INFORMS JOURNAL ON COMPUTING

Mixed-Integer Programming Versus Constraint Programming for Scheduling Problems: New Results and Outlook

Journal:	*INFORMS Journal on Computing*
Manuscript ID	JOC-2020-12-OA-320
Manuscript Type:	Original Article
Date Submitted by the Author:	12-Dec-2020
Complete List of Authors:	Naderi, Bahman; Ruiz, Rubén; Universitat Politecnica de Valencia, Roshanaei, Vahid; University of Toronto, Department of Operations Management & Statistics
Keywords:	Shop Scheduling, Flow Shop, Job Shop, Parallel Machines, Benchmarks

SCHOLARONE™
Manuscripts

Suitability for IJOC

This paper fits the mission and scope of IJOC in general as it explores the performance of modern Constraint Programming (CP) and Mixed Integer Linear (MILP) solvers and models for several well-known scheduling problems. In order to do so, extensive and exhaustive computational experiments are carried out, exploring the bridges between operations research and computing.

Suitability for this IJOC Area

This paper fits the mission and scope of IJOC in the Modeling: Methods & Analysis area as it shows how advances in Constraint Programming off-the-shelf exact solvers are better than traditional Mixed Integer Linear Programming (MILP) solvers for scheduling problems. This goes completely against the common belief that MILP solvers are the best off-the-shelf exact tecniques for scheduling problems. Many authors in the literature benchmark their exact techniques (often branch and bound methods) against MILP solvers and models.
This paper shows that this is no longer true and CP moldes should be used instead.

Mixed-Integer Programming Versus Constraint Programming for Scheduling Problems: New Results and Outlook

Bahman Naderi[a], Rubén Ruiz[b,*], Vahid Roshanaei[c]

[a] *Department of Mechanical, Automotive, and Materials, Faculty of Engineering, University of Windsor, Windsor, Canada*

[b] *Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, 46021, València, Spain*

[c] *Department of Operations Management & Statistics, Rotman School of Management, University of Toronto, Toronto, Canada*

Abstract

Constraint Programming (CP) has been given a new lease of life after new CP-based procedures have been incorporated into state-of-the-art solvers, most notably the CP Optimizer from IBM. Classical CP solvers were only capable of finding integer solutions without being able to provide any bounds for these solutions. There were also problems when trying to prove optimality. New versions, however, provide bounds and optimality guarantees, effectively making CP a viable alternative to more traditional mixed-integer programming (MILP) models and solvers. We capitalize on these developments and conduct an extensive computational evaluation of MILP and CP models on 11 select scheduling problems. We carefully chose these 11 problems to represent a wide variety of scheduling problems that occur in different service and manufacturing settings. We also consider basic and well-studied simplified problems. These scheduling settings range from pure sequencing (e.g., flow shop and open shop) or joint assignment-sequencing (i.e., distributed flow shop and hybrid flow shop) to pure assignment (i.e., parallel machine) scheduling problems. We present MILP and CP models for each variant of these problems and evaluate their performance over 17 relevant and standard benchmarks that we identified in the literature. The computational campaign encompasses almost 38,000 experiments and evaluates the MILP and CP models along six dimensions of problem characteristics, objective function, decision variables, input parameters, time limit and quality of bounds. We establish the areas that each one of these models excels and recognize their conceivable reasons. The obtained results indicate that CP sets new limits with regard to the maximum problem size that can be solved using exact techniques.

Keywords: Shop scheduling, flow shop, Job shop, Open shop, Parallel machines, Benchmarks, Constraint programming, Mixed integer programming.

*Corresponding author. Tel: +34 96 387 70 00. Ext: 74946. Fax: +34 96 387 74 99

Email addresses: bahman_naderi62@yahoo.com (Bahman Naderi), rruiz@eio.upv.es (Rubén Ruiz), v.roshanaei@utotonto.ca (Vahid Roshanaei)

1. Introduction

Mathematical programming is, by far, the most widely employed approach for modeling scheduling problems. Mathematical programs, often in the form of mixed integer linear programming (MILP) models, are the first choice for researchers as the rich literature confirms (Naderi and Ruiz, 2010; Stafford, 1988; Stafford et al., 2005; Pan, 1997; Tseng et al., 2004; Tseng and Stafford, 2008; Wilson, 1989; Manne, 1960; Wagner, 1959; Roshanaei, 2012). A much less studied alternative is Constraint Programming or Constraint Propagation (CP). Recently, the effectiveness of CP models has also been recognized by researchers (Samarghandi and Behroozi, 2017; Ku and Beck, 2016; Malapert et al., 2012; Laborie et al., 2018; Bukchin and Raviv, 2018). Despite this recent trend towards CP, MILP models are still actively studied (Androutsopoulos et al., 2020; Unsal and Oguz, 2019; Valicka et al., 2019).

As far as scheduling optimization is concerned, there are two main CP approaches for formulating a scheduling problem: classical ones versus more modern CP solvers like IBM ILOG CPLEX Optimization Studio CP Optimizer. The classical CP approach uses integer variables and global constraints such as disjunctive and/or cumulative while the CP Optimizer approach uses interval variables and interval-specific global functions (CPOptimizer, 2017; Laborie, 2015). Laborie et al. (2018) discuss the advantages of the CP Optimizer approach over classical CP for scheduling problems. Samarghandi and Behroozi (2017) Ku and Beck (2016), and Malapert et al. (2012) use the classical CP and Laborie et al. (2018) use the CP Optimizer.

CP models and techniques are well known for producing high quality integer solutions. However, CP techniques have had, until relatively recent times, one major drawback: a lack of optimality proofs in most cases. Furthermore, the lack of bounding mechanisms in the tree search algorithms employed inside CP methods results in a lack of any bounds on the quality of their integer solutions. As a result, traditional CP techniques can only be considered heuristics in most cases. However, more modern CP approaches, like the aforementioned CP Optimizer (starting with CPLEX 12.2) include automatic search procedures that are complete, i.e., given an optimization problem, CP will either find a proven optimal solution or will prove that the problem has no feasible solution. Additionally, bounds are provided and optimality gaps for any integer solution found are given. As such, CP models can be viewed as complete models and their performance can therefore be compared with mixed-integer programs (MILPs) that have been the primary modelling tool for the exact solution to scheduling problems in the literature.

As of late, the interest in the applications of CP for scheduling problems has been growing rapidly. Malapert et al. (2012) solve open shop scheduling problems with a CP-based algorithm and show that this algorithm provides state-of-the-art results. Ku and Beck (2016) solve job shop scheduling problems using a classical CP approach. Their results show that MILP performs similarly to CP for problems of moderate sizes. However, CP is superior to MILP for the larger instances. Samarghandi and Behroozi (2017) study a no-wait flow shop and develop two CP models, based on the classical CP approach, along with three MILP models. They conclude that one of the MILP models outperforms the other models including two of the CP models they developed. Laborie et al. (2018) develop a CP model based on the CP Optimizer approach that produces state-of-the-art results for job shop and resource-constrained project scheduling problems. Gedik et al. (2018) proposed CP-based optimization methods for the unrelated parallel machines scheduling problem with makespan

criterion. Meng et al. (2020) also developed a CP model for distributed job shop scheduling and compare it with MILP models.

CP-based models and algorithms have also been applied in other combinatorial optimization problems such as assembly line balancing (Bukchin and Raviv, 2018), preventive signaling maintenance crew scheduling problems (Pour et al., 2018), cooperative flight departures (Schefers et al., 2018), operating theatres (Wang et al., 2015; Doulabi et al., 2016), resource availability cost problems (Kreter et al., 2018) and container scheduling (Qin et al., 2020).

This paper studies different variants of scheduling problems including parallel machines, flow shops, job shops and open shops as they are the most prevalent shop scheduling problems in the literature. We consider makespan as the most widely considered scheduling objective. We exclude single machine problems from this study as very effective exact approaches exist for most regular scheduling objectives. The extant literature on flow shop scheduling is very rich and many different variants have been studied. To gain a deeper understanding of how these variants contribute to the complexity of flow shops and to the performance of CP and MILP models, we study the following eight flow shop variants while also considering different optimization objectives. 1) permutation flow shops, 2) non-permutation flow shops, 3) hybrid flow shops, 4) distributed flow shops, 5) total completion time permutation flow shops, 6) total tardiness permutation flow shops, 7) no-wait flow shops and 8) sequence-dependent setup times permutation flow shops.

The choice of all these problems is motivated by many factors. These problems provide opportunities to analyze performance from different angles. We consider the permutation flow shop to minimize makespan as a basic problem that has been studied extensively in the literature and for which many different mathematical models and exact approaches have been proposed. The other problems are different in either the objective function, problem characteristics or problem decisions. We consider the permutation flow shop with total completion time and total tardiness minimization so as to study how a change in the objective function affects performance. As the problem characteristics and decisions are the same, we are able to analyze the impact of different objectives on computational time and on solution quality of these models. As for the problem characteristics, we consider the no-wait restriction and sequence-dependent setup times on top of the basic problem. These characteristics do not change the problem decisions (the solution representation can still be a simple permutation) so we are able to analyze the impact of these characteristics on the performance while the objectives and decisions remain unaltered.

Scheduling problems commonly entail two decisions: assignment and sequencing. Some of the problems we study can be considered as pure sequencing while others are pure assignment problems. For example, the open shop is a pure sequencing problem with no additional precedence constraints. The flow and job shops are sequencing problems with precedence constraints. The permutation flow shop needs one job sequence while the job shop requires one job sequence for each machine, similar to non-permutation flow shops. Therefore, they have more sequencing decisions to make when compared to the permutation flow shop. The distributed and hybrid flow shops include both sequencing and assignment dimensions. More specifically, the distributed flow shop requires one assignment for each job while the hybrid flow shop needs one assignment for each job at each stage, which means more assignments for each job. Finally, the parallel-machine scheduling problem is a pure assignment problem, i.e., the sequencing decision does not impact the objective function (makespan). With this

set of selected problems, we are able to comprehensively evaluate the performance and limits of both CP and MILP models.

This study makes several contributions to the literature:

- We exploit the capabilities of modern CP solvers (CP Optimizer version 12.8) and conduct a comprehensive comparative evaluation between the performance of the best known MILP and CP models for 11 select shop scheduling problems. For each of the shop scheduling problems stated earlier, we present one MILP and one CP model. Generally, we adapt the best MILP models from the literature, for example, the models presented in Naderi and Ruiz (2010) for the distributed permutation flow shop problem, and develop the CP models using the CP Optimizer approach (CPOptimizer, 2017), except for job shops where an excellent CP model already exists (Laborie et al., 2018).

- With these models we solve standard and well-known benchmarks taken from the existing literature and accurately scrutinize their performance based on two measures: optimality gaps and relative percentage deviation (RPD) from the best solutions known. While the average optimality gap of each single instance is easily obtainable through comparing the primal bound (the objective function value of the integer solution) with the dual bound (the objective function of the relaxation), it is essential to also compare it to the best integer solution known in the literature for each instance. This allows for the calculation of the RPD and average RPD. This gives us a comparative figure to be used with other proposed algorithms in the literature. Obtaining the best known solutions for several benchmarks and 11 different problems in the literature is far from trivial. We provide these best solutions in the accompanying online materials for this paper.

- We conduct an extensive experimental campaign with a total of 37,974 experiments. This allows us to identify the areas in which each model excels. As we will later show, the use of CP models instead of MILP models results in a vast reduction in the average optimality gap and RPD in all the problems except in parallel machine scheduling problems. Time- and cost-savings achieved by the CP models in general, and the CP Optimizer in particular, imply a paradigm shift in the formulation of exact solutions for scheduling problems which has far-reaching implications for the research and manufacturing communities. We investigate the root cause for this excellence using six dimensions: problem characteristics, objective function, size of the instance in terms of number of jobs and machines, decision variables involved in shop scheduling and time limit. This detailed analysis brings further contributions:

In this paper we show that MILP models are the best candidates only for solving parallel machine scheduling problems and achieve very low average optimality gaps and RPDs. For all the other 10 considered scheduling problems, CP models are clearly better and sometimes by significant margins. In fact, the use of CP models instead of MILP models results in vast reductions in the average optimality gap and RPD values. Furthermore, we show that our CP models extend the capability of exact techniques to solve larger problem instances, previously unattainable by exact off-the-shelf techniques.

We structure this paper as follows. We present preliminaries on modeling paradigms used to formulate scheduling problems in Section §2. In Section §3, we present our mixed-integer

and constraint programming models for the selected scheduling problems and discuss their structural constituents. In section §4, we (i) provide a summary of the benchmarks used to solve these scheduling problems, (ii) present our results, and (iii) discuss the areas in which each of our models excel. We conclude the paper in Section §5 and provide future directions for the current study. We include detailed results in the Appendix.

2. Preliminaries

In order for the comparative study to be self-contained, some basic preliminaries are given for the MILP and CP models, where the emphasis is placed on the different dimensions and possibilities for variable definition.

2.1. Integer programming models

The vast majority of scheduling models use binary variables to decide on the sequence/assignment of jobs and use continuous variables to transform the sequence into a schedule. Each modelling approach can be seen as a different alternative to how these variables are defined. In the following, we loosely group the most popular modelling approaches: position-, time-, sequence- and Manne-based. For the sake of simplicity, we explain them using a numerical example with four jobs and one single machine.

- Position-based (Wagner, 1959): This approach models the sequencing problem as an assignment problem, where jobs are assigned to the positions in the sequence. The number of positions is equal to the number of jobs. For the example with four jobs, the model needs 16 binary variables: e_{jk} takes value 1 if job j is assigned to position k-th in the sequence; and 0 otherwise.

$$\begin{array}{c} \\ \text{Job 1} \\ \text{Job 2} \\ \text{Job 3} \\ \text{Job 4} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Position 1} & \text{Position 2} & \text{Position 3} & \text{Position 4} \end{array} \\ \begin{pmatrix} e_{11} & e_{12} & e_{13} & e_{14} \\ e_{21} & e_{22} & e_{23} & e_{24} \\ e_{31} & e_{32} & e_{33} & e_{34} \\ e_{41} & e_{42} & e_{43} & e_{44} \end{pmatrix} \end{array}$$

- Sequence-based (Wilson, 1989): Here the variables determine the immediate preceding job for each job. In a complete sequence, each job has one immediate preceding and one immediate succeeding job, with the exception of the first and last jobs. For the first job, we additionally consider a dummy job 0 as the first job in the sequence. For the numerical example, the model needs 16 binary variables: $z_{jj'}$ takes value 1 if job j' immediately proceeds job j ($j' \neq j$); and 0 otherwise.

$$\begin{array}{c} \\ \text{Job 0} \\ \text{Job 1} \\ \text{Job 2} \\ \text{Job 3} \\ \text{Job 4} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{Job 0} & \text{Job 1} & \text{Job 2} & \text{Job 3} & \text{Job 4} \end{array} \\ \begin{pmatrix} - & z_{01} & z_{02} & z_{03} & z_{04} \\ - & - & z_{12} & z_{13} & z_{14} \\ - & z_{21} & - & z_{23} & z_{24} \\ - & z_{31} & z_{33} & - & z_{34} \\ - & z_{41} & z_{42} & z_{43} & - \end{pmatrix} \end{array}$$

- Manne-based (Manne, 1960): A different alternative uses variables to determine the relative sequence of each pair of jobs. That is, it specifies whether or not a job precedes another job, but not necessarily immediately, as it does in the sequence-based approach. For the numerical example, this approach requires only 6 binary variables: $x_{jj'}$ takes value 1 if job j' proceeds job j ($j' > j$); and 0 otherwise. It also does not need the dummy job 0. Obviously, if job j proceeds job j', then job j' does not precede job j. Thus, we have $x_{jj'} = 1 - x_{j'j}$ and need either the upper or lower triangular part of the decision variable matrix to represent a valid sequence.

$$\begin{array}{c} \\ \text{Job 1} \\ \text{Job 2} \\ \text{Job 3} \\ \text{Job 4} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Job 1} & \text{Job 2} & \text{Job 3} & \text{Job 4} \end{array} \\ \begin{pmatrix} - & x_{12} & x_{13} & x_{14} \\ - & - & x_{23} & x_{24} \\ - & - & - & x_{34} \\ - & - & - & - \end{pmatrix} \end{array}$$

- Time-based (Bowman, 1959): Different from the previous alternatives, this one directly schedules jobs with the binary variables, without the need for additional variables for constructing the schedule. The continuous time horizon is discretized into smaller time periods and variables determine in which time periods a job is being processed. To this end, we need variables for each job j and time period t: q_{jt} takes value 1 if job j is processed at time period t; and 0 otherwise. For the numerical example, this model requires $4T$ binary variables where T is the number of time periods, which needs to be set a priory and depending on the scheduling problem studied. Setting T ranges from being relatively straightforward to rather complex.

$$\begin{array}{c} \\ \text{Job 1} \\ \text{Job 2} \\ \text{Job 3} \\ \text{Job 4} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Period 1} & \text{Period 2} & \text{Period 3} & \cdots \end{array} \\ \begin{pmatrix} q_{11} & q_{12} & q_{13} & \cdots \\ q_{21} & q_{22} & q_{23} & \cdots \\ q_{31} & q_{32} & q_{33} & \cdots \\ q_{41} & q_{42} & q_{43} & \cdots \end{pmatrix} \end{array}$$

The time-based approach is clearly ineffective when the processing times are long and the number of required time periods increases as the number of binary variables grows very quickly. However, this approach is helpful for scheduling problems with preemptive jobs, i.e., jobs can be interrupted during processing. It is also useful for project scheduling problems (Tofighian and Naderi, 2015). The Manne-based models approach requires the smallest number of binary variables as it defines one binary variable for each pair of jobs, whereas the sequence-based approach requires two binary variables for each pair. Thus, we can conclude that the Manne-based model is generally better than the sequence-based model. However, this is not a universal observation and it is not always the case. The sequence-based approach is suitable for scheduling problems with sequence-dependent processing times, (i.e., the processing time of a job depends on its immediate preceding job) or when jobs have travel times in between stages. An important drawback of the Manne and sequence-based approaches is that disjunctive constraints are needed in their formulation (constraints with Big

M values), which often results in poor performance due to having weak linear relaxations. The position-based approach, although requiring as many binary variables as the sequence-based model, is competitive with the Manne-based approach as it does not require disjunctive constraints for some scheduling problems. Usually, performance drops significantly when the formulation includes disjunctive constraints, which is hard to avoid for some scheduling problems. Therefore, generally speaking, we can conclude that the Manne-based approach is the most competitive and versatile modelling alternative for formulating scheduling problems with different structures, while some of the other approaches are superior for other scheduling problems in particular. This is in line with the extensive literature on MILP models for scheduling problems (Stafford, 1988; Stafford et al., 2005; Wilson, 1989; Pan, 1997; Tseng et al., 2004; Tseng and Stafford, 2008). As a result, we select the Manne-based modelling approach to formulate the scheduling problems as MILP models in this paper.

2.2. Constraint programming models

As for CP, we use the CP Optimizer approach for the scheduling problems (CPOptimizer, 2017). This is in line with the latest publications for different applications: dry bulk terminals (Unsal and Oguz, 2019), container yard scheduling (Qin et al., 2020) and operating room scheduling (Younespour et al., 2019) just to cite a few. Conversely, the classical CP approach uses integer variables and disjunctive constraints to model scheduling problems which are not effective in handling some features such as optional operations and setup times (Laborie et al., 2018).

Interval and *sequencing* variables are the biggest differences between CP Optimizer and MILP models. In the CP Optimizer (referred to as CP in short from now on), for each operation we define one interval variable which is an interval of time during which a job is processed. The start and end points of the interval variable that fall within a larger interval $[\alpha, \beta)$ are decisions for the model. We can formally define an interval variable x as a decision variable whose domain (s, e) is a subset of $\{[\alpha, \beta)|\alpha, \beta \in Z, \alpha \leq \beta\}$ where s and e are the start and end points of the interval, respectively, and $l = e - s$ is its length (see Figure 1). In many applications, the length of the interval variable is fixed and known in advance. Interval variables can be optional; i.e., part of the problem is deciding if the interval is present or absent in the solution. The optional interval variable is a subset of $\{\perp\} \cup \{[\alpha, \beta)|\alpha, \beta \in Z, \alpha \leq \beta\}$ where $x = \perp$ means the interval is absent and $x = [s, e)$ means it is present. The global function *PrecenceOf(x)* returns 1 if an optional interval variable x exists; and 0 otherwise. A sequence variable is also defined for a set of interval variables and a value for the sequence variable is a permutation of the present intervals in that sequence variable (Laborie et al., 2018).

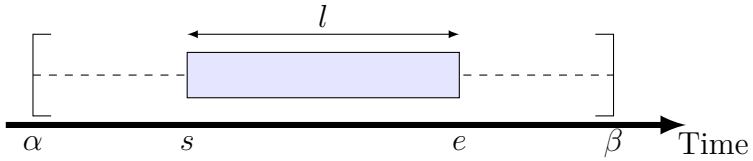


Figure 1: An interval variable with a length of $l = e - s$ within (α, β) where s and e are the start and end time of the interval variable respectively.

Although it might be possible to formulate complex sequencing scenarios with disjunctive constraints, the interval and sequence variables enable us to provide a common structure for many different settings and to exploit them by using the automatic search algorithm in CP (CPOptimizer, 2017). Let p be a sequence variable or a set of interval variables. We can use the following global constraints over this common structure:

- The global constraint *NoOverlap(p)* creates a chain of non-overlapping intervals from variables p. *NoOverlap(p,S)* also represents non-overlapping constraints with a transition matrix S which is the minimum distance between consecutive intervals.
- The global constraint *SameSequence(p,p')* also creates the first-in-first-out relationship between two sequence variables p and p'.
- The global constraint *Alternative(x,p)* creates an alternative constraint among present interval variables p. That is, interval variable x is exactly one of the interval variables in p (e.g. same start and end values).
- The global constraint *EndBeforStart(x,x')* creates a temporal constraint in which interval variable x must end before the start interval variable x'.
- The global constraint *EndAtStart(x,x')* creates a temporal constraint so that interval variable x' must start right after interval variable x ends.
- The global function *Endof(x)* returns the end of interval variable x.

The CP facilitates the formulation of a scheduling problem as a real scheduling problem, rather than a series of interconnected binary/integer decisions as in most MILP models. We can apply complex sequencing aspects as global constraints for the CP directly over the interval/sequence variables.

3. Models

The two main generic assumptions across all scheduling problems are: 1) a job can be processed by at most one machine at the same time, and 2) a machine can process at most one job at a time. The verbal description of a scheduling model is as follows:

minimize	Objective	
subject to	Assign jobs to machines if required,	(1)
	Ensure type-1 non-overlapping if applicable,	(2)
	Ensure type-2 non-overlapping,	(3)
	Calculate objective,	(4)
	Define decision variables.	(5)

Overlap type-1 refers to the overlap among the operations for the same job as a job cannot be processed by more than one machine at any time. Overlap type-2 refers to the overlap among the operations of the different jobs on any machine, as machines cannot process more than one job at any time. Assignment constraints (1) exist in problems with multiple

machines per stage. Overlap constraints (2) appear in problems with multi-operation jobs. They also ensure that the precedence among the operations of jobs, if any, are met. Table 1 contains the notation used in the models. Table 2 also shows the different decision variables.

Table 1: Sets and parameters for the MILP models presented.

Sets:	
$\mathcal{J}$	Set of jobs, $j \in \mathcal{J}$
$\mathcal{I}$	Set of stages $i \in \mathcal{I}$
$\mathcal{F}$	Set of factories, $f \in \mathcal{F}$
$\mathcal{M}_i$	Set of machines at stage i, $k \in \mathcal{M}_i$
Parameters:	
D_j	Due date of job j
P_{ji}	Processing time of job j on machine i
$S_{jj'i}$	Setup time of job j after job j' on machine i

Table 2: Decision variables used in the MILP models presented.

Sequencing	
$x_{jj'}$	1 if job j is processed after job j', 0 otherwise
$x_{ijj'}$	1 if machine i processes job j after job j', 0 otherwise
$h_{jii'}$	1 if for job j visits machine i after machine i', 0 otherwise
$z_{jj'}$	1 if job j is processed immediately after job j', 0 otherwise
Scheduling	
c_{ji}	Completion time of job j on machine i
t_j	Tardiness of job j
$C_{\max}$	Makespan
Assignment	
w_{jik}	1 if job j is assigned to machine k at stage i, 0 otherwise.
y_{jk}	1 if job j is assigned to machine k, 0 otherwise.
q_{jf}	1 if job j is assigned to factory f, 0 otherwise.

3.1. Permutation flow shop (FSP)

In a (permutation or regular) flow shop problem (FSP) a set of jobs has to be processed on a set of stages, each one with only one machine. There are as many operations per job as stages (machines). Each job visits all stages, starting from stage 1 all the way to the last one. In permutation flow shops, we assume that the sequence of jobs at all stages is the

same. That is, once a job is scheduled ahead of another job, all operations for the first job are processed first at all stages. We model the permutation flow shop using mixed-integer and constraint programming models.

3.1.1. MILP model

We use the Manne-based modelling approach to formulate FSP. Since the sequence is kept fixed on all stations, there is no additional need for optimizing the sequence of operations for a job at each stage (i.e., machine). The number of binary sequencing variables is $\frac{|\mathcal{J}|^2-|\mathcal{J}|}{2}$. The mixed integer programming (MILP) model for the FSP is as follows:

$$
\begin{aligned}
\text{minimize} \quad & C_{\max}, && && (\text{MILP}_{\text{FSP}})\\
\text{subject to} \quad & c_{j1} \geq P_{j1} && \forall j \in \mathcal{J}, && (6)\\
& c_{ji} \geq c_{ji-1} + P_{ji} && \forall j \in \mathcal{J}, i \in \mathcal{I} \setminus 1, && (7)\\
& c_{ji} \geq c_{j'i} + P_{ji} - M(1 - x_{jj'}) && \forall i \in \mathcal{I}, j, j' \in \mathcal{J} : j > j', && (8)\\
& c_{j'i} \geq c_{ji} + P_{j'i} - M(x_{jj'}) && \forall i \in \mathcal{I}, j, j' \in \mathcal{J} : j > j', && (9)\\
& C_{\max} \geq c_{ji} && \forall j \in \mathcal{J}, i \in \mathcal{I}, && (10)\\
& c_{ji} \geq 0 && \forall j \in \mathcal{J}, i \in \mathcal{I}, && (11)\\
& x_{jj'} \in \{0, 1\} && \forall j, j' \in \mathcal{J} : j > j'. && (12)
\end{aligned}
$$

Constraints (6) ensure that the completion time of each job j on the first machine, $c_{j1} \geq 0$ is greater than its processing time on that machine, P_{j1}. Constraints (7) indicate the completion time of job j on machines in different manufacturing stages (overlap type-1). Specifically, these constraints ensure that the difference between the completion times of job j at stages i and $i-1$ is at least as large as its processing time on machine i, P_{ji}. Constraints (8) and (9) ensure that no two operations for two jobs j and j' can be processed at the same stage (machine) at the same time. Constraints (10) calculate makespan $C_{\max}$, which is the maximum completion time of all jobs on all stages. Constraints (11) and (12) define the nature of the decision variables.

3.1.2. CP model

The constraint programming (CP) model for the FSP can be written as follows:

$$
\begin{aligned}
\text{minimize} \quad & C_{\max}, && && (\text{CP}_{\text{FSP}})\\
\text{subject to} \quad & Task_{ji} = \text{IntveralVar}\Big(P_{ji}\Big) && j \in \mathcal{J}, i \in \mathcal{I}, && (13)\\
& \text{EndBeforeStart}\Big(Task_{ji}, Task_{ji-1}\Big) && \forall j \in \mathcal{J}, i \in \mathcal{I} \setminus 1, && (14)\\
& SV_i = \text{SequenceVar}\Big(Task_{ji} : j \in \mathcal{J}\Big) && \forall i \in \mathcal{I}, && (15)\\
& \text{NoOverlap}\Big(SV_i\Big) && \forall i \in \mathcal{I}, && (16)\\
& \text{SameSequence}\Big(SV_i, SV_{i-1}\Big) && \forall i \in \mathcal{I} \setminus 1, && (17)\\
& C_{\max} = \max_{(j)} \Big(\text{EndOf}(Task_{j|\mathcal{I}|})\Big). && && (18)
\end{aligned}
$$

Constraints (13) defines the interval variables, one for each job at each stage. We assume that the domain for interval variables is $(\alpha, \beta) = (0, M)$ where M is a large positive number. Constraints (14) control overlap type-1. Constraints (16) do the same for overlap type-2. To create the same sequence for all stages, we need to convert the interval variables into a sequence variable for each stage using constraints (15) and limit the search to the same sequence using constraints (17). This is so because the input argument of global constraint "SameSequence" is a sequence variable. Constraint (18) is the objective calculation which uses the global function "EndOf" over interval variables of jobs at the last stage $|\mathcal{I}|$.

3.2. Non-permutation (general) flow shop (N-FSP)

Unlike the permutation FSP, the sequence of jobs at different stages is not necessarily the same in the non-permutation FSP (N-FSP). We thus need to generate a sequence or permutation of jobs for each stage i using the binary sequencing variables $x_{ijj'} \in \{0, 1\}$. This new sequencing requirement increases the number of binary sequencing variables from $\frac{|\mathcal{J}|^2-|\mathcal{J}|}{2}$ in the FSP to $\frac{|\mathcal{J}|^2-|\mathcal{J}|}{2} \times |\mathcal{I}|$ in the N-FSP, which makes the MILP model's number of binary variables very sensitive to the number of stages.

3.2.1. MILP model

Replacing binary sequencing variable $x_{jj'}$ with $x_{ijj'}$ allows for the design of a separate sequence for each stage. This change has an impact on the constraints that avoid overlap type-2. We replace constraint sets (8), (9) and (12) in the MILP model with constraint sets (19), (20) and (21). The MILP for N-FSP therefore becomes:

$$
\begin{aligned}
&\text{minimize} && C_{\max}, && && (\text{MILP}_{\text{N-FSP}})\\
&\text{subject to} && \text{Constraints (6), (7), (10), and (11)}, \\
& && c_{ji} \geq c_{j'i} + P_{ji} - M(1 - x_{ijj'}) && \forall i \in \mathcal{I}, j, j' \in \mathcal{J} : j > j', && (19)\\
& && c_{j'i} \geq c_{ji} + P_{j'i} - M(x_{ijj'}) && \forall i \in \mathcal{I}, j, j' \in \mathcal{J} : j > j', && (20)\\
& && x_{ijj'} \in \{0, 1\} && \forall i \in \mathcal{I}, j, j' \in \mathcal{J} : j > j'. && (21)
\end{aligned}
$$

3.2.2. CP model

The CP model for the N-FSP is much simpler than the one developed for the FSP. Constraint sets (15), (16) and (17) from CP_{FSP} are removed. Relaxing the "SameSequence" requirement significantly simplifies the model, enabling us to directly use the interval variables to ensure that overlap type-2 is avoided. The resulting simplified CP model is:

$$
\begin{aligned}
&\text{minimize} && C_{\max}, && && (\text{CP}_{\text{N-FSP}})\\
&\text{subject to} && \text{Constraints (13), (14), and (18)}, \\
& && \text{NoOverlap}\big(Task_{ji} : j \in \mathcal{J}\big) && \forall i \in \mathcal{I}. && (22)
\end{aligned}
$$

3.3. Total completion time (TCT-FSP)

The manufacturing shop layout and the processing route for jobs in the Total Completion Time (TCT) FSP (TCT-FSP) is exactly the same as the FSP, with the only change being the objective function. Unlike the FSP that minimizes $C_{\max}$, the TCT-FSP minimizes the total completion time of jobs: the sum of the completion times for jobs at their last manufacturing

stage, $|\mathcal{I}|$. While the set of constraints remains unchanged, we can calculate the TCT based solely on continuous variables c_{ji} without having to resort to the auxiliary continuous variable $C_{\max}$ that captures the maximum completion time of all jobs.

3.3.1. MILP model

$$\begin{aligned} \text{minimize} \quad & \sum_{j\in\mathcal{J}} c_{j,|\mathcal{I}|} && (\text{MILP}_{\text{TCT-FSP}}) \\ \text{subject to} \quad & \text{Constraints (6) - (9), (11), and (12).} \end{aligned}$$

3.3.2. CP model

$$\begin{aligned} \text{minimize} \quad & \sum_{j\in\mathcal{J}} \text{EndOf}(Task_{j|I|}) && (\text{CP}_{\text{TCT-FSP}}) \\ \text{subject to} \quad & \text{Constraints (13) - (17).} \end{aligned}$$

As can be observed, the global function "EndOf" captures the completion time of jobs $j \in \mathcal{J}$ at their last manufacturing stage and thus the changes to the previous models are minimal in the case of the TCT-FSP.

3.4. Total Tardiness (TT-FSP)

The Total Tardiness FSP (TT-FSP) differs from the models that we previously presented in terms of the objective function. The TT-FSP considers due dates, D_j for each job $j \in \mathcal{J}$ and ensures that the amount of total tardiness, t_j is minimized. Ideally, we aim to achieve a total tardiness equal to 0, which would indicate that the completion time of each job $j \in \mathcal{J}$ in its last manufacturing stage $|\mathcal{I}|$, $c_{j|\mathcal{I}|}$, is lower than its due date D_j. Again, given that this is only a change in the objective function, the changes required are minor.

3.4.1. MILP model

$$\begin{aligned} \text{minimize} \quad & \sum_{j\in\mathcal{J}} t_j, && (\text{MILP}_{\text{TT-FSP}}) \\ \text{subject to} \quad & \text{Constraints (6) - (9), (11), and (12),} \\ & t_j \geq c_{j|\mathcal{I}|} - D_j && \forall j \in \mathcal{J}, \quad (23) \\ & t_j \geq 0 && \forall j \in \mathcal{J}. \quad (24) \end{aligned}$$

Constraint sets (23) calculate the tardiness for each job and constraint sets (24) ensure that tardiness cannot be negative, i.e., a job finishing before its due date is not considered tardy.

3.4.2. CP model

The CP model for the TT-FSP does not require auxiliary variables as the available functions in the CP Optimizer allow us to directly calculate the total tardiness in the objective:

$$\begin{aligned} \text{minimize} \quad & \sum_{j\in\mathcal{J}} \max\Big\{\text{EndOf}(Task_{j|I|}) - D_j, 0\Big\}, && (\text{CP}_{\text{TT-FSP}}) \\ \text{subject to} \quad & \text{Constraints (13) - (17).} \end{aligned}$$

3.5. No-wait flow shop (NW-FSP)

No-wait flow shops (NW-FSP) share the same sequencing variables, constraints, and objective function with the FSP with the only difference being that in the NW-FSP we must ensure that the end time of an operation of a job at any stage coincides with the start of the next operation for the same job at that stage, i.e., there is no job waiting between stages. To put it differently, the NW-FSP ensures all operations for the same job are processed one after another with no idle time between them. The optimal solution for the NW-FSP automatically enforces "SameSequence" for all operations for a job–FSP with no idle time requirement between any operations for a job. To transform the FSP into a NW-FSP, we only need to enforce the no-wait restriction in addition to avoiding type-1 overlapping.

3.5.1. MILP model

The MILP model for the NW-FSP with $C_{\max}$ objective function is the following:

$$\begin{aligned} \text{minimize} \quad & C_{\max}, && (\text{MILP}_{\text{NW-FSP}}) \\ \text{subject to} \quad & \text{Constraints (6) and (8) - (12)}, \\ & c_{ji} = c_{ji-1} + P_{ji} && \forall j \in \mathcal{J}, i \in \mathcal{I} \setminus 1. \qquad (25) \end{aligned}$$

Constraint (25) dictates that operations for a job are carried out with no idle time between stages.

3.5.2. CP model

The CP model for the NW-FSP with $C_{\max}$ objective function is detailed below:

$$\begin{aligned} \text{minimize} \quad & C_{\max}, && (\text{CP}_{\text{NW-FSP}}) \\ \text{subject to} \quad & \text{Constraints (13), (18), and (22)}, \\ & \text{EndAtStart}\Big(Task_{ji}, Task_{ji-1}\Big) && \forall j \in \mathcal{J}, i \in \mathcal{I} \setminus 1. \qquad (26) \end{aligned}$$

The global constraint "EndAtStart" in constraint set (26) ensures that all operations for a job are executed uninterruptedly from the first stage to the last.

3.6. Sequence-dependent setup FSP (SDST-FSP)

All previous models only consider the processing times of jobs on machines, P_{ji}. The SDST-FSP is an interesting variation in which a setup operation is required before executing an operation for a job on a machine. Furthermore, the length of this setup operation depends on the sequence of jobs, $S_{jj'i}$. We optimize this problem with the $C_{\max}$ objective function.

3.6.1. MILP model

We formulated the previous sequencing models using the Manne-based modelling approach as we did not require a knowledge of the immediate predecessor and successor of a job. With the inclusion of sequence-dependent setup times, we need to use the sequence-based approach to formulate the problem. The number of binary sequencing variables is $|\mathcal{J}|^2$. To differentiate between the sequencing variables, we use binary sequencing variable $z_{jj'} \in \{0,1\} \mid j \neq j'$ instead of $x_{jj'} \in \{0,1\} \mid j > j'$. This change appears to be insignificant; however, it drastically influences the formulation and its computational efficiency and solution effectiveness. Using $z_{jj'} \in \{0,1\} \mid j \neq j'$, requires the definition of a dummy job 0 as the first job in the sequence. The MILP with the $C_{\max}$ objective function for the SDST-FSP is the following:

$$\begin{aligned}
&\text{minimize} && C_{\max}, && && (\text{MILP}_{\text{SDST-FSP}})\\
&\text{subject to} && \text{Constraints (7), (10), and (11),} \\
& && \sum_{j' \in \{0,\mathcal{J}\}\setminus j} z_{jj'} = 1 && \forall j \in \mathcal{J}, && (27)\\
& && \sum_{j \in \mathcal{J}\setminus j'} z_{jj'} \leq 1 && \forall j' \in \mathcal{J}, && (28)\\
& && \sum_{j \in \mathcal{J}} z_{j0} = 1, && && (29)\\
& && c_{ji} \geq c_{j'i} + P_{ji} + S_{jj'i} - M(1 - z_{jj'}) && \forall i \in \mathcal{I}, j, j' \in \mathcal{J} : j \neq j', && (30)\\
& && z_{jj'} \in \{0,1\} && \forall j, j' \in \mathcal{J} : j \neq j'. && (31)
\end{aligned}$$

Constraints (27), (28) and (29) determine the sequence of jobs where each job follows exactly one job by constraint (27) and precedes at most one job by constraint (28). The last job in the sequence does not precede any job. The dummy job is the only job that definitely precedes one job by constraint (29). Constraints (30) avoid overlapping type-2. Specifically, these constraints ensure that the completion time of the newly arrived job j at machine i is greater than that of the incumbent job j', plus the setup time that is performed after job j' for job j on machine i (i.e., $S_{jj'i}$), plus the processing time of job j on machine i, P_{ji}.

3.6.2. CP model

The CP with the $C_{\max}$ objective function for the SDST-FSP is now defined as:

$$\begin{aligned}
&\text{minimize} && C_{\max}, && && (\text{CP}_{\text{SDST-FSP}})\\
&\text{subject to} && \text{Constraints (13), (14), and (16) - (18),} \\
& && \text{NoOverlap}\big(Task_{ji} : j \in \mathcal{J}, S_i\big) && \forall i \in \mathcal{I}. && (32)
\end{aligned}$$

S_i is the $|\mathcal{J}| \times |\mathcal{J}|$ matrix of setup times at stage i. We need to convert interval variables into sequence variables so as to consider setups and type-2 overlap using the global function "NoOverlap".

3.7. Hybrid flow shop (H-FSP)

The hybrid flow shop scheduling problem (H-FSP) is a significant extension of the FSPs in that we have a different shop floor layout. Specifically, we may have more than one machine

at each stage. Note that that we consider functionally identical machines with the same processing performance, i.e., each operation for a job can be processed by any machine at any stage with the same processing time. To accommodate such a change in resource (machine) arrangements on the shop floor, we require defining *assignment variables* in order to decide which machine at each stage is going to process a job. Therefore, the H-FSP is an assignment/sequencing problem unlike the previous sequencing-only problems.

3.7.1. MILP model

To assign each job j to one of the machines k at stage i, we use binary assignment variables $w_{jik} \in \{0,1\}$. For any set of jobs assigned to any machine at each stage, we ensure that no overlapping occurs using variables $x_{jj'i} \in \{0,1\}$. The MILP model for the H-FSP with the $C_{\max}$ objective function is as follows:

$$\begin{aligned}
&\text{minimize} && C_{\max}, && (\text{MILP}_{\text{H-FSP}})\\
&\text{subject to:} && \text{Constraints (6), (7), (10), and (11),}\\
&&& \sum_{k\in\mathcal{M}_i} w_{jik} = 1 && \forall j \in \mathcal{J}, i \in \mathcal{I}, && (33)\\
&&& c_{ji} \geq c_{j'i} + P_{ji} - M(3 - x_{jj'i} - w_{jik} - w_{j'ik}) && \forall i \in \mathcal{I}, k \in \mathcal{K}_i, j, j' \in \mathcal{J} : j > j', && (34)\\
&&& c_{j'i} \geq c_{ji} + P_{j'i} - M(2 + x_{jj'i} - w_{jik} - w_{j'ik}) && \forall i \in \mathcal{I}, k \in \mathcal{K}_i, j, j' \in \mathcal{J} : j > j', && (35)\\
&&& w_{jik}, x_{jj'i} \in \{0,1\} && \forall i \in \mathcal{I}, k \in \mathcal{K}_i, j, j' \in \mathcal{J} : j > j'. && (36)
\end{aligned}$$

Constraints (33) ensure every job is assigned to exactly one machine at each stage. Constraints (34) and (35) ensure that there are no type-1 and type-2 overlaps respectively.

3.7.2. CP model

Similar to the previous problems, the CP model for the H-FSP with the $C_{\max}$ objective is significantly simpler when compared to its MILP counterpart:

$$\begin{aligned}
&\text{minimize} && C_{\max}, && (\text{CP}_{\text{H-FSP}})\\
&\text{subject to} && \text{Constraints (14) and (18),}\\
&&& Task^*_{jik} = \text{IntervalVar}\Big(P_{ji}, \text{Optional}\Big) && j \in \mathcal{J}, i \in \mathcal{I}, k \in \mathcal{K}_i, && (37)\\
&&& \text{Alternative}\Big(Task_{ji}, Task^*_{jik} : k \in \mathcal{K}_i\Big) && \forall j \in \mathcal{J}, i \in \mathcal{I}, && (38)\\
&&& \text{NoOverlap}\Big(Task^*_{jik} : j \in \mathcal{J}\Big) && \forall i \in \mathcal{I}, k \in \mathcal{K}_i. && (39)
\end{aligned}$$

Constraint (37) defines an optimal interval variable for each operation on each machine. Constraint (38) selects one interval variable for each operation that determines the assignment.

3.8. Distributed flow shop (D-FSP)

The distributed flow shop (D-FSP) generalizes the FSP to multiple independent processing lines or factories where each one of them has an identical number of machines in series (i.e.,

a FSP). Each job is to be assigned to one of these processing lines and processed at different stages on that line. Once assigned, jobs cannot change their processing line. That is, once the job-to-line assignment decision is made, constraints to avoid the type-2 overlap among jobs assigned to different processing lines are no longer required.

3.8.1. MILP model

We use the binary assignment variables $q_{jf} \in \{0,1\}$ to assign each job j to processing line f. The MILP model for the D-FSP is as follows:

$$\begin{aligned}
\text{minimize} \quad & C_{\max}, && && (\text{MILP}_{\text{D-FSP}})\\
\text{subject to} \quad & \text{Constraints (6) and (7) and (10) - (12)}, && &&\\
& \sum_{f\in\mathcal{F}} q_{jf} = 1 && \forall j \in \mathcal{J}, && (40)\\
& c_{ji} \geq c_{j'i} + P_{ji} - M(3 - x_{jj'} - q_{jf} - q_{j'f}) && \forall i \in \mathcal{I}, f \in \mathcal{F}, j, j' \in \mathcal{J} : j > j', && (41)\\
& c_{j'i} \geq c_{ji} + P_{j'i} - M(2 + x_{jj'} - q_{jf} - q_{j'f}) && \forall i \in \mathcal{I}, f \in \mathcal{F}, j, j' \in \mathcal{J} : j > j', && (42)\\
& q_{jf} \in \{0,1\} && \forall i \in \mathcal{I}, f \in \mathcal{F}. && (43)
\end{aligned}$$

Constraint (40) assigns each job to exactly one of the existing processing lines. Constraints (41) and (42) avoid type-2 overlapping for the jobs assigned to the same processing line.

3.8.2. CP model

$$\begin{aligned}
\text{minimize} \quad & C_{\max}, && && (\text{CP}_{\text{D-FSP}})\\
\text{subject to} \quad & \text{Constraints (14) and (16) - (18)}, && &&\\
& Task^*_{jif} = \text{IntveralVar}\Big(P_{ji}, \text{Optional}\Big) && j \in \mathcal{J}, i \in \mathcal{I}, f \in \mathcal{F}, && (44)\\
& \text{PresenceOf}\Big(Task^*_{j1f}\Big) \leq \text{PresenceOf}\Big(Task^*_{jif}\Big) && \forall j \in \mathcal{J}, i \in \mathcal{I} \setminus 1, f \in \mathcal{F}, && (45)\\
& \text{Alternative}\Big(Task_{ji}, Task^*_{jif} : f \in \mathcal{F}\Big) && \forall j \in \mathcal{J}, i \in \mathcal{I}, && (46)\\
& \text{NoOverlap}\Big(Task^*_{jif} : j \in \mathcal{J}\Big) && \forall i \in \mathcal{I}, f \in \mathcal{F}. && (47)
\end{aligned}$$

Constraint (44) defines one interval variable for each operation at each stage. Constraint (45) ensures that if a job is assigned to a line, all its operations are processed on that line and constraint (46) assigns one line to each job.

3.9. Job shop problem (JSP)

So far we have discussed flow shop variants in that jobs and machines have common characteristics: (i) all machines are disposed in series and (ii) all jobs have the same (linear) processing route, requiring the service of all these machines in the same order. We now discuss another widely-occurring manufacturing shop floor setting –the job shop scheduling problem (JSP)– with different characteristics. Machines in JSPs are not necessarily disposed serially and can be laid out according to the machining requirements of jobs. Unlike jobs

with an equal number of operations on each processing line in flow shops, the JSP processes jobs with a varying number of operations and machining requirements. Thus, the processing route of each job might be unique. In FSPs, the stage before stage i is always stage $i-1$, whereas in JSPs, it can be any of the other stages. Let i' indicate the stage before stage i in the processing route of a given job (which is given as input data). Therefore, both MILP and CP models require modifications in order to accommodate such non-ordered sets of stages in the JSP. To avoid type-1 overlapping, we replace $i-1$ with i'.

3.9.1. MILP model

The MILP for the JSSP is as follows:

$$\begin{aligned} \text{minimize} \quad & C_{\max}, && (\text{MILP}_{\text{JSP}}) \\ \text{subject to} \quad & \text{Constraints (6), (10), (11), (19) and (21)}, \\ & c_{ji} \geq c_{ji'} + P_{ji} && \forall j \in \mathcal{J}, i \in \mathcal{I}. \quad (48) \end{aligned}$$

3.9.2. CP model

The CP for the JSSP is as follows:

$$\begin{aligned} \text{minimize} \quad & C_{\max}, && (\text{CP}_{\text{JSP}}) \\ \text{subject to} \quad & \text{Constraints (13), (18), and (22)}, \\ & \text{EndBeforeStart}\left(Task_{ji}, Task_{ji'}\right) && \forall j \in \mathcal{J}, i \in \mathcal{I}. \quad (49) \end{aligned}$$

3.10. Open shop problem (OSP)

The open shop scheduling problem (OSP) is a relaxed variant of the JSP in that there is no *technical precedence* among the operations for a job. As long as all operations for a job are processed, irrespective of their sequence, the job is considered completed. With such a relaxation in technical precedence among operations for a job, the processing route of a job turns into a decision variable, i.e., the processing route of each job is determined during the optimization run. Therefore, there are two sequencing decisions: the order of operations for different jobs at each stage and the order of operations for a job. Such a change requires modifications to type-1 non-overlapping constraints. Specifically, type-1 non-overlapping constraints will resemble type-2 non-overlapping constraints.

3.10.1. MILP model

In addition to type-2 non-overlapping constraints that require binary sequencing variables $x_{ijj'} \in \{0,1\}$, we define a new sequencing variable $h_{jii'} \in \{0,1\}$ to enforce type-1 non-overlapping constraints as disjunctive constraints (50) and (51) as follows:

$$\begin{aligned} \text{minimize} \quad & C_{\max}, && (\text{MILP}_{\text{OSP}}) \\ \text{subject to} \quad & \text{Constraints (6), (10), (11), (19), and (20)}, \\ & c_{ji} \geq c_{ji'} + P_{ji} - M(1 - h_{jii'}) && \forall j \in \mathcal{J}, i, i' \in \mathcal{I} : i > i', \quad (50) \\ & c_{ji'} \geq c_{ji} + P_{ji'} - M(h_{jii'}) && \forall j \in \mathcal{J}, i, i' \in \mathcal{I} : i > i', \quad (51) \\ & h_{jii'}, x_{ijj'} \in \{0,1\} && \forall j \in \mathcal{J}, i, i' \in \mathcal{I} : i > i'. \quad (52) \end{aligned}$$

3.10.2. CP model

We require two "NoOverlap" global constraints for type-1 and type-2 non-overlapping constraints:

$$\begin{aligned} \text{minimize} \quad & C_{\max}, && && (\text{CP}_{\text{OSP}}) \\ \text{subject to} \quad & \text{Constraints (13), (18), and (22)}, \\ & \text{NoOverlap}\Big(Task_{ji} : i \in \mathcal{I}\Big) && \forall j \in \mathcal{J}. && (53) \end{aligned}$$

3.11. Parallel machine problem

All previous shop floor scheduling models that we presented had multiple operations per job. In the parallel machine scheduling problem (PMSP), we schedule a set of single-operation jobs on multiple machines that are disposed in parallel. We minimize $C_{\max}$. Since we are scheduling single-operation jobs, we do not need to include sequencing variables and constraints and we can calculate the $C_{\max}$ only by using assignment variables. In the PMSP, we simply find the $C_{\max}$ by summing over the processing times of the jobs assigned to each machine. Unlike the previous models, the PMSP is a *pure assignment* problem.

3.11.1. MILP model

We use binary assignment variables $y_{jk} \in \{0, 1\}$ to assign each job j to a machine k. The MILP model for the PMSP is as follows:

$$\begin{aligned} \text{minimize} \quad & C_{\max} && && (\text{MILP}_{\text{PMSP}}) \\ \text{subject to} \quad & \sum_{k \in \mathcal{K}} y_{jk} = 1 && \forall j \in \mathcal{J}, && (54) \\ & C_{\max} \geq \sum_{j \in \mathcal{J}} P_{jk} y_{jk} && \forall k \in \mathcal{K}, && (55) \\ & y_{jk} \in \{0, 1\} && \forall j \in \mathcal{J}, \forall k \in \mathcal{K}. && (56) \end{aligned}$$

3.11.2. Constraint programming model

We use "NoOverlap" global constraints for type-2 non-overlapping. The CP model for the PMSP is as follows:

$$\begin{aligned} \text{minimize} \quad & C_{\max} && && (\text{CP}_{\text{PMSP}}) \\ \text{subject to} \quad & Task^*_{jk} = \text{IntveralVar}\Big(P_{ji}, \text{Optional}\Big) && j \in \mathcal{J}, k \in \mathcal{K}, && (57) \\ & \text{Alternative}\Big(Task_j, Task^*_{jk} : k \in \mathcal{K}\Big) && \forall j \in \mathcal{J}, && (58) \\ & \text{NoOverlap}\Big(Task^*_{jk} : j \in \mathcal{J}\Big) && \forall k \in \mathcal{K} && (59) \\ & C_{\max} = \max_{(j)} \Big(\text{EndOf}(Task_j)\Big). && && (60) \end{aligned}$$

4. Experimental evaluation

Testing all the previously mentioned models is a major undertaking. For the tests we use an OpenStack virtualization platform supported by 12 1U blade servers, each one with four 12-core AMD Opteron Abu Dhabi 6344 processors running at 2.6 GHz. and with 256 GB

of RAM, for a total of 576 cores and 3 TBytes of RAM. In this cluster we run hundreds of virtual machines with 4 virtual processors and 16 GBytes of RAM memory for each. The virtual machines run Windows 10 Enterprise 64 bits. This allows for a massive paralellization of the experimental load, which despite the resources, took several months to conclude. We program the models in Python 3 linked with IBM ILOG CPLEX 12.8.1 and IBM ILOG CP Optimizer 12.8.1 using concert programming technology. The goal that we seek to achieve with these experiments is to ascertain which of the MILP and CP models performs better for each scheduling problem that we introduced and modeled in the previous section. To this end, we conduct extensive experiments on existing benchmark instances from the literature for each of the problems considered. Specifically, we consider a total of 6,329 instances. To have a more accurate picture, and considering the maximum amount of CPU time given is, of course, an important factor, we consider varying maximum CPU time limits (timelimits) for all models. We consider three timelimits of 600, 1,800 and 3,600 seconds for each model. Each one of these timelimits is tested independently, i.e., from scratch, and not from a single run of 3,600 in which the solution quality is measured after 600, 1,800 and 3,600 seconds. The reason is subtle, but important. Modern solvers incorporate heuristic components like the heuristic presolver. A good heuristic solution at the start of a run is going to influence the data points taken later at 1,800 and 3,600 seconds, generating dangerous auto-correlations in the experiments. The consideration of short (600), medium (1,800), and long (3,600) timelimits has two advantages: (i) it allows researchers with limited computing power to compare the solutions for their new algorithms and those of existing ones in a shorter CPU time, and (ii) it helps those practitioners who are in need of practical, efficient (fast) and effective (superior solution quality) algorithms for their shop floor scheduling problems decide which algorithm is best suited to their needs. As a result, the total number of runs, considering both models, for all scheduling problems is 37,974 ($2 \times 3 \times 6,329$).

We use two different performance measures. The first measure is the *optimality gap* (Gap) of the integer solution (upper bound) obtained by these models. The optimality gap captures the deviation of the best integer solution found by each model in relation to the best bound obtained from its LP relaxation (lower bound). The second measure is the *Relative Percentage Deviation* (RPD). Unlike the optimality gap, the RPD compares the relative distance of the best integer solution for each model to that of the *best known integer solution* among all tested methods from the literature. Note that the gap we provide is based on the difference between the upper bound and lower bound of each of the CP and MILP models, but the RPD analysis encompasses the distance between the best integer solution for the tested models with the best known integer solution from the literature. It has to be noted that some models might provide worse optimality gaps, but the quality of their integer solutions might be better; in any practical setting it is the integer solution that is used for scheduling and not a better bound (lower bound given by the LP relaxation). We provide the formula for the RPD and Gap as follows:

$$\text{RPD} = \left(\frac{\text{Upper Bound}-\text{Best upper bound}}{\text{Best upper bound}}\right) \times 100, \qquad \text{Gap} = \left(\frac{\text{Upper Bound}-\text{Lower bound}}{\text{Upper bound}}\right) \times 100.$$

4.1. Standard benchmarks

We make use of many famous benchmarks from the literature. Taillard benchmarks (Taillard, 1993) have been widely used to compare the performance of algorithms developed

for basic scheduling problems: flow shops, job shops and open shops over the past three decades.[1] However, this benchmark has almost been solved, where most instances have either known optimal solutions or upper bounds that are very close to lower bounds. Using this benchmark is not advisable as we observe very small differences between the tested models. Vallada et al. (2008) proposed a larger and much more difficult set of 240 instances. Therefore, for the FSP we use the original 120 instances of Taillard plus these harder 240 instances for a total of 360 instances. These are used for most tested flow shop problems in this paper (the permutation flow shop, non-permutation flow shop, no-wait flow shop and TCT flow shop). We also use 480 Taillard-based instances by Ruiz et al. (2005) for the SDST flow shop, 600 Taillard-based instances by Naderi and Ruiz (2010) for the distributed flow shop, 1,440 instances by Pan et al. (2017) for the hybrid flow shop and 540 instances by Vallada et al. (2008) for total tardiness flow shop. As for the parallel-machine problem, we use benchmarks of Fanjul-Peyro and Ruiz (2010) that include 1,400 instances. There are several benchmarks for job shops in the literature, but compared to the previous ones they have fewer instances each, so we amalgamate all of them and consider a total of 237 instances. Similarly, for the open shop scheduling problem, we use three benchmarks for a total of 192 instances. Table 3 summarizes the standard benchmarks for the problems studied in this paper and their size ranges.

Table 3: Datasets for scheduling problems.

Problem	Dataset	Symbol	# instances	Instance sizes	
				$\|\mathcal{J}\|$	$\|\mathcal{I}\|$
Flow shop (FSP)					
Non-permutation flow shop (N-FSP)	Taillard (1993)	TFS	120	20-500	5-20
No-wait flow shop (NW-FSP)	Vallada et al. (2015)	VRF	240	100-800	20-60
TCT flow shop* (TCT-FSP)					
SDST flow shop* (SDST-FSP)	Ruiz et al. (2005)	RMA	480	20-500	5-20
Distributed flow shop* (D-FSP)	Naderi and Ruiz (2010)	NR	600	20-500	5-20
Hybrid flow shop (H-FSP)	Pan et al. (2017)	PRA	1440	50-200	5-10
Total tardiness flow shop (TT-FSP)	Vallada et al. (2008)	VRM	540	50-350	10-50
Parallel machine (PMSP)	Fanjul-Peyro and Ruiz (2010)	FR	1400	100-1000	5-20
Job shops (JSSP)	Demirkol et al. (1998)	DUM	80	20-50	15-20
	Taillard (1993)	TJS	80	15-100	15-20
	Lawrence (1984)	LA	40	10-30	5-15
	Applegate and Cook (1991)	ORB	10	10	10
	Storer et al. (1992)	SWV	20	20-50	10-15
	Yamada and Nakano (1992)	YN	4	20	20
	Fattahi et al. (2007)	FMJ	3	6-20	5-10
Open shops (OSP)	Taillard (1993)	TOS	60	5-20	5-20
	Brucker et al. (1997)	BHJ	52	3-8	3-8
	Guéret and Prins (1999)	GP	80	3-10	3-10

* Based on Taillard (1993)

4.2. Results

The basic problem in this study is the permutation flow shop with makespan minimization. We compare the CP and MILP models using six different performance criteria: (i) problem

[1] This benchmark has been cited in excess of 2,420+ times (at the time of writing this paper).

characteristics, (ii) objective function, (iii) decision variables, (iv) timelimit, (v) problem size and (vi) performance measures. Analysis of each one of these criteria provides unique insights into the performance of the CP and MILP models. Below, we provide a definition for each of these criteria:

1. **Problem characteristics**: We consider the basic flow shop problem and two extensions in which the models share the same type of decision variables: the permutation flow shop (FSP), no-wait flow shop (NW-FSP) and SDST flow shop (SDST-FSP).
2. **Objective function**: We compare the flow shop problem with three different objectives: makespan (FSP), total completion time (TCT-FSP) and total tardiness (TT-FSP).
3. **Problem decisions**: There are two main decisions in scheduling problems: *sequencing* and *assignment*. We consider seven essentially different scheduling problems that encompass different decisions, ranging from pure sequencing (OSP), to a mix of the two decisions (e.g., H-FSP), to pure assignment (PMSP).
4. **Problem size**: We also analyze the convergence of the models over different problem sizes of basic flow shop problems. We consider the number of jobs and machines versus the studied performance measures.
5. **Timelimit**: We also analyze the convergence behavior of these models to better understand how their upper and lower bounds improve over time. We choose to perform this analysis for two problems: The FSP (sequencing decision) and H-FSP (sequencing and assignment decisions).
6. **Bounds**: We analyze performance measures and break them down into two parts: upper and lower bound distance from the best known bounds.

4.2.1. Impact of problem characteristics: flow shop (FSP), No-wait flow shop (N-FSP), and setup times flow shop (SDST-FSP)

We report the average optimality gap, RPD and number of feasible and optimal solutions found by each model in Table 4. Due to extreme performance variability, we cannot fairly compare these models in terms of the average optimality gap. The CP model is able to find integer feasible and optimal solutions for the FSP in 100% of the cases (120 out of 120) and 35% of the cases (42 out of 120) for the TFS instances, respectively. These percentages are 71.6% (86 out of 120) and 0% (0 out of 120) for the MILP model, respectively. While capable of solving all instances for the TFS benchmark, the CP model yields an average optimality gap of 4.35%. The CP model maintains its superior performance for the VFR benchmark that consists of much larger instances, while the performance of the MILP deteriorates to a level of non-performance. Specifically, the CP model finds integer feasible solutions for 100% (240 out of 240) of the instances in the VFR benchmark, whereas the MILP model finds integer feasible solutions for only 2% (5 out of 240) of the problem instances. The average optimality gap of these five solved instances is almost five times higher than that of the CP model that solves all the difficult VFR instances (80.87% versus 16.26%). The average optimality gap of the CP for all 240 instances is 14.8%. An important observation is that the CP model is a state-of-the-art exact technique for solving the FSP and can thus be used to solve industrial-scale permutation FSPs. Of course, we refer to off-the-shelf exact techniques.

In addition to the FSP, we illustrate the performance of the MILP and CP models on the N-FSP and SDST-FSP problems in Table 4. As stated earlier, the only difference between the FSP and the N-FSP is the restriction that ensures that there are no idle times between

Table 4: Average results for flow shop problems: Problem characteristics.

Problem	Dataset	#	CP					MILP				
			State(#)		Gap	Time	RPD	State(#)		Gap	Time	RPD
			Feas.	Opt.				Feas.	Opt.			
FSP	TFS	120	78	42	**4.35**	2710	**2.91**	86	0	63.07	3600	6.00
	VFR	240	240	0	**14.80**	3600	**10.97**	5	0	-	3600	-
N-FSP	TFS	120	120	0	**36.33**	3600	**5.48**	120	0	80.23	3600	26.30
	VFR	240	240	0	**67.51**	3600	-	132	0	94.07	3600	-
SDST-FSP	RMA	480	476	4	**31.34**	3589	**6.77**	441	0	87.63	3600	15.86

Feas. and Opt. indicate number of feasible and optimal solutions, respectively. Time in seconds. Gap and RPD in percentage.

two operations for a job on two successive machines on the shop floor. The enforcement of such a restriction results in significantly worse performance measures for both models. The general finding from comparing the performance of the MILP and CP models for the N-FSP is largely similar to that of the FSP in that the CP model demonstrates higher solvability and lower average optimality gaps when compared to the MILP model. The difference, however, is that the CP average optimality gap in the N-FSP substantially increases (compared to its performance in the FSP). The CP average optimality gaps increase 8 fold (from 4.35% to 36.33%) and 4 times (from 14.8% to 67.51%) for the TFS and VFR benchmarks, respectively. Another notable difference is that the MILP model can solve all instances of the TFS benchmarks (from 86 to 120 in TFS). The MILP model obtains an average gap of 80.23% and an average RPD of 26.30%—almost five times more than the average RPD for the CP model, which is 5.48%. As can be seen, the CP model is superior to the MILP on both performance measures and benchmarks. The VFR benchmark is notably more difficult than the TFS for the N-FSP.

For the SDST-FSP, we use the RMA benchmark, which is based on TFS instances. Again, the CP model clearly outperforms the MILP model in terms of solvability, average optimality gap, and average RPD. The CP and MILP models solve 99.2% and 91.9% of instances, yielding average optimality gaps of 31.34% and 87.63%, respectively. The average RPDs of the CP and MILP models are 6.77% and 15.86%, respectively. An interesting observation is that the quality of the integer solutions produced by the MILP moves closer to that of the CP—the average RPD of the MILP model is now only twice as large as that of the CP model. We can also see that the consideration of sequence-dependent setup times deteriorates the CP model's performance by 26.99% when compared to the FSP (31.34% − 4.25%). Despite this complexity, the performance of the CP model at a 31.34% optimality gap is much more acceptable than that of the MILP model at 87.63%. Again, we can conclude that the CP is superior to the MILP for this problem.

4.2.2. Objective impact: Makespan, Total Completion Time, and Total Tardiness

The results of the MILP and CP models developed for the FSP with different objective functions: Makespan (FSP), Total Completion Time (TCT) and Total Tardiness (TT) are shown in Table 5. Using the same datasets as in the FSP, we find that the CP model developed for the TCT-FSP demonstrates higher solvability and a lower average optimality gap. The TCT objective function makes the problem substantially more difficult to solve.

The CP solves 100% of the instances for both the TFS and VFR benchmarks, whereas these percentages are 75% and 3.75% for the MILP model, respectively. The average optimality gap in the CP model is 26.25% and 58.28% on TFS and VFR benchmarks, respectively. When solving the TCT-FSP we find that the CP model can no longer solve any of our instances to optimality, demonstrating the difficulty in solving the TCT-FSP, even though TCT is considered as a completion-time related objective similar to makespan. The performance of the MILP model bears a close resemblance to that of the C_{max} FSP with an average gap of 62.67%, and is again unsuitable for TCT-FSPs. As for the quality of the integer solutions, the CP model obtains an average RPD of 6.67%, whereas the MILP model yields an average RPD of 10.45%. We see that a change in objective function does not change the outcome: CP models are superior to MILP models.

Table 5: General results for flow shop problems: Objective functions.

Objective	Dataset	#	CP					MILP				
			State(#)		Gap	Time	RPD	State(#)		Gap	Time	RPD
			Feas.	Opt.				Feas.	Opt.			
FSP	TFS	120	78	42	**4.35**	2710	**2.91**	86	0	63.07	3600	6.00
	VFR	240	240	0	**14.80**	3600	**10.97**	5	0	-	3600	-
TCT-FSP	TFS	120	120	0	**26.25**	3600	**6.67**	90	0	62.67	3624	10.45
	VFR	240	240	0	**58.28**	3600	-	9	0	76.28	3600	-
TT-FSP	VRM	540	476	64	**80.35**	3198	**25.37**	167	0	97.57	3600	81.70

Feas. and Opt. indicate number of feasible and optimal solutions, respectively. Time in seconds. Gap and RPD in percentage.

We now address the impact of the TT objective function on the performance of the CP and MILP models. The inclusion of the TT objective function worsens the average performance of both models in terms of average optimality gaps, RPD, and solvability. This is by far the worst performance by our models in any problem variants that we have tested so far. The CP and MILP models achieve average gaps of 80.35% and 97.57% respectively, and yield average RPD values of 25.37% and 81.70% with respect to the best known integer solutions in the literature respectively. The rate of solvability is also significantly worse for both models at 88% and 31% for the CP and MILP models, respectively. Further analysis of individual instances reveals an interesting observation. When the amount of total tardiness is zero in any instance, the CP model can quickly find the optimal solution (the CP has found 64 optimal solutions for the TT objective function), but when the optimal solution entails having a positive amount of tardiness, both models perform quite poorly. The large average optimality gap in both models causes the decision maker to believe that these models are not efficient for solving the TT-FSP; however, the average RPD of the CP model demonstrates that its performance is 25.37% worse than the best known integer feasible solutions in the literature. Note that best integer solutions have been obtained from different studies that include different algorithms; there is no single study in the literature that includes all the best integer solutions. As such, an average RPD of 25.37% is reasonable from a practical perspective given the fact that the CP is compared against the best ad-hoc algorithms in the literature that were specifically devised for this problem, while our CP model runs in an off-the-shelf commercial CP solver (CP Optimizer). The conclusion after studying these

three objective functions is that the CP model is consistently better than the MILP.

4.2.3. Problem decision dimensions: Sequencing and assignment

We proceed to an analysis of the performance of the CP and MILP models under varying scheduling decisions or dimensions: assignment and sequencing. As discussed earlier, scheduling problems can range from pure sequencing to pure assignment problems. Table 6 depicts the average gap and RPD for these models in different scheduling problems. The general finding from comparing the performance of the MILP and CP models for these problems is that as we move away from pure sequencing problems to assignment problems, the performance of the CP model deteriorates (see Figure 2).

Table 6: Average results for scheduling problems with different problem dimensions.

Objective	Dataset	#	CP					MILP				
			State(#)		Gap	Time	RPD	State(#)		Gap	Time	RPD
			Feas.	Opt.				Feas.	Opt.			
OSP	TOS	60	0	60	**0.00**	1	**0.00**	37	23	28.72	2126	1.64
	GP	80	0	80	**0.00**	6	**0.00**	7	73	0.01	105	**0.00**
	BHJ	52	3	49	**0.11**	316	**0.00**	14	38	1.13	708	0.01
JSP	TJS	80	40	40	**2.50**	1760	**0.75**	80	0	47.40	3600	275.72
	DUM	80	62	18	**7.15**	2939	**3.54**	80	0	51.75	3600	20.71
	LA	40	1	39	**0.07**	116	**0.00**	33	7	27.34	2688	0.50
	Others	42	20	22	**3.07**	1361	**0.96**	35	7	23.97	2076	4.45
FSP	TFS	120	78	42	**4.35**	2710	**2.91**	86	0	63.07	3600	6.00
	VFR	240	240	0	**14.80**	3600	**10.97**	5	0	-	3600	-
N-FSP	TFS	120	87	33	**5.88**	2877	**3.90**	110	0	75.27	3600	15.34
	VFR	240	240	0	**26.43**	3600	**10.49**	45	0	81.50	3600	21.58
D-FSP	NR	600	580	20	**57.01**	3553	**10.60**	404	14	57.40	3570	136.09
H-FSP	PRA	1440	1440	0	**66.47**	3600	**6.21**	1128	0	83.80	3600	212.32
PMSP	FR	1400	1364	36	78.74	3514	1.73	538	862	**0.23**	1523	**-0.10**

Feas. and Opt. indicate number of feasible and optimal solutions, respectively. Time in seconds. Gap and RPD in percentage.

More specifically, for the N-FSP, both the MILP and CP models perform, to a large extent, similarly to how they perform in the FSP. The difference, however, is that the average optimality gap of the CP model in the N-FSP is a bit large when compared to the FSP. However, the results in the N-FSP are worse for the MILP when compared to those of the FSP. OSP benchmarks include smaller instances as the OSP has been a challenging problem. Surprisingly, the CP model solves 98.44% of the instances (189 instances out of 192) to optimality with average computation times of 1, 6 and 316 seconds for the TOS, GP and BHJ benchmarks respectively. Comparatively, the MILP solves 134 out of 192 instances with average CPU times of 2s126, 105 and 708 seconds for TOS, GP, and BHJ, respectively. For the JSP, the average optimality gap of the CP model across all 242 instances is just 3.73% and that of the MILP model is 41.46%. The CP optimally solves 119 instances, whereas the

MILP only solves 14 smaller instances. The results for the JSP are somewhat expected as CP solvers have been finely tuned for the JSP problem (Laborie et al., 2018).

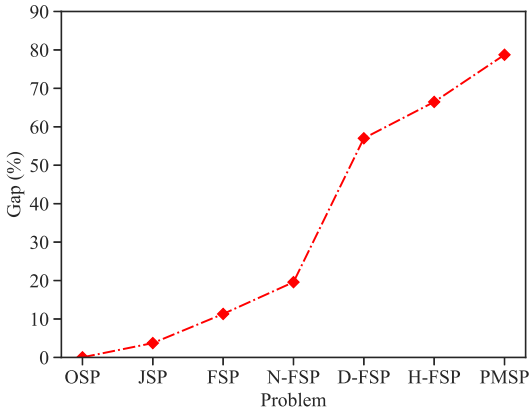


Figure 2: Average gap of the CP in the selected scheduling problems

The D-FSP is the first scheduling problem in Table 6 that includes assignments in the form of assignment of jobs to factories. This decision, one per job, worsens the average gap performance of the CP model remarkably, by 52.66% (= 57.01% − 4.35%—average gap of FSP). Note that the benchmark NR is also based on the TFS benchmark and hence the comparison between the FSP and D-FSP is fair. The average RPD of the CP model increases from 2.91% in the FSP to 10.6% in the D-FSP. When making the transition from the D-FSP to the H-FSP (which includes assignment decisions for each job at each processing stage), the average optimality gap of the CP model again increases from 57.01% to 66.47%. Contrary to our expectations, the MILP models are not competitive at all, with an average RPD of 136.09% and 212.32% for the D-FSP and H-FSP, respectively. This is by far the worst performance of the MILP with respect to the CP and the best known integer solutions. Summing up, in the case of the D-FSP and H-FSP problems, CP models are vastly superior to MILP ones, but in some cases and scenarios the average gaps start to be of questionable quality in practice. Note that in the case of the H-FSP, an average RPD of just 6.21% for the CP is still a very good result as we are comparing it against the best known solutions from the literature and these solutions come from the best performance of different published methods and algorithms.

So far we have discussed scheduling problems that are either pure sequencing or that jointly optimize assignment and sequencing variables. We now focus on the PMSP that is a pure assignment problem. We can see that solving the PMSP represents a turning point for the CP model (see Table 6). The PMSP is the only scheduling problem in which the MILP model is superior to the CP model. While both models find feasible solutions for 100% of the instances, the MILP and CP models solve 61.6% and 2.6% of instances to optimality, respectively. In addition to solving more instances to optimality, the MILP model has an average optimality gap of just 0.23%, which is much lower than that of the CP with 78.74%. This indicates a clear superiority of the MILP over the CP if we ignore their average RPDs. Comparing the average RPD of these two models demonstrates that the performance of the CP is not as inferior as its average optimality gap insinuates. In fact, the CP yields an

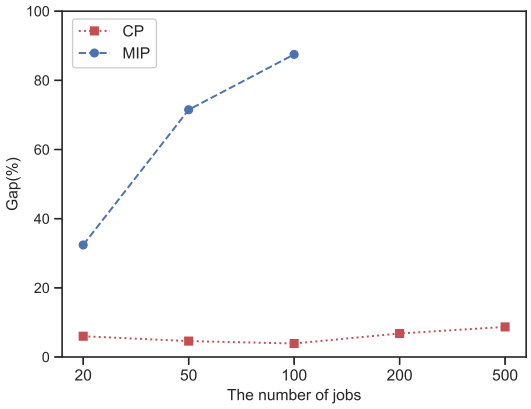
average RPD of 1.73% and the MILP yields an average RPD of -0.10% (this negative number indicates that the solution provided by the MILP model is lower (i.e., better) than most of the best existing solutions from the literature—This turns the numerator of the RPD formula into a negative number). This means that the CP model only generates poor lower bounds for the PMSP, resulting in large gap values. However, when compared with the best known upper bounds from the literature, the average RPD is much lower. In any case, it has to be acknowledged that MILP models are much better for the PMSP.

4.2.4. Impact of problem size on performance

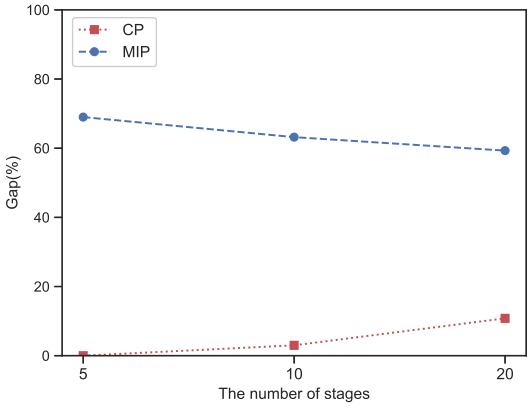
Up to this point we have shown average results across all benchmarks, without commenting on the effect that instance size has on the performance of the CP and MILP models. Figure 3 shows the effect of the number of jobs and machines on the tested models for the FSP. As illustrated in Figure 3(a), the MILP model is very sensitive to an increase in the number of jobs, whereas the CP model's performance remains almost constant regardless of the number of jobs. The MILP model fails to find feasible solutions for problem instances exceeding 100 jobs, but the CP model can find integer feasible solutions for instances of up to 500 jobs. It is remarkable that the CP remains robust with respect to the increase in the number of jobs and achieves similar average optimality gaps for significantly different job sizes. Unlike the increase in the number of jobs, the CP model's average optimality gap deteriorates as we increase the number of stages, whereas the average optimality gap of the MILP model ameliorates (see Figure 3(b)). The number of stages creates more precedence relations for the models and this result implies that the CP is influenced by the precedence relations, more than by the number of jobs, which is commonly considered to be the problem size indicator of importance in the scheduling literature. Despite the observable improvement pattern in the MILP's performance due to the increase in the number of stages, its performance is still vastly inferior to that of the CP model for the problem instance sizes considered.

4.2.5. Impact of timelimit on convergence

Also worthy of consideration is the effect of the maximum CPU time given to the models on their performance. More specifically, we are interested in analyzing how the upper bound and lower bound of these two models change as additional CPU time becomes available. Recall that all instances have been solved with three maximum CPU time limits (timelimits): 600, 1,800, 3,600 seconds. Figure 4 shows the average optimality gap for the FSP (no assignment decisions) and H-FSP (joint sequencing-assignment decisions). When solving the FSP, the allocation of more CPU time to the CP model improves the average optimality gap, whereas additional allocation of CPU time to the MILP model does not necessarily improve the average optimality gap (see Figure 4(a)). The profile of convergence is different under the H-FSP (see Figure 4(b)). The allocation of more CPU time to the MILP model improves the average optimality gap, but it does not improve the average gap in the CP model. An interesting observation is that for some problems, additional CPU times do not necessarily improve results, indicating that an allotment of 600 seconds is already enough for some scheduling problems if the CP approach is used.

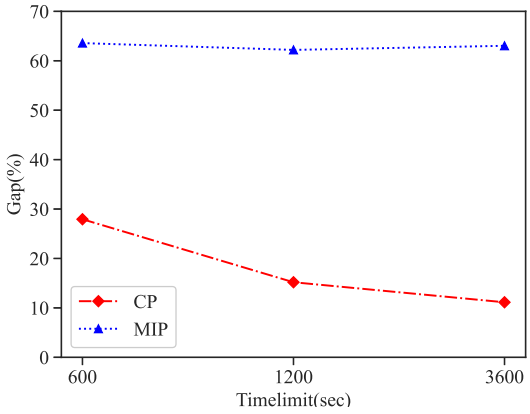


(a) Number of jobs ($|\mathcal{J}|$)

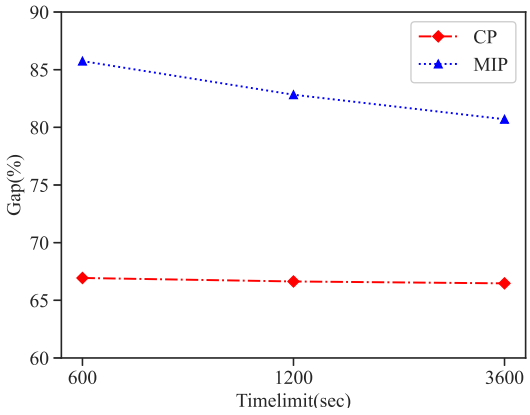


(b) Number of machines ($|\mathcal{I}|$)

Figure 3: Average Gap vs. problem size for the MILP and CP models in permutation flow shops (FSP).



(a) flow shops (FSP)



(b) Hybrid flow shops (H-FSP)

Figure 4: Average Gap vs. timelimit for the MILP and CP models in permutation (FSP) and hybrid flow shops (H-FSP).

5. Conclusions and future research

It is common, almost traditional, for authors to compare algorithms against mixed-integer programming (MILP) models in the scheduling literature. MILP models are often used as a baseline and their normally poor performance used as a justification for the introduction of metaheuristics. As a result, the literature on metaheuristic techniques is incredibly rich in the shop scheduling area. Constraint Programming (CP) models have gained momentum over the past two decades. Originally, CP models were used for Constraint Satisfaction Problems, mainly to determine whether a problem had a feasible solution or not. Also, CP models were used for optimization problems in that they could verify the optimality of their integer solutions but could not provide any bound for their feasible integer solutions. That changed with the introduction of paradigm-altering versions of CP engines, and more specifically, the CP Optimizer, which can provide optimality gaps for its integer feasible solutions. We took advantage of these characteristics and conducted a comparative evaluation between the performance of MILP and CP models for many different shop scheduling problems. The selected scheduling problems have very active research communities and the results of this research will help these areas in the design of more efficient algorithms. The tested scheduling problems comprise different settings occurring in various service and manufacturing industries. These problems include permutation flow shop scheduling (FSP), FSP with sequence-dependent setup times, non-permutation FSP, no-wait FSP with C_{max} minimization objective, FSP with total completion time and total tardiness objective functions, distributed FSP, hybrid flow shop, parallel machine scheduling, job shop scheduling and open shop scheduling. These problems range from pure sequencing, to joint sequencing-assignment, to pure assignment problems. Apart from the consideration of these problems because of their practical applications, the selection is also motivated by the fact that these problems have established and well-known benchmarks, for which high quality solutions exist. The third consideration when choosing these shop scheduling problems is that, due to their NP-hardness and overall difficulty, the vast majority of the extant literature consists of either heuristics or metaheuristics that provide no bounds or any optimality guarantee for the integer solutions found. In the absence of strong exact and general techniques in the literature on these scheduling problems, most researchers have chosen to compare their algorithms against (i) other sub-optimal algorithms and/or (ii) with the bounds obtained from the mathematical or constraint programming models developed for these problems. With that being said, it is of paramount importance to employ the best exact models in order to have the best possible bounds for comparison.

For each of the shop scheduling problems stated earlier, we have presented a MILP model and a CP model. With them, we solved standard and well-known benchmarks and evaluated their performance based on two measures: optimality gap and relative percentage deviation (RPD). It is essential to use the best known integer solution in the entire literature for each instance so as to be able to calculate the RPD. We reviewed all the papers published in the past 20 years that used these standard benchmarks so as to find the best known integer solution for each instance in each benchmark. The accompanying electronic excel file contains all these best integer solutions in the literature for future reference. After 37,956 results, we have obtained many improved integer solutions against which researchers can compare the performance of their algorithms. We have also provided insights on which scheduling problem models yield large average optimality gaps but at the same time low RPDs with respect to

http://joc.pubs.informs.org

best known integer solutions in the literature.

After an extensive computational campaign on around 38,000 problem instances gathered from the extant literature, we have shown the following:

- The no-wait or sequence-dependent setup time variants significantly hamper the performance of CP, while the impact on MILP is just the opposite. For example, the no-wait variant results in marginal performance improvements for MILP models.

- In the case of flow shop problems, CP models yield the lowest average optimality gaps for makespan but the highest for total tardiness. For total completion time, the optimality gap of the CP model is six times greater than that of the CP model with makespan. This suggests a large variability in the expected results in relation to the objective function to be optimized.

- CP models perform well in pure sequencing problems and are a superior alternative for solving joint sequencing-assignment problems. MILP models are superior when the scheduling problems only involve assignment decisions.

- The performance of MILP models is very sensitive to an increase in the number of jobs while the performance of the CP model is robust with respect to the number of jobs, but it is moderately influenced by the number of machines that each job must visit. The consideration of higher number of stages results in more precedence relations for models and this result implies that CP models are influenced by the precedence relations, more than the number of jobs, which is commonly considered as the problem size indicator in the scheduling literature. This seems to hold only for MILP models.

- Allocating more time to CP and MILP models improves performance in pure sequencing and pure assignment problems, respectively. In joint sequencing-assignment problems, the CP model obtains a lower optimality gap given the tested computational time limit, but the convergence trend indicates that for larger time limits, the MILP model surpasses the CP model for such problems. For shorter computational times, the CP model is preferable.

Recently, it has been shown that effective decomposition approaches can be developed when CP and MILP models are hybridized (see Roshanaei et al. (2020); Booth et al. (2016)). Future studies could include multiple factories, each of which including the shop floor scheduling problems that we studied. It would be an interesting avenue for future research if one could solve these shop floor scheduling problems with a solver such as SCIP that can hybridize the strength of both the CP and MILP models. We are currently planning to solve some of these scheduling problems, including the hybrid flow shop and distributed flow shop with logic-based Benders decomposition.

Acknowledgments

Rubén Ruiz is partially supported by the Spanish Ministry of Science, Innovation and Universities under grant "OPTEP-Port Terminal Operations Optimization" (No. RTI2018-094940-B-I00) financed with FEDER funds.

AppendixA. Detailed results

The following tables offer more detailed results on the different scheduling problems tested. In particular, the results are also broken down by number of jobs. Flow shop (FSP) is shown in Table A.7, non-permutation flow shop (N-FSP) in Table A.8, total completion time flow shop (TCT-FSP) in Table A.9, total tardiness flow shop (TT-FSP) in Table A.10, no-wait flow shop (NW-FSP) in Table A.11, SDST flow shop in Table A.12, distributed flow shop (D-FSP) in Table A.13, hybrid flow shops (H-FSP) in Table A.14, job shop (JSP) in Table A.15, open shop (OSP) in Table A.16 and parallel machines (PMSP) in Table A.17.

Table A.7: Results of CP and MILP models on the permutation flow shop (FSP).

Bench	$\|\mathcal{J}\|$	#	CP							MILP						
			State(#)				Gap	Time	RPD	State(#)				Gap	Time	RPD
			Er	Un	Fe	Op				Er	Un	Fe	Op			
TFS	20	30	0	0	11	19	**4.7**	1643	**0.28**	0	0	30	0	32.3	3600	1.12
	50	30	0	0	17	13	**3.24**	2264	**1.15**	0	0	30	0	70.74	3600	6.51
	100	30	0	0	20	10	**2.19**	2444	**1.37**	0	4	26	0	86.16	3600	10.28
	200	20	0	0	20	0	**3.69**	3601	**3.36**	0	20	0	0	-	3600	-
	500	10	0	0	10	0	**7.92**	3602	**8.41**	5	5	0	0	-	3600	-
Average		120	0	0	78	42	**4.35**	2710	**2.91**	5	29	86	0	63.07	3600	6.00
VFR	100	30	0	0	30	0	**16.26**	3600	**5.72**	0	25	5	0	80.87	3606	23.02
	200	30	0	0	30	0	**15.17**	3600	**9.06**	0	30	0	0	-	3600	-
	300	30	0	0	30	0	**15.38**	3600	**11.19**	2	28	0	0	-	3600	-
	400	30	0	0	30	0	**15.00**	3600	**11.86**	20	10	0	0	-	3600	-
	500	30	0	0	30	0	**14.8**	3600	**12.43**	25	5	0	0	-	3600	-
	600	30	0	0	30	0	**14.35**	3600	**12.64**	30	0	0	0	-	3600	-
	700	30	0	0	30	0	**13.91**	3600	**12.53**	30	0	0	0	-	3600	-
	800	30	0	0	30	0	**13.51**	3600	**12.36**	30	0	0	0	-	3600	-
Average		240	0	0	240	0	**14.80**	3600	**10.97**	137	98	5	0	-	3600	-

Er: Memory error, Un: Unknown, Fe: Feasible, Op: Optimal. Gap and RPD are both in %. Time is in seconds.

Table A.8: Results of CP and MILP models on the non-permutation flow shop (N-FSP).

Bench	$\|\mathcal{J}\|$	#	CP							MILP						
			State(#)				Gap	Time	RPD	State(#)				Gap	Time	RPD
			Er	Un	Fe	Op				Er	Un	Fe	Op			
TFS	20	30	0	0	17	13	**6.03**	2078	**0.00**	0	0	30	0	46.88	3600	8.14
	50	30	0	0	20	10	**4.63**	2421	**2.05**	0	0	30	0	75.53	3600	18.06
	100	30	0	0	20	10	**3.84**	2684	**2.89**	0	0	30	0	86.92	3600	17.67
	200	20	0	0	20	0	**6.63**	3601	**6.45**	0	0	20	0	91.76	3600	17.48
	500	10	0	0	10	0	**8.30**	3602	**8.71**	10	0	0	0	-	-	-
Average		120	0	0	87	33	**5.88**	2877	**3.90**	10	0	110	0	75.27	3600	15.34
VFR	100	30	0	0	30	0	**19.89**	3600	**8.95**	0	0	30	0	74.91	3609	22.17
	200	30	0	0	30	0	**17.83**	3600	**11.24**	15	0	15	0	88.10	3625	20.99
	300	30	0	0	30	0	**16.69**	3600	**11.73**	30	0	0	0	-	-	-
	400	30	0	0	30	0	**15.59**	3600	**11.57**	30	0	0	0	-	-	-
	500	30	0	0	30	0	**16.68**	3600	**10.79**	30	0	0	0	-	-	-
	600	30	0	0	30	0	**27.87**	3600	**10.32**	30	0	0	0	-	-	-
	700	30	0	0	30	0	**40.41**	3600	**9.84**	30	0	0	0	-	-	-
	800	30	0	0	30	0	**56.51**	3600	**9.46**	30	0	0	0	-	-	-
Average		240	0	0	240	0	**26.43**	3600	**10.49**	195	0	45	0	81.50	3617	21.58

Er: Memory error, Un: Unknown, Fe: Feasible, Op: Optimal. Gap and RPD are both in %. Time is in seconds.

Table A.9: Results of CP and MILP models on the total completion time flow shop (TCT-FSP).

Bench	$\|\mathcal{J}\|$	#	CP							MILP						
			State(#)				Gap	Time	RPD	State(#)				Gap	Time	RPD
			Er	Un	Fe	Op				Er	Un	Fe	Op			
TFS	20	30	0	0	30	0	**17.24**	3600	**0.56**	0	0	30	0	36.88	3600	1.28
	50	30	0	0	30	0	**21.85**	3600	**2.70**	0	0	30	0	68.28	3601	9.72
	100	30	0	0	30	0	**24.55**	3600	**4.63**	0	0	30	0	82.84	3612	20.34
	200	20	0	0	20	0	**29.66**	3601	**7.65**	0	20	0	0	-	3611	-
	500	10	0	0	10	0	**37.97**	3602	**17.83**	7	3	0	0	-	3698	-
Average		120	0	0	120	0	**26.25**	3601	**6.67**	7	23	90	0	62.67	3624	10.45
VFR	100	30	0	0	30	0	**32.10**	3601	-	0	21	9	0	76.28	3609	-
	200	30	0	0	30	0	**36.94**	3602	-	0	30	0	0	-	3610	-
	300	30	0	0	30	0	**51.80**	3603	-	6	24	0	0	-	3611	-
	400	30	0	0	30	0	**55.31**	3604	-	20	10	0	0	-	3612	-
	500	30	0	0	30	0	**70.95**	3605	-	29	1	0	0	-	3613	-
	600	30	0	0	30	0	**72.15**	3606	-	30	0	0	0	-	3614	-
	700	30	0	0	30	0	**73.12**	3607	-	30	0	0	0	-	3615	-
	800	30	0	0	30	0	**73.83**	3608	-	30	0	0	0	-	3616	-
Average		240	0	0	240	0	**58.28**	3605	-	145	86	9	0	76.28	3613	-

Er: Memory error, Un: Unknown, Fe: Feasible, Op: Optimal. Gap and RPD are both in %. Time is in seconds.

Table A.10: Results of CP and MILP models on the total tardiness flow shop (TT-FSP).

Bench	$\|\mathcal{J}\|$	#	CP							MILP						
			State(#)				Gap	Time	RPD	State(#)				Gap	Time	RPD
			Er	Un	Fe	Op				Er	Un	Fe	Op			
VRM	50	135	0	0	125	10	**71.18**	3337	**15.66**	0	0	135	0	90.67	3602	43.93
	150	135	0	0	122	13	**84.64**	3267	**25.44**	0	128	7	0	99.59	3644	86.17
	250	135	0	0	115	20	**82.46**	3096	**29.58**	0	120	15	0	100.00	3648	97.87
	350	135	0	0	114	21	**83.10**	3093	**30.79**	45	80	10	0	100.00	3671	98.82
Average		540	0	0	476	64	**80.35**	3198	**25.37**	45	328	167	0	97.57	3641	81.70

Er: Memory error, Un: Unknown, Fe: Feasible, Op: Optimal. Gap and RPD are both in %. Time is in seconds.

Table A.11: Results of CP and MILP models on the no-wait flow shop (NW-FSP).

Bench	$\|\mathcal{J}\|$	#	CP							MILP						
			State(#)				Gap	Time	RPD	State(#)				Gap	Time	RPD
			Er	Un	Fe	Op				Er	Un	Fe	Op			
TFS	20	30	0	0	30	0	**27.60**	3600	**0.49**	0	0	30	0	41.75	3600	1.41
	50	30	0	0	30	0	**30.89**	3600	**1.99**	0	0	30	0	77.54	3600	10.22
	100	30	0	0	30	0	**32.22**	3600	**4.42**	0	0	30	0	89.36	3602	19.33
	200	20	0	0	20	0	**41.79**	3600	**8.84**	0	0	20	0	94.40	3611	29.23
	500	10	0	0	10	0	**49.14**	3600	**11.67**	0	0	10	0	98.10	3682	71.33
Average		120	0	0	120	0	**36.33**	3601	**5.48**	0	0	120	0	80.23	3619	26.30
VFR	100	30	0	0	30	0	**55.36**	3600	-	0	0	30	0	84.12	3607	-
	200	30	0	0	30	0	**58.00**	3600	-	0	0	30	0	91.93	3627	-
	300	30	0	0	30	0	**64.05**	3600	-	0	0	30	0	95.08	3659	-
	400	30	0	0	30	0	**64.49**	3600	-	0	1	29	0	96.81	3706	-
	500	30	0	0	30	0	**70.18**	3600	-	0	20	10	0	98.04	3772	-
	600	30	0	0	30	0	**73.69**	3600	-	0	27	3	0	98.47	3852	-
	700	30	0	0	30	0	**72.82**	3600	-	10	20	0	0	-	3852	-
	800	30	0	0	30	0	**81.52**	3600	-	13	17	0	0	-	3955	-
Average		240	0	0	240	0	**67.51**	3600	-	23	85	132	0	94.07	3754	-

Er: Memory error, Un: Unknown, Fe: Feasible, Op: Optimal. Gap and RPD are both in %. Time is in seconds.

Table A.12: Results of CP and MILP models on the SDST flow shop (SDST-FSP).

Bench	$\|\mathcal{J}\|$	#	CP							MILP						
			State(#)				Gap	Time	RPD	State(#)				Gap	Time	RPD
			Er	Un	Fe	Op				Er	Un	Fe	Op			
RMA	20	120	0	0	116	4	**26.53**	3520	**1.93**	0	0	120	0	68.44	3600	3.45
	50	120	0	0	120	0	**28.72**	3601	**4.98**	0	0	120	0	85.54	3602	13.74
	100	120	0	0	120	0	**29.22**	3601	**5.89**	0	0	120	0	92.16	3603	19.14
	200	80	0	0	80	0	**33.41**	3603	**7.81**	0	2	78	0	94.99	3613	22.83
	500	40	0	0	40	0	**38.81**	3621	**13.26**	37	0	3	0	97.03	3716	20.14
Average		480	0	0	476	4	**31.34**	3589	**6.77**	37	2	441	0	87.63	3627	15.86

Er: Memory error, Un: Unknown, Fe: Feasible, Op: Optimal. Gap and RPD are both in %. Time is in seconds.

Table A.13: Results of CP and MILP models on the distributed flow shop (D-FSP).

Bench	$\|\mathcal{J}\|$	#	CP							MILP						
			State(#)				Gap	Time	RPD	State(#)				Gap	Time	RPD
			Er	Un	Fe	Op				Er	Un	Fe	Op			
NR	20	150	0	0	130	20	17.48	3345	1.07	0	0	136	14	**13.41**	3401	**0.85**
	50	150	0	0	150	0	**44.54**	3601	**8.34**	0	0	150	0	45.22	3603	9.97
	100	150	0	0	150	0	**63.83**	3602	**10.49**	0	0	115	0	76.89	3613	71.08
	200	100	0	0	100	0	**74.37**	3604	**15.14**	2	0	3	0	94.10	3664	462.49
	500	50	0	0	50	0	**84.84**	3613	**17.95**	50	0	0	0	-	-	-
Average		600	0	0	580	20	**57.01**	3553	**10.60**	52	0	404	14	57.40	3570	136.09

Er: Memory error, Un: Unknown, Fe: Feasible, Op: Optimal. Gap and RPD are both in %. Time is in seconds.

Table A.14: Results of CP and MILP models on the hybrid flow shop (H-FSP).

Bench	$\|\mathcal{J}\|$	#	CP							MILP						
			State(#)				Gap	Time	RPD	State(#)				Gap	Time	RPD
			Er	Un	Fe	Op				Er	Un	Fe	Op			
PRA	50	360	0	0	360	0	**44.28**	3601	**7.90**	0	0	360	0	57.13	3602	40.75
	100	360	0	0	360	0	**65.97**	3601	**6.32**	0	0	360	0	88.09	3609	213.62
	150	360	0	0	360	0	**75.22**	3601	**5.68**	0	0	208	0	94.15	3620	276.94
	200	360	0	0	360	0	**80.41**	3601	**4.94**	0	0	200	0	95.85	3636	317.97
Average		1440	0	0	1440	0	**66.47**	3601	**6.21**	0	0	1128	0	83.80	3617	212.32

Er: Memory error, Un: Unknown, Fe: Feasible, Op: Optimal. Gap and RPD are both in %. Time is in seconds.

Table A.15: Results of CP and MILP models on the job shop (JSP).

Bench	$\|\mathcal{J}\|$	#	CP							MILP						
			State(#)				Gap	Time	RPD	State(#)				Gap	Time	RPD
			Er	Un	Fe	Op				Er	Un	Fe	Op			
TJS	15	10	0	0	1	9	**0.74**	676	**0.00**	0	0	10	0	7.23	3462	0.56
	20	20	0	0	18	2	**7.49**	3281	**1.34**	0	0	20	0	22.61	3600	5.24
	30	20	0	0	19	1	**4.21**	3424	**2.35**	0	0	20	0	43.89	3601	13.19
	50	20	0	0	2	18	**0.07**	891	**0.07**	0	0	20	0	65.00	3601	23.38
	100	10	0	0	0	10	**0.00**	529	**0.00**	0	0	10	0	98.25	3604	1336.22
Average		80	0	0	40	40	**2.50**	1760	**0.75**	0	0	80	0	47.40	3574	275.72
DUM	20	20	0	0	20	0	**10.42**	3600	**2.77**	0	0	20	0	28.57	3600	7.95
	30	20	0	0	18	2	**7.52**	3275	**3.88**	0	0	20	0	48.85	3601	17.59
	40	20	0	0	12	8	**5.52**	2508	**3.66**	0	0	20	0	60.68	3601	24.47
	50	20	0	0	12	8	**5.15**	2371	**3.85**	0	0	20	0	68.92	3601	32.84
Average		80	0	0	62	18	**7.15**	2939	**3.54**	0	0	80	0	51.75	3601	20.71
LA	10	10	0	0	0	10	**0.00**	5	**0.00**	0	0	3	7	0.04	17	**0.00**
	15	15	0	0	0	15	**0.00**	60	**0.00**	0	0	15	0	17.35	3533	0.27
	20	10	0	0	1	9	**0.29**	397	**0.01**	0	0	10	0	37.98	3600	1.74
	30	5	0	0	0	5	**0.00**	2	**0.00**	0	0	5	0	53.99	3600	**0.00**
Average		40	0	0	1	39	**0.07**	116	**0.00**	0	0	33	7	27.34	2688	0.50
ORB	10	10	0	0	0	10	**0.00**	17	**0.00**	0	0	6	4	0.76	500	**0.00**
SWV	20	10	0	0	8	2	**7.62**	3201	**3.38**	0	0	10	0	43.19	3600	11.85
	50	10	0	0	5	5	**1.16**	1801	**1.17**	0	0	10	0	75.29	3601	14.09
YN	20	4	0	0	4	0	**11.64**	3600	**2.22**	0	0	4	0	20.82	3600	4.85
ABZ	10	2	0	0	0	2	**0.00**	7	**0.00**	0	0	0	2	**0.00**	30	**0.00**
	20	3	0	0	3	0	**7.18**	3600	**1.90**	0	0	3	0	28.42	3600	6.90
FMJ	6	1	0	0	0	1	**0.00**	0	**0.00**	0	0	0	1	**0.00**	0	**0.00**
	10	1	0	0	0	1	**0.00**	19	**0.00**	0	0	1	0	0.11	151	**0.00**
	20	1	0	0	0	1	**0.00**	2	**0.00**	0	0	1	0	47.11	3600	2.40
Average		42	0	0	20	22	**3.07**	1361	**0.96**	0	0	35	7	23.97	2076	4.45

Er: Memory error, Un: Unknown, Fe: Feasible, Op: Optimal. Gap and RPD are both in %. Time is in seconds.

Table A.16: Results of CP and MILP models on the open shop (OSP).

Bench	$\|\mathcal{J}\|$	#	CP							MILP						
			State(#)				Gap	Time	RPD	State(#)				Gap	Time	RPD
			Er	Un	Fe	Op				Er	Un	Fe	Op			
TOS	4	10	0	0	0	10	**0.00**	1	**0.00**	0	0	0	10	**0.00**	0	**0.00**
	5	10	0	0	0	10	**0.00**	2	**0.00**	0	0	0	10	**0.00**	2	**0.00**
	7	10	0	0	0	10	**0.00**	2	**0.00**	0	0	7	3	2.15	1952	**0.00**
	10	10	0	0	0	10	**0.00**	1	**0.00**	0	0	10	0	27.49	3600	**0.00**
	15	10	0	0	0	10	**0.00**	1	**0.00**	0	0	10	0	64.56	3600	1.86
	20	10	0	0	0	10	**0.00**	2	**0.00**	0	0	10	0	78.10	3601	8.01
Average		60	0	0	0	60	**0.00**	1	**0.00**	0	0	37	23	28.72	2126	1.64
GP	3	10	0	0	0	10	**0.00**	0	**0.00**	0	0	0	10	**0.00**	0	**0.00**
	4	10	0	0	0	10	**0.00**	1	**0.00**	0	0	0	10	**0.00**	0	**0.00**
	5	10	0	0	0	10	**0.00**	1	**0.00**	0	0	0	10	**0.00**	0	**0.00**
	6	10	0	0	0	10	**0.00**	2	**0.00**	0	0	0	10	**0.00**	1	**0.00**
	7	10	0	0	0	10	**0.00**	2	**0.00**	0	0	0	10	**0.00**	1	**0.00**
	8	10	0	0	0	10	**0.00**	5	**0.00**	0	0	0	10	**0.00**	5	**0.00**
	9	10	0	0	0	10	**0.00**	13	**0.00**	0	0	2	8	0.02	26	**0.00**
	10	10	0	0	0	10	**0.00**	24	**0.00**	0	0	5	5	0.05	810	**0.00**
Average		80	0	0	0	80	**0.00**	6	**0.00**	0	0	7	73	0.01	105	**0.00**
BHJ	3	8	0	0	0	8	**0.00**	0	**0.00**	0	0	0	8	**0.00**	0	**0.00**
	4	9	0	0	0	9	**0.00**	0	**0.00**	0	0	0	9	**0.00**	0	**0.00**
	5	9	0	0	0	9	**0.00**	1	**0.00**	0	0	0	9	**0.00**	1	**0.00**
	6	9	0	0	0	9	**0.00**	5	**0.00**	0	0	1	8	0.01	11	**0.00**
	7	9	0	0	0	9	**0.00**	277	**0.00**	0	0	5	4	1.77	959	0.04
	8	8	0	0	3	5	**0.68**	1614	**0.00**	0	0	8	0	5.00	3276	0.02
Average		52	0	0	3	49	**0.11**	316	**0.00**	0	0	14	38	1.13	708	0.01

Er: Memory error, Un: Unknown, Fe: Feasible, Op: Optimal. Gap and RPD are both in %. Time is in seconds.

Table A.17: Results of CP and MILP models on the paralell machines scheduling problem (PMSP).

Bench	$\|\mathcal{J}\|$	#	CP							MILP						
			State(#)				Gap	Time	RPD	State(#)				Gap	Time	RPD
			Er	Un	Fe	Op				Er	Un	Fe	Op			
FR	100	350	0	0	317	33	57.48	3275	0.85	0	0	44	306	**0.25**	488	**-0.13**
	200	350	0	0	347	3	74.81	3575	2.06	0	0	82	268	**0.26**	1010	**-0.12**
	500	350	0	0	350	0	88.73	3602	2.15	0	0	207	143	**0.26**	2280	**-0.10**
	1000	350	0	0	350	0	93.95	3604	1.86	0	0	205	145	**0.14**	2315	**-0.06**
Average		1400	0	0	1364	36	78.74	3514	1.73	0	0	538	862	**0.23**	1523	**-0.10**

Er: Memory error, Un: Unknown, Fe: Feasible, Op: Optimal. Gap and RPD are both in %. Time is in seconds.

References

Androutsopoulos, Konstantinos N., Eleftherios G. Manousakis, Michael A. Madas. 2020. Modeling and solving a bi-objective airport slot scheduling problem. *European Journal of Operational Research* **284**(1) 135 – 151.

Applegate, David, William Cook. 1991. A computational study of the job-shop scheduling problem. *ORSA Journal on Computing* **3**(2) 149–156.

Booth, Kyle E. C., Tony T. Tran, J. Christopher Beck. 2016. Logic-based decomposition methods for the travelling purchaser problem. Claude-Guy Quimper, ed., *Integration of AI and OR Techniques in Constraint Programming. International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2016)*. Springer, 55–64.

Bowman, E.H. 1959. Schedule-sequencing problem. *Operations Research* **7**(5) 612–614.

Brucker, Peter, Johann Hurink, Bernd Jurisch, Birgit Wöstmann. 1997. A branch & bound algorithm for the open-shop problem. *Discrete Applied Mathematics* **76**(1-3) 43–59.

Bukchin, Yossi, Tal Raviv. 2018. Constraint programming for solving various assembly line balancing problems. *Omega* **78** 57 – 68.

CPOptimizer. 2017. *IBM ILOG CPLEX Optimization Studio CP Optimizer User's Manual*. International Business Machines Corporation (IBM), 12th ed. URL https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.studio.help/pdf/usrcpoptimizer.pdf.

Demirkol, Ebru, Sanjay Mehta, Reha Uzsoy. 1998. Benchmarks for shop scheduling problems. *European Journal of Operational Research* **109**(1) 137 – 141.

Doulabi, S.H.H., L.-M. Rousseau, G. Pesant. 2016. A constraint-programming-based branch-and-price-and-cut approach for operating room planning and scheduling. *INFORMS Journal on Computing* **28**(3) 432–448.

Fanjul-Peyro, Luis, Rubén Ruiz. 2010. Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research* **207**(1) 55 – 69.

Fattahi, P., M.S. Mehrabad, F. Jolai. 2007. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of intelligent manufacturing* **18**(3) 331–342.

Gedik, Ridvan, Darshan Kalathia, Gokhan Egilmez, Emre Kirac. 2018. A constraint programming approach for solving unrelated parallel machine scheduling problem. *Computers & Industrial Engineering* **121** 139 – 149.

Guéret, C., C. Prins. 1999. A new lower bound for the open-shop problem. *Annals of Operations Research* **92**(0) 165–183.

Kreter, Stefan, Andreas Schutt, Peter J. Stuckey, Jürgen Zimmermann. 2018. Mixed-integer linear programming and constraint programming formulations for solving resource availability cost problems. *European Journal of Operational Research* **266**(2) 472 – 486.

Ku, Wen-Yang, J. Christopher Beck. 2016. Mixed integer programming models for job shop scheduling: A computational analysis. *Computers & Operations Research* **73** 165 – 173.

Laborie, P. 2015. Modeling and solving scheduling problems with CP optimizer. doi: 10.13140/RG.2.1.3984.0166.

Laborie, Philippe, Jérôme Rogerie, Paul Shaw, Petr Vilím. 2018. IBM ILOG CP optimizer for scheduling. 20+ years of scheduling with constraints at IBM/ILOG. *Constraints* **23**(2) 210–250.

Lawrence, S. 1984. Resource constrained project scheduling. *Technical report - Carnegie-Mellon University* .

Malapert, Arnaud, Hadrien Cambazard, Christelle Guéret, Narendra Jussien, André Langevin, Louis-Martin Rousseau. 2012. An optimal constraint programming approach to the open-shop problem. *INFORMS Journal on Computing* **24**(2) 228–244.

Manne, A.S. 1960. On the job-shop scheduling problem. *Operations Research* **8**(2) 219 – 223.

Meng, Leilei, Chaoyong Zhang, Yaping Ren, Biao Zhang, Chang Lv. 2020. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Computers & Industrial Engineering* **142** 106347.

Naderi, B., Rubén Ruiz. 2010. The distributed permutation flowshop scheduling problem. *Computers & Operations Research* **37**(4) 754 – 768.

Pan, C.H. 1997. A study of integer programming formulations for scheduling problems. *International Journal of Systems Science* **28**(1) 33–41.

Pan, Quan-Ke, Rubén Ruiz, Pedro Alfaro-Fernández. 2017. Iterated search methods for earliness and tardiness minimization in hybrid flowshops with due windows. *Computers & Operations Research* **80** 50 – 60.

Pour, Shahrzad M., John H. Drake, Lena Secher Ejlertsen, Kourosh Marjani Rasmussen, Edmund K. Burke. 2018. A hybrid constraint programming/mixed integer programming framework for the preventive signaling maintenance crew scheduling problem. *European Journal of Operational Research* **269**(1) 341 – 352.

Qin, Tianbao, Yuquan Du, Jiang Hang Chen, Mei Sha. 2020. Combining mixed integer programming and constraint programming to solve the integrated scheduling problem of container handling operations of a single vessel. *European Journal of Operational Research* **285**(3) 884 – 901.

Roshanaei, V. 2012. Mathematical modelling and optimization of flexible job shops scheduling problem. *Electronic Theses and Dissertations. 157.* URL https://scholar.uwindsor.ca/etd/157.

Roshanaei, V., K.E.C. Booth, D. Aleman, D. Urbach, J. C. Beck. 2020. Branch-and-check approaches for multi-level or planning and scheduling. *International Journal of Production Economics* **220** 107433.

Ruiz, Rubén, Concepción Maroto, Javier Alcaraz. 2005. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research* **165**(1) 34 – 54.

Samarghandi, Hamed, Mehdi Behroozi. 2017. On the exact solution of the no-wait flow shop problem with due date constraints. *Computers & Operations Research* **81** 141 – 159.

Schefers, Nina, Juan José Ramos González, Pau Folch, José Luis Munoz-Gamarra. 2018. A constraint programming model with time uncertainty for cooperative flight departures. *Transportation Research Part C: Emerging Technologies* **96** 170 – 191.

Stafford, E.F. 1988. On the development of a mixed-integer linear-programming model for the flowshop sequencing problem. *Journal of the Operational Research Society* **39**(12) 1163–1174.

Stafford, E.F., F.T. Tseng, J.N.D. Gupta. 2005. Comparative evaluation of milp flowshop models. *Journal of the Operational Research Society* **56**(1) 88–101.

Storer, Robert H., S. David Wu, Renzo Vaccari. 1992. New search spaces for sequencing problems with application to job shop scheduling. *Management Science* **38**(10) 1495–1509.

Taillard, E. 1993. Benchmarks for basic scheduling problems. *European Journal of Operational Research* **64**(2) 278 – 285.

Tofighian, Ali Asghar, B. Naderi. 2015. Modeling and solving the project selection and scheduling. *Computers & Industrial Engineering* **83** 30 – 38.

Tseng, F.T., E.F. Stafford. 2008. New milp models for the permutation flowshop problem. *Journal of the Operational Research Society* **59**(10) 1373–1386.

Tseng, F.T., E.F. Stafford, J.N.D. Gupta. 2004. An empirical analysis of integer programming formulations for the permutation flowshop. *Omega—International Journal of Management Science* **32**(4) 285–293.

Unsal, Ozgur, Ceyda Oguz. 2019. An exact algorithm for integrated planning of operations in dry bulk terminals. *Transportation Research Part E: Logistics and Transportation Review* **126** 103 – 121.

Valicka, Christopher G., Deanna Garcia, Andrea Staid, Jean-Paul Watson, Gabriel Hackebeil, Sivakumar Rathinam, Lewis Ntaimo. 2019. Mixed-integer programming models for optimal constellation scheduling given cloud cover uncertainty. *European Journal of Operational Research* **275**(2) 431 – 445.

Vallada, Eva, Rubén Ruiz, Jose M. Framinan. 2015. New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research* **240**(3) 666 – 677.

Vallada, Eva, Rubén Ruiz, Gerardo Minella. 2008. Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers & Operations Research* **35**(4) 1350 – 1373.

Wagner, H.M. 1959. An integer linear-programming model for machine scheduling. *Naval Research Logistics* **6**(2) 131 – 140.

Wang, Tao, Nadine Meskens, David Duvivier. 2015. Scheduling operating theatres: Mixed integer programming vs. constraint programming. *European Journal of Operational Research* **247**(2) 401 – 413.

Wilson, J.M. 1989. Alternative formulations of a flowshop scheduling problem. *Journal of the Operational Research Society* **40**(4) 395–399.

Yamada, T., R. Nakano. 1992. A genetic algorithm applicable to large-scale job-shop instances. *In R. Männer & B. Manderick (Eds.), Parallel instance solving from nature 2. Amsterdam: Elsevier* 281–290.

Younespour, Maryam, Arezoo Atighehchian, Kamran Kianfar, Ehsan T. Esfahani. 2019. Using mixed integer programming and constraint programming for operating rooms scheduling with modified block strategy. *Operations Research for Health Care* **23** 100220.