IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 November 2023, accepted 14 December 2023, date of publication 21 December 2023, date of current version 28 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3345793

RESEARCH ARTICLE

A Constraint Programming Formulation of the Multi-Mode Resource-Constrained Project Scheduling Problem for the Flexible Job Shop Scheduling Problem

FRANCISCO YURASZECK[1], ELIZABETH MONTERO[2], (Member, IEEE), DARÍO CANUT-DE-BON[3,4], NICOLÁS CUNEO[4], AND MAXIMILIANO ROJEL[4]
[1]Department of Engineering Sciences, Universidad Andres Bello, Viña del Mar 2531015, Chile
[2]Department of Computer Science, Universidad Técnica Federico Santa María, Valparaíso 2390123, Chile
[3]Dirección de Programas, Investigación y Desarrollo de la Armada de Chile, Valparaíso 2360035, Chile
[4]Faculty of Engineering, Universidad Andres Bello, Viña del Mar 2531015, Chile

Corresponding author: Francisco Yuraszeck (francisco.yuraszeck@unab.cl)

ABSTRACT In this work, a constraint programming (CP) formulation of the multi-mode resource-constrained project scheduling problem (MMRCPSP) is proposed for solving the flexible job shop scheduling problem (FJSSP) under the makespan minimization criterion. The resulting CP model allows us to tackle the classical instances of the FJSSP (such as where the operations of a given job follow a linear order). It can also handle FJSSP instances where the precedence relationships between operations are defined by an arbitrary directed acyclic graph (sequencing flexibility). The performance of our approach was tested using 271 classical FJSSP instances and 50 FJSSP instances with sequencing flexibility. We establish the validity of our approach by achieving an average relative percentage deviation of 3.04% and 0.18% when compared to the best-known lower and upper bounds, respectively. Additionally, we were able to contribute to the literature with ten new lower bounds and two new upper bounds. Our CP approach is relatively simple yet competitive and can be quickly applied and adapted by new practitioners in the area.

INDEX TERMS Constraint programming, flexible job shop, FJSSP, job shop, JSSP, multi-mode resource-constrained project scheduling problem, MMRCPSP, sequencing flexibility.

I. INTRODUCTION

In the scheduling and combinatorial optimization literature, one of the most studied problems in the last decades is the well-known job shop scheduling problem (JSSP). In fact, according to Scopus's database, at the time of writing this paper, the term "job shop" returns over +13,000 results for the query of the term within the article title, abstract, and/or keywords.

In a JSSP environment, there exist n jobs that must be processed without preemptions at m machines. The operation of job j at machine i is denoted by O_{ji}. In this context, the route that must follow the jobs at the machines is known in advance, thus, the problem consists of establishing the sequence of the operations at the machines (i.e. the order in which the operations of the jobs will be sorted in a determined machine) to minimize a given criterion. Other common assumptions are that both machines and jobs are all available at time zero, also machines can not perform more than one operation at a time. Additionally, the makespan or C_{max} (the total length of the schedule) is, by far, the most researched regular objective function criterion, where a regular objective criterion is a function that is non-decreasing in terms of the completion time of the jobs. As first shown by [1], the shortest-length schedule problem for an m machine JSSP is NP-complete for $m \geq 2$.

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Ching Ying.

Within this framework, different extensions for the JSSP have been proposed in order to map the characteristics of intricate real-world shop situations. This paper will focus on the flexible job shop scheduling problem (FJSSP), where each operation can be processed on any of the machines from a subset of the set of machines [2]. Thus, as opposed to the JSSP, in an FJSSP it is also necessary to allocate the job's operations to the machines. The FJSSP is an NP-hard (non-deterministic polynomial time) problem since it takes the JSSP to a particular case. In terms of representation, the problem of minimizing the makespan in an FJSSP can be denoted as $FJSSP_m//C_{max}$ following the three-field notation proposed in [3].

Additionally, as a consequence of mass customization and automation, [2] claims that the FJSSP is no longer just another extension of the JSSP but a shop scheduling problem on its own. Indeed, we consider that the abundant literature related to the FJSSP reinforces this affirmation.

Because of its high computational complexity, the solution methods to tackle the FJSSP are mainly dominated by heuristic and metaheuristic algorithms. Nevertheless, research in exact methods such as mixed integer linear programming (MILP) and constraint programming (CP) has also been conducted. In Section II we will explore the solution methods further.

In this paper, we propose a CP formulation for the FJSSP that relies on the well-known multi-mode resource-constrained project scheduling problem (MMRCPSP). The MMRCPSP extends the resource-constrained project scheduling problem (RCPSP). An MMRCPSP involves the simultaneous scheduling of multiple project instances into an optimal schedule. These instances have activities that can be executed in various modes and are subject to different types of resources (renewable and/or non-renewable), time, and precedence constraints [4]. In this regard, we consider that a *project instance* is equivalent to a job, an *activity* is analogous to a job operation, and a *renewable resource* is equivalent to a machine. In our approach, we do not consider a homologous to a non-renewable resource. To the best of our knowledge, an MMRCPSP approach has not been applied before to tackle the FJSSP.

To assess the performance of our approach, we conducted a computational study using 321 FJSSP instances of varying sizes and characteristics: 271 *classical* FJSSP instances (where the operations of a given job follow a linear order), and 50 FJSSP instances with *sequencing flexibility* (the precedence relationships between the operations of a specific job are defined by an arbitrary directed acyclic graph). We contrast our results with the best-known up-to-date lower bounds and upper bounds available in the literature. Our approach demonstrates its validity by achieving an average relative percentage deviation of 3.04% and 0.18% when compared to the best-known lower bound and upper bound, respectively. Also, we contribute with ten new lower bounds and two new upper bounds. Lastly, the proposed MMRCPSP formulation can readily be adapted to address related shop scheduling problems and alternative objective criteria.

The remainder of the paper is organized as follows. In Section II we review the literature related to the FJSSP, with an emphasis on the solution methods applied. Then, in Section III we present a CP formulation based on the MMRCPSP to tackle the FJSSP. Next, in Section IV we evaluate the performance of our approach when solving the FJSSP, comparing its performance with the best-known lower bounds and upper bounds reported in the literature. Finally, in Section V we conclude and highlight possible lines for future research.

II. LITERATURE REVIEW

The FJSSP under C_{max} minimization, is an NP-hard problem that has a wide range of applications in the real world. This extension of the classical JSSP arises as a response to the flexibility that introduces the chance that the job's operations must also be assigned to machines. Indeed, in manufacturing is common that some job operations can be accomplished in more than one machine. Moreover, these machines could be technologically different, and thus, the processing times could be different from machine to machine. In this regard, the three-field notation ($\alpha/\beta/\gamma$) introduced by [3] is a standard and compact form of representing the specific characteristics of the shop scheduling problem under study.

The literature relative to the FJSSP is abundant. We refer the reader to a recent survey of [2] that presents and classifies the different criteria, constraints, configurations, and solution approaches that have been considered for the FJSSP in the last 30 years of research.

In terms of solution methods applied to solve the FJSSP, these are mainly based on approximate methods such as heuristics and metaheuristics. Metaheuristics follows certain rules to explore the most promising areas within the solution space with the aim of finding high-quality solutions. However, this comes at the expense of increased computing time compared to heuristics [5]. In fact, metaheuristics have proven to be powerful for solving hard scheduling problems. In this context, [6] proposed an effective Quantum Annealing (QA) metaheuristic for solving the FJSSP in a time-efficient manner. In [7], a discrete improved grey wolf optimization (DIGWO) algorithm proves effective in solving the FJSSP. Moreover, in [8] the authors introduce a framework that generalizes literature results related to local search algorithms for the JSSP and FJSSP. A Greedy Randomized Adaptive Search Procedure (GRASP) is proposed by [9] to tackle efficiently large instances of the generalized FJSSP (where hard constraints are taken into account such as machine capacity, time lags, and sequence-dependent setup times, among others). Despite the success of the use of metaheuristics to solve hard scheduling problems, they also have some drawbacks. These include a lack of guarantee to find the optimal solution, difficulty in handling constraints, sensitivity to the representation of the problem, and complex coding, among others. In general, the effectiveness of these metaheuristics methods relies on the implementation and fine-tuning of parameters, as they integrate both problem representation and solution strategy within unified frameworks [10].

In contrast, the mathematical modeling approach (where MILP and CP belong) separates the problem representation from the solution strategy [11]. Moreover, for both exact methods, in some small-sized scheduling instances, it is possible to prove the optimality of a determined solution. However, they may face challenges in terms of scalability for large instances. In this regard, [12] presents a MILP and a CP model to formulate the multiresource FJSSP with arbitrary precedence graphs, in order to assess the efficiency of both methods. In [9], three approaches are compared when solving the generalized FJSSP: GRASP, MILP, and CP. The computational results indicate that CP outperforms MILP, and GRASP outperforms CP. Earlier, in [10], a mixed integer programming (MIP) and a CP approach were used to address the FJSSP with parallel batch processing machines, where CP incomparably outperforms the MIP approach. To our knowledge, usually, CP has proven to be more competitive than MILP approaches in solving complex shop scheduling problems. Indeed, the computational results reported in [13] and [14] concerning other related shop scheduling problems support this claim.

Regarding the characteristics of the problem, an FJSSP assumes that the route of a job is sequential or linear. This is what we call a *classical* FJSSP instance. However, in real-world manufacturing, a job operation may have more than one predecessor or more than one successor. This extension has been described in the literature using terms like sequencing flexibility, non-linear routes, and arbitrary precedence constraints, among others. Hereinafter, we refer to this case as *sequencing flexibility* adopting the terminology used by [15]. In Section III we address both cases in detail.

In the same way, as for the FJSSP, the literature related to the MMRCPSP is vast. The MMRCPSP is a generalization of the resource-constrained project scheduling problem (RCPSP) where each activity may have more choice by performing different modes (i.e., the activity duration and resource requirements) [16]. Thus, every mode represents a feasible compromise between the time duration of a task and resource demand. The MMRCPSP is an NP-hard optimization problem as same as the RCPSP [17]. Hence, large-sized instances of the MMRCPSP cannot be optimally solved in a reasonable computational time [18].

Because of its computational complexity, the solution methods proposed for tackling the MMRCPSP mostly rely on heuristics and metaheuristics. In [19], the authors provide a brief survey of the literature, showing arguments in favor of the use of metaheuristic approaches to solve the MMRCPSP. A similar summary can also be found in [20]. On the other hand, the authors in [18] claim that exact optimization algorithms are appropriate for smaller data sets. Hybrid metaheuristics have been applied as well in the context of the MMRCPSP. In [4], a hybrid approach (or matheuristic) is proposed to handle the MMRCPSP, combining CP and a metaheuristic-based algorithm.

Finally, to the best of our knowledge, the MMRCPSP has not been applied to make an equivalent representation of the FJSSP. Moreover, metaheuristics approaches for handling the FJSSP are usually tailor-made and often require complex computational coding and calibrations of multiple parameters. Then, this paper is motivated to provide a simple yet competitive approach to handling FJSSP instances through the resolution of an MMRCPSP using a CP solver which can be quickly used by new practitioners in the area.

III. CONSTRAINT PROGRAMMING FORMULATION

The following CP formulation for the FJSSP is based on the "**sched_rcpspmm.mod**" file available in the examples folder of the IBM ILOG CPLEX Optimization Studio 22.1.1 installation. Since the FJSSP instances we are addressing do not involve non-renewable resources, we have chosen to exclude this part from the original MMRCPSP formulation. Additionally, hereinafter, a renewable resource (or simply resource) is analogous to a machine, and a task is analogous to a job operation.

We now proceed to describe the elements of the proposed CP model.

A. INDICES, SETS, AND PARAMETERS

- t: Index for tasks
- r: Index for resources
- $NbTasks$: Number of tasks
- $NbRsrcs$: Number of resources
- $RsrcIds$: Set of resources $\{1, \ldots, NbRsrcs\}$
- A_r: Availability of the resource $r \in RsrcIds$

B. TUPLES

- *Task*: A tuple with the following fields: {***id***: identifier of the task, ***succs***: set of immediately successors of the task}
- *Mode*: A tuple with the following fields: {***taskId***: identifier of the task, ***id***: identifier to determine the mode type, ***pt***: processing time for the mode, $dmd_{RsrcIds}$: number of resources that belong to the set *RsrcIds* used per time period during the execution of the mode}

C. TUPLESETS

- *Tasks*: A tupleset that stores instances of the *Task* tuple
- *Modes*: A tupleset that stores instances of the *Mode* tuple

D. DECISION VARIABLES

- $task_{t \in Tasks}$: Interval variable between the start and the end of the *Tasks* t
- $mode_{m \in Modes}$: An optional interval variable of size pt if the $task_t$ is performed under the $mode_m$

E. CUMUL FUNCTION EXPRESSION

A cumul expression is a built-in function available in Optimization Programming Language (OPL) that can be used to model and track cumulative resource consumption. This feature is useful in the context of the FJSSP to avoid overlapping of tasks in a resource. Thus, we define the following function:

$$Usage_r = \sum_{m\ in\ Modes} pulse(mode_m, m.dmd_r), m.dmd_r > 0$$

where $Usage_r$ tracks the cumulative consumption over time of the resource $r \in RsrcIds$ through the sum of individual contributions of the optional interval variables $mode_m$.

F. MATHEMATICAL FORMULATION

$$\min\{\max_{t \in Tasks} \text{endOf}(task_t)\} \tag{1}$$

$$\text{subject to: } Usage_r \leq A_r \quad \forall r \in RsrcIds \tag{2}$$

$$alternative(task_t, mode_m) \quad \forall t \in Tasks, \forall m \in Modes, m.taskId = t.id \tag{3}$$

$$endBeforeStart\ (task_{t1}, task_{t2}) \quad \forall t1 \in Tasks, t2id \in t1.succs \tag{4}$$

The objective function (1) is used to minimize the makespan C_{max}, which is computed by the expression $max_{t \in Tasks} endOf(task_t)$ that corresponds to the task that finishes last. Constraints set (2) guarantee that resource availability is respected. In the context of the FJSSP, this ensures that each resource (machine) does not perform multiple tasks (job operations) simultaneously. Constraints set (3) ensures that each task is executed in exactly one mode. Finally, constraint sets (4) impose that precedence relationships between operations (route conditions) are respected.

IV. NUMERICAL EXPERIMENTS

All the computational experiments were performed on a Hewlett-Packard laptop having an Intel i7-10750H 2.60 GHz CPU with six cores and 16 GB of DDR4 RAM running at 2933 MHz. For solving the FJSSP instances we used the software IBM ILOG CP Optimizer 22.1.1 with the default Auto-search strategy (a combined search approach automatically controlled by the solver). We ran each instance once. We run our experiments using two types of instances. The *classical* FJSSP instances (eight sets) and the FJSSP instances with *sequencing flexibility* (two sets). The problem instances are detailed in Sections IV-A and IV-B. We compare our results against the best-known bounds in literature. All the details pertaining to these bounds are presented in Section IV-C. Additionally, quality indicators to analyze our results are presented in Section IV-D.

A. CLASSICAL FJSSP INSTANCES

In a classical FJSSP instance, the operations of a given job follow a linear order. This means that each operation has at most one predecessor and one successor. In this context, the following set of 271 instances of different sizes are widely used for testing the performance of different solution approaches:

- *BRdata*: 10 instances introduced by [21].
- *BCdata*: 21 instances from [22].
- *DPdata*: 18 instances proposed by [23].
- *KCdata*: 4 instances by [24].
- *Fdata*: 20 instances from [25].
- *HUdata*: 198 instances divided into three sets of 66 instances each (*HU-edata*, *HU-rdata*, and *HU-vdata*) with different degrees of flexibility (the average number of candidate machines for each operation) from [26].

TABLE 1. Description of the FJSSP benchmark instances.

Instance Set	Size	Jobs (n)	Machines (m)	Flexibility (F)
BRdata	10	{10,15,20}	[4,15]	[1.43, 4.10]
BCdata	21	{10,15}	[11,18]	[1.07, 1.30]
DPdata	18	{10,15,20}	{5,8,10}	[1.13, 5.02]
KCdata	4	{4,10,15}	{5,7,10}	[5.00, 10.00]
Fdata	20	[2,12]	[2,8]	[1.50, 2.67]
HU-edata	66	[6,30]	[4,15]	[1.12, 1.21]
HU-rdata	66	[6,30]	[4,15]	[1.88, 2.08]
HU-vdata	66	[6,30]	[4,15]	[2.09, 6.70]
DA-data	30	[4,12]	[5,10]	[2.67, 5.19]
Y-data	20	[4,17]	[7,26]	[2.53, 6.08]

An important aspect that characterizes an FJSSP instance is what is known as its *flexibility* (F), which corresponds to the average number of candidate machines for each operation [2]. In Table 1, we summarize the range of *flexibility* for each set, along with other benchmark features. For instance, within the *BRdata* set, the instance with the lowest *flexibility* is $MK08$ with $F = 1.43$ (322 modes in total for 225 job operations), whereas the instance with the highest *flexibility* is $MK02$ with $F = 4.10$ (238 modes in total for 58 job operations).

B. FJSSP INSTANCES WITH SEQUENCING FLEXIBILITY

Here we consider those FJSSP instances where an arbitrary directed acyclic graph defines the precedence relationships between operations. The 50 instances below were proposed by [15]:

- *Ydata*: 20 instances composed of two independent sequences of operations followed by an assembling operation.
- *DAdata*: 30 instances composed of jobs whose precedences are given by an arbitrary directed acyclic graph.

In Figure 1 we provide a graphical representation of the operations' precedence constraints of instance DAFJS23. In this context, each job operation is denoted by a node or vertex, and the precedence relations between them through a directed arc. For illustrative purposes, we filled with the same color the nodes that belong to a particular job (i.e. green for J_1). For instance, in the case of J_1, as soon as operation 1 is finished, it is possible to start both operations 2 and 8 simultaneously. To the extent that the availability of

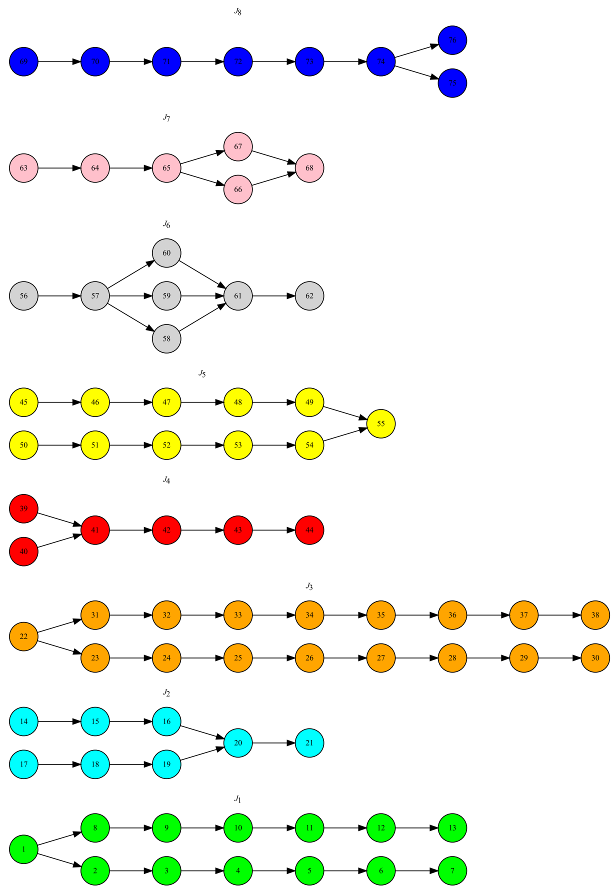

FIGURE 1. Graphical representation of the operations' precedence constraints of instance DAFJS23.

machines allows it. With the aim of clarity, other issues of the instances like the routing flexibility are not displayed.

C. BEST-KNOWN BOUNDS FOR THE FJSSP

There are numerous references that report the best-known lower bound ($Best_{LB}$) and the best-known upper bound ($Best_{UB}$) found until specific dates for the FJSSP instances. In this context, a recent survey conducted by [2] summarizes the best bounds achieved for the FJSSP in the last decades. This information is used as our primary point of reference. However, some of these bounds are not updated and/or present some typos in the article. Indeed, for the *Fdata* set, authors of [27] provide better bounds than the original reported by [2]. Also, for the *Ydata* and *DAdata*, we detected some errors in the reported bounds that are attributed to the CPLEX solver. Indeed, some best solutions in these last two sets of instances that are attributable to the Beam Search method of [28] are not considered.

Taking all these factors into consideration, we made efforts to update the $Best_{LB}$ and $Best_{UB}$ for each FJSSP instance based on the sets detailed in Section IV-A and Section IV-B. All details are listed in Tables 2- 9 and Tables 11- 12 in Section IV, pointing out the source of information.

TABLE 2. Test set *BRdata*.

Instance	n	m	o	p	F	LB	UB	Time	RPD_{LB}	RPD_{UB}
MK01	10	6	55	115	2.09	**40** [2]	**40** [2]	0.27	0.00%	0.00%
MK02	10	6	58	238	4.10	**26** [2]	**26** [2]	9.49	0.00%	0.00%
MK03	15	8	150	451	3.01	**204** [2]	**204** [2]	0.31	0.00%	0.00%
MK04	15	8	90	157	1.74	**60** [2]	**60** [2]	1.47	0.00%	0.00%
MK05	15	4	106	181	1.71	**172** [2]	**172** [2]	958.87	0.00%	0.00%
MK06	10	15	150	490	3.27	**57** [2]	**57** [2]	34.68	0.00%	0.00%
MK07	20	5	100	283	2.83	**139** [2]	**139** [2]	3.81	0.00%	0.00%
MK08	20	10	225	322	1.43	**523** [2]	**523** [2]	0.15	0.00%	0.00%
MK09	20	10	240	606	2.53	**307** [2]	**307** [2]	1.19	0.00%	0.00%
MK10	20	15	240	716	2.98	189 [2]	193 [2]	2,196.47	3.70%	1.55%

TABLE 3. Test set *BCdata*.

Instance	n	m	o	p	F	LB	UB	Time	RPD_{LB}	RPD_{UB}
mt10c1	10	11	100	110	1.10	**927** [2]	**927** [2]	3.35	0.00%	0.00%
mt10cc	10	12	100	120	1.20	**908** [2]	**908** [2]	1.01	0.00%	0.00%
mt10x	10	11	100	130	1.30	**918** [2]	**918** [2]	0.94	0.00%	0.00%
mt10xx	10	12	100	120	1.20	**918** [2]	**918** [2]	1.32	0.00%	0.00%
mt10xxx	10	13	100	130	1.30	**918** [2]	**918** [2]	4.30	0.00%	0.00%
mt10xy	10	12	100	110	1.10	**905** [2]	**905** [2]	0.95	0.00%	0.00%
mt10xyz	10	13	100	112	1.12	**847** [2]	**847** [2]	1.58	0.00%	0.00%
setb4c9	15	11	150	165	1.10	**914** [2]	**914** [2]	3.15	0.00%	0.00%
setb4cc	15	12	150	180	1.20	**907** [2]	**907** [2]	2.60	0.00%	0.00%
setb4x	15	11	150	195	1.30	**925** [2]	**925** [2]	6.33	0.00%	0.00%
setb4xx	15	12	150	180	1.20	**925** [2]	**925** [2]	10.07	0.00%	0.00%
setb4xxx	15	13	150	195	1.30	**925** [2]	**925** [2]	6.26	0.00%	0.00%
setb4xy	15	12	150	165	1.10	**910** [2]	**910** [2]	5.76	0.00%	0.00%
setb4xyz	15	13	150	180	1.20	**902** [2]	**902** [2]	2.64	0.00%	0.00%
seti5c12	15	16	225	240	1.07	**1169** [2]	**1169** [2]	18.20	0.00%	0.00%
seti5cc	15	17	225	255	1.13	**1135** [2]	**1135** [2]	36.22	0.00%	0.00%
seti5x	15	16	225	270	1.20	**1198** [2]	**1198** [2]	6.08	0.00%	0.00%
seti5xx	15	17	225	255	1.13	**1194** [2]	**1194** [2]	6.96	0.00%	0.00%
seti5xxx	15	18	225	270	1.20	**1194** [2]	**1194** [2]	10.29	0.00%	0.00%
seti5xy	15	17	225	240	1.07	**1135** [2]	**1135** [2]	43.67	0.00%	0.00%
seti5cxyz	15	18	225	255	1.13	**1125** [2]	**1125** [2]	25.49	0.00%	0.00%

D. MEASURE METRICS

To evaluate the performance of our approach, we consider three main measure metrics:

- Relative Percentage Deviation (RPD_{LB}) calculated over the $Best_{LB}$ documented:

$$RPD_{LB} = \frac{UB \; - \; Best_{LB}}{Best_{LB}} \tag{5}$$

- Relative Percentage Deviation (RPD_{UB}) calculated over the $Best_{UB}$ found in the literature:

$$RPD_{UB} = \frac{UB \; - \; Best_{UB}}{Best_{UB}} \tag{6}$$

- Best Upper Bounds Percentage ($\% \; BEST_{UB}$) attained over the number of instances of a benchmark set:

$$\% \; BEST_{UB} = \frac{Number \; of \; BEST_{UB}}{Number \; of \; instances} \tag{7}$$

Since the best-known bounds for the FJSSP were attained under different hardware, software, and time limit conditions, we consider that is difficult to make a direct comparison between these approaches and our proposal. Thus, in **https://github.com/yuraszeck/fjssp**, we report detailed results for different time limits at 600-second intervals (until reaching 3,600 seconds) to establish a balance between solution quality and the time allocated to our approach. In Table 10 and Table 13 we summarize our findings.

E. RESULTS FOR CLASSICAL FJSSP INSTANCES

Tables 2- 9 show the results obtained in the classical FJSSP instances. For each problem instance, we show the number

TABLE 4. Test set *DPdata*.

Instance	n	m	o	p	F	LB	UB	Time	RPD_{LB}	RPD_{UB}
01a	10	5	196	221	1.13	**2505** [2]	**2505** [2]	15.75	0.00%	0.00%
02a	10	5	196	332	1.69	**2228** [2]	**2228** [2]	112.60	0.22%	0.22%
03a	10	5	196	501	2.56	**2228** [2]	**2228** [2]	29.87	0.00%	0.00%
04a	10	5	196	221	1.13	**2503** [2]	**2503** [2]	27.60	0.00%	0.00%
05a	10	5	196	332	1.69	2192 [2]	2,203 [2]	460.43	1.09%	0.59%
06a	10	5	196	501	2.56	2163 [2]	2171 [2]	2,756.95	1.20%	0.83%
07a	15	8	293	364	1.24	2216 [2]	2,254 [2]	382.02	4.96%	3.19%
08a	15	8	293	710	2.42	**2061** [2]	**2061** [2]	237.01	0.19%	0.19%
09a	15	8	293	1182	4.03	**2061** [2]	**2061** [2]	1,048.60	0.05%	0.05%
10a	15	8	293	363	1.24	2212 [2]	2,241 [2]	314.49	5.42%	4.06%
11a	15	8	293	708	2.42	2018 [2]	2037 [2]	1,853.51	2.18%	1.23%
12a	15	8	293	1184	4.04	1969 [2]	1,984 [2]	3,497.16	2.89%	2.12%
13a	20	10	387	518	1.34	2197 [2]	2236 [2]	1,050.20	3.14%	1.34%
14a	20	10	387	1156	2.99	**2161** [2]	**2161** [2]	2,224.30	0.09%	0.09%
15a	20	10	387	1941	5.02	**2161** [2]	**2161** [2]	3,215.72	0.05%	0.05%
16a	20	10	387	518	1.34	2193 [2]	2,231 [2]	1,664.99	3.47%	1.70%
17a	20	10	387	1156	2.99	2088 [2]	2,105 [2]	3,507.33	2.83%	2.00%
18a	20	10	387	1938	5.01	2057 [2]	2,070 [2]	2,768.17	3.65%	3.00%

TABLE 5. Test set *KCdata*.

Instance	n	m	o	p	F	LB	UB	Time	RPD_{LB}	RPD_{UB}
Kacem1	4	5	12	60	5.00	**11** [2]	**11** [2]	0.02	0.00%	0.00%
Kacem2	10	7	29	203	7.00	**11** [2]	**11** [2]	0.07	0.00%	0.00%
Kacem3	10	10	30	300	10.00	**7** [2]	**7** [2]	0.06	0.00%	0.00%
Kacem4	15	10	56	560	10.00	**11** [2]	**11** [2]	0.80	0.00%	0.00%

TABLE 6. Test set *Fdata*.

Instance	n	m	o	p	F	LB	UB	Time	RPD_{LB}	RPD_{UB}
SFJS1	2	2	4	8	2.00	**66** [2]	**66** [2]	0.02	0.00%	0.00%
SFJS2	2	2	4	6	1.50	**107** [2]	**107** [2]	0.02	0.00%	0.00%
SFJS3	3	2	6	10	1.67	**221** [27]	**221** [2]	0.02	0.00%	0.00%
SFJS4	3	2	6	10	1.67	**355** [27]	**355** [2]	0.02	0.00%	0.00%
SFJS5	3	2	6	12	2.00	**119** [27]	**119** [2]	0.02	0.00%	0.00%
SFJS6	3	3	9	15	1.67	**320** [27]	**320** [2]	0.02	0.00%	0.00%
SFJS7	3	5	9	18	2.00	**397** [2]	**397** [2]	0.03	0.00%	0.00%
SFJS8	3	4	9	18	2.00	**253** [27]	**253** [2]	0.03	0.00%	0.00%
SFJS9	3	3	9	18	2.00	**210** [2]	**210** [2]	0.02	0.00%	0.00%
SFJS10	4	5	12	20	1.67	**516** [27]	**516** [2]	0.02	0.00%	0.00%
MFJS1	5	6	15	33	2.20	**468***	**468** [2]	0.07	0.00%	0.00%
MFJS2	5	7	15	39	2.60	**446***	**446** [2]	0.08	0.00%	0.00%
MFJS3	6	7	18	48	2.67	**466***	**466** [2]	0.08	0.00%	0.00%
MFJS4	7	7	21	56	2.67	**554***	**554** [2]	0.08	0.00%	0.00%
MFJS5	7	7	21	55	2.62	**514***	**514** [2]	0.08	0.00%	0.00%
MFJS6	8	7	24	62	2.58	**634***	**634** [27]	0.08	0.00%	0.00%
MFJS7	8	7	32	74	2.31	**879***	**879** [2]	0.26	0.00%	0.00%
MFJS8	9	8	36	86	2.39	**884***	**884** [2]	0.20	0.00%	0.00%
MFJS9	11	8	44	103	2.34	**1055***	**1055** [27]	19.40	0.00%	0.00%
MFJS10	12	8	48	112	2.33	**1196***	**1196** [27]	110.01	0.00%	0.00%

TABLE 7. Test set *HU-edata*.

Instance	n	m	o	p	F	LB	UB	Time	RPD_{LB}	RPD_{UB}
abz5	10	10	100	113	1.13	**1167** [2]	**1167** [2]	0.43	0.00%	0.00%
abz6	10	10	100	114	1.14	**925** [2]	**925** [2]	1.15	0.00%	0.00%
abz7	20	15	300	337	1.12	604 [2]	610 [2]	2,310.21	1.82%	0.82%
abz8	20	15	300	340	1.13	625 [2]	636 [2]	321.66	2.72%	0.94%
abz9	20	15	300	339	1.13	**644** [2]	**644** [2]	3,343.96	0.00%	0.00%
car1	11	5	55	66	1.20	**6176** [2]	**6176** [2]	0.88	0.00%	0.00%
car2	13	4	52	63	1.21	**6327** [2]	**6327** [2]	0.50	0.00%	0.00%
car3	12	5	60	71	1.18	**6856** [2]	**6856** [2]	0.45	0.00%	0.00%
car4	14	4	56	67	1.20	**7789** [2]	**7789** [2]	0.06	0.00%	0.00%
car5	10	6	60	70	1.17	**7229** [2]	**7229** [2]	0.22	0.00%	0.00%
car6	8	9	72	84	1.17	**7990** [2]	**7990** [2]	0.42	0.00%	0.00%
car7	7	7	49	58	1.18	**6123** [2]	**6123** [2]	0.23	0.00%	0.00%
car8	8	8	64	75	1.17	**7689** [2]	**7689** [2]	0.19	0.00%	0.00%
la01	10	5	50	60	1.20	**609** [2]	**609** [2]	0.07	0.00%	0.00%
la02	10	5	50	60	1.20	**655** [2]	**655** [2]	0.14	0.00%	0.00%
la03	10	5	50	59	1.18	**550** [2]	**550** [2]	0.47	0.00%	0.00%
la04	10	5	50	59	1.18	**568** [2]	**568** [2]	0.16	0.00%	0.00%
la05	10	5	50	60	1.20	**503** [2]	**503** [2]	0.07	0.00%	0.00%
la06	15	5	75	86	1.15	**833** [2]	**833** [2]	0.07	0.00%	0.00%
la07	15	5	75	86	1.15	**762** [2]	**762** [2]	0.21	0.00%	0.00%
la08	15	5	75	86	1.15	**845** [2]	**845** [2]	0.07	0.00%	0.00%
la09	15	5	75	86	1.15	**878** [2]	**878** [2]	0.15	0.00%	0.00%
la10	15	5	75	87	1.16	**866** [2]	**866** [2]	0.14	0.00%	0.00%
la11	20	5	100	113	1.13	**1103** [2]	**1103** [2]	0.19	0.00%	0.00%
la12	20	5	100	114	1.14	**960** [2]	**960** [2]	0.27	0.00%	0.00%
la13	20	5	100	114	1.14	**1053** [2]	**1053** [2]	0.14	0.00%	0.00%
la14	20	5	100	113	1.13	**1123** [2]	**1123** [2]	0.14	0.00%	0.00%
la15	20	5	100	113	1.13	**1111** [2]	**1111** [2]	0.84	0.00%	0.00%
la16	10	10	100	113	1.13	**892** [2]	**892** [2]	0.24	0.00%	0.00%
la17	10	10	100	113	1.13	**707** [2]	**707** [2]	0.35	0.00%	0.00%
la18	10	10	100	113	1.13	**842** [2]	**842** [2]	1.08	0.00%	0.00%
la19	10	10	100	113	1.13	**796** [2]	**796** [2]	0.47	0.00%	0.00%
la20	10	10	100	113	1.13	**857** [2]	**857** [2]	0.24	0.00%	0.00%
la21	15	10	150	173	1.15	**1009** [2]	**1009** [2]	69.59	0.00%	0.00%
la22	15	10	150	173	1.15	**880** [2]	**880** [2]	4.22	0.00%	0.00%
la23	15	10	150	171	1.14	**950** [2]	**950** [2]	11.59	0.00%	0.00%
la24	15	10	150	174	1.16	**908** [2]	**908** [2]	11.78	0.00%	0.00%
la25	15	10	150	174	1.16	**936** [2]	**936** [2]	7.31	0.00%	0.00%
la26	20	10	200	227	1.14	**1106** [2]	**1106** [2]	1,429.11	0.99%	0.99%
la27	20	10	200	227	1.14	**1181** [2]	**1181** [2]	140.90	0.00%	0.00%
la28	20	10	200	226	1.13	**1142** [2]	**1142** [2]	911.69	0.00%	0.00%
la29	20	10	200	227	1.14	**1107** [2]	**1107** [2]	1,234.74	0.00%	0.00%
la30	20	10	200	227	1.14	**1188** [2]	**1188** [2]	3,599.70	0.42%	0.42%
la31	30	10	300	341	1.14	**1532** [2]	**1532** [2]	173.40	0.59%	0.59%
la32	30	10	300	341	1.14	**1698** [2]	**1698** [2]	3.07	0.00%	0.00%
la33	30	10	300	339	1.13	**1547** [2]	**1547** [2]	17.38	0.00%	0.00%
la34	30	10	300	339	1.13	**1599** [2]	**1599** [2]	9.87	0.00%	0.00%
la35	30	10	300	339	1.13	**1736** [2]	**1736** [2]	1.48	0.00%	0.00%
la36	15	15	225	258	1.15	**1160** [2]	**1160** [2]	22.37	0.00%	0.00%
la37	15	15	225	258	1.15	**1397** [2]	**1397** [2]	1.08	0.00%	0.00%
la38	15	15	225	257	1.14	**1141** [2]	**1141** [2]	54.41	0.00%	0.00%
la39	15	15	225	257	1.14	**1184** [2]	**1184** [2]	7.36	0.00%	0.00%
la40	15	15	225	258	1.15	**1144** [2]	**1144** [2]	12.92	0.00%	0.00%
mt06	6	6	36	42	1.17	**55** [2]	**55** [2]	0.06	0.00%	0.00%
mt10	10	10	100	113	1.13	**871** [2]	**871** [2]	2.37	0.00%	0.00%
mt20	20	5	100	113	1.13	**1088** [2]	**1088** [2]	1.65	0.00%	0.00%
orb1	10	10	100	114	1.14	**977** [2]	**977** [2]	3.55	0.00%	0.00%
orb2	10	10	100	113	1.13	**865** [2]	**865** [2]	2.16	0.00%	0.00%
orb3	10	10	100	114	1.14	**951** [2]	**951** [2]	5.41	0.00%	0.00%
orb4	10	10	100	114	1.14	**984** [2]	**984** [2]	2.98	0.00%	0.00%
orb5	10	10	100	114	1.14	**842** [2]	**842** [2]	1.13	0.00%	0.00%
orb6	10	10	100	114	1.14	**958** [2]	**958** [2]	1.05	0.00%	0.00%
orb7	10	10	100	113	1.13	**389** [2]	**389** [2]	1.45	0.00%	0.00%
orb8	10	10	100	114	1.14	**894** [2]	**894** [2]	0.40	0.00%	0.00%
orb9	10	10	100	114	1.14	**933** [2]	**933** [2]	1.40	0.00%	0.00%
orb10	10	10	100	114	1.14	**933** [2]	**933** [2]	2.11	0.00%	0.00%

of jobs (n), the number of machines (m), the number of operations (o), the number of modes (p), the *flexibility* factor ($F = p/o$), the best-known lower and upper bounds, and the time required by our approach to finding its best relative percentage deviations for lower and upper bounds.

In Table 2 we can observe that for all but one instance in set *BRdata*, our approach was able to find the best lower and upper bounds from literature in low times. The hardest problem instance in this set was MK10, the largest one. In most other cases the approach was able to reach the lower and upper bounds in very low execution times (lower than 10 seconds). Only in two cases, instance MK06 and instance MK05 it required around 34 and 958 seconds respectively. These cases are not the largest ones, but possibly some structurally complex cases.

In Table 3 we can observe that in all instances in set *BCdata* our approach reached the best lower and upper bounds from literature. Larger times are required for larger problem instances, but the time required to solve these problem instances is at most around 43 seconds.

In Table 4 we can observe that only in three out of 18 instances our approach was able to find the best lower and upper bounds from literature. These were found for instances 01a, 03a, and 04a, all these considering 10 jobs and 5 machines. Moreover, these best bounds were found using at most 29.87 seconds. The hardest problem instances in this set were 07a and 10a, both considering 15 jobs and 8 machines. For these, the best bounds were at most 5.42% closer to the best bound from literature and were obtained using at most 400 seconds.

The results for instances in set *KCdata* are shown in Table 5. Regardless of the size of the problem instance, in this set our approach reached the best lower and upper bounds from literature in less than one second in all cases.

The results for instances in set *Fdata* are shown in Table 6. In a very similar way that for results in set *KCdata*, here

TABLE 8. Test set *HU-rdata*.

Instance	n	m	o	p	F	LB	UB	Time	RPD_{LB}	RPD_{UB}
abz5	10	10	100	198	1.98	**954** [2]	**954** [2]	1.70	0.00%	0.00%
abz6	10	10	100	194	1.94	**807** [2]	**807** [2]	0.28	0.00%	0.00%
abz7	20	15	300	587	1.96	493 [2]	522 [2]	438.46	8.72%	2.68%
abz8	20	15	300	579	1.93	507 [2]	535 [2]	177.24	8.88%	3.18%
abz9	20	15	300	584	1.95	517 [2]	536 [2]	1,578.96	6.00%	2.24%
car1	11	5	55	111	2.02	**5034** [2]	**5034** [2]	85.42	0.24%	0.24%
car2	13	4	52	100	1.92	**5985** [2]	**5985** [2]	1,498.07	0.00%	0.00%
car3	12	5	60	121	2.02	**5622** [2]	**5622** [2]	119.83	0.09%	0.09%
car4	14	4	56	110	1.96	**6514** [2]	**6514** [2]	32.97	0.00%	0.00%
car5	10	6	60	115	1.92	**5615** [2]	**5615** [2]	25.71	0.00%	0.00%
car6	8	9	72	140	1.94	**6147** [2]	**6147** [2]	0.14	0.00%	0.00%
car7	7	7	49	102	2.08	**4425** [2]	**4425** [2]	0.36	0.00%	0.00%
car8	8	8	64	123	1.92	**5692** [2]	**5692** [2]	0.70	0.00%	0.00%
la01	10	5	50	96	1.92	**570** [2]	**570** [2]	631.25	0.00%	0.00%
la02	10	5	50	94	1.88	**529** [2]	**529** [2]	74.49	0.00%	0.00%
la03	10	5	50	99	1.98	**477** [2]	**477** [2]	109.31	0.00%	0.00%
la04	10	5	50	101	2.02	**502** [2]	**502** [2]	0.55	0.00%	0.00%
la05	10	5	50	103	2.06	**457** [2]	**457** [2]	14.56	0.00%	0.00%
la06	15	5	75	141	1.88	**799** [2]	**799** [2]	9.66	0.00%	0.00%
la07	15	5	75	147	1.96	**749** [2]	**749** [2]	156.86	0.00%	0.00%
la08	15	5	75	145	1.93	**765** [2]	**765** [2]	9.16	0.00%	0.00%
la09	15	5	75	144	1.92	**853** [2]	**853** [2]	33.29	0.00%	0.00%
la10	15	5	75	147	1.96	**804** [2]	**804** [2]	878.80	0.00%	0.00%
la11	20	5	100	203	2.03	**1071** [2]	**1071** [2]	2.94	0.00%	0.00%
la12	20	5	100	199	1.99	**936** [2]	**936** [2]	3.88	0.00%	0.00%
la13	20	5	100	198	1.98	**1038** [2]	**1038** [2]	0.76	0.00%	0.00%
la14	20	5	100	197	1.97	**1070** [2]	**1070** [2]	5.33	0.00%	0.00%
la15	20	5	100	198	1.98	**1089** [2]	**1089** [2]	513.12	0.00%	0.00%
la16	10	10	100	201	2.01	**717** [2]	**717** [2]	0.38	0.00%	0.00%
la17	10	10	100	193	1.93	**646** [2]	**646** [2]	0.15	0.00%	0.00%
la18	10	10	100	199	1.99	**666** [2]	**666** [2]	0.27	0.00%	0.00%
la19	10	10	100	196	1.96	**700** [2]	**700** [2]	1.25	0.00%	0.00%
la20	10	10	100	199	1.99	**756** [2]	**756** [2]	0.33	0.00%	0.00%
la21	15	10	150	301	2.01	808 [2]	825 [2]	79.70	4.08%	1.94%
la22	15	10	150	306	2.04	741 [2]	753 [2]	126.77	2.70%	1.06%
la23	15	10	150	306	2.04	816 [2]	831 [2]	66.66	3.19%	1.32%
la24	15	10	150	297	1.98	775 [2]	795 [2]	800.26	3.74%	1.13%
la25	15	10	150	302	2.01	768 [2]	779 [2]	118.44	2.21%	0.77%
la26	20	10	200	391	1.96	1056 [2]	1057 [2]	778.68	0.95%	0.85%
la27	20	10	200	392	1.96	**1085** [2]	**1085** [2]	908.90	0.28%	0.28%
la28	20	10	200	402	2.01	1075 [2]	1076 [2]	2,192.46	0.37%	0.28%
la29	20	10	200	399	2.00	993 [2]	994 [2]	315.85	0.70%	0.60%
la30	20	10	200	392	1.96	1068 [2]	1071 [2]	810.66	1.31%	1.03%
la31	30	10	300	576	1.92	**1520** [2]	**1520** [2]	2,940.79	0.07%	0.07%
la32	30	10	300	585	1.95	**1657** [2]	**1657** [2]	1,815.48	0.06%	0.06%
la33	30	10	300	581	1.94	**1497** [2]	**1497** [2]	625.41	0.07%	0.07%
la34	30	10	300	584	1.95	**1535** [2]	**1535** [2]	356.82	0.07%	0.07%
la35	30	10	300	592	1.97	**1549** [2]	**1549** [2]	1,245.54	0.06%	0.06%
la36	15	15	225	439	1.95	**1023** [2]	**1023** [2]	9.93	0.00%	0.00%
la37	15	15	225	437	1.94	**1062** [2]	**1062** [2]	515.55	0.00%	0.00%
la38	15	15	225	444	1.97	**954** [2]	**954** [2]	8.57	0.00%	0.00%
la39	15	15	225	436	1.94	**1011** [2]	**1011** [2]	71.55	0.00%	0.00%
la40	15	15	225	441	1.96	**955** [2]	**955** [2]	1,386.81	0.00%	0.00%
mt06	6	6	36	74	2.06	**47** [2]	**47** [2]	0.04	0.00%	0.00%
mt10	10	10	100	196	1.96	**686** [2]	**686** [2]	0.92	0.00%	0.00%
mt20	20	5	100	194	1.94	**1022** [2]	**1022** [2]	6.07	0.00%	0.00%
orb1	10	10	100	196	1.96	**746** [2]	**746** [2]	1.14	0.00%	0.00%
orb2	10	10	100	197	1.97	**696** [2]	**696** [2]	1.00	0.00%	0.00%
orb3	10	10	100	195	1.95	**712** [2]	**712** [2]	1.01	0.00%	0.00%
orb4	10	10	100	194	1.94	**753** [2]	**753** [2]	0.68	0.00%	0.00%
orb5	10	10	100	198	1.98	**639** [2]	**639** [2]	1.28	0.00%	0.00%
orb6	10	10	100	196	1.96	**754** [2]	**754** [2]	0.64	0.00%	0.00%
orb7	10	10	100	197	1.97	**302** [2]	**302** [2]	1.94	0.00%	0.00%
orb8	10	10	100	195	1.95	**639** [2]	**639** [2]	1.78	0.00%	0.00%
orb9	10	10	100	194	1.94	**694** [2]	**694** [2]	0.21	0.00%	0.00%
orb10	10	10	100	198	1.98	**742** [2]	**742** [2]	1.23	0.00%	0.00%

TABLE 9. Test set *HU-vdata*.

Instance	n	m	o	p	F	LB	UB	Time	RPD_{LB}	RPD_{UB}
abz5	10	10	100	467	4.67	**859** [2]	**859** [2]	0.57	0.00%	0.00%
abz6	10	10	100	429	4.29	**742** [2]	**742** [2]	0.15	0.00%	0.00%
abz7	20	15	300	1951	6.50	**492** [2]	**492** [2]	952.22	0.20%	0.20%
abz8	20	15	300	1973	6.58	506 [2]	507 [2]	110.56	0.59%	0.39%
abz9	20	15	300	1994	6.65	**497** [2]	**497** [2]	878.37	0.40%	0.40%
car1	11	5	55	151	2.75	**5005** [2]	**5005** [2]	103.59	0.02%	0.02%
car2	13	4	52	119	2.29	**5929** [2]	**5929** [2]	82.57	0.00%	0.00%
car3	12	5	60	156	2.60	**5597** [2]	**5597** [2]	544.62	0.00%	0.00%
car4	14	4	56	117	2.09	**6514** [2]	**6514** [2]	823.69	0.00%	0.00%
car5	10	6	60	170	2.83	4909 [2]	4910 [2]	130.97	0.14%	0.12%
car6	8	9	72	294	4.08	**5486** [2]	**5486** [2]	0.08	0.00%	0.00%
car7	7	7	49	169	3.45	**4281** [2]	**4281** [2]	0.09	0.00%	0.00%
car8	8	8	64	254	3.97	**4613** [2]	**4613** [2]	0.16	0.00%	0.00%
la01	10	5	50	142	2.84	**570** [2]	**570** [2]	24.67	0.00%	0.00%
la02	10	5	50	134	2.68	**529** [2]	**529** [2]	4.03	0.00%	0.00%
la03	10	5	50	128	2.56	**477** [2]	**477** [2]	70.87	0.00%	0.00%
la04	10	5	50	119	2.38	**502** [2]	**502** [2]	1.36	0.00%	0.00%
la05	10	5	50	119	2.38	**457** [2]	**457** [2]	5.95	0.00%	0.00%
la06	15	5	75	182	2.43	**799** [2]	**799** [2]	19.91	0.00%	0.00%
la07	15	5	75	182	2.43	**749** [2]	**749** [2]	6.49	0.00%	0.00%
la08	15	5	75	194	2.59	**765** [2]	**765** [2]	164.27	0.00%	0.00%
la09	15	5	75	190	2.53	**853** [2]	**853** [2]	4.98	0.00%	0.00%
la10	15	5	75	200	2.67	**804** [2]	**804** [2]	4.33	0.00%	0.00%
la11	20	5	100	253	2.53	**1071** [2]	**1071** [2]	7.25	0.00%	0.00%
la12	20	5	100	248	2.48	**936** [2]	**936** [2]	2.25	0.00%	0.00%
la13	20	5	100	245	2.45	**1038** [2]	**1038** [2]	3.27	0.00%	0.00%
la14	20	5	100	244	2.44	**1070** [2]	**1070** [2]	1.19	0.00%	0.00%
la15	20	5	100	263	2.63	**1089** [2]	**1089** [2]	64.08	0.00%	0.00%
la16	10	10	100	470	4.70	**717** [2]	**717** [2]	0.20	0.00%	0.00%
la17	10	10	100	479	4.79	**646** [2]	**646** [2]	0.18	0.00%	0.00%
la18	10	10	100	479	4.79	**663** [2]	**663** [2]	0.17	0.00%	0.00%
la19	10	10	100	480	4.80	**617** [2]	**617** [2]	0.41	0.00%	0.00%
la20	10	10	100	487	4.87	**756** [2]	**756** [2]	0.16	0.00%	0.00%
la21	15	10	150	715	4.77	800 [2]	825 [2]	986.42	0.13%	0.13%
la22	15	10	150	677	4.51	**733** [2]	**733** [2]	238.07	0.27%	0.27%
la23	15	10	150	680	4.53	**809** [2]	**809** [2]	1,863.25	0.12%	0.12%
la24	15	10	150	727	4.85	**773** [2]	**773** [2]	185.15	0.13%	0.13%
la25	15	10	150	725	4.83	**751** [2]	**751** [2]	2,636.48	0.27%	0.27%
la26	20	10	200	917	4.59	**1052** [2]	**1052** [2]	2,998.60	0.00%	0.00%
la27	20	10	200	915	4.58	**1084** [2]	**1084** [2]	429.54	0.00%	0.00%
la28	20	10	200	897	4.49	**1069** [2]	**1069** [2]	1,465.37	0.00%	0.00%
la29	20	10	200	881	4.41	**993** [2]	**993** [2]	319.43	0.10%	0.10%
la30	20	10	200	930	4.65	**1068** [2]	**1068** [2]	1,276.78	0.09%	0.09%
la31	30	10	300	1380	4.60	**1520** [2]	**1520** [2]	2,409.32	0.00%	0.00%
la32	30	10	300	1362	4.54	**1657** [2]	**1657** [2]	615.48	0.06%	0.06%
la33	30	10	300	1354	4.51	**1497** [2]	**1497** [2]	208.41	0.07%	0.07%
la34	30	10	300	1387	4.62	**1535** [2]	**1535** [2]	408.09	0.00%	0.00%
la35	30	10	300	1394	4.65	**1549** [2]	**1549** [2]	369.99	0.06%	0.06%
la36	15	15	225	1507	6.70	**948** [2]	**948** [2]	1.02	0.00%	0.00%
la37	15	15	225	1492	6.63	**986** [2]	**986** [2]	0.90	0.00%	0.00%
la38	15	15	225	1479	6.57	**943** [2]	**943** [2]	0.83	0.00%	0.00%
la39	15	15	225	1470	6.53	**922** [2]	**922** [2]	1.41	0.00%	0.00%
la40	15	15	225	1458	6.48	**955** [2]	**955** [2]	0.91	0.00%	0.00%
mt06	6	6	36	103	2.86	**47** [2]	**47** [2]	0.04	0.00%	0.00%
mt10	10	10	100	448	4.48	**655** [2]	**655** [2]	0.16	0.00%	0.00%
mt20	20	5	100	262	2.62	**1022** [2]	**1022** [2]	77.44	0.00%	0.00%
orb1	10	10	100	446	4.46	**695** [2]	**695** [2]	0.28	0.00%	0.00%
orb2	10	10	100	440	4.40	**620** [2]	**620** [2]	0.53	0.00%	0.00%
orb3	10	10	100	467	4.67	**648** [2]	**648** [2]	0.28	0.00%	0.00%
orb4	10	10	100	471	4.71	**753** [2]	**753** [2]	0.22	0.00%	0.00%
orb5	10	10	100	477	4.77	**584** [2]	**584** [2]	0.27	0.00%	0.00%
orb6	10	10	100	478	4.78	**715** [2]	**715** [2]	0.21	0.00%	0.00%
orb7	10	10	100	456	4.56	**275** [2]	**275** [2]	0.33	0.00%	0.00%
orb8	10	10	100	455	4.55	**573** [2]	**573** [2]	0.18	0.00%	0.00%
orb9	10	10	100	447	4.47	**659** [2]	**659** [2]	0.23	0.00%	0.00%
orb10	10	10	100	451	4.51	**681** [2]	**681** [2]	0.22	0.00%	0.00%

most problem instances were solved instantaneously except the two largest ones that required around 19 and 110 seconds respectively. Moreover, our approach was able to find 10 new lower bounds. These values are accompanied by a *star* (*) in Table 6.

Tables 7, 8 and 9 show the results obtained when solving the *HUdata*. All these test sets consider 66 instances.

Table 7 show the results obtained in data set *Hu-edata*. Here, our approach was able to solve 61 out of 66 cases. Moreover, the solution times were lower and closer to 1 second in 39 cases. There are just a few cases that were solved in very high execution times. Also, cases abz7, abz8, and abz9 resulted very complex. The first ones were not solved, while instance abz9 was solved, but the approach required more than 3,000 seconds to reach the best solution. These instances are the largest in their category. On the other side, instances la26, la27, la28, la29, and la30 were also complex to solve. Only three of these cases were optimally solved but required more than 140 seconds. It is interesting here to note that these problem instances are not the largest in their corresponding set, hence, the complexity of these problem instances should come only from their structure.

Table 8 show the results obtained in data set *Hu-rdata*. Here, our approach was able to solve only 46 out of 66 cases. In 25 instances, the solving times were either lower than or approaching one second. There are a few cases that were solved with very high execution times. In 8 cases, the time was higher than 100 seconds and in two cases the time required was higher than 1,000 seconds. Complex instances for the approach in this set range from cases with 10 jobs and 5 machines to 20 jobs and 5 machines. Also, there are complex instances with 15 jobs and 15 machines.

TABLE 10. **Results for classical FJSSP instances presented in Section IV-A. The best upper bounds percentage (% $BEST_{UB}$) is reported for the time limit configuration of 3,600 seconds.**

Instance Set	% $BEST_{UB}$	TL: 600[s]		TL: 1,200[s]		TL: 1,800[s]		TL: 2,400[s]		TL: 3,000[s]		TL: 3,600[s]	
		RPD_{LB}	RPD_{UB}	RPD_{LB}	RPD_{UB}	RPD_{LB}	RPD_{UB}	RPD_{LB}	RPD_{UB}	RPD_{LB}	RPD_{UB}	RPD_{LB}	RPD_{UB}
BRdata	90%	0.53%	0.32%	0.48%	0.26%	0.48%	0.26%	0.37%	0.16%	0.37%	0.16%	0.37%	0.16%
BCdata	100%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
DPdata	16.67%	1.92%	1.32%	1.84%	1.24%	1.79%	1.19%	1.78%	1.18%	1.76%	1.16%	1.75%	1.15%
KCdata	100%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Fdata	100%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
HU-edata	92.42%	0.23%	0.19%	0.14%	0.10%	0.14%	0.10%	0.11%	0.07%	0.11%	0.06%	0.10%	0.06%
HU-rdata	69.69%	0.71%	0.32%	0.68%	0.29%	0.67%	0.28%	0.66%	0.27%	0.66%	0.27%	0.66%	0.27%
HU-vdata	77.27%	0.06%	0.06%	0.05%	0.05%	0.05%	0.05%	0.05%	0.04%	0.04%	0.04%	0.04%	0.04%

TABLE 11. **Test set *DAdata*.**

Instance	n	m	o	p	F	LB	UB	Time	RPD_{LB}	RPD_{UB}
DAFJS01	4	5	26	82	3.15	**257** [29]	**257** [29]	0.23	0.00%	0.00%
DAFJS02	4	5	25	79	3.16	**289** [29]	**289** [29]	0.21	0.00%	0.00%
DAFJS03	4	10	55	279	5.07	**576** [29]	**576** [29]	0.03	0.00%	0.00%
DAFJS04	4	10	43	220	5.12	**606** [29]	**606** [29]	0.03	0.00%	0.00%
DAFJS05	6	5	39	104	2.67	**384** [29]	**384** [29]	0.87	0.00%	0.00%
DAFJS06	6	5	44	136	3.09	**404** [29]	**404** [29]	42.86	0.00%	0.00%
DAFJS07	6	10	85	431	5.07	**505** [29]	**505** [29]	1.45	0.00%	0.00%
DAFJS08	6	10	85	403	4.74	**628** [29]	**628** [29]	0.04	0.00%	0.00%
DAFJS09	8	5	45	135	3.00	324 [29]	460 [29]	121.32	42.28%	0.22%
DAFJS10	8	5	58	168	2.90	337 [29]	516 [29]	743.71	54.01%	0.58%
DAFJS11	8	10	113	534	4.73	**658** [29]	**658** [29]	1.08	0.00%	0.00%
DAFJS12	8	10	117	603	5.15	530 [29]	588 [29]	3,395.92	11.70%	0.68%
DAFJS13	10	5	62	193	3.11	306 [29]	633*	931.83	106.86%	0.00%
DAFJS14	10	5	69	206	2.99	367 [29]	708 [29]	581.98	94.01%	0.56%
DAFJS15	10	10	120	595	4.96	512 [29]	626 [29]	209.02	23.83%	1.28%
DAFJS16	10	10	120	602	5.02	641 [29]	642 [29]	2,310.78	0.31%	0.16%
DAFJS17	12	5	82	246	3.00	309 [29]	771 [29]	517.67	150.16%	0.26%
DAFJS18	12	5	74	231	3.12	328 [29]	766 [29]	946.09	133.54%	0.00%
DAFJS19	8	7	70	283	4.04	**512** [29]	**512** [29]	7.03	0.00%	0.00%
DAFJS20	10	7	92	361	3.92	434 [29]	660 [29]	1,334.78	52.07%	0.00%
DAFJS21	12	7	107	425	3.97	504 [29]	755 [29]	2,348.36	50.60%	0.53%
DAFJS22	12	7	116	450	3.88	464 [29]	659 [29]	2,853.72	43.32%	0.91%
DAFJS23	8	9	76	367	4.83	450 [29]	460*	170.01	2.22%	0.00%
DAFJS24	8	9	92	463	5.03	476 [29]	533 [29]	177.22	12.82%	0.75%
DAFJS25	10	9	123	619	5.03	584 [29]	689 [29]	3,589.80	19.69%	1.45%
DAFJS26	10	9	119	606	5.09	565 [29]	681 [29]	2,999.11	22.48%	1.62%
DAFJS27	12	9	127	625	4.92	503 [29]	768 [29]	3,178.62	54.08%	0.91%
DAFJS28	8	10	91	457	5.02	**535** [29]	**535** [29]	5.11	0.00%	0.00%
DAFJS29	8	10	95	468	4.93	609 [29]	618 [29]	13.71	1.81%	0.32%
DAFJS30	10	10	98	509	5.19	467 [29]	519 [29]	2,987.10	11.35%	0.19%

TABLE 12. **Test set *Ydata*.**

Instance	n	m	o	p	F	LB	UB	Time	RPD_{LB}	RPD_{UB}
YFJS01	4	7	40	104	2.60	**773** [29]	**773** [29]	0.23	0.00%	0.00%
YFJS02	4	7	40	104	2.60	**825** [29]	**825** [29]	0.17	0.00%	0.00%
YFJS03	6	7	24	63	2.63	**347** [29]	**347** [29]	0.09	0.00%	0.00%
YFJS04	7	7	28	71	2.54	**390** [29]	**390** [29]	0.14	0.00%	0.00%
YFJS05	8	7	32	81	2.53	**445** [29]	**445** [29]	0.14	0.00%	0.00%
YFJS06	9	7	36	95	2.64	**446** [29]	**446** [29]	1.09	0.00%	0.00%
YFJS07	9	7	36	93	2.58	**444** [29]	**444** [29]	0.32	0.00%	0.00%
YFJS08	9	12	36	100	2.78	**353** [29]	**353** [29]	0.07	0.00%	0.00%
YFJS09	9	12	36	219	6.08	**242** [29]	**242** [29]	0.16	0.00%	0.00%
YFJS10	10	12	40	113	2.83	**399** [29]	**399** [29]	0.13	0.00%	0.00%
YFJS11	10	10	50	134	2.68	**526** [29]	**526** [29]	0.35	0.00%	0.00%
YFJS12	10	10	50	133	2.66	**512** [29]	**512** [29]	0.41	0.00%	0.00%
YFJS13	10	10	50	137	2.74	**405** [29]	**405** [29]	0.23	0.00%	0.00%
YFJS14	13	26	221	641	2.90	**1317** [29]	**1317** [29]	0.86	0.00%	0.00%
YFJS15	13	26	221	648	2.93	**1239** [29]	**1239** [29]	0.82	0.00%	0.00%
YFJS16	13	26	221	633	2.86	**1222** [29]	**1222** [29]	2.27	0.00%	0.00%
YFJS17	17	26	289	1328	4.60	**1133** [29]	**1133** [29]	2.84	0.00%	0.00%
YFJS18	17	26	289	1362	4.71	**1220** [29]	**1220** [29]	1.66	0.00%	0.00%
YFJS19	17	26	289	1347	4.66	**926** [29]	**926** [29]	18.37	0.00%	0.00%
YFJS20	17	26	289	1343	4.65	**968** [29]	**968** [29]	11.57	0.00%	0.00%

Finally, Table 9 show the results obtained in data set *Hu-vdata*. In this test set, our approach was able to solve 51 out of 66 cases. Here, again, the solution times were very low, lower or close to 1 second, in 29 cases. Also, there are a few cases that were solved with very high execution times. In 8 cases, the time was higher than 100 seconds, and in three cases the time required was higher than 1,000 seconds. Complex instances for the approach in this set range from cases with 12 jobs and 5 machines to 30 jobs and 10 machines. Again, neither the size nor the flexibility of the test cases determine the ability of the approach to solve these problems and it could be mostly related to specific features of the problems being solved.

All in all, in Table 10 we summarize our findings for the classical FJSSP instances:

- We proved optimality in all instances from sets *BCdata*, *KCdata*, and *Fdata* within the first 600 seconds. Indeed, the average time needed to reach the optimal solution in these 55 instances was 7.3 seconds. Moreover, in the *Fdata* set we contribute to the literature with 10 new lower bounds.
- For the *BRdata* set we reached the $BEST_{UB}$ in 9 out of 10 instances. In only instance *MK*10 we could not reach the $BEST_{UB}$ within the time limit, however, the results in terms of RPD_{LB} and RPD_{UB} ($\leq$ 0.53% on average) seem satisfactory.
- Overall, the *DPdata* set was the most challenging, where we reached 3 out of 18 $BEST_{UB}$ with both RPD values less than or equal to 1.92% on average. It is worth emphasizing that, to the author's knowledge, optimality has been proven in only 8 out of 18 instances within this benchmark (44.44%).
- For all the 3 sets of *HUdata* both RPD_{LB} and RPD_{UB} present consistent results with values lower than or equal to 0.71% on average.
- Although it is natural to expect improvements in the results as the computational time limit increases, it can be noticed the competitiveness of our approach even for relatively short execution times.

F. RESULTS FOR FJSSP INSTANCES WITH SEQUENCING FLEXIBILITY

In Table 11 we present detailed results obtained for *DAdata* set. Only 11 cases were optimally solved in this test set. Anyway, two new upper bounds were found in this test set. These are marked with a *star* (*) in Table 11. Execution times were very low in all these cases. On the other side, specially high differences in lower bounds were found in these cases, higher than 10% in 16 cases and higher than 100% in three cases.

Table 12 show the detailed results obtained for set Ydata. All cases were optimally solved in this test set. Execution times were lower than and close to one second in 15 cases, and lower than 19 seconds in all cases. Large differences in lower

TABLE 13. Results for FJSSP instances with sequencing flexibility presented in Section IV-B. The best upper bounds percentage (% $BEST_{UB}$) is reported for the time limit configuration of 3,600 seconds.

Instance Set	% $BEST_{UB}$	TL: 600[s]		TL: 1,200[s]		TL: 1,800[s]		TL: 2,400[s]		TL: 3,000[s]		TL: 3,600[s]	
		RPD_{LB}	RPD_{UB}	RPD_{LB}	RPD_{UB}	RPD_{LB}	RPD_{UB}	RPD_{LB}	RPD_{UB}	RPD_{LB}	RPD_{UB}	RPD_{LB}	RPD_{UB}
DAdata	43.33%	29.94%	0.61%	29.80%	0.51%	29.74%	0.47%	29.70%	0.44%	29.63%	0.38%	29.57%	0.33%
Ydata	100%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

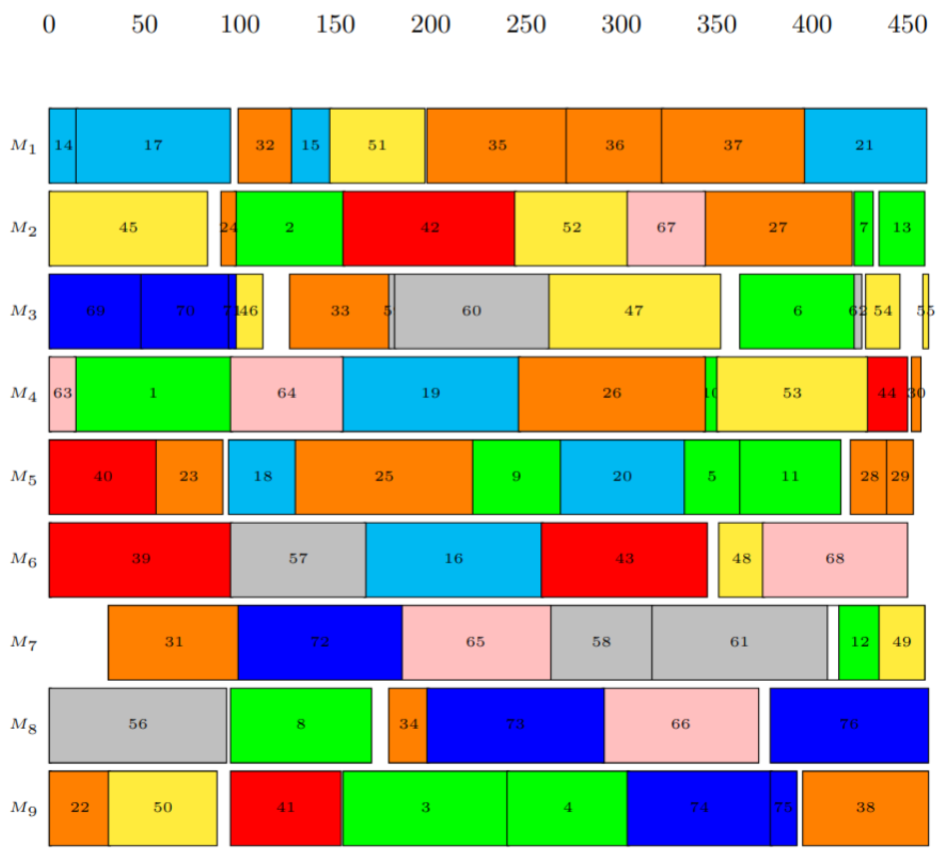

FIGURE 2. A Gantt chart for a feasible solution of instance *DAFJS*23 with C_{max} = 460.

bounds were found in these cases, higher than 10% in 16 cases and higher than 100% in three cases. Higher execution times were required in larger problem instances in this test set.

Summarized results for FJSSP instances with sequencing flexibility, grouped by benchmark set, are provided in Table 13. The attained results allow us to make the following conclusions:

- We found the optimal solution in all *Ydata* instances in an average computational time of 2.1 seconds.
- For the *DAdata* set, we reach the $BEST_{UB}$ in 13 out of 30 instances. In this context, we contribute with two new $BEST_{UB}$ for the instances *DAFJS*13 and *DAFJS*23. In Figure 2 we provide a Gantt Chart for the instance *DAFJS*23 with a $C_{max} = 460$ based on the precedence constraints previously shown in Figure 2. In Table 14 we present the details of the attained solution. Finally, although the average RPD_{LB} values are higher than those obtained for classical FJSSP instances, the average RPD_{UB} is competitive, with values averaging lower than or equal to 0.61%.

V. CONCLUSION AND FUTURE WORKS

In this paper, we proposed a CP formulation of the MMRCPSP to solve the FJSSP under the makespan minimization criterion. The FJSSP is an NP-hard problem that extends the well-known JSSP by allowing each operation to be executed on any machine within a specified subset of the available machines.

To evaluate our approach, we conducted computational experiments comparing our results to the best-known solutions in the literature. To do this we consider two categories of FJSSP instances: 271 classical instances where the operations of a given job follow a linear order, and 50 instances with sequencing flexibility. In these cases, the precedence relationships between operations are defined by a directed acyclic graph. In this context, we reach a competitive

TABLE 14. Tabulated results for a feasible solution of instance *DAFJS*23 with $C_{max} = 460$.

Task	Job	Mode	Machine	Starting	Finishing
1	J_1	1	4	13	95
2	J_1	3	2	97	154
3	J_1	1	9	154	240
4	J_1	1	9	240	303
5	J_1	1	5	332	362
6	J_1	1	3	362	422
7	J_1	1	2	422	431
8	J_1	3	8	95	168
9	J_1	1	5	221	268
10	J_1	1	4	343	350
11	J_1	5	5	362	414
12	J_1	1	7	414	435
13	J_1	2	2	435	458
14	J_2	1	1	0	14
15	J_2	5	1	126	147
16	J_2	4	6	165	258
17	J_2	1	1	14	94
18	J_2	1	5	94	129
19	J_2	3	4	153	246
20	J_2	2	5	268	332
21	J_2	1	1	395	459
22	J_3	1	9	0	31
23	J_3	3	5	55	90
24	J_3	1	2	90	97
25	J_3	1	5	129	221
26	J_3	3	4	246	343
27	J_3	1	2	343	420
28	J_3	3	5	420	439
29	J_3	3	5	439	452
30	J_3	1	4	452	456
31	J_3	6	7	31	99
32	J_3	1	1	99	126
33	J_3	1	3	126	178
34	J_3	4	8	178	198
35	J_3	3	1	198	271
36	J_3	3	1	271	321
37	J_3	1	1	321	395
38	J_3	1	9	395	460
39	J_4	1	6	0	95
40	J_4	5	5	0	55
41	J_4	1	9	95	152
42	J_4	2	2	154	244
43	J_4	3	6	258	344
44	J_4	1	4	428	449
45	J_5	1	2	0	82
46	J_5	1	3	97	111
47	J_5	2	3	261	351
48	J_5	1	6	351	374
49	J_5	5	7	435	458
50	J_5	5	9	31	87
51	J_5	1	1	147	196
52	J_5	1	2	244	303
53	J_5	2	4	350	428
54	J_5	1	3	428	445
55	J_5	6	3	458	460
56	J_6	4	8	0	92
57	J_6	1	6	95	165
58	J_6	1	7	262	316
59	J_6	2	3	178	181
60	J_6	1	3	181	261
61	J_6	4	7	316	407
62	J_6	2	3	422	425
63	J_7	1	4	0	13
64	J_7	3	4	95	153
65	J_7	1	7	184	262
66	J_7	1	8	290	371
67	J_7	6	2	303	343
68	J_7	2	6	374	449
69	J_8	1	3	0	48
70	J_8	1	3	48	94
71	J_8	3	3	94	97
72	J_8	1	7	99	184
73	J_8	2	8	198	290
74	J_8	1	9	303	378
75	J_8	4	9	378	391
76	J_8	2	8	378	460

average RPD_{UB} of 0.18% over the 321 instances considered, contributing with ten new lower bounds and two new upper bounds to the literature.

In terms of future lines of research, other operational constraints and optimization criteria could be considered for the FJSSP. Additionally, given the generality of our MMRCPSP approach, it can be readily adapted to address other closely related problems subsumed by the FJSSP, such as the flexible flow shop scheduling problem (FFSSP) and the flexible open shop scheduling problem (FOSSP).

REFERENCES

[1] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, no. 2, pp. 117–129, May 1976.

[2] S. Dauzère-Pérès, J. Ding, L. Shen, and K. Tamssaouet, "The flexible job shop scheduling problem: A review," *Eur. J. Oper. Res.*, May 2023. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S037722172300382X and https://service.elsevier.com/app/answers/detail/a_id/22801/supporthub/sciencedirect/

[3] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and heuristic in deterministic sequencing and scheduling: A survey," *Ann. Discrete Math.*, vol. 5, pp. 287–326, 1979.

[4] A. Ahmeti and N. Musliu, "Hybridizing constraint programming and meta-heuristics for multi-mode resource-constrained multiple projects scheduling problem," in *Proc. 13th Int. Conf. Pract. Theory Automated Timetablin*, vol. 1, 2021, pp. 188–206.

[5] R. Pellerin, N. Perrier, and F. Berthaut, "A survey of hybrid metaheuristics for the resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 280, no. 2, pp. 395–416, Jan. 2020.

[6] P. Schworm, X. Wu, M. Glatt, and J. C. Aurich, "Solving flexible job shop scheduling problems in manufacturing with quantum annealing," *Prod. Eng.*, vol. 17, no. 1, pp. 105–115, Feb. 2023.

[7] X. Kong, Y. Yao, W. Yang, Z. Yang, and J. Su, "Solving the flexible job shop scheduling problem using a discrete improved grey wolf optimization algorithm," *Machines*, vol. 10, no. 11, p. 1100, Nov. 2022.

[8] K. Tamssaouet and S. Dauzère-Pérès, "A general efficient neighborhood structure framework for the job-shop and flexible job-shop scheduling problems," *Eur. J. Oper. Res.*, vol. 311, no. 2, pp. 455–471, Dec. 2023.

[9] V. Boyer, J. Vallikavungal, X. C. Rodríguez, and M. A. Salazar-Aguilar, "The generalized flexible job shop scheduling problem," *Comput. Ind. Eng.*, vol. 160, Oct. 2021, Art. no. 107542.

[10] A. M. Ham and E. Cakici, "Flexible job shop scheduling problem with parallel batch processing machines: MIP and CP approaches," *Comput. Ind. Eng.*, vol. 102, pp. 160–165, Dec. 2016.

[11] G. M. Kopanos, C. A. Méndez, and L. Puigjaner, "MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry," *Eur. J. Oper. Res.*, vol. 207, no. 2, pp. 644–655, Dec. 2010.

[12] G. A. Kasapidis, S. Dauzère-Pérès, D. C. Paraskevopoulos, P. P. Repoussis, and C. D. Tarantilis, "On the multiresource flexible job-shop scheduling problem with arbitrary precedence graphs," *Prod. Oper. Manage.*, vol. 32, no. 7, pp. 2322–2330, Jul. 2023.

[13] F. Yuraszeck, G. Mejía, J. Pereira, and M. Vilà, "A novel constraint programming decomposition approach for the total flow time fixed group shop scheduling problem," *Mathematics*, vol. 10, no. 3, p. 329, Jan. 2022.

[14] G. Mejía and F. Yuraszeck, "A self-tuning variable neighborhood search algorithm and an effective decoding scheme for open shop scheduling problems with travel/setup times," *Eur. J. Oper. Res.*, vol. 285, no. 2, pp. 484–496, Sep. 2020.

[15] E. G. Birgin, P. Feofiloff, C. G. Fernandes, E. L. de Melo, M. T. I. Oshiro, and D. P. Ronconi, "A MILP model for an extended version of the flexible job shop problem," *Optim. Lett.*, vol. 8, no. 4, pp. 1417–1431, Apr. 2014.

[16] X. Chu, S. Li, F. Gao, C. Cui, F. Pfeiffer, and J. Cui, "A data-driven meta-learning recommendation model for multi-mode resource constrained project scheduling problem," *Comput. Oper. Res.*, vol. 157, Sep. 2023, Art. no. 106290.

[17] J. Blazewicz, J. K. Lenstra, and A. H. G. R. Kan, "Scheduling subject to resource constraints: Classification and complexity," *Discrete Appl. Math.*, vol. 5, no. 1, pp. 11–24, Jan. 1983.

[18] R. Nemati-Lafmejani, H. Davari-Ardakani, and H. Najafzad, "Multi-mode resource constrained project scheduling and contractor selection: Mathematical formulation and metaheuristic algorithms," *Appl. Soft Comput.*, vol. 81, Aug. 2019, Art. no. 105533.

[19] S. Elloumi and P. Fortemps, "A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 205, no. 1, pp. 31–41, Aug. 2010.

[20] F. Zaman, S. Elsayed, R. Sarker, and D. Essam, "Hybrid evolutionary algorithm for large-scale project scheduling problems," *Comput. Ind. Eng.*, vol. 146, no. May, May 2020, Art. no. 106567.

[21] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Ann. Oper. Res.*, vol. 41, no. 3, pp. 157–183, Sep. 1993.

[22] C. J. B. Barnes, "Flexible job shop scheduling by tabu search," Univ. Texas, Austin, TX, USA, Tech. Rep., 1996.

[23] S. Dauzère-Pérès and J. Paulli, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search," *Ann. Oper. Res.*, vol. 70, pp. 281–306, Apr. 1997.

[24] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Trans. Syst., Man Cybern. C, Appl. Rev.*, vol. 32, no. 1, pp. 1–13, Feb. 2002.

[25] P. Fattahi, M. S. Mehrabad, and F. Jolai, "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems," *J. Intell. Manuf.*, vol. 18, no. 3, pp. 331–342, Jul. 2007.

[26] J. Hurink, B. Jurisch, and M. Thole, "Tabu search for the job-shop scheduling problem with multi-purpose machines," *OR Spektrum*, vol. 15, no. 4, pp. 205–215, Dec. 1994.

[27] Y. Yuan, H. Xu, and J. Yang, "A hybrid harmony search algorithm for the flexible job shop scheduling problem," *Appl. Soft Comput.*, vol. 13, no. 7, pp. 3259–3272, Jul. 2013.

[28] E. G. Birgin, J. E. Ferreira, and D. P. Ronconi, "List scheduling and beam search methods for the flexible job shop scheduling problem with sequencing flexibility," *Eur. J. Oper. Res.*, vol. 247, no. 2, pp. 421–440, Dec. 2015.

[29] G. A. Kasapidis, D. C. Paraskevopoulos, P. P. Repoussis, and C. D. Tarantilis, "Flexible job shop scheduling problems with arbitrary precedence graphs," *Prod. Oper. Manage.*, vol. 30, no. 11, pp. 4044–4068, Nov. 2021.

FRANCISCO YURASZECK received the Ph.D. degree in industrial engineering from Pontificia Universidad Católica de Valparaíso, Chile, in 2022. He is currently a Research Professor with the Department of Engineering Sciences, Universidad Andrés Bello, Chile. His research interests include shop scheduling, metaheuristics, and exact methods of optimization, such as constraint programming and mixed-integer linear programming.

ELIZABETH MONTERO (Member, IEEE) received the Ph.D. degree from the University of Nice Sophia Antipolis, France. She is currently an Academician of computing science with Universidad Técnica Federico Santa María, Chile. She has published over 50 technical papers in high-level heuristic search conferences, such as GECCO, PPSN, and CEC, and over 20 Q1 and Q2 WoS papers. Her research interests include the foundations and application of heuristic search methods, parameter setting problems, automatic algorithm generation, and their applications to combinatorial optimization problems.

DARÍO CANUT-DE-BON received the B.S. degree in electronic engineering and the M.Eng. degree in industrial engineering from Pontificia Universidad Católica de Valparaíso, Chile, in 2003 and 2012, respectively. From 2004 to 2011, he was a Research and Development Engineer with the Directorate of Programs, Research and Development, Chilean Navy, where his research interests include digital electronics mainly applied to underwater acoustics. Since 2011, he has been a Naval Operations Analyst with Anti-Submarine Warfare. Since 2019, he has been a Professor with the Engineering School, Universidad Andrés Bello, Viña del Mar, Chile.

NICOLÁS CUNEO is currently pursuing the bachelor's degree in industrial civil engineering with the Faculty of Engineering, Universidad Andrés Bello, Chile. He has recently begun working on his degree project on topics related to production scheduling, under the supervision of Prof. Francisco Yuraszeck.

MAXIMILIANO ROJEL is currently pursuing the bachelor's degree in industrial civil engineering with the Faculty of Engineering, Universidad Andrés Bello, Chile. He has recently initiated his degree project, focusing on subjects pertaining to operations research and production scheduling, under the supervision of Prof. Francisco Yuraszeck.