IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 23, 2019, accepted November 11, 2019, date of publication November 21, 2019,
date of current version December 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2954905

Bounding Strategies for the Parallel Processors Scheduling Problem With No-Idle Time Constraint, Release Date, and Delivery Time

LOTFI HIDRI, ALI M. AL-SAMHAN, AND MOHAMMED M. MABKHOT
Department of Industrial Engineering, King Saud University, Riyadh 11400, Saudi Arabia

Corresponding author: Lotfi Hidri (lhidri@ksu.edu.sa)

This work was supported by the Deanship of Scientific Research at King Saud University through the Research Group under Grant RG-1439-56.

ABSTRACT The identical parallel processors scheduling problem with no-idle time, release date, and delivery time is addressed in this paper. The problem considers a family of tasks that has to be processed by identical parallel processors without idle time. Each task is ready for processing from a release date (arrival time) in an available processor. After completing the processing, a task is delivered during a delivery time. There is no-idle time in each processor from the first treated task until the last one. This is the no-idle processor time constraint, which is faced in real life problems. In these problems, minimizing the consumed energy during the processing of tasks is a crucial issue. Building a feasible schedule satisfying all the already mentioned constraints and minimizing the makespan (maximum completion time) is the objective. The studied scheduling problem is proofed to be NP-Hard in the strong sense. Therefore, a family of efficient heuristics solving the addressed problem are proposed. These heuristics are composed of two phases: Phase 1 and Phase 2. The building of a feasible schedule is performed during phase 1, while in the second phase (phase 2) an improvement procedure is proposed. In order to evaluate the quality of the proposed heuristics, a tight lower bound is developed. The optimal solution of the parallel processors scheduling problem with release date and delivery time is the basic used algorithm while developing the proposed procedures (heuristics and lower bound). In order to assess the performance and the efficiency of the proposed procedures, an extensive experimental study is carried out. During this experimental study the relative mean gap is not exceeding 0.7%, which provides strong evidence of the performance of the developed procedures.

INDEX TERMS Identical parallel processors, makespan, no-idle time, release date, delivery time, lower bound, heuristics.

I. INTRODUCTION

In scheduling theory, the idle time corresponds to the duration separating the completion of a task and the beginning of the next one in the same processor (machine). Generally, while studying scheduling problems, the idle processor time is assumed to be without cost. However, in several real life encountered problems such as in manufacturing and in parallel computing [27], this idle time is the source of high costs. Indeed, a running processor without processing any task is a waste of energy such as for a furnace. Even, stopping and restarting a running processor incurs high costs, indeed this strategy impacts the life cycle of the processors [31]. Therefore, an additional constraint which is the no-idle processor time should be considered for such applications. In this case, the schedules to be taken into account are only those with no-idle time. In addition, the main concern while handling problems related to the management of power, is the elimination of idle times, and schedules without idle times are required [28]. In this work, the parallel processors scheduling problem with no-idle time constraint is studied. In addition, the tasks to be processed are subject to release date and

The associate editor coordinating the review of this manuscript and approving it for publication was Kai Li.

delivery time restrictions. The release date might correspond to the arrival time of the task to the system, and the delivery time models for example the cooling time of a treated part. The objective to be minimized is the maximum completion time (or makespan).

The studied problem models several real life applications such as the parallel computing. The parallel computing is the usage of identical parallel processors (more than two processors) for processing several tasks at the same time [1], [5]. In parallel computing, small problems resulting from dividing large ones are processed simultaneously [2], [15]. The processing of small problems in parallel instead of treating the large problem using only one processor, allows shortening the consumed time while solving complex problems. However, the utilization of parallel computation centers is largely recognized as a high electrical energy consumer across the world [29], [30]. In this context, it has been shown throughout statistics studies that the percentage of released greenhouse gazes, due to the computing power consumption is 2%, and the increase is expected to reach 6% each year [18]. Therefore, reducing the consumed electrical energy is a crucial issue, and one of the proposed solutions is to adopt the no-idle processors time constraint while scheduling the tasks.

Parallel computing allows spectacular advances in several fields such as for optimization, medicine, aerospace engineering, civil engineering, management, biology, chemistry, mechanical engineering, high performance computing [11], [12], [16]. The key point in these advances is the simulation of large scale phenomena, which becomes possible thanks to high performance parallel computing. Balancing between the positif impacts of parallel computing and the consumed power triggers the emergence of the high performance green computing research field. This research field focuses in proposing new innovative solutions (hardware and algorithms) that reduce the parallel computing energy consumption.

Furthermore, the addressed scheduling problem models several industrial and manufacturing systems. These systems are characterized by a high energy consumption. Indeed, 50% of the total consumed energy in the world is intended to the industrial sector [32]. Moreover, the manufacturing sector for example in China consumes 81.32% of the total industrial energy [33]. Consequently, manufactures are forced to take some urgent actions in order to save energy. This can be performed by improving energy efficiency throughout the production schedule during the manufacturing process. Thus, energy efficient scheduling [34] attracts a lot of attention, and allows to save energy without extra cost invested in new equipments. This can be performed by selecting schedules with no-idle time.

The no-idle time constraint is encountered in several manufacturing systems such as ceramic industry, glassmaking, fiberglass processing, and integrated circuits. The no-idle time constraint is considered in different types of shops, for example in [35] the permutation flow shop with no-idle time is addressed. In addition, the hybrid flow shop scheduling problem with no-idle time is studied in [36], and the no-idle mixed shops is presented in [37]. In this work, the parallel processors with no-idle time is studied. Thus, the literature review will be restricted to the parallel machine and the single machine shops, both with no-idle time constraint.

The parallel processors scheduling problem and its variants attracted a lot of attention during the last years and an extensive literature was presented [4], [7], [17]. Authors in [5] provide a detailed literature review. Surprisingly, the parallel processors scheduling problem with no-idle time, release date and delivery time, is not studied in literature, to the best of our knowledge. Only the particular case with one processor, no-idle time, release date, and delivery time is addressed in few works, exactly in four papers. The author in the first paper [22] proposed several complexity results with no-idle time constraint. In addition, the author proofed that for some particular cases, certain algorithms designed originally to solve the problem with idle time, are also valid for solving the problem with no-idle time, after adjusting the release dates. Efficient heuristics are proposed in [14] for the single processor scheduling problem with no-idle time, release date and delivery time. In addition, a worst case study is proposed for all the developed heuristics. Authors in the third paper [3], proposed the adaptation of the well known Branch and Bound algorithm of Carlier [23], which is designed for the single machine without idle time constraint. This adaptation is based on some interesting results. The single processor scheduling problem with no-idle time and with release date (without delivery time) is examined in the fourth paper [13]) for several regular criterions. In the latter paper, the author proposed a constraint programming based algorithm to solve the studied problem. More recently, authors in [38] addressed the problem of identical parallel processors with homogeneously non-idling constraint, release date, due date, and unit-time job. In this work, a polynomial algorithm is proposed for solving the addressed problem.

The current examined problem is proofed to be NP-Hard in the strong sense. Indeed, this problem is a generalization of the well known parallel processor scheduling problem with release date and delivery time [4], [7], [8], [26]. In this work, the exact solution of the parallel scheduling problem with release date and delivery time will be used systematically in solving the parallel processor problem with no-idle time constraint. Using exact solution of a well studied problem (even for an NP-Hard) in solving more complex problems is encountered in literature and several examples are provided. Indeed, the Branch and Bound based exact solution of the one process scheduling problem with release date and delivery time [23], is embedded in several heuristics and exact solutions for solving more complex problems such as the non-permutation flow shop [40], job shop [39], and the parallel processors scheduling problems. In addition, the exact solution of the parallel processors problem with delivery time and release date is utilized in solving the two-stage hybrid flow shop scheduling problem [24], [25]. The exploration of the proposed literature for the parallel processors problem allows to determine the most efficient exact solution for the latter problem. Indeed, the branch and bound based exact solution presented in [7] is the most efficient one since it is able to solve large instance problems within a short CPU time. For this reason, this exact solution will be adopted within this work.

Within this research work, a family of heuristics are proposed. These heuristics are composed of two phases. The first one is intended to build an initial feasible solution, while the second one is an improvement phase. The two phases are developed using the provided Branch and Bound algorithm in [7]. In the first phase, this Branch and Bound is used to generate a feasible solution for the parallel processors scheduling problem with release date and delivery time. For this generated solution, each task in each processor is right shifted such that all the idle times are omitted. In the improvement phase different algorithms are used to solve iteratively a two processors scheduling problem. These two processors are the most and the least loaded ones. In order to assess the proposed heuristics, a new lower bound is developed.

The organisation of this paper is as follows: The addressed problem is introduced and defined in Section 2. A family of heuristics and a lower bound are presented in section 3. In section 4, an extensive experimental study is carried out and the performance of the proposed procedures is assessed. Finally, the summary of the performed work in this paper, and the future directions are presented in the conclusion.

II. PROBLEM DEFINITION

The parallel processors problem with no-idle time, release date, and delivery time is formally defined as follows. A set $M = \{M_1, M_2, \ldots, M_m\}$ of m identical parallel processors, has to process a set $J = \{1, 2, \ldots, n\}$ of n tasks ($n > m$). Each task $j \in J$ is ready to be processed from time r_j, this is the release date. Task $j \in J$ has to be processed in a processor during p_j units of time, this is the processing time. The duration separating the completing of processing of task $j \in J$ and the exiting of the system is q_j, this is the delivery time (it corresponds for example to a cooling period).

The processing of all tasks on the identical parallel processors is performed under the following assumptions:

- Processors are available for treating tasks from time 0.
- Preemption is not allowed during the processing of a task. In other term, the interruption of processing before finishing totally the task underway, is forbidden.
- A task is processed entirely by one processor (no splitting of tasks).
- At the same time, a processor treats at most one task.
- The release dates r_j, the processing times p_j, and the delivery times q_j are assumed to be deterministic and integral.

In addition, between the finishing and the starting of two consecutive tasks there is no idle time, this is the no-idle time constraint. A feasible schedule is an assignment of tasks to processors without violating the above mentioned assumptions. Let c_j be the finishing processing date of task j relatively to a feasible schedule σ, then $C_j = c_j + q_j$ denotes the completion time of task j. The purpose is to determine a feasible schedule that minimizes the maximum completion time (or makespan) $C_{max} = \max_{1 \le j \le n} (C_j)$. Based on Graham's notation [10], the studied problem is denoted $P_m, NI/r_j, q_j/C_{max}$. The no-idle constraint is indicated by NI (No Idle) notation in the processors field.

In the sequel, an example illustrating a feasible schedule for the studied problem, is presented.

Example 1: For this example: $n = 5$ and $m = 2$, release dates, processing times, and delivery times are displayed in Table 1.

TABLE 1. Data of example 1.

j	1	2	3	4	5
r_j	5	2	3	7	8
p_j	3	6	3	9	7
q_j	4	3	16	6	2

A feasible schedule, corresponding to the data presented in Example 1, is displayed in Figure 1. This feasible schedule has a makespan $C_{max} = 23$.

Proposition 1: The problem $P_m, NI/r_j, q_j/C_{max}$ is NP-Hard in the strong sense.

Proof: When relaxing the no-idle time constraint for the problem $P_m, NI/r_j, q_j/C_{max}$, then the obtained problem is $P_m/r_j, q_j/C_{max}$, which is NP-Hard in the strong sense [7], [9].

III. LOWER BOUND AND HEURISTICS

A. LOWER BOUND

This subsection is reserved to the development of a new lower bound for the addressed scheduling problem ($P_m, NI/r_j, q_j/C_{max}$). This lower bound as well as other procedures, are based on the optimal solution of the problem $P_m/r_j, q_j/C_{max}$. The lower bound is presented over the following lemma 1.

Lemma 1: Assume that C^*_{max} is the optimal value of an optimal schedule for $P_m/r_j, q_j/C_{max}$, then C^*_{max} is a lower bound for the problem $P_m, NI/r_j, q_j/C_{max}$.

Proof: Let C^{NI}_{max} be the optimal value of an optimal schedule σ_{NI} for the problem $P_m, NI/r_j, q_j/C_{max}$. The optimal schedule σ_{NI} (for $P_m, NI/r_j, q_j/C_{max}$) is also a feasible schedule for the problem $P_m/r_j, q_j/C_{max}$. Therefore, $C^*_{max} \le C^{NI}_{max}$. This means that C^*_{max} is a valid lower bound for the studied scheduling problem.

This lower bound is denoted LB, in other term $LB = C^*_{max}$.

Since the problem $P_m/r_j, q_j/C_{max}$ is NP-Hard, then it may happen that the optimal solution is not obtained using the exact procedure [7]. In this case the following remark (Remark 1) is useful.

Remark 1: If L is a lower bound for the problem $P_m/r_j, q_j/C_{max}$, then it is also a lower bound for the problem $P_m, NI/r_j, q_j/C_{max}$.

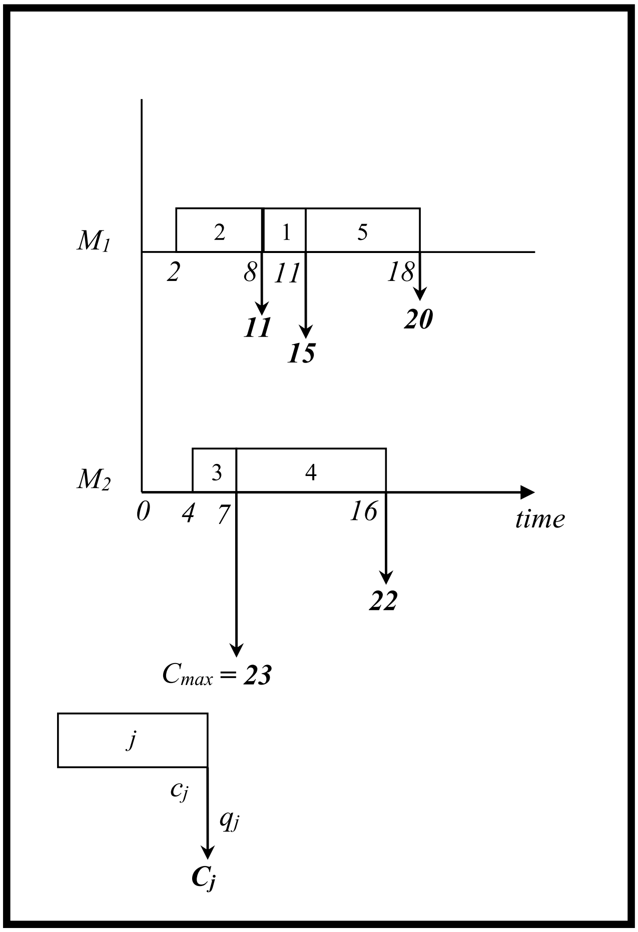

FIGURE 1. Gantt chart of a feasible schedule having a makespan equal to 23.

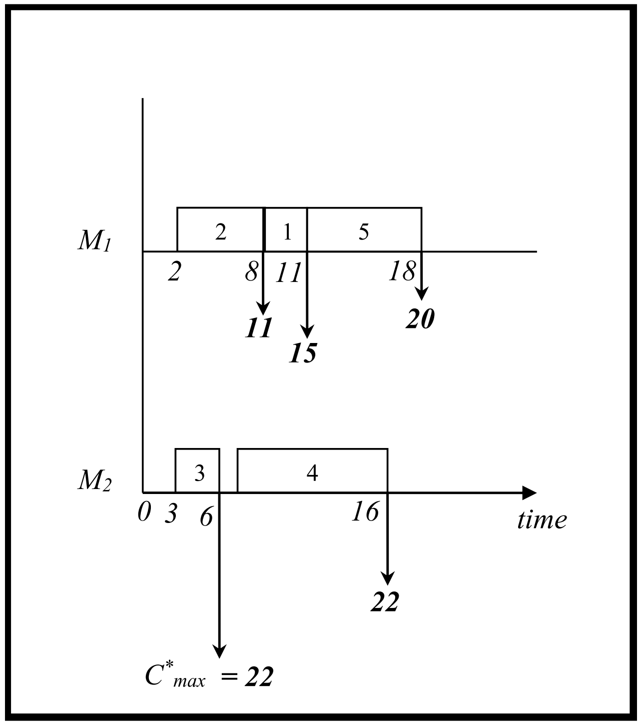

FIGURE 2. Gantt chart of an optimal schedule of $P_m/r_j, q_j/C_{max}$ having $C^*_{max} = LB = 22$.

Proof: Since L is a lower bound for the problem $P_m/r_j, q_j/C_{max}$ then $L \leq C^*_{max}$. According to the latter lemma $C^*_{max} \leq C^{NI}_{max}$. Thus, $L \leq C^{NI}_{max}$ and consequently L is a lower bound for the problem $P_m, NI/r_j, q_j/C_{max}$.

It is worth noting that in case where the exact procedure fails to solve the problem $P_m/r_j, q_j/C_{max}$ within a fixed time limit, then it returns the best obtained lower bound (the reader is referred to [7]).

For Example 1, the lower bound $LB = 22$, which is at the same time the optimal solution of the problem $P_m/r_j, q_j/C_{max}$ for the presented data. The corresponding schedule is displayed in Figure 2.

B. HEURISTICS

This section is dedicated to the development of a family of heuristics. These heuristics are composed of two consecutive phases. The first phase is intended to the development of an initial feasible schedule, while the second phase is an improvement one. During the two phases, the optimal solution of the problem $P_m/r_j, q_j/C_{max}$ as well as the well known Schrage's algorithm (will be introduced later) are used. The combination of the two latter procedures (exact solution and Schrage's algorithm) results into four heuristics that will be detailed in the sequel. It is worth noting that the exact procedure presented in [7] may fail solving optimally the problem $P_m/r_j, q_j/C_{max}$ within a time limit. In this case, the best reached feasible schedule is returned by the proposed procedure in [7].

1) HEURISTIC H_{EP-EP}

Phase 1: The first phase in the development of heuristic H_{EP-EP} is performed firstly by solving exactly the problem $P_m/r_j, q_j/C_{max}$ using the procedure presented in [7]. In the sequel, this procedure will be denoted *EP* (Exact Procedure). Let S be the optimal obtained schedule and C^*_{max} the optimal corresponding value. This schedule S satisfies only one of the following three conditions:

- Within the schedule S there is no idle time, in this case the obtained schedule S is also an optimal schedule for the problem $P_m, NI/r_j, q_j/C_{max}$ and the procedure is halted ($LB = C^*_{max}$).
- The schedule S presents idle times, in this case the first action to be taken is to right shift all the tasks in order to eliminate the idle times. The obtained schedule after right shifting the tasks is denoted S^R. If the schedules S and S^R have the same makespan's value C^*_{max}, then the optimal solution for the problem $P_m, NI/r_j, q_j/C_{max}$ is reached and the procedure is stopped ($LB = C^*_{max}$).
- The schedule S contains idle times and the right shifting results into the schedule S^R with a makespan satisfying:

TABLE 2. Data of example 2.

j	1	2	3	4	5	6	7	8	9	10
r_j	1	11	10	3	3	5	9	5	2	7
p_j	8	6	10	1	6	4	10	8	3	10
q_j	8	10	1	7	1	18	3	3	8	10

$C_{max} > C^*_{max}$. In this case, the obtained schedule is denoted S_1^R and the second phase is triggered.

Phase 2: The schedule S_1^R is the input of the second phase and some useful notations in the second phase are presented as follows.

- The set of the scheduled tasks on processor $M_k(k = 1, \ldots, m)$, relatively to schedule S_1^R is denoted J_k.
- The maximum completion time in processor $M_k(k = 1, \ldots, m)$, relatively to schedule S_1^R is denoted C_k with $C_k = max\{C_j, j \in J_k\}$, where C_j is the completion time of task j relatively to S_1^R.

In the sequel, the main procedure used in the second phase for each heuristic is presented. In this procedure, first and without loss of generality, the completion times in each processor are assumed to satisfy: $C_1 \leq C_2 \leq \ldots \leq C_m$. Phase 2 selects at the beginning the most and the least loaded processors (M_1, M_m) and the scheduled tasks on them $(J_1 \bigcup J_m)$. This allows to setup a two parallel processors scheduling problem $P_2/r_j, q_j/C_{max}$. This problem is solved using the procedure presented in [7] and a right shifting operation is performed whenever an idle time appears in the obtained schedule. The resulting schedule after the right shifting has a makespan C^1_{max} which satisfies: $C^1_{max} \leq C_m$. If $C^1_{max} < C_m$ then an improvement is detected and the maximum completion times in the processors are sorted in the increasing order. In the case where $C^1_{max} = C_m$, the current schedule on (M_1, M_m) is maintained.

Following the first step, an iterative procedure selecting at each iteration two processors (M_m, M_k), $k = 2, \ldots, m-1$ and the scheduled tasks $(J_m \bigcup J_k)$ results into a two parallel processors scheduling problem $P_2/r_j, q_j/C_{max}$ which is solved and right shifted. The obtained schedule's makespan is denoted C^k_{max}. For each iteration an update is performed if an improvement is detected: $C^k_{max} < C_m$. This procedure is repeated until no improvement is detected.

The current heuristic is denoted H_{EP-EP} mentioning that the optimal solution is used during the two phases. The obtained maximum completion time (upper bound) at the end of H_{EP-EP} is denoted UB_{EP-EP}.

To illustrate the two phases for heuristic H_{EP-EP}, the following example (Example 2) is presented.

Example 2: For this example: $n = 10$ and $m = 3$, release dates, processing times, and delivery times are presented in Table 2.

During the first phase (Phase 1), the problem $P_m/r_j, q_j/C_{max}$ is solved using the exact procedure in [7] and the obtained schedule is presented in Figure 3. This schedule has a maximum completion time $C^*_{max} = 28$.

The schedule displayed in Figure 3 presents three idle times distributed as follows.

- On processor M_2, the time interval [5, 7], separating the two consecutive tasks 9 and 10,
- On processor M_3, the time interval [4, 5], separating the two consecutive tasks 2 and 3,
- On processor M_3, the time interval [9, 11], separating the two consecutive tasks 3 and 1.

According to phase 1 procedure, a right shifting is performed in order to eliminate all the idle times, and the obtained schedule is presented in Figure 4. The maximum completion time (makespan) of this schedule is $C_{max} = 29$. Therefore, the condition three is satisfied and phase 2 is activated.

At the beginning of Phase 2, the subsets of tasks as well as the maximum completion times on each processor are identified and presented as follows.

- For processor M_1: $C_1 = 26$ and $J_1 = \{6, 9, 8\}$,
- For processor M_2: $C_2 = 28$ and $J_2 = \{10, 5, 7\}$
- For processor M_3: $C_3 = 29$ and $J_3 = \{2, 3, 1, 4\}$.

The iterative procedure in phase 2 starts by selecting the most and the least loaded processors as well as the scheduled jobs in these two processors. In our case, the requested processors are M_1 and M_3, and the related subset of tasks is $J_1 \bigcup J_3 = \{1, 2, 3, 4, 6, 9, 8\}$. The resulted two processors scheduling problem $P_2/r_j, q_j/C_{max}$ is solved using *EP* (the exact procedure provided in [7]). The obtained schedule is depicted in Figure 5.

An improvement is detected, and the new distribution of tasks as well as the new maximum completion times for the processors are presented as follows.

- For processor M_1: $C_1 = 28$ and $J_1 = \{6, 1, 4\}$,
- For processor M_2: $C_2 = 28$ and $J_2 = \{10, 5, 7\}$
- For processor M_3: $C_3 = 27$ and $J_3 = \{2, 3, 1, 4\}$.

Recall that for the considered data, and according to phase 1, the problem $P_m, NI/r_j, q_j/C_{max}$ has a lower bound $LB = 28$. Since, the feasible schedule presented in Figure 5, has a maximum completion time $UB_{EP-EP} = 28 = LB$, then this schedule is an optimal one for $P_m, NI/r_j, q_j/C_{max}$ and the whole procedure is halted.

The heuristic H_{EP-EP} is totally based on an exact procedure solving the $P_m/r_j, q_j/C_{max}$. The latter problem is NP-Hard and the exact procedure is a time consuming one for certain data. Thus, including other simple heuristics, returning a near optimal solution for $P_m/r_j, q_j/C_{max}$ within a short time, are required to have an accurate assessment. In this context, the Schrage's heuristic, which is a dispatching rule, is adopted. This heuristic is selected due to its time complexity which is in $O(nlogn)$ time. Combinations of Schrage's algorithm with the exact procedure are performed, for example Schrage in phase 1 and exact solution in phase 2, and three other heuristics are developed. These heuristics have the same logic as for H_{EP-EP}, in terms of phases and the content of these phases. More details for these heuristics are presented in the sequel.

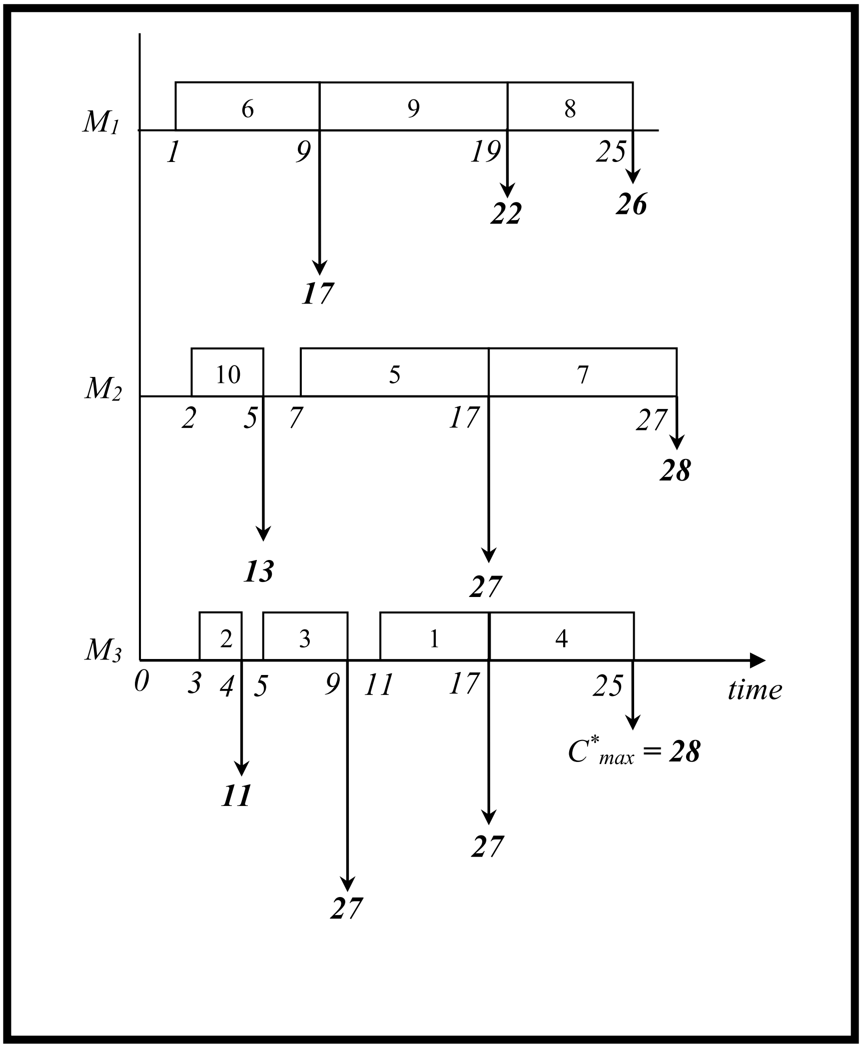

FIGURE 3. Gantt chart of an optimal schedule of $P_m/r_j, q_j/C_{max}$ for example 2.

2) HEURISTIC H_{MS-MS}

The Schrage's algorithm is an iterative procedure, intended to provide a near optimal solution for the problem $P_m/r_j, q_j/C_{max}$. At each iteration, the task with the largest delivery time (q_j) is scheduled in the first available processor. Therefore, the main effort for this algorithm is sorting the delivery time (q_j) in the decreasing order. Thus the Schrage's algorithm time complexity is in $O(nlogn)$ time.

The Schrage's algorithm is illustrated over the following example (Example 3).

Example 3: The number of processors and tasks are respectively $n = 5$ and $m = 2$. The release dates, processing times, and delivery times are presented in Table 3.

TABLE 3. Data of example 3.

j	1	2	3	4	5
r_j	1	2	6	4	5
p_j	2	3	2	3	2
q_j	15	11	10	8	7

Applying Schrage's algorithm yields the schedule displayed in Figure 6. The maximum completion time of this feasible schedule is $C_{max} = 18$.

Since the no-idle time is a mandatory constraint for the studied problem $P_m, NI/r_j, q_j/C_{max}$, then a modified version of Schrage's algorithm is proposed in this section. This Modified version of Shcrage's algorithm (MS) consists on scheduling among the unscheduled tasks the one with the largest delivery time (q_j), on one of the first available processors. The selection of the processor is performed according to the following procedure.

In iteration $i(i = 1, \ldots, n)$, consider:

1) Task i with the i^th largest delivery time among the unscheduled tasks (without loss of generality).
2) MA_i the set of available processors for treating task i.
3) s_i^k is the earliest starting time of task i on processor $M_k \in MA_i$.

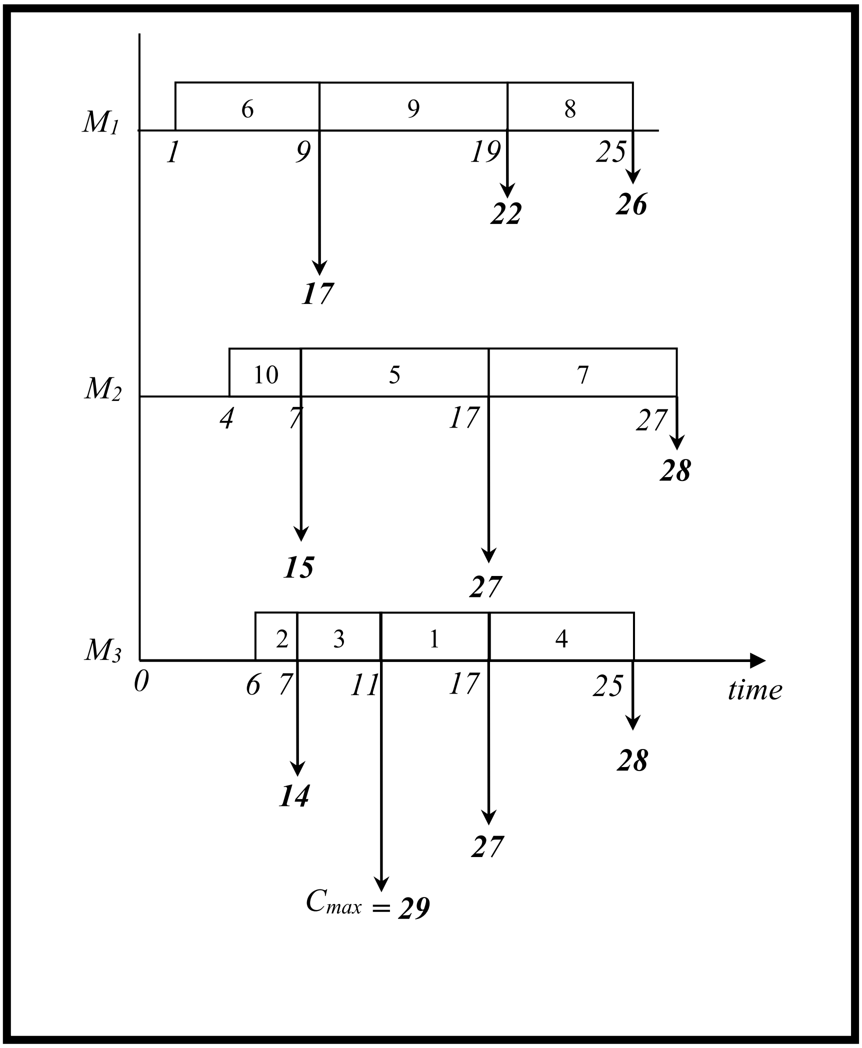

FIGURE 4. Feasible schedule obtained after the Right shifting procedure.

4) The completion time of i in processor M_k is $C_i^k = s_i^k + p_i + q_i$.
5) if j_k is the last scheduled task on M_k, then $a_k = s_i^k + q_{j_k}$. The new completion time of j_k after eliminating idle time (if it exists) is a_k.
6) the selected processor is the one with smallest a_k.

Indeed, $s_i + q_k$ represents the completion time of task j_k right shifted until the stating time of task i. Therefore, MS aims to minimize the increasing of the completion time when right shifting the tasks with idle time.

For Example 3, the iterations of Modified Shrage's algorithm MS are as follows.

1) iteration 1: Task 1 is the candidate and it is scheduled on M_1, with starting time $s_1 = 1$ and completion time $C_1 = 18$.
2) iteration 2: Task 2 is the candidate and it is scheduled on M_2, with starting time $s_2 = 2$ and completion time $C_2 = 16$.
3) iteration 3: Task 3 is the candidate and it can be scheduled in either M_1 or M_2. The two processors have one scheduled task. The earliest starting time for task 3 is 6. Comparison $s_3 + q_1 = 6 + 15 = 21$ and $s_3 + q_2 = 6 + 11 = 17$ yields an advantage for processor M_2 and task 3 is scheduled on M_2 instead of M_1 as for the previous feasible schedule.
4) iteration 4: Task 4 is the candidate and it is scheduled on M_1, with starting time $s_4 = 4$ and completion time $C_2 = 15$.
5) iteration 5: Task 5 is the candidate and it is scheduled on M_1, with starting time $s_4 = 7$ and completion time $C_2 = 17$.

The obtained schedule is presented in Figure 7.

Observing that the two last feasible schedules(depicted in Figure 6 and Figure 7) have the same maximum completion time $C_{max} = 18$. However, when the right shifting procedure is applied for both of them, then for the first one (Schrage's algorithm): $C_{max} = 21$ and for the second

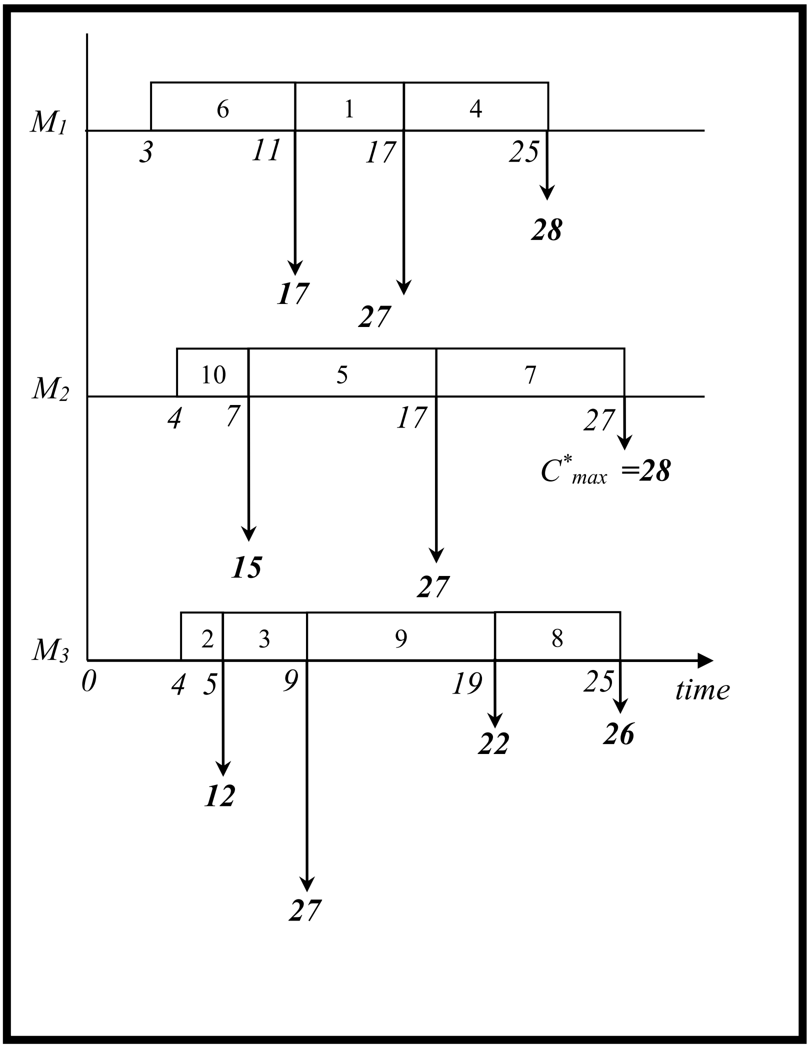

FIGURE 5. Gantt chart of a feasible schedule having $LB = C^*_{max} = 28$.

one (Modified Schrage's algorithm): $C_{max} = 19$. This is the main reason for developing the Modified Schrage's algorithm (*MS*).

The Modified Schrage's algorithm (*MS*) is used in the development of the second heuristic in the same way as for the heuristic H_{EP-EP}. In other terms, in Phase 1 and Phase 2 the *MS* procedure is used instead of the exact procedure *EP*. The resulting heuristic and the corresponding maximum completion time are denoted respectively, H_{MS-MS} and UB_{MS-MS}.

3) HEURISTICS H_{MS-EP} AND H_{EP-MS}

The combination of the Modified Schrage's algorithm (*MS*) and the exact procedures *EP* yields two other variants which are presented below.

1) (*MS*) used in phase 1 and exact procedure used in Phase 2 results into the heuristic H_{MS-EP} and the corresponding maximum completion time is denoted UB_{MS-EP}.
2) The usage of exact procedure *EP* used in phase 1 and (*MS*) used in Phase 2, products the heuristic H_{EP-MS} with maximum completion time denoted UB_{EP-MS}.

IV. COMPUTATIONAL EXPERIMENTS

A. TEST PROBLEMS

The performances of the four proposed heuristics H_{EP-EP}, H_{MS-MS}, H_{EP-MS}, H_{MS-EP}, and the lower bound *LB* are assessed over an extensive experimental study. This experimental study is carried out using test problems as introduced in [4] and in [8]. Three classes of instances are generated and denoted respectively: Class A, Class B and Class C. It is worth noting that the combination of several different problem sizes (n and m), processing times, delivery times, and release date distributions, yields a highly diversified test problems. In so doing, we propose a method for an unbiased experimental analysis of the performance and efficiency of the proposed procedures (heuristics and lower bound).

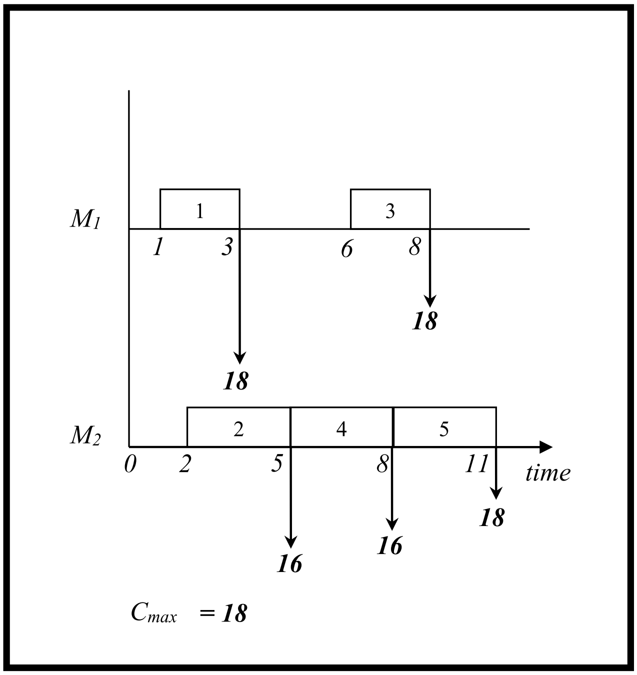

FIGURE 6. Feasible schedule produced by Schrage's algorithm.

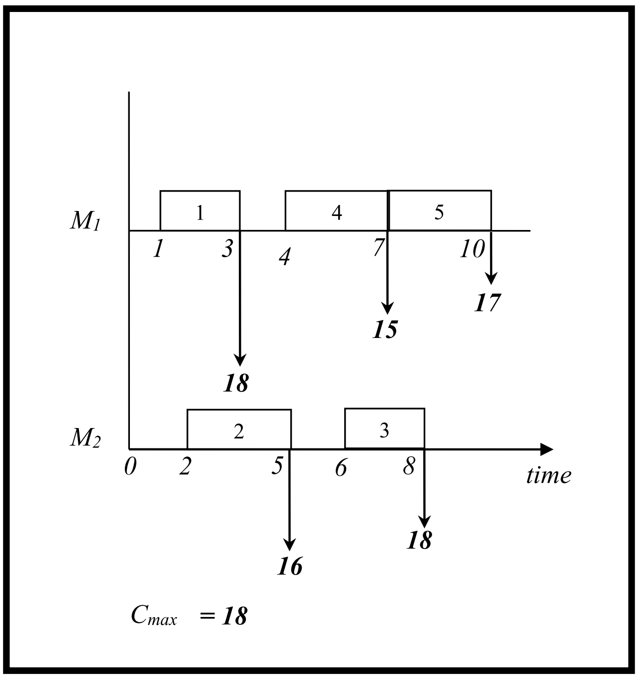

FIGURE 7. Enhanced feasible schedule produced by Modified Schrage's algorithm.

1) CLASS A

For Class A the number of tasks n and processors m are generated as follows.

- $n \in \{10, 20, 40, 50, 200\}$
- $m \in \{2, 3, 5, 8\}$

The release dates r_i, processing times p_i, and delivery times q_i are generated as follows.

- p_i uniformly generated in $[1, p_{max}]$, with $p_{max} = 10$.
- r_i uniformly generated in $[1, r_{max}]$,
- q_i uniformly generated in $[1, q_{max}]$, where r_{max} and q_{max} depend on n, m, and a parameter K as: $r_{max} = q_{max} = \left\lceil \frac{nK}{m} \right\rceil$,
- with $K \in \{1, 3, 5, 7, 10, 13, 17, 22, 27, 33\}$

For each combination of n, m, and K several instances are generated (as in [4]) and 2000 instances are obtained for class A.

2) CLASS B

The generation of instances for Class B is similar to class A except for p_i, r_i, which are generated as follows ([8]).

- p_i uniformly generated in $[1, n]$.
- r_i uniformly generated in $[1, n]$,

The number of generated instance for Class B is 2000 instances by considering different combinations.

3) CLASS C

The test problems in Class C are generated as follows.

- $n \in \{10, 20, 40, 50, 200\}$
- $m \in \{2, 3, 5, 8\}$
- p_i uniformly generated in $[1, 50]$.

The generation of release date r_i and delivery time q_i are uniformly generated. This generation is performed according to three following sub-classes:

- Small-Large(SL):$r_i \in [1, 20]$ and $q_i \in [1, 50]$.
- Medium-Medium(MM): $r_i \in [1, 50]$ and $q_i \in [1, 50]$.
- Large-Small(LS): $r_i \in [1, 50]$ and $q_i \in [1, 20]$.

Each subclass (SL, MM, LL) contains 2000 instances which results into 6000 instances for Class C.

The lower bound LB as well as the heuristics H_{EP-EP}, H_{MS-MS}, H_{EP-MS}, H_{MS-EP}, are coded in C language over a quad-core (1.8 GHz) Personal Computer with 16 GB RAM. The results are assessed throughout the following performance measures (metrics).

- TLB: required average time to compute LB.
- $RG = 100(UB - LB)/LB$: the relative gap.
- Gap: the average relative gap.
- $Time$: required average time for the heuristics.
- NIt: The Phase 2 average number of iterations.

The relative gap RG is measuring the maximum relative deviation of the studied heuristic's value UB relatively to the optimal solution (which is not available). Indeed, if C^*_{max} is the optimal value, then $C^*_{max} \geq LB$ and $RG = 100(UB - LB)/LB \geq 100(UB - C^*_{max})/LB$. The more RG is close to 0, the more the heuristic is efficient.

The obtained results (detailed and average) are presented as follows.

- For Class A: in Tables 4, 5,6.
- For Class B: in Tables 7, 8,9.
- For Class C: seeking clarity, for this class only the average results are presented in Table 10. The detailed results are displayed in the Appendix V, as below.

TABLE 4. Class A: detailed results for H_{EP-EP} and H_{MS-MS}.

			H_{EP-EP}			H_{MS-MS}		
m	n	TLB	$Time$	Gap	NIt	$Time$	Gap	NIt
2	10	0	0.01	0.63	0	0.01	5.82	1.08
2	20	0	0.01	0.92	0	0.02	7.45	1.08
2	40	0.01	0.03	0.2	0	0.02	8.88	1.12
2	50	0.02	0.04	0.49	0	0.02	9.07	1.09
2	200	1.47	1.6	0.34	0	0.17	11.06	1.16
3	10	0	0.02	2.22	0.35	0.02	4.14	1.7
3	20	0	0.02	0.89	0.36	0.03	5.94	1.78
3	40	0.01	0.04	0.41	0.44	0.04	5.89	1.71
3	50	0.01	0.06	0.57	0.56	0.04	6.8	1.51
3	200	1.25	1.51	0.93	0.85	0.22	8.59	2.08
5	10	0	0.02	2.07	0.56	0.01	1.69	1.96
5	20	0	0.04	1.92	1.03	0.03	2.84	2.14
5	40	0	0.05	0.55	0.68	0.06	3.6	3.25
5	50	0.01	0.07	0.87	0.95	0.05	4.37	2.67
5	200	0.81	1.15	1.09	1.39	0.29	5.43	2.91
8	10	0	0.01	0	0	0.02	0.3	4.55
8	20	0.01	0.06	1.83	1.56	0.05	1.73	3.32
8	40	0.11	0.17	0.67	1.54	0.07	1.9	3.99
8	50	0	0.09	1.2	2.17	0.09	1.74	4.17
8	200	0.44	0.98	1.52	3.4	0.43	2.61	4.92

TABLE 5. Class A: detailed results for H_{EP-MS} and H_{MS-EP}.

			H_{EP-MS}			H_{MS-EP}		
m	n	TLB	$Time$	Gap	NIt	$Time$	Gap	NIt
2	10	0	0.01	0.07	0.4	0.02	0.6	0.99
2	20	0	0.01	0.42	1	0.02	1.44	1
2	40	0.01	0.02	0.04	1	0.03	0.89	1
2	50	0.02	0.04	0.08	1	0.06	0.76	1
2	200	1.47	2.26	0.13	1	3.28	1.51	1.03
3	10	0	0.01	0.29	1.22	0.01	1.02	1.38
3	20	0	0.02	0.22	1.23	0.03	2.13	2.12
3	40	0.01	0.03	0.07	1.45	0.05	1.93	1.79
3	50	0.01	0.05	0.06	1.13	0.06	1.85	2.21
3	200	1.25	1.76	0.28	1.39	1.23	3.69	2.18
5	10	0	0.01	0	1.28	0.02	0.82	2.95
5	20	0	0.03	0	2.03	0.04	0.8	2.73
5	40	0	0.02	0.17	2.17	0.07	1.24	2.41
5	50	0.01	0.05	0.2	4.65	0.06	2.21	2.43
5	200	0.81	1.26	0.43	2.74	0.54	2.95	2.76
8	10	0	0.01	0	1	0.02	0	1
8	20	0.01	0.04	0.01	1.57	0.05	0.69	2.38
8	40	0.11	0.19	0.02	1.81	0.09	0.81	3.28
8	50	0	0.08	0.08	3.96	0.11	0.67	4.02
8	200	0.44	1.2	0.47	9.53	0.57	1.68	4.82

TABLE 6. Global results for Class A.

$Heuristic$	$Time$	Gap	NIt
H_{EP-EP}	**0.3855**	**0.7385**	**0.705**
H_{MS-MS}	**0.0875**	**5.203**	**2.25**
H_{EP-MS}	**0.3785**	**0.1775**	**2.525**
H_{MS-EP}	**0.4075**	**0.823**	**2.68**

- For subclass *SL*: in Tables 11,12.
- For subclass *MM*: in Tables 13,14.
- For subclass *LS*: in Tables 15,16.

B. NUMERICAL RESULTS

For class A and according to Tables 4-6, the average consumed time *TLB* while computing *LB* is not exceeding 1.47s, which is a very short time. Recall that *LB* is computed using an exact procedure *EP* ([7]). Despite using exact procedures to generate *LB*, the consumed time is short and acceptable. This is an additional justification for the usage of exact procedures to solve more complex problems. In addition, for each number of processor m, the *TLB* average time is increasing as the number of tasks n increases, and the maximum is reached for $n = 200$. For each number of tasks n, *TLB* is almost insensitive to the variation of the number of processors m (for $n = 40$: *TLB* varies from 0s to 0.11s). Remarkably, the maximum *TLB* is reached for the smallest number of processors: $m = 2$. This is due to the relative weakness of the *EP* while treating small number of processors.

TABLE 7. Class B: detailed results for H_{EP-EP} and H_{MS-MS}.

			H_{EP-EP}			H_{MS-MS}		
m	n	TLB	$Time$	Gap	NIt	$Time$	Gap	NIt
2	10	0	0.01	0	0	0.02	16.69	1.01
2	20	0.05	0.05	0	0	0.03	12.26	1.21
2	40	0.05	0.05	0	0	0.02	7.23	1.16
2	50	0.16	0.17	0	0	0.03	5.84	1.26
2	200	16.62	16.78	0	0	0.35	1.13	1.39
3	10	0	0.01	0	0	0.02	9.39	1.96
3	20	0.01	0.01	0	0	0.05	17.7	1.96
3	40	0.13	0.13	0	0	0.05	10.99	2.03
3	50	0.13	0.13	0	0	0.08	8.82	2.18
3	200	76.05	99.53	0.01	0.89	0.75	1.94	4.99
5	10	0	0.01	0	0	0.02	0.82	2.69
5	20	0.01	0.02	0	0	0.09	18.82	2.54
5	40	25.39	25.44	0.08	0.64	0.21	15.63	3.05
5	50	12.39	12.42	0	0.24	0.22	13.77	6.32
5	200	86.95	91.85	0.01	1.74	0.9	3.9	5.87
8	10	0	0.01	0	0	0.01	0	1.24
8	20	0	0.01	0	0	0.05	0.71	2.75
8	40	72.51	72.68	1.08	3.73	0.43	24.99	5.04
8	50	55.33	55.41	0.28	2.87	0.36	20.89	7.28
8	200	100	142.64	0.06	5.1	1.41	6.4	9.43

Based on Tables 4-5, the average time *Time* while running the four heuristics is increasing as the number of tasks n increases. The maximum $Time = 3.28s$ is obtained for $n = 200$, $m = 2$, and H_{MS-EP}. As remarked previously, when using *EP* the largest consumed time *Time* is reached for $m = 2$. For each one of the proposed heuristics, the distribution of *Time* is as follows.

- For H_{EP-EP}, $Time \in [0.01, 1.6]$ and the average is 0.3855s.
- For H_{MS-MS}, $Time \in [0.01, 0.43]$ and the average is 0.0875s.
- For H_{EP-MS}, $Time \in [0.01, 2.26]$ and the average is 0.3785s.
- For H_{MS-EP}, $Time \in [0.01, 3.28]$ and the average is 0.4075s.

Which indicates that the heuristics using partially (in one phase) or totally (in both phases) the *EP* procedure, are the most time consuming heuristics. Although *EP* is and exact procedure, the consumed time while running the heuristics still short.

According to Table 6, the respective averages *Time*, for the heuristics H_{EP-EP}, H_{MS-MS}, H_{EP-MS}, and H_{MS-EP} are 0.3855s, 0.0875s, 0.3785s, and 0.4075s. These times are short and are not exceeding 0.4075s despite the usage of

TABLE 8. Class B: detailed results for H_{EP-MS} and H_{MS-EP}.

			H_{EP-MS}			H_{MS-EP}		
m	n	TLB	$Time$	Gap	NIt	$Time$	Gap	NIt
2	10	0	0.01	0	0	0.02	2.4	1
2	20	0.05	0.05	0	0	0.03	0.83	1
2	40	0.05	0.05	0	0	0.11	0	1
2	50	0.16	0.17	0	0	0.09	0.9	1
2	200	16.62	20.13	0	0	19.43	0	1
3	10	0	0.01	0	0	0.02	2.99	1.44
3	20	0.01	0.01	0	0	0.05	5.83	2.72
3	40	0.13	0.13	0	0	0.19	0.67	3.73
3	50	0.13	0.13	0	0	0.59	1.13	3.55
3	200	76.05	76.37	0.01	2	49.31	0.46	3.54
5	10	0	0.01	0	2	0.02	0	1.59
5	20	0.01	0.02	0	2	0.08	7.54	3.67
5	40	25.39	25.42	0.14	3.94	0.76	3.16	6.66
5	50	12.39	12.43	0.05	4	0.41	1.68	7.84
5	200	86.95	87.51	0.04	4	67.88	0.45	8.44
8	10	0	0.01	0	4	0.01	0	1.32
8	20	0	0.01	0	4	0.05	0	2.82
8	40	72.51	72.77	1.16	6.94	0.47	6.96	10.34
8	50	55.33	55.47	0.28	7	0.37	7.46	8.18
8	200	100	101.17	0.19	7	125.34	0.58	18.65

TABLE 9. Global results for Class B.

$Heuristic$	$Time$	Gap	NIt
H_{EP-EP}	**25.868**	**0.076**	**0.7605**
H_{MS-MS}	**0.255**	**9.896**	**3.268**
H_{EP-MS}	**22.594**	**0.0935**	**2.344**
H_{MS-EP}	**13.2615**	**2.152**	**4.4745**

the exact procedure *EP* for three of the proposed heuristics. The least average time *Time* (0.0875*s*) is reached as expected for H_{MS-MS}, since the *MS* is a polynomial procedure (time complexity is $Onln(n)$). The maximum average time which is reached for H_{MS-EP} is 0.4075*s*. This is signify that the initial provided schedule from phase 1, when using the *MS* is far from the optimal solution and the *EP* has to provide more effort to enhance the latter one.

Based on Tables 4-5, the average relative gap *Gap* is ranging:

- For H_{EP-EP}, from 0 to 2.22,
- For H_{MS-MS}, from 0.3 to 11.06,
- For H_{EP-MS}, from 0 to 0.47,
- For H_{MS-EP}, from 0 to 3.69.

The heuristic H_{EP-MS} is presenting the minimum *Gap* while H_{MS-MS} is providing the maximum one. The two remaining ones (H_{EP-EP} and H_{MS-EP}) are quite similar in term of *Gap*. Thus, when the *EP* is involved then relative gap *Gap* becomes small. In addition, the average gap *Gap* for each class (n, m) is significantly more important for the heuristics where the first phase is performed via *MS* procedure.

According to Table 6, the minimum average relative gap $Gap = 0.1775$ is reached for H_{EP-MS}, while the maximum $Gap = 5.203$ is obtained when using H_{MS-MS}, which is far from the other heuristics ($5.203 \gg 0.823 > 0.7385 > 0.1775$). The heuristic H_{EP-MS} is outperforming the two other ones (H_{EP-EP} and H_{MS-EP}) in terms of average time *Time* and average gap *Gap*. Therefore, the obtained relative gap *Gap* is very small for the three heuristics using the *EP* procedure. This is a proof of the efficiency of the proposed heuristics (except H_{MS-MS}) and the proposed lower bound *LB*, since *Gap* is involving both the heuristics and the lower bound.

To get more insight on phase 2, the average number of iteration *NIt* during phase 2 is presented. Based on Table 6, *NIt* is almost 3 iterations for all the heuristics except H_{EP-EP}, where $NIt \approx 1$. Thus, phase 2 is increasing the performance of the proposed procedures.

TABLE 10. Average results for Class C.

	H_{EP-EP}		
	$Time$	Gap	$Iter$
SL	13.427	0.057	0.426
MM	14.439	0.092	0.518
LS	11.1935	0.057	0.415
Average	**13.02**	**0.07**	**0.45**
	H_{MS-MS}		
	$Time$	Gap	$Iter$
SL	0.207	4.3335	2.8375
MM	0.2155	9.2965	2.9235
LS	0.2195	15.267	3.048
Average	**0.21**	**9.63**	**2.94**
	H_{EP-MS}		
	$Time$	Gap	$Iter$
SL	13.4465	0.0655	2.3145
MM	14.458	0.0945	2.576
LS	11.155	0.059	2.7755
Average	**13.02**	**0.07**	**2.56**
	H_{MS-EP}		
	$Time$	Gap	$Iter$
SL	3.4665	0.72	3.791
MM	3.7805	1.451	4.5835
LS	2.0485	2.409	4.49
Average	**3.10**	**1.53**	**4.29**

TABLE 11. Class C, subclass *SL*: detailed results for H_{EP-EP} and H_{MS-MS}.

			H_{EP-EP}			H_{MS-MS}		
m	n	TLB	$Time$	Gap	NIt	$Time$	Gap	NIt
2	10	0	0.01	0	0	0.01	6.77	1
2	20	0.03	0.03	0	0	0.02	5.99	1.01
2	40	0.1	0.1	0	0	0.02	3.02	1.18
2	50	0.95	0.95	0	0	0.03	2.47	1.25
2	200	12.57	12.72	0	0	0.3	0.54	1.41
3	10	0	0.01	0	0	0.02	2.78	1.62
3	20	3.75	3.75	0.01	0.02	0.06	8.86	1.86
3	40	1.42	1.43	0	0	0.06	5.17	2.34
3	50	1.71	1.71	0	0	0.06	4.21	2.15
3	200	15.78	15.93	0	0	0.49	0.99	2.62
5	10	0	0.01	0	0	0.01	0.07	2.72
5	20	0.04	0.05	0	0	0.07	3.76	3.38
5	40	11.14	11.15	0.04	0.2	0.2	8.46	4
5	50	24.59	24.84	0.06	0.62	0.19	7.27	3.98
5	200	33.22	33.54	0.01	0.43	0.75	1.88	4.03
8	10	0	0.01	0	0.01	0.01	0	1.88
8	20	0	0.01	0	0	0.03	0.16	2.79
8	40	35.33	35.4	0.43	1.69	0.36	9.37	5.01
8	50	60.7	60.81	0.53	2.85	0.37	11.49	5.93
8	200	65.61	66.08	0.06	2.7	1.08	3.41	6.59

TABLE 12. Class C, subclass *SL*: detailed results for H_{EP-MS} and H_{MS-EP}.

			H_{EP-MS}			H_{MS-EP}		
m	n	TLB	$Time$	Gap	NIt	$Time$	Gap	NIt
2	10	0	0.01	0	0	0.02	0.31	1
2	20	0.03	0.05	0	0	0.07	0.04	1
2	40	0.1	0.1	0	0	0.24	0	1
2	50	0.95	0.95	0	0	1.11	0.04	1
2	200	12.57	12.86	0	0	21.19	0	1
3	10	0	0.01	0	0	0.02	0.29	2.14
3	20	3.75	3.75	0.01	0.9	0.37	1.41	2.7
3	40	1.42	1.42	0	2	5.99	0.68	3.31
3	50	1.71	1.7	0	2	0.57	0.43	3.23
3	200	15.78	16.16	0	2	12	0.1	3.01
5	10	0	0.01	0	2	0.01	0	1.3
5	20	0.04	0.05	0	2	0.08	0.89	3.51
5	40	11.14	11.19	0.04	3.82	5.21	1.63	6.59
5	50	24.59	24.68	0.07	4	11.55	1.2	6.46
5	200	33.22	33.52	0.01	4	6.15	0.16	6.96
8	10	0	0.01	0	1.63	0.01	0	2.1
8	20	0	0.01	0	1	0.03	0	1.8
8	40	35.33	35.45	0.45	6.94	0.4	3.47	7.84
8	50	60.7	60.91	0.64	7	1.68	3.35	9.49
8	200	65.61	66.09	0.09	7	2.63	0.4	10.38

TABLE 13. Class C, subclass *MM*: detailed results for H_{EP-EP} and H_{MS-MS}.

			H_{EP-EP}			H_{MS-MS}		
m	n	TLB	$Time$	Gap	NIt	$Time$	Gap	NIt
2	10	0.01	0.01	0	0	0.01	15.79	1
2	20	1.08	1.08	0.03	0	0.03	12.08	1.07
2	40	0.72	0.72	0	0	0.02	6.25	1.32
2	50	0.18	0.18	0	0	0.03	4.57	1.51
2	200	14.44	14.55	0	0	0.29	1.14	1.52
3	10	0	0.01	0.01	0.02	0.03	11.55	1.78
3	20	0.45	0.45	0	0	0.05	16.52	1.93
3	40	1.3	1.3	0	0	0.05	9.56	2.47
3	50	3.36	3.37	0	0.03	0.08	7.24	2.3
3	200	21.42	21.57	0	0	0.44	1.85	2.91
5	10	0	0.01	0.05	0.04	0.03	1.22	2.06
5	20	1.79	1.81	0.05	0.04	0.11	17	3.12
5	40	18.32	18.34	0.06	0.36	0.21	14.97	3.82
5	50	15.3	15.33	0.03	0.32	0.19	12.81	3.39
5	200	39.23	39.86	0.02	0.75	0.68	3.27	4.11
8	10	0	0.01	0.06	0.07	0.01	0	1.17
8	20	0	0.01	0	0	0.11	2.19	4.19
8	40	51.09	51.18	0.75	2.69	0.47	22.34	6.12
8	50	63.34	63.49	0.72	3.39	0.38	19.93	6.13
8	200	55.05	55.5	0.06	2.65	1.09	5.65	6.55

TABLE 14. Class C, subclass *MM*: detailed results for H_{EP-MS} and H_{MS-EP}.

			H_{EP-MS}			H_{MS-EP}		
m	n	TLB	$Time$	Gap	NIt	$Time$	Gap	NIt
2	10	0.01	0.01	0	0	0.03	0.31	1
2	20	1.08	1.09	0.03	0.31	2.11	0.49	1
2	40	0.72	0.72	0	1	1.41	0.24	1
2	50	0.18	0.18	0	1	0.37	0.05	1
2	200	14.44	14.75	0	1	31.37	0.02	1
3	10	0	0.01	0	1	0.04	0.99	2.14
3	20	0.45	0.45	0	1	0.22	2.35	3.24
3	40	1.3	1.33	0	1	0.36	2.19	3.26
3	50	3.36	3.34	0	1.58	4.13	1.37	3.5
3	200	21.42	21.83	0	2	17.04	0.3	3.5
5	10	0	0.01	0	1.02	0.03	0.07	2.49
5	20	1.79	1.81	0.05	3.46	0.13	4.01	5.53
5	40	18.32	18.39	0.08	4	4.33	2.91	6.83
5	50	15.3	15.32	0.04	4	1.99	1.74	7.47
5	200	39.23	39.57	0.04	3.99	6.94	0.39	7.6
8	10	0	0.01	0	3.34	0.01	0	1.77
8	20	0	0.01	0	1	0.12	0.26	3.66
8	40	51.09	51.3	0.76	6.82	0.54	5.86	11.9
8	50	63.34	63.53	0.8	7	1.64	4.61	11.98
8	200	55.05	55.5	0.09	7	2.8	0.86	11.8

TABLE 15. Class C, subclass *LS*: detailed results for H_{EP-EP} and H_{MS-MS}.

			H_{EP-EP}			H_{MS-MS}		
m	n	TLB	$Time$	Gap	NIt	$Time$	Gap	NIt
2	10	0	0.01	0	0	0.02	25.79	1.05
2	20	0.1	0.1	0	0	0.03	19.71	1.19
2	40	0.11	0.11	0	0	0.02	11.11	1.44
2	50	0.1	0.1	0	0	0.03	9.82	1.5
2	200	8.83	8.95	0	0	0.27	1.93	1.48
3	10	0	0.01	0.04	0.04	0.03	15.45	1.73
3	20	1.54	1.54	0	0	0.06	29.01	2.16
3	40	1.08	1.08	0	0	0.06	17.06	2.34
3	50	0.82	0.82	0	0	0.08	13.96	2.38
3	200	12.17	12.31	0	0.01	0.46	3.47	2.56
5	10	0	0.01	0	0	0.04	1.27	2.14
5	20	0.26	0.28	0	0	0.12	20.68	3.41
5	40	16.21	16.24	0.05	0.34	0.24	27.42	4.12
5	50	20.81	21.06	0.05	0.45	0.21	23.45	4.67
5	200	23.64	23.9	0.01	0.37	0.68	6.38	4.51
8	10	0	0.01	0.05	0.07	0.01	0	1.21
8	20	0	0.01	0.08	0.07	0.08	3.75	2.98
8	40	30.57	30.64	0.39	1.7	0.44	31.62	6.21
8	50	51.48	51.57	0.42	2.83	0.37	33.34	6.37
8	200	53.61	55.12	0.05	2.42	1.14	10.12	7.51

For class B and based to Tables 7-8, one can observe easily that the consumed time TLB running LB is much higher than class A. Indeed, TLB reaches a maximum of $100s$ for $n = 200$ and $m = 8$. This is explained by the hardness of test problems of the Class B to be handled by the EP procedure. Compared to Class A, the average time TLB is presenting the same behavior for class B. Indeed, TLB increases as n increases for each fixed m. Contrary to Class A, for each n, an increasing of TLB is observed when m increases. This increasing is more important for large value of n.

Furthermore, the average time $Time$ is becoming much important compared to test problems of Class A, and $Time$ reaches a maximum of $142.64s$ for $n = 200$ and $m = 8$. Average time $Time$ is increasing when the number of tasks n increases for class B. In addition, $Time$ is not presenting a regular variation (increases or decreases) against m for fixed value of n. A comparative study of the behavior of each one of the developed heuristics according to $Time$, yields the following distributions.

- For H_{EP-EP}, $Time \in [0.01, 142.64]$ and the average is $25.868s$.
- For H_{MS-MS}, $Time \in [0.01, 1.41]$ and the average is $0.255s$.
- For H_{EP-MS}, $Time \in [0.01, 101.17]$ and the average is $22.594s$.
- For H_{MS-EP}, $Time \in [0.01, 125.34]$ and the average is $13.2615s$.

Based on the latter distributions, one can conclude that the maximum time running the heuristics using the EP dose not exceed 1.5 minute and the average consumed time is not

TABLE 16. Class C, subclass *LS*: detailed results for H_{EP-MS} and H_{MS-EP}.

			H_{EP-MS}			H_{MS-EP}		
m	n	TLB	$Time$	Gap	NIt	$Time$	Gap	NIt
2	10	0	0.01	0	0	0.02	0.11	1
2	20	0.1	0.11	0	0	0.16	0.4	1
2	40	0.11	0.11	0	0	0.18	0.22	1
2	50	0.1	0.1	0	0	0.17	0.42	1
2	200	8.83	9.03	0	0	18.02	0.09	1
3	10	0	0.01	0.01	0.97	0.04	1.09	2.16
3	20	1.54	1.55	0	2	0.2	3.98	3.57
3	40	1.08	1.09	0	2	0.58	4.03	3.62
3	50	0.82	0.83	0	2	0.92	2.34	3.54
3	200	12.17	12.45	0	2	9.33	0.55	4.3
5	10	0	0.01	0	2	0.04	0.05	1.47
5	20	0.26	0.28	0	2	0.12	3.7	4.98
5	40	16.21	16.32	0.07	3.74	0.61	5.87	6.67
5	50	20.81	20.86	0.07	4	1.05	4.66	7.34
5	200	23.64	23.86	0.01	4	3.33	0.89	7.3
8	10	0	0.01	0	3.46	0.02	0	1.24
8	20	0	0.02	0.08	6.34	0.13	0.1	3.08
8	40	30.57	30.69	0.39	7	0.46	8.14	11.3
8	50	51.48	51.63	0.47	7	0.65	9.1	11.85
8	200	53.61	54.13	0.08	7	4.94	2.44	12.38

exceeding 0.5 minute. This is an additional justification of the usage of the *EP* procedure while developing the heuristics.

According to Table 9, the average relative gap *Gap* is similar for the heuristics H_{EP-EP} and H_{EP-EP} is reached for H_{EP-MS}, with respective values 0.076 and 0.0935. These two values are quite small and proofed that two heuristics H_{EP-EP} and H_{EP-MS} as well as the lower bound *LB*, are performant.

The effect of the second phase (phase 2) is assessed over the average number of iterations *NIt* presented in Table 9. The *NIt* values range from 1 to 5 according to the used heuristic. This shows the impact of the second phase in improving the quality of the proposed heuristics.

For Class C and based on Table 10, the heuristics H_{EP-EP} and H_{EP-MS} outperform the remaining ones. These two heuristics are presenting a very small relative gap $Gap = 0.07$ and an average time $Time = 13.02s$. For more details, about Class C the reader is referred to the Appendix.

As a general conclusion, the heuristics H_{EP-EP} and H_{EP-MS} are performing well in term of makespan value (measured throughout the relative gap *Gap*)with $Gap \leq 0.07$, and in term of short consumed time.

V. CONCLUSION AND FUTURE DIRECTIONS

The parallel processors scheduling problem with no-idle time constraint, release date, and with delivery time, is studied in this paper. This problem is an interesting one from theoretical and practical point of views. Indeed, this problem is proofed to be NP-Hard in strong sense in addition to modeling real life problems. An exact procedure solving the parallel machine scheduling problem with release date and delivery time is used to derive new lower bound and several heuristics for the studied problem. The proposed heuristics are of two phases, where the first phase is constructive and the second one is an improvement phase. An extensive experimental survey is carried out over three classes of instances(10000 instances) with up to 200 tasks and 8 processors. This computational study shows the efficiency of the proposed procedures since the maximum mean relative gap dose not exceed 0.7%. The average consumed time while running the proposed procedures remains satisfactory. As a future directions, evolutionary meta-heuristics will be considered and developed to reduce the consumed time while running the heuristics as well as the lower bound. In addition, a low complexity lower bounds will be proposed. The presented procedures will be integrated to an exact Branch and Bound algorithm, in order to solve optimally the studied scheduling problem.

APPENDIX
DETAILED NUMERICAL RESULTS OF CLASS C

See Table 11–16.

REFERENCES

[1] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proc. Spring Joint Comput. Conf. (AFIPS)*. Montvale, NJ, USA: AFIPS Press, 1967, pp. 483–485.
[2] A. J. Bernstein, "Analysis of programs for parallel processing," *IEEE Trans. Electron. Comput.*, vol. EC-15, no. 5, pp. 757–763, Oct. 1966.
[3] J. Carlier, F. Hermès, A. Moukrim, and K. Ghédira, "Exact resolution of the one-machine sequencing problem with no machine idle time," *Comput. Ind. Eng.*, vol. 59, no. 2, pp. 193–199, Sep. 2010.
[4] J. Carlier, "Scheduling jobs with release dates and tails on identical machines to minimize the makespan," *Eur. J. Oper. Res.*, vol. 29, no. 3, pp. 298–306, Jun. 1987.
[5] D. E. Culler, J. P. Singh, and A. Gupta, *Parallel Computer Architecture—A Hardware/Software Approach*. San Mateo, CA, USA: Morgan Kaufmann, 1999.
[6] E. B. Edis, C. Oguz, and I. Ozkarahan, "Parallel machine scheduling with additional resources: Notation, classification, models and solution methods," *Eur. J. Oper. Res.*, vol. 230, no. 3, pp. 449–463, Nov. 2013.
[7] A. Gharbi and M. Haouari, "Minimizing makespan on parallel machines subject to release dates and delivery times," *J. Scheduling*, vol. 5, no. 4, pp. 329–355, Jul. 2002.
[8] A. Gharbi and M. Haouari, "An approximate decomposition algorithm for scheduling on parallel machines with heads and tails," *Comput. Oper. Res.*, vol. 34, no. 3, pp. 868–883, Mar. 2007.
[9] M. R. Garey and D. S. Johnson, *Computers and Intractability :A Guide to the Theory of NP-Completeness*. New York, NY, USA: Freeman, 1979.
[10] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Ann. Math.*, vol. 5, pp. 287–326, Jan. 1979.
[11] W. Hao, F. Xudong, W. Guangqian, L. Tiejian, and G. Jie, "A common parallel computing framework for modeling hydrological processes of river basins," *Parallel Comput.*, vol. 37, nos. 6–7, pp. 302–315, Jun./Jul. 2011.
[12] T. Wang and Q. Kemao, "Parallel computing in experimental mechanics and optical measurement: A review (II)," *Opt. Lasers Eng.*, vol. 104, no. 1, pp. 181–191, May 2018.
[13] A. Jouglet, "Single-machine scheduling with no idle time and release dates to minimize a regular criterion," *J. Scheduling*, vol. 15, no. 2, pp. 217–238, Apr. 2012.
[14] I. Kacem and H. Kellerer, "Approximation algorithms for no idle time scheduling on a single machine with release times and delivery times," *Discrete Appl. Math.*, vol. 164, pp. 154–160, Feb. 2014.
[15] S. H. Roosta, *Parallel Processing and Parallel Algorithms: Theory and Computation*. Berlin, Germany: Springer, 2012.
[16] G. Wenjing and K. Qian, "Parallel computing in experimental mechanics and optical measurement: A review," *Opt. Lasers Eng.*, vol. 50, no. 4, pp. 608–617, Apr. 2012.
[17] B. Veltman, B. J. Lageweg, and J. K. Lenstra, "Multiprocessor scheduling with communication delays," *Parallel Comput.*, vol. 16, nos. 2–3, pp. 173–182, Dec. 1990.
[18] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, and W. Dai, "Energy efficient task allocation and energy scheduling in green energy powered edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 89–99, Jun. 2019.
[19] A. A. Bertossi and A. Fusiello, "Rate-monotonic scheduling for hard-real-time systems," *Eur. J. Oper. Res.*, vol. 96, no. 3, pp. 429–443, Feb. 1997.

[20] E. L. Lawler, J. K. Lenstra, A. H. G. RinnooyKan, and D. Shmoys, "Sequencing and scheduling: Algorithms and complexity," in *Handbooks in Operations Research and Management Science*, vol. 4. New York, NY, USA: Elsevier, 1993, pp. 445–522.
[21] R. Kostadis, B. Nawaf, and E. Robert, "Deterministic batch scheduling without static partitioning, in *Proc. Job Scheduling Strategies Parallel Process. IPPS/SPDP Workshop (JSSPP)*. San Juan, Puerto Rico: Springer, Apr. 1999, pp. 220–237.
[22] P. Chrétienne, "On single-machine scheduling without intermediate delays," *Discrete Appl. Math.*, vol. 156, no. 13, pp. 2543–2550, Jul. 2008.
[23] J. Carlier, "The one-machine sequencing problem," *Eur. J. Oper. Res.*, vol. 11, no. 1, pp. 42–47, Sep. 1982.
[24] M. Haouari, L. Hidri, and A. Gharbi, "Optimal scheduling of a two-stage hybrid flow shop," *Math. Methods Oper. Res.*, vol. 64, no. 1, pp. 107–124, Aug. 2006.
[25] L. Hidri, S. Elkosantini, and M. M. Mabkhot, "Exact and heuristic procedures for the two-center hybrid flow shop scheduling problem with transportation times," *IEEE Access*, vol. 6, pp. 21788–21801, 2018.
[26] L. Hidri, A. Gharbi, and M. Haouari, "Energetic reasoning revisited: Application to parallel machine scheduling," *J. Scheduling*, vol. 11, no. 4, pp. 239–252, Aug. 2008.
[27] K. Landis, "Group technology and cellular manufacturing," School Bus., Univ. Southern California, Los Angeles, CA, USA, Project Tech. Rep. IOM 581, 1993.
[28] S. Irani and K. R. Pruhs, "Algorithmic problems in power management," *ACM SIGACT News*, vol. 36, no. 2, pp. 63–76, Jun. 2005.
[29] V. Sundriyal and M. Sosonkina, "Modeling of the CPU frequency to minimize energy consumption in parallel applications," *Sustain. Comput., Inform. Syst.*, vol. 17, pp. 1–8, Mar. 2018.
[30] A. Fonseca and B. Cabral, "Understanding the impact of task granularity in the energy consumption of parallel programs," *Sustain. Comput., Inform. Syst.*, vol. 17, pp. 69–80, Mar. 2018.
[31] C. Lu, L. Gao, X. Li, Q. Pan, and Q. Wang, "Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm," *J. Cleaner Prod.*, vol. 144, pp. 228–238, Feb. 2017.
[32] K. Fang, N. Uhan, F. Zhao, and J. W. Sutherland, "A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction," *J. Manuf. Syst.*, vol. 30, no. 4, pp. 234–240, Oct. 2011.
[33] K. Li and B. Lin, "The efficiency improvement potential for coal, oil and electricity in China's manufacturing sectors," *Energy*, vol. 86, pp. 403–413, Jun. 2015.
[34] C. Gahm, F. Denz, M. Dirr, and A. Tuma, "Energy-efficient scheduling in manufacturing companies: A review and research framework," *Eur. J. Oper. Res.*, vol. 248, no. 3, pp. 744–757, Feb. 2016.
[35] J.-F. Chen, L. Wang, and Z.-P. Peng, "A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100557.
[36] M. Yazdani and B. Naderi, "Modeling and scheduling no-idle hybrid flow shop problems," *J. Optim. Ind. Eng*, vol. 10, no. 21, pp. 59–66, Dec. 2016.
[37] Q.-K. Pan and R. Ruiz, "An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem," *Omega*, vol. 44, pp. 41–50, Apr. 2014.
[38] P. Chrétienne and A. Quilliot, "A polynomial algorithm for the homogeneously non-idling scheduling problem of unit-time independent jobs on identical parallel machines," *Discrete Appl. Math.*, vol. 243, pp. 132–139, Jul. 2018.
[39] A. Gharbi and M. Labidi, "Extending the single machine-based relaxation scheme for the job shop scheduling problem," *Electron. Notes Discrete Math.*, vol. 36, pp. 1057–1064, Aug. 2010.
[40] M. Labidi, A. Kooli, T. Ladhari, A. Gharbi, and U. S. Suryahatmaja, "A computational study of the two-machine no-wait flow shop scheduling problem subject to unequal release dates and non-availability constraints," *IEEE Access*, vol. 6, pp. 16294–16304, 2018.

LOTFI HIDRI received the B.S. degree in mathematics from the Tunisian College of Science, in 1993, and the M.S. degree in energetic engineering from the National Engineering School, in 1999, and the Ph.D. degree in operations research from the Tunisian Higher Institute of Management, in 2007. Since 2012, he has been a Faculty Member with the Industrial Engineering Department, King Saud University. His main research interest is focused in scheduling and transportation.

ALI M. AL-SAMHAN received the B.Sc. and M.Sc. degrees in industrial engineering from King Saud University, Riyadh, Saudi Arabia, and the Ph.D. degree from the School of Manufacturing and Mechanical Engineering, Birmingham University, U.K. He is currently a Full Professor and the Chairman of the Industrial Engineering Department, King Saud University. He was the Vice Dean of scientific research at King Saud University. He holds many registered patents and has published articles in leading journals of Industrial and Manufacturing Engineering. His research interests focus on engineering materials and manufacturing processes, monitoring and control of manufacturing systems using artificial intelligence, product design and development, automation and mechatronics, and biomechanical engineering. He is serving as a Consultant for leading manufacturers in Saudi Arabia and a member of SASO committee for industrial standardization. He is also leading a number of projects in industry 4.0 and smart manufacturing funded from different organizations in Saudi Arabia.

MOHAMMED M. MABKHOT received the B.Sc. degree in industrial engineering from King Khalid University, Abha, Saudi Arabia, and the M.Sc. degree in industrial engineering from King Saud University, Riyadh, Saudi Arabia, where he is currently pursuing the Ph.D. degree in industrial engineering. His master's thesis focuses on development of knowledge-based decision support systems for disruption and risks management in reconfigurable manufacturing system. His research interests include the management of smart manufacturing systems to response to unpredictable change in customer demands and to failures of production resources using artificial intelligence.