequalities for the problem. We also provided very tight lower bounds and have embedded them within an exact search procedure. We have reported the results of extensive computational experiments that prove that the proposed approaches consistently provide optimal solution for instances with up to 25 jobs while requiring short CPU times.

For future research, the investigated algorithm could be incorporated within more complex and realistic scheduling problems with unavailability constraints. More precisely, this research can be extended to the case where we have multiple unavailability periods.

REFERENCES

[1] J. O. Achugbue and F. Y. Chin, “Complexity and solutions of some three-stage flow shop scheduling problems,” *Math. Oper. Res.*, vol. 7, no. 4, pp. 532–544, 1982.

[2] T. P. Bagchi, J. N. D. Gupta, and C. Sriskandarajah, "A review of TSP based approaches for flowshop scheduling," *Eur. J. Oper. Res.*, vol. 169, no. 3, pp. 816–854, 2006, doi: 10.1016/j.ejor.2004.06.040.

[3] F. B. Chihaoui, I. Kacem, A. B. Hadj-Alouane, N. Dridi, and N. Rezg, "No-wait scheduling of a two-machine flow-shop to minimise the makespan under non-availability constraints and different release dates," *Int. J. Prod. Res.*, vol. 49, no. 21, pp. 6273–6286, 2011, doi: 10.1080/00207543.2010.531775.

[4] J. Carlier, "The one-machine sequencing problem," *Eur. J. Oper. Res.*, vol. 11, no. 1, pp. 42–47, 1982, doi: 10.1016/S0377-2217(82)80007-6.

[5] T. C. E. Cheng and Z. Liu, "Approximability of two-machine no-wait flowshop scheduling with availability constraints," *Oper. Res. Lett.*, vol. 31, no. 4, pp. 319–322, 2003, doi: 10.1016/S0167-6377(02)00230-4.

[6] T. C. E. Cheng and Z. Liu, "Approximation for two-machine no-wait flowshop scheduling with availability constraints," *Inf. Process. Lett.*, vol. 88, no. 4, pp. 161–165, 2003, doi: 10.1016/j.ipl.2003.08.002.

[7] J. Cheng, G. Steiner, and P. Stephenson, "A computational study with a new algorithm for the three-machine permutation flow-shop problem with release times," *Eur. J. Oper. Res.*, vol. 130, no. 3, pp. 559–575, 2001, doi: 10.1016/S0377-2217(99)00415-4.

[8] J. Cheng, G. Steiner, and P. Stephenson, "Fast algorithms to minimize the makespan or maximum lateness in the two-machine flow shop with release times," *J. Scheduling*, vol. 5, no. 1, pp. 71–92, 2002, doi: 10.1002/jos.94.

[9] L. A. Hall, "A polynomial approximation scheme for a constrained flowshop scheduling problem," *Math. Oper. Res.*, vol. 19, no. 1, pp. 68–85, 1994, doi: 10.1287/moor.19.1.68.

[10] K. N. Kashyrskikh, C. N. Potts, and S. V. Sevastianov, "A 3/2-approximation algorithm for two-machine flow-shop sequencing subject to release dates," *Discrete Appl. Math.*, vol. 114, nos. 1–3, pp. 255–271, 2001, doi: 10.1016/S0166-218X(00)00374-7.

[11] M. Y. Kovalyov and F. Werner, "A polynomial approximation scheme for problem $F2|r_j|C_{max}$," *Oper. Res. Lett.*, vol. 20, no. 2, pp. 75–79, 1997, doi: 10.1016/S0167-6377(96)00049-1.

[12] C. N. Potts, "Analysis of heuristics for two-machine flow-shop sequencing subject to release dates," *Math. Oper. Res.*, vol. 10, no. 4, pp. 576–584, 1985, doi: 10.1287/moor.10.4.576.

[13] R. Tadei, J. N. D. Gupta, F. D. Croce, and M. Cortesi, "Minimising makespan in the two-machine flow-shop with release times," *J. Oper. Res. Soc.*, vol. 49, no. 1, pp. 77–85, 1998, doi: 10.1057/palgrave.jors.2600481.

[14] C. Chu, "A branch-and-bound algorithm to minimize total tardiness with different release dates," *Naval Res. Logistics*, vol. 39, no. 2, pp. 1520–1650, 1992, doi: 10.1002/1520-6750(199203)39:2<265::AID-NAV3220390209>3.0.CO.2-L.

[15] A. Gharbi and M. Labidi, "Jackson's semi-preemptive scheduling on a single machine," *Comput. Oper. Res.*, vol. 37, no. 12, pp. 2082–2088, 2010, doi: 10.1016/j.cor.2010.02.008.

[16] A. Gharbi, M. Labidi, and M. Haouari, "An exact approach for single machine scheduling with unavailability periods," *Eur. J. Ind. Eng.*, vol. 9, no. 2, pp. 244–260, 2015, doi: 10.1504/EJIE.2015.068654.

[17] P. C. Gilmore and R. E. Gomory, "Sequencing a one-state variable machine: A solvable case of the traveling salesman problem," *Oper. Res.*, vol. 12, no. 5, pp. 655–679, 1964, doi: 10.1287/opre.12.5.655.

[18] M. Haouari and M. Kharbeche, "An assignment-based lower bound for a class of two-machine flow shop problems," *Comput. Oper. Res.*, vol. 40, no. 7, pp. 1693–1699, 2013, doi: 10.1016/j.cor.2013.01.001.

[19] M. Haouari and T. Ladhari, "Minimizing maximum lateness in a two-machine flowshop," *J. Oper. Res. Soc.*, vol. 51, no. 9, pp. 1100–1106, 2000, doi: 10.2307/254231.

[20] F. Hnaien, F. Yalaoui, and A. Mhadhbi, "Makespan minimization on a two-machine flowshop with an availability constraint on the first machine," *Int. J. Prod. Econ.*, vol. 164, pp. 95–104, Jun. 2015, doi: 10.1016/j.ijpe.2015.02.025.

[21] W. A. Horn, "Some simple scheduling algorithms," *Naval Res. Logistics Quart.*, vol. 21, no. 1, pp. 177–185, 1974, doi: 10.1002/nav.3800210113.

[22] I. Kacem and M. Haouari, "Approximation algorithms for single machine scheduling with one unavailability period," *Quart. J. Oper. Res.*, vol. 7, no. 1, pp. 79–92, Mar. 2009, doi: 10.1007/s10288-008-0076-6.

[23] A. Kooli and M. Serairi, "A mixed integer programming approach for the single machine problem with unequal release dates," *Comput. Oper. Res.*, vol. 51, pp. 323–330, Nov. 2014, doi: 10.1016/j.cor.2014.06.013.

[24] C. Koulamas, "On the complexity of two-machine flowshop problems with due date related objectives," *Eur. J. Oper. Res.*, vol. 106, no. 1, pp. 95–100, 1998, doi: 10.1016/S0377-2217(98)00323-3.

[25] M. A. Kubzin and V. A. Strusevich, "Two-machine flow shop no-wait scheduling with a nonavailability interval," *Naval Res. Logistics*, vol. 51, no. 4, pp. 613–631, 2004, doi: 10.1002/nav.10118.

[26] J. B. Lasserre and M. Queyranne, "Generic scheduling polyhedra and a new mixed-integer formulation for single-machine scheduling," in *Proc. 2nd IPCO Conf.*, 1992, pp. 136–149.

[27] C. Y. Lee, "Two-machine flowshop scheduling with availability constraints," *Eur. J. Oper. Res.*, vol. 114, no. 2, pp. 420–429, 1999, doi: 10.1016/S0377-2217(97)00452-9.

[28] Y. Ma, C. Chu, and C. Zuo, "A survey of scheduling with deterministic machine availability constraints," *Comput. Ind. Eng.*, vol. 58, no. 2, pp. 199–211, 2010, doi: 10.1016/j.cie.2009.04.014.

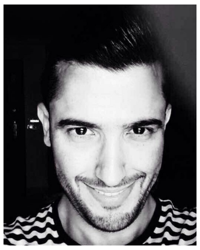
[29] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. New York, NY, USA: Springer, 2008.

[30] C. N. Potts and L. N. van Wassenhove, "A branch and bound algorithm for the total weighted tardiness problem," *Oper. Res.*, vol. 33, no. 2, pp. 363–377, 1985, doi: 10.1287/opre.33.2.363.

[31] M. A. Rakrouki, T. Ladhari, and V. T'kindt, "Coupling genetic local search and recovering beam search algorithms for minimizing the total completion time in the single machine scheduling problem subject to release dates," *Comput. Oper. Res.*, vol. 39, no. 6, pp. 1257–1264, 2012, doi: 10.1016/j.cor.2010.12.007.

[32] G. Schmidt, "Scheduling with limited machine availability," *Eur. J. Oper. Res.*, vol. 121, no. 1, pp. 1–15, 2000, doi: 10.1016/S0377-2217(98)00367-1.

[33] M. L. Espinouse, P. Formanowicz, and B. Penz, "Minimizing the makespan in the two-machine no-wait flow-shop with limited machine availability," *Comput. Ind. Eng.*, vol. 37, nos. 1–2, pp. 497–500, 1999, doi: 10.1016/S0360-8352(99)00127-8.



MOHAMED LABIDI received the B.Sc. degree in computer science, and the master's and Ph.D. degrees in operations research from the High Institute of Management, Tunisia.

He was a Lecturer in computer science with the High School of Economic and Commercial Sciences, Tunisia, for four years. Since 2011, he has been an Assistant Professor with the Industrial Engineering Department, College of Engineering, King Saud University.

His primary areas of research are operations research and combinatorial optimization. His main interest is scheduling problems with special focus on the job shop problem.

ANIS KOOLI received the B.Sc. degree in computer science from the National School of Computer Science, Tunisia, the master's degree in applied mathematics from the National School of Engineers, Tunisia, and the Ph.D. degree in operations research from the High Institute of Management, Tunisia.

He was a Lecturer in computer science with the High School of Economic and Commercial Sciences, Tunisia, for five years. Since 2013, he has been an Assistant Professor in computer science with the Higher School of Economic and Commercial Sciences of Tunis (Ecole Supérieure des Sciences Economiques et Commerciales de Tunis), University of Tunis, Tunisia.

His primary areas of research are operations research and combinatorial optimization. His main interest is scheduling problems with special focus on the resource constrained project scheduling problem and flow shop scheduling problem.

TALEL LADHARI received the B.S. degree in computer Science, the M.S. degree in operations research, and the Ph.D. degree in management science from the University of Tunis, Tunisia, in 1996, 1999, and 2003, respectively.

Since 2011, he has been a Professor of operations research with the Higher School of Economic and Commercial Sciences (Ecole Supérieure des Sciences Economiques et Commerciales de Tunis). His research interests include the design of exact and heuristic methods to solve hard combinatorial optimization problems, and the application of the multiple-criteria decision-making in many fields.

ANIS GHARBI received the B.Sc. degree in mathematics from the Faculty of Science of Tunis, University of Tunis, the master's degree in operations research from the High Institute of Management of Tunis, University of Tunis, the Ph.D. degree in management science from the High Institute of Management of Tunis, University of Tunis, and the Habilitation degree in management science from the High Institute of Management, University of Tunis. He is currently a Professor of operations research with the Industrial Engineering Department, College of Engineering, King Saud University, Saudi Arabia.

His research interests focus on the design, implementation, and analysis of optimization algorithms for hard combinatorial problems with applications on machine scheduling and bin packing. He has conducted his research within the Combinatorial Optimization Research Group-ROI, Polytechnic School of Tunisia.

UMAR S. SURYAHATMAJA received the B.S. degree in industrial engineering from Gadjah Mada University, Indonesia, in 2008, and the M.S. degree in industrial engineering from King Saud University, Riyadh, Saudi Arabia, in 2017, where he is currently pursuing the Ph.D. degree.