Improved Formulations and Branch-and-Cut Algorithm for the Unrelated Parallel Machines Scheduling Problem with a Common Server and Job-Sequence Dependent Setup Times

Youssouf Hadhbi, Nathalie Grangeon, Laurent Deroussi, Sylvie Norre

Improved Formulations and Branch-and-Cut Algorithm for the Unrelated Parallel Machines Scheduling Problem with a Common Server and Job-Sequence Dependent Setup Times

Youssouf Hadhbi
Clermont Auvergne University, CNRS, Clermont Auvergne
INP, LIMOS, 63000 Clermont-Ferrand, France
Email: youssouf.hadhbi@uca.fr

Laurent Deroussi
Clermont Auvergne University, CNRS, Clermont Auvergne
INP, LIMOS, 63000 Clermont-Ferrand, France
Email: laurent.deroussi@uca.fr

Nathalie Grangeon
Clermont Auvergne University, CNRS, Clermont Auvergne
INP, LIMOS, 63000 Clermont-Ferrand, France
Email: nathalie.grangeon@uca.fr

Sylvie Norre
Clermont Auvergne University, CNRS, Clermont Auvergne
INP, LIMOS, 63000 Clermont-Ferrand, France
Email: sylvie.norre@uca.fr

***Abstract*—In this work, we focus on a non-preemptive unrelated parallel machines scheduling problem with a common server and job-sequence dependent setup times. This problem arises when planning the production of some mechanical parts of automobile, hydraulic and electrical sectors. It's well known to be NP-Hard. We first propose new mixed integer linear programming formulations for the problem. We then compare them with the only state-of-the-art formulation for the same problem. Based on these results, we devise a Branch-and-Cut algorithm along with computational results are presented to evaluate the performance of our approach. Moreover, we provide a warm starting algorithm for the problem, and further show its influence on boosting the Branch-and-Cut algorithm.**

***Index Terms*—Job-sequence dependent, unrelated parallel machines, setup time, machine eligibility, scheduling with a common server, MILP, branch-and-cut, warm starting algorithm.**

I. Introduction

In this paper, we study a variant of an unrelated parallel machines scheduling problem. Here, we consider a job-sequence dependent setup times, and a common server which is used to manage the setup processing over all machines. It can be stated as follows. We first consider an interval of T_{max} periods T numeroted from 1 to T_{max}. Let M be a set of parallel machines such that each machine $i \in M$ is available along all periods in T. These machines are unrelated given that they have different characteristics, we mention speed, configuration, and quality of work. A common server is considered as a common operator for ensuring the setup process operations for a set N of n jobs over all machines. Notice that all jobs N are available at period 0. Each job $k \in N$ is specified by

- a priority weight $w_k \in \mathbb{R}^*$,
- a processing time $p_i^k \in \mathbb{N}$ on each machine $i \in M$,
- a subset of machines M_k that are capable of processing job k such that $p_i^k > 0$ for each $i \in M_k$ and $p_i^k = +\infty$ for each $i \in M \setminus M_k$,
- a setup time $s_{j,k} \in \mathbb{N}$ of job k after job $j \in N_0 \setminus \{k\}$ if they are assigned to the same machine that is to say the job-sequence dependent setup times,

where N_0 denotes the set of jobs in N with an additional dummy-job 0 (i.e., $N_0 = N \cup \{0\}$) such that the dummy-job 0 precedes the first jobs assigned to the differents machines.
Notice that the common server can be unavailable on some periods. For this, we consider a parameter a_t which equals to 1 if the server is available at period t, an 0 if not.
The problem aims at assigning each job k to one and only one of its qualified machines M_k while satisfying the following constraints:

- *No preemption of processing*: that is to say the job k is processed one and only one time on one of its qualified machines M_k. This means that we have one completion period of processing for each job j,
- *No preemption of setup*: the common server must finish the setup operation of each job k before starting the setup operation for another job even if they are assigned to different machines,
- *Server availability*: the common server can ensure the setup operation of at most one job k at period $t \in T$ if and only if it is available at period t.

Finally, the total weighted completion time of processing for the different jobs N is minimized.
This problem occurs in real industrial flexible manufacturing systems and more precisely when planning and dimensioning the production of some mechanical parts of automobile, hydraulic and electrical sectors in the context of Industry 4.0.

It has been shown to be NP-Hard [10]. There exist several variants of scheduling problem that are related to this problem and have been well studied in the literature while some technological constraints are relaxed like the no preemption of setup and server availability constraints. Here we focus on the original variant called unrelated parallel machines scheduling problem with sequence-dependent setup times. In this context, a sequence-dependent setup time is considered after each completion of processing of each job, we need to regulate machines to process the next job and this needs a setup time which depends only on each consecutive pair of jobs assigned to the same machine. Allahverdi et al. [1] presented a detailed state of the art for a generalized parallel machines scheduling problem considering a setup process. Rabadi et al. [16] proposed a metaheuristic for the problem based on a randomized priority search that has been shown to be very performant compared with a partitioning heuristic proposed by Al Salem et al. [2]. Helal et al. [11] developed a tabu search algorithm for solving the problem and showed that it gives better results than the partitioning heuristic proposed by Al Salem et al. [2]. Arnaout et al. [3] developed an Ant Colony Optimization algorithm for the same problem. This has been shown to be efficient compared with the two last approaches proposed by Al Salem et al. [2] and Helal et al. [11]. Chen [6] developed hybrid metaheuristics for the problem to minimize the weighted number of tardy jobs. They are based on a variable neighborhood descent and a tabu search algorithm. The results have shown the effectiveness of this last approach. Fang et al. [8] have developed a hybrid metaheuristic for the problem. Their approach is based on an adaptive large neighborhood search algorithm with learning automata and a tabu search algorithm. The problem has been divided into two stages. The authors have shown that their approach showed excellent performances compared with ant colony optimization and a worm optimization algorithm proposed by Arnaout et al. [3]. Moreover, their approach has shown to be able to beat an adaptive large neighborhood search algorithm with learning automata. Some exact algorithms are also provided to solve the problem. Fanjul-Peyro et al. [9] introduced a new formulation for the problem. Moreover, the same authors developed a Branch-and-Check algorithm for the problem, and compared it with a hybrid decomposition method based on a Bender decomposition algorithm and a Branch-and-Check algorithm. A Branch-and-Price algorithm was developed by Pereira et al. [15] for the solution of the problem. The results showed that the approach is able to solve large sized instances to optimality with a reasonable computational time. On the other hand, there exists another variant of this problem such that the setup time between each pair of jobs depends on the assigned machine. This is known under the name unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. Lee et al. [13] developed a tabu search algorithm for the problem which has been shown to be performant compared with an existing simulated annealing algorithm. Moreover, the results showed that more than 50 % of small instances are solved to optimality. Exact algorithms have also been used to solve this variant. Avalos-Rosales et al. [4] proposed a new formulation and a metaheuristic based on a multi-start algorithm and variable neighbourhood descent metaheuristic for the problem. A branch-and-bound algorithm (B&B) was developed by Rocha et al. [18] while a greedy randomized adaptive search procedure is used to provide a starting upper bound for the problem.

However, all these previous works did not take into account the presence of a single server as a resource for managing the setup process and shared by all jobs and machines. Recently, Bektur et al. [5] proposed a new formulation and metaheuristics based on a simulated annealing and a tabu search algorithm for solving this problem without taking into account the existence of some unavailability periods for the common server. Only the instances with the number of jobs up to 10 and two machines are solved to the optimality. Huang et al. [12] presented a new formulation for this problem without taking into account the unavailability of the server in some periods.

Recently, Raboudi et al. [17] proposed the first formulation for this problem taking into account all the technological constraints that have been considered in our study. The results have shown the limitation of their formulation such that it suffers from a tractability point of view and it's not able to solve small instances to optimality with a number of jobs up to 7 and 6 machines. To the best of our knowledge, the problem has first been identified by Raboudi et al. [17]. It remains challenging to propose new tractable formulations for the problem. For this, we aim to propose several mixed integer linear programming formulations (MILP) for the problem, and further compare them with the formulation of Raboudi et al. [17]. Using this, we devise a Branch-and-Cut (B&C) algorithm for each formulation to solve the problem, and further show that we are able to outperform the results of Raboudi et al. [17]. Moreover, we propose a warm-start algorithm based on an Iterated Local Search algorithm [14]. This will be used as a preprocessing for the Branch-and-Cut algorithm by giving a feasible solution (if possible) for the problem and an initial upper bound for the Branch-and-Cut algorithm. At the end, we show the influence of the warm-start algorithm.

The remainder of this paper is organized as follows. In Section II, we introduce 3 new MILP for the problem. These will be compared in Section III with an existing formulation of the problem. A Branch-and-Cut algorithm is presented in Section IV. We then present an extensive computational study in Section V using 4 classes of instances. Finally, we summarize our results and future outlook in Section VI.

II. Mixed Integer Linear Programming Formulations

In what follows, we introduce three different MILP for the problem.

A. Formulation I

First, we introduce the first formulation which is based on the following variables. For each job $k \in N$ and machine

$i \in M$, let u_i^k be a variable which takes 1 if job k is assigned to machine i, and 0 if not. For each machine $i \in M$, job $j \in N$ and job $k \in N$, let $x_{j,k}^i$ be a variable which takes 1 if jobs j is processed immediately before job k on machine i, and 0 if not. For each job $j \in N$ and period $t \in T$, variable y_t^k will take 1 if the setup operation of job k is performed at period t, and 0 if not. We also consider the variables q_k^j for each two distinct jobs $j, k \in N$ which takes 1 if the job k is processed after job j even if they are not assigned to the same machine, and 0 if not. This is equivalent to say that the starting period of the setup operation for job k is performed after the ending period of the setup operation of job j. We will denote by b_t^k (resp. e_t^k) the binary variable which takes 1 if t is the starting period (resp. the ending period) of the setup operation for job k, and 0 if not. The completion period of processing for each job $k \in N$ is denoted by $c_k \in \mathbb{R}^+$.

The problem is equivalent to the following MILP

$$\min \sum_{k \in N} w_k c_k, \tag{1}$$

subject to

$$\sum_{i \in M_k} u_i^k = 1, \forall k \in N, \tag{2}$$

$$\sum_{i \in M \setminus M_k} u_i^k = 0, \forall k \in N, \tag{3}$$

$$\sum_{j \in N_0 \setminus \{k\}} x_{j,k}^i = u_i^k, \forall k \in N, \tag{4}$$

$$\sum_{j \in N \setminus \{k\}} x_{k,j}^i \leq u_i^k, \forall k \in N, \tag{5}$$

$$\sum_{k \in N} x_{0,k}^i \leq 1, \forall i \in M, \tag{6}$$

$$\sum_{k \in N} y_t^k \leq a_t, \forall t \in T, \tag{7}$$

$$\sum_{t \in T} y_t^k = \sum_{j \in N_0 \setminus \{k\}} \sum_{i \in M} s_k^j x_{j,k}^i, \forall k \in N, \tag{8}$$

$$b_t^k \leq y_t^k, \forall k \in N \text{ and } t \in T, \tag{9}$$

$$y_t^k \leq b_t^k + \sum_{t'=1}^{t-1} y_{t'}^k, \forall k \in N \text{ and } t \in T, \tag{10}$$

$$\sum_{t \in T} b_t^k = 1, \forall k \in N, \tag{11}$$

$$\sum_{j \in N \setminus \{k\}} \sum_{i \in M} p_j^i x_{j,k}^i \leq \sum_{t \in T} t b_t^k, \forall k \in N, \tag{12}$$

$$(t+1) y_t^k + B(y_t^k - 1) \leq \sum_{t \in T} t e_t^k, \forall k \in N \text{ and } t \in T', \tag{13}$$

$$\sum_{t \in T} e_t^k = 1, \forall k \in N, \tag{14}$$

$$c_j + B(\sum_{i \in M} x_{j,k}^i - 1) \leq \sum_{t \in T} t b_t^k, \forall k \in N, \forall j \in N, \tag{15}$$

$$\sum_{t \in T} t e_t^k - \sum_{t \in T} t b_t^k \geq \sum_{j \in N_0 \setminus \{k\}} s_{j,k}^i x_{j,k}^i, \forall k \in N, \tag{16}$$

$$\sum_{t \in T} t e_t^k \leq \sum_{t \in T} t b_t^j + B q_k^j, \forall k \in N, \forall j \in N, \tag{17}$$

$$\sum_{t \in T} t e_t^j \leq \sum_{t \in T} t b_t^k + B(1 - q_k^j), \forall k \in N, \forall j \in N, \tag{18}$$

$$c_k = \sum_{t \in T} t e_t^k + \sum_{i \in M_k} p_k^i u_i^k, \forall k \in N, \tag{19}$$

$$c_k \leq T_{max}, \forall k \in N, \tag{20}$$

$$u_i^k, x_{j,k}^i, y_t^k, q_k^j, b_t^k, e_t^k, c_k \geq 0, \tag{21}$$

$$u_i^k, x_{j,k}^i, y_t^k, q_k^j, b_t^k, e_t^k \in \{0, 1\}. \tag{22}$$

where $T' = \{1, ..., T_{max} - 1\}$, and B a large integer number (eg., $B = T_{max}$ is feasible).

As mentioned before, the objective function (1) consists in minimizing the total weighted completion time of processing for the different jobs in N. Equations (2) ensure that each job $k \in N$ is assigned to only one qualified machine $i \in M_k$. A job $k \in N$ cannot be assigned to a non qualified machine $i \in M \setminus M_k$ as shown by equations (3). Equations (4) express the fact that a job k is preceded by one job $j \in N_0 \setminus \{k\}$ on a machine $i \in M$ if and only if job k is assigned to machine i. Moreover, constraints (5) ensure that a job k can be considered as the predecessor of at most one job $j \in N \setminus \{k\}$ on machine i if and only if it is assigned to machine i. Constraints (6) ensure that the dummy-job can precede at most one job on each machine $i \in M$. The common server can ensure the setup operation of at most one job at each period $t \in T$ if and only it is available at period t as noticed in constraints (7). However, variables y_t^k are forced to be equal to 0 for each $k \in N$ when the common server is not available at period t. The number of periods in which the setup is performed for job k must be equal to its setup time as shown in equations (8). Constraints (9) ensure that a period t cannot be a starting period of setup for a job k if the setup operation is not performed at period t. Constraints (10) ensure that a period t can be a starting period of the setup operation of jobs k if the setup of job k is performed at period t and there does not exist a period $t' \in \{1, ..., t-1\}$ in which the setup of job k is performed. Equations (11) express the fact that each job k has one and only one starting period of setup operation. The starting period of setup for a job k is forced to be greater than the processing time of a job j which is processed immediately before job k on a shared qualified machine in $M_k \cap M_j$. This is ensured by constraints (12). In a similar way, we ensure in constraints (13) that the setup operation of each job $k \in N$ is accomplished after the last period t in which the setup is performed for job k. Moreover, equations (14) ensure that each job k has only one ending period of setup operation. Constraints (15) ensure the non-overlapping of the processing operation with the setup operation for two distinct jobs that are processed one after the other and immediately. The gap between the ending period and the starting period of setup for job j is greater than its setup time as shown in constraints (16). Constraints (17) and (18) ensure the no preemption of setup constraints. The constraints (19) compute the completion period for each job $k \in K$. This completion period is imposed to be smaller

than the maximum time of processing over machines T_{max} by contraints (20). Inequalities (21) are the trivial inequalities, and constraints (22) are the integrality constraints.

B. Formulation II

Here we propose a new MILP for the problem which can be seen as an improved formulation for the last formulation while sharing some variables and constraints with the formulation I. For this, we keep the same objective function, and the variables u_i^k, $x_{j,k}^i$, q_k^j, y_t^k and c_k. We consider two new variables $b^k \in \mathbb{R}^+$ and $e^k \in \mathbb{R}^+$ respectively to express the start period and the end period of the setup process for each job $k \in N$. The constraints (2)-(8) and (20) of formulation I are also used in this new formulation. However, the rest of constraints of formulation I are replaced by the following constraints

$$b^k \leq ty_t^k + B(1 - y_t^k), \forall k \in N \text{ and } t \in T, \tag{23}$$

$$ty_t^k - B\sum_{t'=1}^{t-1} y_{t'}^k \leq b_k, \forall k \in N \text{ and } t \in T, \tag{24}$$

$$\sum_{j \in N \setminus \{k\}} \sum_{i \in M} p_j^i x_{j,k}^i \leq b^k, \forall k \in N, \tag{25}$$

$$(t+1)y_t^k \leq e^k, \forall k \in N \text{ and } t \in \{1, ..., T_{max} - 1\}, \tag{26}$$

$$c_j + B(\sum_{i \in M} x_{j,k}^i - 1) \leq b^k, \forall k \in N, \forall j \in N \setminus \{k\}, \tag{27}$$

$$e^k - b^k \geq \sum_{j \in N_0 \setminus \{k\}} s_{j,k}^i x_{j,k}^i, \forall k \in N, \tag{28}$$

$$e^k \leq b^j + Bq_k^j, \forall k \in N, \forall j \in N \setminus \{k\}, \tag{29}$$

$$e^j \leq b^k + B(1 - q_k^j), \forall k \in N, \forall j \in N \setminus \{k\}, \tag{30}$$

$$c_k = e^k + \sum_{i \in M_k} p_k^i u_i^k, \forall k \in N, \tag{31}$$

$$c_k, b^k, e^k \geq 0, \forall k \in N. \tag{32}$$

As noticed in the program presented above, constraints (9) and (10) are replaced by constraints (23)-(25) to express the fact that each job k has one starting period of setup and should be equal to the smallest period t in which the setup is performed for job k. In a similar way, constraints (13)-(14) are also replaced by constraints (26) to ensure that the setup is accomplished after the last period t in which the setup is performed. Constraints (15) are replaced by (27). Constraints (16) are replaced by constraints (28) to impose the gap between the ending period and the starting period of setup for each job k. We replace the constraints (17) by (29), and (18) by (30) to ensure the no preemption of the setup for the different jobs in N. The constraints (31) compute the completion period for each job $k \in N$ as done in (19). Inequalities (32) are the trivial inequalities.

C. Formulation III

In what follows, we introduce a new formulation in order to reduce the number of variables for the formulation II. For this, we keep the same objective function, and the variables u_i^k, $x_{j,k}^i$, q_k^j, y_t^k and c_k. However, this caused a huge increase in the number of constraints. The constraints (2)-(8) and (20) of formulation I are still used in this formulation. However, the other ones are replaced by the following constraints

$$c_j + B(\sum_{i \in M} x_{j,k}^i - 1) \leq ty_t^k + B(1 - y_t^k), \forall k, j \in N \text{ and } t \in T, \tag{33}$$

$$c_k - \sum_{i \in M_k} p_k^i u_i^k \leq ty_t^j + B(1 - y_t^j + q_k^j), \forall k, j \in N \text{ and } t \in T, \tag{34}$$

$$c_j - \sum_{i \in M_j} p_j^i u_i^j \leq ty_t^k + B(2 - y_t^k - q_k^j), \forall k, j \in N \text{ and } t \in T, \tag{35}$$

$$(t+1)y_t^k + \sum_{i \in M_k} p_k^i u_i^k \leq c_k, \forall k \in N \text{ and } t \in T. \tag{36}$$

The variables b^k of formulation II are replaced by $ty_t^k + B(1 - y_t^k)$. The variables e^k are also replaced by $c_k - \sum_{i \in M_k} p_k^i u_i^k$. Constraints (27) are replaced by (33) for ensuring the non-overlapping of the processing operation with the setup operation for two jobs k and j if job j is processed immediately before job k on a shared qualified machine. The no preemption of the setup constraints are now ensured by constraints (34) and (35). The constraints (36) compute the completion period for the different jobs in N.

III. Comparative Study

Based on the previous results, we compare our three formulations with an existing formulation proposed by Raboudi et al. [17] in terms of their number of binary variables, continuous variables, and number of constraints as shown in Table I. We notice that in the formulation III, we use less number of variables compared with the other ones. On the other hand, the formulation II contains less number of constraints compared with the other ones. As a consequence, and in terms of model size (number of variables * number of constraints), the formulation may suffer less from a tractability point of view compared with the other formulations. More details about the differences between these different formulations will be highlighted in the rest of this paper.

IV. Branch-and-Cut Algorithm

Using the theoretical results presented in the previous sections, we devise a Branch-and-Cut algorithm for each formulation. This algorithm consists in solving a sequence of linear programs using a cutting-plane algorithm and provides an optimal solution for the linear relaxation at each node of the well known *branch-and-bound* tree. This solution may not be feasible for the problem if it does not satisfy the integrality constraints. For this, we perform the branching phase which consists in dividing the problem into subproblems. We continue this process until an optimal solution is obtained for the problem.

V. Computational Study

The B&C algorithm has been implemented in Java using CPLEX 12.9 [7] as a MILP solver while allowing the generation of its proper cuts and benefiting from its parallelism

TABLE I
MODEL SIZE: COMPARISON BETWEEN DIFFERENT FORMULATIONS.

	Number of Constraints	Number of Binary Variables	Number of Continuous Variables
Raboudi et al. [17]	$\|N\|(1+4\|M\|+2\|M\|\|N\|+2\|T\|+2\|N\|)+\|M\|+\|T\|+1$	$\|N\|(\|M\|+\|N_0\|\|M\|+\|N\|+\|M\|\|T\|)$	$4\|N\|$
MILP I	$\|N\|(4+2\|M\|+3\|T\|+3\|N\|)+\|M\|+\|T\|$	$\|N\|(\|N\|+\|M\|+\|M\|\|N_0\|+3\|T\|)$	$\|N\|$
MILP II	$\|N\|(2+2\|M\|+3\|T\|+3\|N\|)+\|M\|+\|T\|$	$\|N\|(\|M\|+\|M\|\|N_0\|+\|T\|+\|N\|)$	$3\|N\|$
MILP III	$\|N\|(2\|M\|+3\|N\|\|T\|+\|T\|)+\|M\|+\|T\|$	$\|N\|(\|M\|+\|M\|\|N_0\|+\|T\|+\|N\|)$	$\|N\|$

TABLE II
INSTANCES CHARACTERISTICS.

Instances / Characteristics	**I**	**II**	**III**	**IV**
$\|M\|$	{2, 3, 4}			
$\|N\|$	{4,5,6,7,8,10, 20, 24}			
w_k	{1,2,3,4,5}			
p_k^i	[10; 350]			
$s_{j,k}$	[0; 10]			
$\|M_k\|$	≥ 1	≥ 1	≥ 2	≥ 2
Server continuous availability	8 hours	24 hours	8 hours	24 hours
Server continuous unavailability	8 hours			
T_{max}	{200, 350, 400, 600, 800, 1000}			

by activating 8 threads. We use a HPC server with a memory size limited to 64 Gb to perform our experiments. The CPU time is limited to 1 hour (3600 s). In addition, we present our computational results using four types of instances as shown in Table II such that for each type of instances we generate 34 instances randomly with a number of machines up to 4, number of jobs up to 24 and 1000 as the maximum makespan (T_{max} hours).

In what follows, we present the performance results of the different formulations using a Branch-and-Cut algorithm. The main goal is to show the effectiveness of our approach compared with the existing one of Raboudi et al. [17].

Table III reports a comparative study between these different algorithms considering different criterias and the same objective function. The results show that MILP II is able to beat the other ones over all types of instances such that using MILP II, we are able to solve more instances to optimality compared with the other formulations (see column 1 in Table III). Almost all of these instances are solved in less CPU time and fewer nodes proceeded in the B&C tree by MILP II (see columns 2, 3 and 4 in Table III). Moreover, and when the optimality is not proven, MILP II enables reducing the gap (i.e., the relative error between the lower bound and the best upper bound obtained at the end of the resolution) for several instances that are solved with a large gap by the other formulations (see columns 5 in Table III). However, there exists a few instances that are not solved by MILP II for which the CPLEX's primal heuristic is not able to produce a feasible solution for these instances. On the other hand, MILP I is shown to be the second best formulation for our study. It's able to solve more instances to optimality using our different types of instances compared with MILP III or that of Raboudi et al. [17]. As a consequence, our computational experiments show clairly the strength of MILP II and I. Moreover, they significantly improve the results yielded by Raboudi et al. [17].

Next, we investigate several enhancements that can be used to speed up and increase the efficiency of the different formulations and even for that of Raboudi et al. [17]. This is based on a warm-start algorithm used to introduce a feasible solution for the problem and initialize the upper bound for the B&C algorithm. For this, we first generate an initial sequence of jobs for each instance, and compute its solution using a greedy-algorithm which consists in selecting the best machine assignment for each job in the sequence. This sequence will be used in an Iterated Local Search algorithm [14] which aims at identifying the best sequence of jobs having the smallest total weighted completion time. Using this, we evaluate the influence of the warm-start algorithm on the Branch-and-Cut algorithm (see Table IV) considering several criterias. We noticed in column 4 of Table IV that using the warm-start algorithm speed-up the resolution of several instances by MILP II, MILP I and that of Raboudi et al. [17]. Also, the warm-start algorithm is able to decrease the gap for several instances (see column 2 in Table IV), and the number of nodes in the B&C tree (see column 3 in Table IV). Moreover, there exist some instances that haven't been solved to optimality by the different formulations, that are solved to optimality when using the warm-start algorithm (see column 1 of Table IV vs column 1 of Table III). There are also some instances that are not solved by the different formulations, that are solved with a significant gap when using the warm-start algorithm (see column 5 of Table IV vs column 6 of Table III). In some sense, the warm-start algorithm is shown to be a promising prospect and yield an efficient Branch-and-Cut algorithm.

VI. CONCLUDING REMARKS

In this paper, we have studied a non-preemptive unrelated parallel machines scheduling problem with a common server and job-sequence dependent setup times. First, we have proposed several MILP for the problem. These have been compared with an existing formulation for the problem. Using this, we have developed an exact algorithm based on a Branch-and-Cut algorithm to solve the problem. The results have shown that some formulations perform very well compared with other ones and that of the existing one of the literature. Furthermore, we have introduced a warm-start algorithm based on a metaheuristic called Iterated Local Search algorithm. This has been shown to be very interesting and allow improving the results yielded by the Branch-and-Cut algorithm.

Finally, it would be interesting to further study other variants of the problem, for example taking into account the existence of multiple servers, and also the unavailability of each machine in some periods. Moreover, through our research we plan to develop some adaptive metaheuristics, hybrid metaheuristics and matheuristics considering large-sized instances for the problem.

TABLE III
BRANCH-AND-CUT ALGORITHM: COMPARISON BETWEEN DIFFERENT FORMULATIONS.

		% of instances solved to optimality	% of instances solved to optimality in less than 10 mn	% of instances solved to optimality with the smaller number of nodes	% of instances solved to optimality with the smaller CPU Time	% of instances not solved to optimality but with the smaller gap	% of instances not solved
Instance I	**Raboudi et al. [17]**	11,76	2,94	0,00	0,00	0,00	17,65
	MILP I	20,59	2,94	0,00	0,00	0,00	47,06
	MILP II	**67,65**	**58,82**	**100**	**100**	**14,7**	**17,65**
	MILP III	2,94	2,94	0,00	0,00	0,00	17,65
Instance II	**Raboudi et al. [17]**	2,94	2,94	0,00	0,00	0,00	17,65
	MILP I	41,18	17,65	0,00	0,00	0,00	41,18
	MILP II	**73,53**	**64,71**	**100**	**100**	**11,7**	**14,71**
	MILP III	0,00	0,00	0,00	0,00	0,00	17,65
Instance III	**Raboudi et al. [17]**	5,88	2,94	0,00	0,00	0,00	17,65
	MILP I	8,82	2,94	0,00	0,00	0,00	55,88
	MILP II	**44,12**	**38,24**	**100**	**100**	**44,1**	**14,71**
	MILP III	0,00	0,00	0,00	0,00	0,00	20,59
Instance IV	**Raboudi et al. [17]**	2,94	8,82	0,00	0,00	0,00	17,65
	MILP I	17,65	2,94	0,00	0,00	0,00	55,88
	MILP II	**70,59**	**64,71**	**100**	**100**	**11,7**	**17,65**
	MILP III	0,00	0,00	0,00	0,00	0,00	17,65

TABLE IV
IMPACT OF THE WARM-START ALGORITHM ON THE BRANCH-AND-CUT ALGORITHM

		% of instances solved to optimality	% of instances for which we improved the gap %	% of instances for which we improved the gap % (or equal) and decreased the B&C tree size	% of instances for which we improved the CPU Time	% of instances not solved
Instances I	**Raboudi et al. [17]**	11,76	79,41	44,12	11,76	8,82
	MILP I	23,53	38,24	35,29	14,71	23,53
	MILP II	73,53	23,53	82,35	47,06	8,82
	MILP III	2,94	67,65	64,71	0,00	8,82
Instances II	**Raboudi et al. [17]**	2,94	64,71	58,82	5,88	11,76
	MILP I	44,12	44,12	79,41	35,29	11,76
	MILP II	79,41	8,82	82,35	67,65	11,76
	MILP III	0,00	82,35	73,53	0,00	11,76
Instances III	**Raboudi et al. [17]**	5,88	58,82	61,76	5,88	14,71
	MILP I	17,65	76,47	88,24	17,65	14,71
	MILP II	50,00	35,29	44,12	41,18	8,82
	MILP III	0,00	82,35	64,71	0,00	14,71
Instances IV	**Raboudi et al. [17]**	2,94	73,53	41,18	0,00	14,71
	MILP I	35,29	64,71	82,35	32,35	14,71
	MILP II	70,59	14,71	70,59	61,76	14,71
	MILP III	0,00	82,35	41,18	0,00	14,71

ACKNOWLEDGMENT

This work was supported by the France-Relance Project in collaboration with InoProd (https://www.inoprod.com/).

REFERENCES

[1] A. Allahverdi, J. N. D Gupta, and T. Aldowaisan, "A review of scheduling research involving setup considerations" in OMEGA International Journal of Management Science, 27, 1999, pp. 219–239.

[2] A. Al-Salem, "Scheduling to minimize makespan on unrelated parallel machines with sequence dependent setup times" in Engineering Journal of University of Qatar, 17, 2004, pp. 177-187.

[3] J.P. Arnaout, G. Rabadi, and R. Musa, "A two-stage Ant Colony optimization algorithm to minimize the makespan on unrelated parallel machines—part II: enhancements and experimentations" in Journal of Intelligent Manufacturing, 25, 2014, pp. 43–53.

[4] O. Avalos-Rosales, F. Angel-Bello, and A. Alvarez, "Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times" in International Journal of Advanced Manufacturing Technology, 76, 2015, pp. 1705–1718.

[5] G. Bektur, and T. Saraç, "A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server" in COR Journal, 103, 2019, pp. 46-63.

[6] J.F. Chen, "Scheduling on unrelated parallel machines with sequence- and machine-dependent setup times and due-date constraints" in Journal of Advanced Manufacturing Technology, 44, 2009, pp. 1204–1212.

[7] I.I. CPLEX: V12. 9, "User's Manual for CPLEX" in International Business Machines Corporation, 46(53), pp. 157.

[8] W. Fang, H. Zhu, and Y. Mei, "Hybrid meta-heuristics for the unrelated parallel machine scheduling problem with setup times" in Knowledge-Based Systems Journal, 241, 2022, 108193 p.

[9] L. Fanjul-Peyro, R. Ruiz, and F. Perea, "Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times" in COR Journal, 101, 2019, pp. 173-182.

[10] P.L. Hammer, E.L. Johnson, and B.H. Korte, "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey", in Annals of Discrete Mathematics, 5, 1979, pp. 287-326.

[11] M. Helal, G. Rabadi, A. Al-Salem, "A tabu search algorithm to minimize the makespan for the unrelated parallel machines scheduling problem with setup times" in IJOR, 3, 2006, 182-192.

[12] S. Huang, L. Cai, and X. Zhang, "Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server" in Computers and Industrial Engineering Journal, 58, 2010, pp. 165-174.

[13] J.H. Lee, J.M. Yu, and D.H. Lee, "A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: minimizing total tardiness" in International Journal of Advanced Manufacturing Technology, 69, 2013, pp. 2081-2089.

[14] H.R. Lourenço, O.C. Martin, T. Stützle, "Iterated Local Search", in Glover, F., Kochenberger, G.A. (eds) Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol 57. Springer, Boston, MA, 2003.

[15] M.J. Pereira Lopes, and J.M.V. De-Carvalho, "A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times" in EJOR, 176, 2007, pp. 1508–1527.

[16] G. Rabadi, R. Moraga, and A. Al-Salem, "Heuristics for the unrelated parallel machine scheduling problem with setup times" in Journal of Intelligent Manufacturing, 17, 2006, pp. 85-97.

[17] H. Raboudi, G. Alpan, F. Mangione, G. Tissot, F. Noels, "Scheduling unrelated parallel machines with a common server and sequence dependent setup times" in IFAC-PapersOnLine, 55, 2022, pp. 2179-2184.

[18] P. Rocha, M. Ravetti, G. Mateus, and P. Pardalos, "Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times" in COR Journal, 35, 2008, pp. 1250-1264.