Mixed-Integer Programming Versus Constraint Programming for Shop 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. 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 guaranteeing the optimality of a solution, but they could not provide bounds for the integer feasible solutions found if interrupted prematurely due to, say, timelimits. New versions, however, provide bounds and optimality guarantees, effectively making CP a viable alternative to more traditional mixed-integer programming (MIP) models and solvers. We capitalize on these developments and conduct a computational evaluation of MIP and CP models on 12 select scheduling problems.[1] We carefully chose these 12 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 (e.g., distributed flow shop and hybrid flow shop) to pure assignment (i.e., parallel machine) scheduling problems. We present MIP 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 6,623 experiments and evaluates the MIP and CP models along five dimensions of problem characteristics, objective function, decision variables, input parameters, and quality of bounds. We establish the areas that each one of these models performs well and recognize their conceivable reasons. The obtained results indicate that CP sets new limits concerning the maximum problem size that can be solved using off-the-shelf exact techniques.

Keywords: Shop scheduling, Flow Shop, Job Shop, Flexible Job Shop, Open Shop, Parallel Machines, Benchmarks, Constraint Programming, Mixed Integer Programming.

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 programming (MIP) 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;

*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@utoronto.ca (Vahid Roshanaei)

[1] We note that comparison is conducted using the *default* settings of solvers.

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 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, MIP 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: classic (integer-based) and modern (interval-based). The classical CP approach uses integer variables and global constraints such as *disjunctive* and/or *cumulative* while the modern approach, as in the IBM CP Optimizer uses interval variables and interval-specific functions (CPOptimizer, 2017; Laborie, 2015). Laborie et al. (2018) discuss the advantages of the CP Optimizer approaches over classical CP approaches 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. In fact, for all our scheduling problems that do not contain "machine allocation" dimension, the two formulations are very similar and there is no real advantage of using an interval-based formulation.[2] The additional advantage of the interval-based formulation is when the problem has an allocation dimension that can be efficiently formulated with optional interval variables and alternative constraints (i.e., D-FSP, H-FSP, and PMSP).

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 bounding mechanism—a mechanism similar to the LP relaxation within mathematical programming solvers, that provides optimality gaps for the integer solutions found before optimality. 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. The idea of calculating bounds for found integer solutions by CP has existed since the seminal work of Hooker and Yan (2002) that calculates bounds via the relaxations of global constraints. Fontaine et al. (2016) have used bounds in hybrid solvers for solving scheduling problems. Despite these early advances, it took a few years for developers of CP solvers to incorporate these ideas. As a result, classical off-the-shelf CP solvers could only provide optimality proofs for those scheduling problems which were optimally solvable within the timelimit; otherwise, no bound could be provided for the integer solutions found before the timelimit is reached. 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 (MIPs) that have been the primary modeling 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 MIP performs similarly to CP for problems of moderate sizes. However, CP is superior to MIP for 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 MIP models. They conclude that one of the MIP models outperforms the other models including two of the CP models they developed. Laborie et al.

[2]In particular, for the JSP problem, both formulations are basically equivalent. So for a problem like the JSP, what really makes the difference between CP solvers is the automatic search, not the formulation (and the CP Optimizer automatic search is known to be particularly efficient for scheduling problems).

(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 compared it with MIP models. Laborie (2018) provides an update on the comparison of different approaches for solving a well-known allocation and scheduling problem, demonstrating that with the recent advances in the automatic search within the CP Optimizer, a standalone simple CP model outperforms all existing approaches for cost-based resource-constrained mixed allocation-sequencing problems.

CP-based models and algorithms have also been applied to 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; Naderi et al., 2021; Roshanaei et al., 2020), resource availability cost problems (Kreter et al., 2018) and container scheduling (Qin et al., 2020). The performance evaluation of CP Optimizer (among other CP versions) has garnered significant attraction. For instance, Laborie and Godard (2007) describe the early version of the automatic search of CP Optimizer and evaluate it on 21 different scheduling benchmarks (including N-FSP, H-FSP, JSP, OSP, and a generalization of SDST-FSP), whereas Vilim et al. (2015) include experiments with 7 classical scheduling benchmarks including JSP, OSP and several of their extensions, including different variants of RCPSP. In these papers, CP Optimizer was compared against the current state-of-the-art problem-specific approaches. The originality and the interest of the present paper is to focus on a direct comparison between a CP Optimizer and two popular MIP solvers, Gurobi 9.1.2 and CPLEX 20.1, two generic exact approaches for solving MIP models.

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 the performance of CP and MIP 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 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 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 can 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, whereas others are pure assignment problems. For example, the open shop is a pure sequencing problem with no precedence constraints among operations of a job. 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 and the flexible job 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: each stage consists of functionally identical machines, which means more assignments for each job. Flexible job shops differ from the previous two scheduling problems in that the processing route of different jobs is arbitrary and the number of operations varies from one job to another. Additionally, the processing time of each operation depends on the processor to which it is assigned. Thus, one needs to make a processor assignment decision for each particular operation and the sequencing decision for the given set of operations assigned to each processor. The heterogeneity in processor performances and the arbitrary route of jobs on the shop floor render the flexible job shop as one of the most intractable scheduling problems. 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 evaluate the performance and limits of both CP and MIP models.

Contributions, Scope, and Scale of this study. This paper offers quadruple contributions to the literature. We study 12 *disjunctive shop* scheduling problems that represent a wide variety of scheduling problems occurring in different service and manufacturing settings.[3] These scheduling settings range diversely from pure sequencing (e.g., flow shop and open shop) to joint assignment-sequencing (e.g., distributed flow shop and hybrid flow shop) to pure assignment (i.e., parallel machine) scheduling problems. **First**, we ascertain *which* solver performs generally better for our select *disjunctive shop* scheduling problems.[4] To do so, we compare MIP models on well-established and state-of-the-art mathematical programming solvers: Gurobi 9.1.2 and CPLEX 20.1. Having compared the two solvers and ascertained the best mathematical programming solver, we compare it against the most popular constraint programming solver, CP Optimizer 20.1. This comparison is especially interesting as the CP Optimizer 20.1 is now providing bounds on the quality of its feasible solutions—a valuable feature that did not exist in CP Optimizer versions before CP Optimizer 12.8. This comparison entails developing equivalent CP models for each scheduling problem. This comparison will reveal as to which solver is *generally* able to achieve better performance. Another purpose that this analysis serves is that in case CP models are recognized as a superior modeling choice, one can determine whether the poor performance of MIP models is ascribable to either the MIP Technology itself as a poor modeling choice or the mathematical programming solvers or both of them.[5] **Second**, we determine the best solver for each *particular* scheduling problem. This

[3]Our comparative study includes disjunctive shop scheduling problems; as such, we do not compare the performance of MIP and CP models on *cumulative* scheduling problems (i.e., a machine can process simultaneously more than one operation. We also only test the performance of our CP models using CP Optimizer 20.1 as this is the most popular CP solver in the literature. The comparison of CP models on different CP solvers does not fall within the scope of this study.

[4]Given the high-level nature of this study, we do not seek to ascertain *why* a certain solver outperforms others.

[5]We note that all these comparisons are based on the default settings of the chosen solvers. Comparing these models on different solver's settings requires a rigorous analysis for each of these scheduling problems and warrants a separate paper for each problem (see e.g., Ku and Beck (2016) for job shop scheduling problem). We thus do not seek to conduct such detailed analyses in this paper.

analysis is especially useful if no solver yields a universally superior performance on all scheduling problems in which case scheduling practitioners need to be selective in choosing their solvers. **Third**, we quantify the exact amount of improvement that each solver can achieve on each scheduling problem and make these results publicly accessible, which will hopefully foster more comparisons between the performance of existing metaheuristics with our new bounds and solutions. Our new bounds and feasible solutions pave the ground for more accurate evaluations of the quality of existing metaheuristics' integer solutions, encouraging hopefully more researchers to employ exact solvers instead of randomized search techniques. **Fourth**, we provide insights on how performance of different scheduling problems are influenced by the choice of, e.g., problem characteristics, objective functions, problem decisions, and problem sizes. We provide these insights by applying our MIP and CP models to 6,623 famous problem instances gathered from the scheduling literature. We believe the balance of the *scope* (the number of scheduling problems and solvers considered) and the *scale* (the number of problem instances considered for each problem) of this paper is sufficient to convey our insights and will be of interest to the general scheduling community.

A summary of findings. In this paper, we show that MIP 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 11 considered scheduling problems, CP models are clearly better and sometimes by significant margins. The use of CP models instead of MIP 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 select 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 works best. We conclude the paper in Section § 5 and provide future directions for the current study. We include some of our alternative CP models for these scheduling problems in Appendix A.

2. Preliminaries

In order for the comparative study to be self-contained, some basic preliminaries are given for the MIP 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 modeling approach can be seen as a different alternative to how these variables are defined. In the following, we loosely group the most popular modeling approaches: position-, time-, sequence- and Manne-based. For ease of exposition, we explain them using a numerical example with four jobs and a single machine.

- **Position-based** (Wagner, 1959): This approach models the sequencing problem as an assignment problem, where jobs are assigned to the positions in a 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 the k-th position 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 succeeds 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 binary 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' succeeds job j ($j' > j$); and 0 otherwise. It also does not need the dummy job 0. Obviously, if job j succeeds job j', then job j' does not necessarily 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 starts its processing 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 priori and based on the scheduling problem studied. Setting T ranges from being relatively straightforward to

rather complex.

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

An ideal model is one that simultaneously possesses a strong LP relaxation and a low number of branching operations (to ensure integrality of solutions), a mutually exclusive scenario for many scheduling models. The existence of a higher number of active constraints in a model renders the LP relaxation of the model likely stronger, at the expense of increasing the number of extreme points (iterations)—a computationally expensive task when Simplex is to be used as the solution technique for solving the LP relaxation. On the other hand, the existence of a high number of binary variables renders the branching operation extremely difficult, because the number of branching operations increases exponentially in the number of binary variables. Note that after each branching, an LP relaxation of the problem (if feasible) must be solved to hopefully find an integral solution. The use of strong LP relaxation makes sense if it causes the pruning procedure within the branch-and-cut of the mathematical programming solver to effectively eliminate those fruitless branches that do not contain the optimal solution. Rarely can we find scheduling models having a tight LP relaxation assisting with branching operations.[6] This is partially attributable to the huge number of disjunctive (big-M) constraints (that must be used in the formulation of scheduling problems to avoid overlapping of jobs) that disproportionately increases the number of extreme points without actually tightening the convex hull of the LP relaxation model. This situation can be exacerbated if a weak LP relaxation of a model is accompanied by an astronomical number of branching operations. As such, an acceptable scheduling model must strike an effective balance between its branching efficiency and LP relaxation quality.

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. Additionally, the discretization of time causes the obtained solutions to be *approximate* rather than being *exact*. However, this approach is helpful for scheduling problems with preemptive jobs, i.e., jobs can be interrupted during processing. For example, it is useful for project scheduling problems (Tofighian and Naderi, 2015). The Manne-based model 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. Empirically, over the years and on different scheduling problems, researchers have shown that Manne-based models with higher branching efficiency are better than the Sequence-based models with fewer big-M constraints for solving scheduling problems, stressing counter-intuitively on the development of models that possess fewer variables than constraints (Stafford, 1988; Stafford et al., 2005; Wilson, 1989; Pan, 1997; Tseng et al., 2004; Tseng and Stafford, 2008; Naderi and Ruiz, 2010; Roshanaei et al., 2013; Ku and Beck, 2016).[7] The sequence-based

[6] May be only scheduling problems whose objective function values are principally influenced by assignment decisions for which strong formulations exist—a problem like Parallel Machine Scheduling Problem.

[7] According to our previous experience with working with some of these select models, we learned that Manne-based modeling leaves no room for competition for other modeling paradigms, especially when the size of jobs and machines is significantly large. Furthermore, Roshanaei and Naderi (2021) and Roshanaei et al. (2020) have recently shown that the time-based modeling approach is substantially inferior to the Manne-based modeling paradigm on a dual resource-constrained operating room scheduling problem. A similar conclusion has been made for JSPs (Ku and Beck, 2016).

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. 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. The issue with the Position- and Sequence-based models is that the solver expends significant branching efforts to find an integer feasible solution. According to the vast literature on the effectiveness of Manne-based models and in view of our previous experiences with these select scheduling models, we assertively declare Manne-based models as being generally superior to other MIP modeling approaches. Hence, all of our MIP models will be Manne based.

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[8]: dry bulk terminals (Unsal and Oguz, 2019), container yard scheduling (Qin et al., 2020) and operating room scheduling (Younespour et al., 2019; Naderi et al., 2021; Roshanaei et al., 2020) 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 MIP 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 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 constraint *PresenceOf(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).

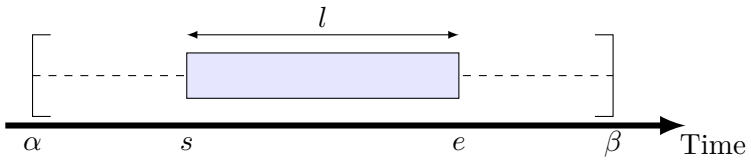


Figure 1: An interval variable with a length of $l = e - s$ within (α, β) where s and e are the start and end time of the interval variable respectively.

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

[8]Note that there are other CP solvers that could be used for our comparison purposes; however, we chose CP Optimizer as the most popular CP solver for our computational comparison purposes. The determination of the best choice of CP solver for our scheduling problems does not fall within the scope of our study. This matter can certainly be pursued as an interesting future study.

different settings and to exploit them by using the automatic search algorithm in CP (CPOptimizer, 2017). Table 1 shows the functions and global constraints used in the CP models (IBM, 2016).

Table 1: Functions and global constraints used in the CP models presented.

Global constraints	
EndBeforeStart	it constrains minimum delay between the end of one interval variable and start of another one.
EndAtStart	it constrains the delay between the end of one interval variable and start of another one.
Alternative	it creates an alternative constraint between interval variables.
NoOverlap	it constrains a set of interval variables not to overlap each others.
SameSequence	it creates a same-sequence constraint between two sequence variables.
Functions	
EndOf	it returns the end of an interval variable.
Pulse	it returns an interval variable (or a fixed interval) whose value is equal to 0 outside the interval and equal to a non-negative constant on the interval.
Element	it returns an element of an array.
PresenceOf	it returns the presence status of an interval variable.

In the integer-based formulation, we convert a scheduling problem into a set of interconnected binary decisions while the interval based CP formulation uses intervals to represent operations and global constraints to apply complex sequencing aspects which are commonly hard to represent by binary decisions.

3. Models

The two main generic assumptions across all *disjunctive shop* 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—job precedence,	(2)
	Ensure type-2 non-overlapping—machine precedence,	(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 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: a processing station with homogeneous or heterogeneous set of machines. 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 2 contains the notation used in the models. Table 3 also shows the different decision variables used in MIP models.

Table 2: Sets and parameters for the MIP 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 3: Decision variables used in the MIP models presented.

Sequencing	
$x_{jj'}$	1 if job j is processed after job j', 0 otherwise ($j > j'$)
$x_{ijj'}$	1 if machine i processes job j after job j', 0 otherwise ($j > j'$)
$h_{jii'}$	1 if for job j visits machine i after machine i', 0 otherwise ($i > i'$)
$z_{jj'}$	1 if job j is processed immediately after job j', 0 otherwise ($j \neq j'$)
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

As per our rationalization in the previous sections, we only present Manne-based versions of our scheduling models. For the SDST-FSP, though, we use the sequence-based modeling paradigm as Manne-based formulation is not possible for this problem.

3.1. Permutation Flow Shop Problem (FSP)

In a (permutation or regular) flow shop problem (FSP) a set of jobs must 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 to the last one. In permutation flow shops, 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. MIP model

It is well known in the literature that Manne-based modeling approach for the permutation flow ship is superior to its position-based counterpart (Pan, 1997; Stafford et al., 2005). As such, we use the Manne-based modeling 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 for FSP is $\frac{|\mathcal{J}|^2-|\mathcal{J}|}{2}$.[9] The mixed-integer programming (MIP) model for the FSP is as follows:

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

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

We develop a constraint programming (CP) model for the FSP as follows:

$$
\begin{aligned}
\text{minimize} \quad & C_{\max}, && && (\text{CP}_{\text{FSP}})\\
\text{subject to} \quad & Task_{ji} = \text{IntervalVar}(P_{ji}) && j \in \mathcal{J}, i \in \mathcal{I}, && (13)
\end{aligned}
$$

[9] According to Ku and Beck (2016) determining the performance of the MIP models based on the number of binary variables and constraints is a good first-step analysis, but it is error prone, hence requiring rigorous empirical evaluation, especially when modern MIP solvers are involved.

$$\text{EndBeforeStart}(Task_{ji}, Task_{ji-1}) \qquad \forall j \in \mathcal{J}, i \in \mathcal{I} \setminus 1, \tag{14}$$
$$\text{NoOverlap}(SV_i) \qquad \forall i \in \mathcal{I}, \tag{15}$$
$$SV_i = \text{SequenceVar}(Task_{ji} : j \in \mathcal{J}) \qquad \forall i \in \mathcal{I}, \tag{16}$$
$$\text{SameSequence}(SV_i, SV_{i-1}) \qquad \forall i \in \mathcal{I} \setminus 1, \tag{17}$$
$$C_{\max} = \max_{(j)} \left(\text{EndOf}(Task_{j|\mathcal{I}|})\right). \tag{18}$$

Constraints (13) define 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 (15) 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 (16) 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 function "EndOf" over interval variables of jobs at the last stage $|\mathcal{I}|$.

3.2. Non-permutation (general) FSPs (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 MIP model's number of binary variables very sensitive to the number of stages.

3.2.1. MIP 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 MIP model with constraint sets (19), (20) and (21). The MIP for N-FSP therefore becomes:

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

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:

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

3.3. Total Completion Time FSP (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. MIP model

The MIP model for TCT objective function ($\text{MIP}_{\text{TCT-FSP}}$) is as follows:

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

3.3.2. CP model

The CP model for TCT ($\text{CP}_{\text{TCT-FSP}}$) is as follows:

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

As can be observed, the 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 FSP (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 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. MIP model

The MIP model for TT objective function ($\text{MIP}_{\text{TT-FSP}}$) is as follows:

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

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 FSP (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 an NW-FSP, we only need to enforce the no-wait restriction in addition to avoiding type-1 overlapping.

3.5.1. MIP model

The MIP model for the NW-FSP with $C_{\max}$ objective function ($\text{MIP}_{\text{NW-FSP}}$) is as follows:

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

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:

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

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 Time 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. MIP model

We formulated the previous sequencing models using the Manne-based modeling approach as we did not require 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 from a mathematical perspective; 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 MIP with the $C_{\max}$ objective function for the SDST-FSP is the following:

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

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:

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

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 constraint "NoOverlap".

3.7. Hybrid FSP (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 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* to decide which machine at each stage processes a job. Therefore, the H-FSP is a joint assignment/sequencing problem unlike the previous sequencing-only problems.

3.7.1. MIP 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 MIP model for the H-FSP with the $C_{\max}$ objective function is as follows:

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

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. This model needs $\frac{|\mathcal{J}|^2-|\mathcal{J}|}{2} \times |\mathcal{I}|$ binary sequencing variables and $|\mathcal{J}| \times \sum_i |\mathcal{M}_i|$ binary assignment variables.

3.7.2. CP model

We initially formulated the H-FSP using global constraint "Alternative" and "NoOverlap" (see Appendix A). Such a formulation yielded poor performance due to a high number of assignment variables. To achieve better performance for the CP model for the H-FSP, one could exploit the special structure within the H-FSP to circumvent the complexity associated with the optimization of assignment decisions and develop a model that is more computationally efficient than some of the pure scheduling problems (see the seminal work of (Laborie et al., 2018) on this matter). We used the global constraint "Pulse" which is widely used in the formulation of cumulative scheduling problems.[10] This CP model for the H-FSP with the $C_{\max}$ objective function is as follows:

$$\begin{aligned}
&\text{minimize} && C_{\max}, && && (\text{CP}_{\text{H-FSP}})\\
&\text{subject to} && \text{Constraints (13), (14) and (18),} && &&\\
& && \sum_{j\in\mathcal{J}} \text{Pulse}(Task_{ji}, 1) \leq |\mathcal{M}_i| && \forall i \in \mathcal{I}. && (37)
\end{aligned}$$

In constraint sets (37), we use function"Pulse" to limit the cardinality of simultaneous jobs being processed at any stage of H-FSP to the number of *identical* machines in each stage.

3.8. Distributed FSP (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., an 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.

[10] As stated in the contribution, we had no intention of studying cumulative scheduling problems. However, since the H-FSP can be efficiently modeled using advances made in the cumulative scheduling literature, we formulated H-FSP as if all machines in each stage i function as a super machine with a cumulative capacity of processing $|\mathcal{M}_i|$ jobs at any point of time.

3.8.1. MIP model

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

$$\begin{aligned}
&\text{minimize} && C_{\max}, && && (\text{MIP}_{\text{D-FSP}})\\
&\text{subject to} && \text{Constraints (6) and (7) and (10) - (12)}, && &&\\
& && \sum_{f\in\mathcal{F}} q_{jf} = 1 && \forall j \in \mathcal{J}, && (38)\\
& && c_{ji} \geq c_{j'i} + P_{ji} - M(3 - x_{jj'} - q_{jf} - q_{j'f}) && \forall i \in \mathcal{I}, f \in \mathcal{F}, j, j' \in \mathcal{J} : j > j', && (39)\\
& && c_{j'i} \geq c_{ji} + P_{j'i} - M(2 + x_{jj'} - q_{jf} - q_{j'f}) && \forall i \in \mathcal{I}, f \in \mathcal{F}, j, j' \in \mathcal{J} : j > j', && (40)\\
& && q_{jf} \in \{0,1\} && \forall i \in \mathcal{I}, f \in \mathcal{F}. && (41)
\end{aligned}$$

Constraint (38) assigns each job to exactly one of the existing processing lines. Constraints (39) and (40) avoid type-2 overlapping for the jobs assigned to the same processing line. This model needs $\frac{|\mathcal{J}|^2-|\mathcal{J}|}{2}$ binary sequencing variables and $|\mathcal{J}| \times |\mathcal{F}|$ binary assignment variables.

3.8.2. CP model

We develop the CP model for the D-FSP ($\text{CP}_{\text{D-FSP}}$) as follows:

$$\begin{aligned}
&\text{minimize} && C_{\max}, && && (\text{CP}_{\text{D-FSP}})\\
&\text{subject to} && \text{Constraints (14) and (18)}, && &&\\
& && Task^*_{jif} = \text{IntervalVar}(P_{ji}, \text{Optional}) && j \in \mathcal{J}, i \in \mathcal{I}, f \in \mathcal{F}, && (42)\\
& && \text{PresenceOf}(Task^*_{j1f}) = \text{PresenceOf}(Task^*_{jif}) && \forall j \in \mathcal{J}, i \in \mathcal{I} \setminus 1, f \in \mathcal{F}, && (43)\\
& && \text{Alternative}(Task_{ji}, Task^*_{jif} : f \in \mathcal{F}) && \forall j \in \mathcal{J}, i \in \mathcal{I}, && (44)\\
& && SV_{if} = \text{SequenceVar}(Task^*_{jif} : j \in \mathcal{J}) && \forall i \in \mathcal{I}, f \in \mathcal{F}, && (45)\\
& && \text{NoOverlap}(SV_{if}) && \forall i \in \mathcal{I}, f \in \mathcal{F}, && (46)\\
& && \text{SameSequence}(SV_{if}, SV_{i-1f}) && \forall i \in \mathcal{I} \setminus 1, f \in \mathcal{F}. && (47)
\end{aligned}$$

Constraint (42) defines one interval variable for each operation at each stage. Constraint (43) ensures that if a job is assigned to a line, all its operations are processed on that line and constraint (44) 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 MIP and CP models require modifications to accommodate such non-ordered sets of stages in the JSP. To avoid type-1 overlapping, we replace $i-1$ with i'.

Table 4: Additional notation for the F-JSP model

Sets:	
$\mathcal{K}_j$	Set of operations of job j, $k \in \mathcal{K}_j$
$\mathcal{I}$	Set of machines, $i \in \mathcal{I}$
$\mathcal{I}_{jk}$	Set of eligible machines for operation O_{jk}
Parameters:	
P_{jki}	The processing times of operation O_{jk} on machine i

3.9.1. MIP model

The Manne-based MIP for the JSP (MIP_{JSP}) as proposed in Ku and Beck (2016)[11] is as follows:

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

Constraints (48) respects the processing route of a job. Note that i' denotes the stage before stage i in the job j' processing route. This model needs $\frac{|\mathcal{J}|^2-|\mathcal{J}|}{2} \times |\mathcal{I}|$ binary sequencing variables.[12]

3.9.2. CP model

The CP for the JSP (CP_{JSP}) is as follows:

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

3.10. Flexible JSP (F-JSP)

In the F-JSP, each operation of a job can be processed by one of the eligible machines for that operation. A such, the F-JSP entails making two decisions: assignment and sequencing. Once operations are assigned to each machine, the sequence of the assigned operations is optimally determined. Table 4 and 5 show the notation and decision variables used in our MIP F-JSP model, respectively.

3.10.1. MIP model

The Manne-based MIP model that we present for the F-JSP ($\text{MIP}_{\text{F-JSP}}$) is due to Roshanaei et al. (2013) and is as follows:

$$\min \text{C}_{\max} \qquad (\text{MIP}_{\text{F-JSP}})$$

[11] For notational consistency, we presented the model with completion times of operations rather than their starting times.

[12] Readers are refereed to the work of Ku and Beck (2016) for rigorous comparison among time-based, position-based, sequenced-based, and Manned-based MIP models for JSP using Gurobi, CPLEX, and SCIP. In this paper, the authors show that Manne-based model is superior to all other MIP models.

Table 5: Decision variables used in the MIP model for F-JSP.

Sequencing	
$x_{jkj'k'}$	1 if the kth operation of job j is processed after the k'th operation of job j', 0 otherwise ($j > j'$)
Scheduling	
c_{jk}	The completion time of the kth operation of job j
Assignment	
z_{jki}	1 if the kth operation of job j is processed by machine i, 0 otherwise

$$\text{s.t.} \sum_{i \in \mathcal{I}_{jk}} z_{jki} = 1 \qquad \forall j \in \mathcal{J}, k \in \mathcal{K}_j \tag{50}$$

$$c_{jk} \geq c_{jk-1} + \sum_{i \in \mathcal{I}_{jk}} P_{jki} z_{jki} \qquad \forall j \in \mathcal{J}, k \in \mathcal{K}_j \tag{51}$$

$$c_{jk} \geq c_{j'k'} + P_{jki} - V(3 - x_{jkj'k'} - z_{jki} - z_{j'k'i}) \forall j > j' \in \mathcal{J}, k \in \mathcal{K}_j, k' \in \mathcal{K}_{j'}, i \in \mathcal{I}_{jk} \cap \mathcal{I}_{j'k'} \tag{52}$$

$$c_{j'k'} \geq c_{jk} + P_{j'k'i} - V(2 + x_{jkj'k'} - z_{jki} - z_{j'k'i}) \; \forall j > j' \in \mathcal{J}, k \in \mathcal{K}_j, k' \in \mathcal{K}_{j'}, i \in \mathcal{I}_{jk} \cap \mathcal{I}_{j'k'} \tag{53}$$

$$\mathrm{C}_{\max} \geq c_{jk} \qquad \forall j \in \mathcal{J}, k \in \mathcal{K}_j \tag{54}$$

$$c_{jk} \geq 0 \qquad \forall j \in \mathcal{J}, k \in \mathcal{I}_j \tag{55}$$

$$x_{jkj'k'}, z_{jki} \in \{0, 1\} \qquad \forall j > j' \in \mathcal{J}, k \in \mathcal{K}_j, k' \in \mathcal{K}_{j'}, i \in \mathcal{I} \tag{56}$$

Constraints (50) assign each operation to an eligible machine. Constraints (51) ensure that there is no overlap in the starting times of operations of a job. Constraints (52) and (53) ensure that operations of different jobs assigned to the same machine will not overlap. Constraints (54) calculate makespan, which corresponds to the maximum completion time of each individual machine. Constraints (55) and (56) define the nature of decision variables.

We cannot provide a precise parametric formula for the required number of variables for F-JSP as it depends on the number of operations of different jobs and the flexibility rate of machines.

3.10.2. CP model

The CP model that we present for the F-JSP ($\text{CP}_{\text{F-JSP}}$) is due to Naderi and Roshanaei (2021) and is as follows:

$$\text{minimize} \quad \mathrm{C}_{\max}$$

$$\text{subject to} \quad \text{Task}_{jki} = \text{IntervalVar}(P_{jki}, \text{Optional}) \qquad \forall j \in \mathcal{J}, k \in \mathcal{K}_j, i \in \mathcal{I}_{jk} \tag{57}$$

$$\text{Alternative}(Task^*_{jk}, \{Task_{jki} : i \in \mathcal{I}_{jk}\}) \qquad \forall j \in \mathcal{J}, k \in \mathcal{K}_j \tag{58}$$

$$\text{EndBeforeStart}(Task_{jk-1}, Task_{jk}) \qquad \forall j \in \mathcal{J}, k \in \mathcal{K}_j \tag{59}$$

$$\text{NoOverlap}(Task_{jki} : j \in \mathcal{J}, k \in \mathcal{K}_j | i \in \mathcal{I}_{jk}) \qquad \forall i \in \mathcal{I} \tag{60}$$

$$\mathrm{C}_{\max} = \max_{j \in \mathcal{J}} (\text{EndOf}(Task^*_{j|\mathcal{K}_j|})) \tag{61}$$

Constraints (57) define interval variables, one for each operation of a job on each eligible machine. The duration of each interval variable is equal to the processing time of that operation on that machine, P_{jki}. The definition of other constraints and variables is clear from their names and one can easily contrast them with those of the MIP ones.

3.11. 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.

3.11.1. MIP 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 (62) and (63) as follows:

$$
\begin{aligned}
\text{minimize} \quad & C_{\max}, && && (\text{MIP}_{\text{OSP}})\\
\text{subject to} \quad & \text{Constraints (6), (10), (11), (19), and (20)}, && &&\\
& c_{ji} \geq c_{ji'} + P_{ji} - M(1 - h_{jii'}) && \forall j \in \mathcal{J}, i, i' \in \mathcal{I} : i > i', && (62)\\
& c_{ji'} \geq c_{ji} + P_{ji'} - M(h_{jii'}) && \forall j \in \mathcal{J}, i, i' \in \mathcal{I} : i > i', && (63)\\
& h_{jii'}, x_{ijj'} \in \{0,1\} && \forall j \in \mathcal{J}, i, i' \in \mathcal{I} : i > i'. && (64)
\end{aligned}
$$

This model needs $\frac{|\mathcal{J}|^2-|\mathcal{J}|}{2} \times |\mathcal{I}| + \frac{|\mathcal{I}|^2-|\mathcal{I}|}{2} \times |\mathcal{J}|$ binary sequencing variables.

3.11.2. CP model

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

$$
\begin{aligned}
\text{minimize} \quad & C_{\max}, && && (\text{CP}_{\text{OSP}})\\
\text{subject to} \quad & \text{Constraints (13), (18), and (22)}, && &&\\
& \text{NoOverlap}(Task_{ji} : i \in \mathcal{I}) && \forall j \in \mathcal{J}. && (65)
\end{aligned}
$$

3.12. Parallel Machine Scheduling Problem (PMSP)

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 thus can calculate the $C_{\max}$ by only 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.

3.12.1. MIP model

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

$$\begin{aligned}
\text{minimize} \quad & C_{\max} && && (\text{MIP}_{\text{PMSP}}) \\
\text{subject to} \quad & \sum_{i \in \mathcal{M}} y_{ji} = 1 && \forall j \in \mathcal{J}, && (66) \\
& C_{\max} \geq \sum_{j \in \mathcal{J}} P_{ji} y_{ji} && \forall i \in \mathcal{M}, && (67) \\
& y_{ji} \in \{0,1\} && \forall j \in \mathcal{J}, \forall i \in \mathcal{M}. && (68)
\end{aligned}$$

The MIP model needs $|\mathcal{J}| \times |\mathcal{M}|$ binary assignment variables.

3.12.2. Constraint programming model

There are two approaches to formulate the PMSP, one interval or integer. In the case of the interval approach, we define $|\mathcal{M}|$ optional interval variables for each job, one on each machine. Then "NoOverlap" and "Alternative". In the integer approach, we define one integer variable for each job that can take a value between 1 to $|\mathcal{M}|$ indicating the machine to which that job is assigned. Then, $C_{\max}$ equals the maximal sum of processing times of jobs assigned to each machine. Note that in PMSP, the sequence of jobs on each machine does not impact makespan (all sequences have the same makespan). We tested both approaches and realized the integer approach outperforms the interval approach significantly. The CP model for the PMSP is as follows:

$$\begin{aligned}
\text{minimize} \quad & C_{\max} && && (\text{CP}_{\text{PMSP}}) \\
\text{subject to} \quad & X_j = \text{IntegerVar}(1, |\mathcal{M}|) && j \in \mathcal{J}, && (69) \\
& C_{\max} = \max_{(j)} \Big(\sum_{i \in \mathcal{M}} P_{ji} (X_j == i) \Big). && && (70) \\
& |\mathcal{M}| C_{\max} \geq \sum_{j \in \mathcal{J}} \text{Element}([P_{j1}, P_{j2}, ..., P_{j|\mathcal{M}|}], X_j) && && (71)
\end{aligned}$$

Constraints (69) define an integer variable for each job. Constraints (70) calculate makespan. Constraints (71) are also a lower bound for makespan. "Element" is a function that returns an element of a given array indexed.

4. Experimental evaluation

Testing all the aforementioned models is a major undertaking. For the tests we use an OpenStack virtualization platform supported by several blade servers, each one with two 18-core Intel Xeon Gold 5220 processors running at 2.2 GHz. and 384 GBytes 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 parallelization of the experimental load, which despite the resources, took several weeks to conclude. We program the models in Python 3.8.10 linked with IBM ILOG CPLEX 20.1 and IBM ILOG CP Optimizer 20.1 using concert programming technology. As for Gurobi, we have employed version 9.1.2. The goal that we seek to achieve with these experiments is to ascertain which of the MIP and CP models performs better for each scheduling problem that we introduced and modeled in the previous section. To this end, we conduct experiments on existing benchmark instances from the literature for each of the problems considered. We note that to keep the findings general, we conduct the comparison

using default settings of solvers. We make no attempts in influencing the search parameters of these solvers and we restrict the comparison to the default setting of these solvers—a decision that is likely adopted by a majority of researchers comparing MIP and CP technology. Specifically, we consider a total of 6,623 instances. The maximum amount of CPU time given for each model is 3,600 seconds.

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 solution found by each model in relation to the best bound obtained from its LP relaxation (lower bound). We note that the optimality gap of both MIP and CP models on any problem instance is provided by the CPLEX and CP Optimizer 20.1 and we make no calculation for the optimality gap.[13] The second measure is the *Relative Percentage Deviation* (RPD). Unlike the optimality gap, the RPD compares the relative distance of the best solution for each model to that of the *best-known solution* in the literature.[14] Note that the gap we provide is based on the difference between the upper bound and lower bound of each of the CP and MIP models, but the RPD analysis encompasses the distance between the best solution for the tested models with the best-known 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 in mathematical programming solvers).

We provide the formula for the RPD and Gap as follows:

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

4.1. Benchmarks and Experimental Plans

In this section, we explain the sources from which we have obtained our benchmarks. We also elaborate on our plan for comparing the performances of different solvers for our scheduling problems.

4.1.1. 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.[15] 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

[13] In the case of CP Optimizer, the lower bound is obtained in different ways, not only the LP relaxation (see Vilim et al., 2015).

[14] We have provided our found best solutions for each problem as an spreadsheet available at `http://soa.iti.es/files/Results_22_9_2021.xlsx`.

[15] This benchmark has been cited in excess of 2,580+ times (at the time of writing this paper).

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. For the F-JSP, we use nine benchmarks for a total of 289 instances that we have gathered from different studies in the literature (see details in Naderi and Roshanaei, 2021). Most instances in the F-JSP literature are considered small instances given the advances made to computing technology and off-the-shelf commercial solvers over the past decade. However, the new 96 F-JSP instances of Naderi and Roshanaei (2021), denoted by BV, are really challenging and constitute the largest benchmark in the F-JSP literature in terms of the number of instances and the size of jobs and machines considered. For the open shop scheduling problem, we use three benchmarks for a total of 192 instances. Table 6 summarizes the standard benchmarks for the problems studied in this paper and their size ranges.

Table 6: 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 (JSP)	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
Flexible Job shops (F-JSP)	Brandimarte (1993)	Brand	10	10-20	4-15
	Hurink et al. (1994)	data-la	129	6-30	5-15
	Naderi and Roshanaei (2021)	BV	96	30-100	10-20
	Dauzère-Pérès and Paulli (1997)	D-Press	18	10-20	5-10
	Others	Others	36	4-20	5-18
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)

4.1.2. Experimental plans

In this section, we follow three objectives. First, we compare the performances of state-of-the-art MIP solvers, CPLEX 20.1 and Gurobi 9.1.2, to ascertain which MIP solver is more suitable for solving our problems. Second, we compare the performance of the best MIP solver with that of the CP Optimizer 20.1 to ascertain which solver is universally best suited for our scheduling problems. Third, once such comparisons were made, we analyze the performance of the best MIP solver and the CP Optimizer on each benchmark. The basic problem in this study is the permutation flow shop with makespan minimization. We compare the CP and MIP models using five different performance criteria: (i) problem characteristics, (ii) objective function, (iii) decision variables, (iv) problem size, and (v) performance measures. Analysis of each one of these criteria provides unique insights into the performance of the CP and MIP 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 functions**: 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. **Bounds**: We analyze performance measures and break them down into two parts: upper and lower bound distance from the best-known bounds.

4.2. Results

4.2.1. Comparison between MIP solvers

We compare the performance of CPLEX 20.1 and Gurobi 9.1.2 in terms of their found number of feasible and optimal solutions and their average optimality gaps on each scheduling problem (see Table 7). The obtained results on 6,623 experiments manifestly demonstrate that CPLEX 20.1 is vastly superior to Gurobi 9.1.2 on almost all the considered performance measures. Specifically, CPLEX 20.1 and Gurobi 9.1.2 find feasible solutions in 72.83% (4,824 out of 6,623) and 59.91% (3,968 out of 6,623) of instances, solves 16.50% (1,093 out of 6,623) and 16.21% (1,074 out of 6,623) of instances to optimality, and achieve a weighted grand average optimality gap of 46.00% and 36.06% on their solved instances, respectively. It might appear unreasonable that we state that the weighted grand average optimality gap of 46.00% obtained by CPLEX 20.1 is better than the lower weighted grand average optimality of 36.09% obtained by Gurobi 9.1.2. The reason for such a statement is that CPLEX finds feasible solutions for much larger instances of different scheduling problems at the expense of a higher average optimality gap (856 more difficult solved instances). On similarly solved instances, Gurobi trivially outperforms CPLEX on the weighted grand average optimality gap at 35.10% versus that of CPLEX at 35.78%—0.68% improvement, which may be attributable to either stronger LP relaxation or better integer solutions obtained by Gurobi.[16] *In view of the provided analysis and in the grand scheme of things, we can conclude that CPLEX 20.1 is better suited for solving our scheduling problems than Gurobi 9.1.2.*

In the previous paragraph, we concluded that CPLEX 20.1 is *generally* a better candidate for solving our scheduling problems. However, care must be taken when these MIP solvers are to be deployed for solving each of our scheduling problems as their performances are highly variable for each scheduling problem. Other than TT-FSP and D-FJSP that Gurobi 9.1.2 is marginally superior to CPLEX 20.1 in terms of feasibility and average optimality gap, researchers and practitioners can safely use CPLEX 20.1 for solving other scheduling problems. If we want to single out one scheduling problem that must not be solved by Gurobi 9.1.2, we can allude to H-FSP in that Gurobi finds feasible solutions for only 32.5% of instances. On the same dataset, CPLEX 20.1 finds feasible solutions for 68.5% of instances—twice as many as feasible solutions than that of Gurobi. PMSP is the only scheduling problem that both CPLEX and Gurobi can efficiently solve with a

[16] We compare GAPs and RPDs of these solvers and determine on which instances, the performance is related to stronger LP relaxation or better integer solutions.

Table 7: Comparison of MIP performances solved via CPLEX 20.1 and Gurobi 9.1.2. Percentages reported under the gap column show the average optimality gap of solved instances and percentages included in parentheses () under the gap column show the average optimality gaps of MIP solvers on similarly solved instances; **bold**: Best performance under each category. We break ties by giving priority to feasibility. That is, on each problem type, we use bold for the MIP solver that has the highest feasibility rate even if its average optimality gap is higher; **underlined numbers**: One MIP solver outperforms the other MIP solver by at least 10% on any performance measure.

Problem	#	MIP solved via `CPLEX 20.1`			MIP solved via `Gurobi 9.1.2`		
		Status (%)		Gap (%)	Status (%)		Gap (%)
		Feasible	Optimal		Feasible	Optimal	
FSP	360	**28.9**	**0.0**	**64.62** (60.71)	23.3	**0.0**	62.85 (**62.85**)
N-FSP	360	**50.8**	**0.0**	**76.92** (69.12)	24.2	**0.0**	74.78 (74.78)
NW-FSP	360	**71.7**	**0.0**	**84.56** (91.19)	43.3	**0.0**	77.68 (**91.11**)
TCT-FSP	360	**34.2**	**0.0**	**64.22** (59.33)	21.7	**0.0**	59.04 (**59.04**)
TT-FSP	540	32.2	**0.0**	92.42 (91.19)	**35.0**	**0.0**	**92.88** (**91.11**)
SDST-FSP	480	**91.0**	**0.0**	**84.34 (81.97)**	74.8	**0.0**	80.43 (80.43)
D-FSP	600	73.8	**1.5**	42.27 (41.11)	**75.5**	**1.5**	**43.26 (40.36)**
H-FSP	1440	**68.5**	**0.0**	**76.92** (62.53)	32.5	**0.0**	59.59 (**59.59**)
F-JSP	289	**97.6**	11.42	**42.32** (38.09)	91.0	**22.2**	34.3 (**34.10**)
JSP	242	**100.0**	5.79	**41.14 (41.14)**	**100.0**	**12.0**	41.2 (41.24)
OSP	192	**100.0**	66.67	**8.93 (8.93)**	**100.0**	**78.1**	9.13 (9.13)
PMSP	1400	**100.0**	**64.93**	0.22 (0.22)	**100.0**	58.7	**0.21 (0.21)**
Total	6,623	**4,824**	**1,093**	**46.00** (35.78)	3,968	1,074	36.09 (**35.10**)
(%)		**72.83**	**16.50**		59.91	16.21	

100% solvability rate and average optimality gaps of ≈ 0.20. The interesting observation is that CPLEX 20.1 has solved more instances of PMSP to optimality (64.93%) than Gurobi 9.1.2 (58.71%), rendering it as a better option for solving PMSP problems. Another interesting observation is the superior performance of Gurobi 9.1.2 in finding more optimal solutions for almost all *scheduling* problems (other than PMSPs). The excellent performance of Gurobi in proving optimality becomes more conspicuous on those scheduling problems with more difficult sequencing decisions, i.e., JSP, OSP, and F-JSP in that the processing route of jobs are mostly non-linear. These two observations together help MIP practitioners design more efficient MIP-based decomposition techniques like classical and/or logic-based Benders decomposition techniques based on the interplay between CPLEX (for solving mostly assignment master problem) and Gurobi (for solving mostly scheduling problems—see e.g., SDST-PMSP of Tran et al., 2016)—the latter significantly helps with developing Benders cuts based on quickly obtainable optimal (rather than sub-optimal) solutions that are indispensable for the design of an exact Benders solution technique.

We have thus far compared the performance of CPLEX and Gurobi based on their number of found feasible and optimal solutions as well as their average optimality gaps. To conduct a fair (an apple to apple) comparison, we even compared the average optimality gap of CPLEX and Gurobi on *comparably solved* instances of each scheduling problem. However, none of these performance metrics can provide us with a piece of compelling evidence as to whether CPLEX is superior to Gurobi even for scheduling problems with comparable average gaps. The reason for this ambiguity is that one of the MIP solvers may possess a tighter LP relaxation (bound) while the other may obtain better integer solutions. Assume CPLEX's LP relaxation and feasible solution for an instance of the FSP is 150 and 300, resulting in a 50% optimality gap. Similarly, assume Gurobi's LP relaxation and feasible solution for the same instance of the FSP is 100 and 200; thus, its optimality gap is also 50%. For the same optimality gap, we can see that Gurobi's hypothetical solution is preferable for practical scheduling purposes as it leads to a lower make-span value. To decide which MIP solver

is more suited for practical purposes, we perform RPD analysis on the integer solutions obtained by each solver on comparably solved problem instances as well as comparing them to best-known solutions from other algorithms in the literature (see Table 9).[17]

4.2.2. Comparison between the best MIP solver versus CP Optimizer

We now compare the performance of the best MIP solver, CPLEX 20.1, with that of the CP Optimizer 20.1 in terms of their found number of feasible and optimal solutions and their average optimality gaps on each scheduling problem (see Table 8). The obtained results demonstrably manifest that CP Optimizer 20.1 is vastly superior to CPLEX 20.1 in terms of finding feasible solutions and achieving lower optimality gaps on almost all (PMSP excluded) scheduling problems. Specifically, CPLEX 20.1 and CP Optimizer 20.1 find feasible solutions in 72.83% (4,824 out of 6,623) and 100.00% (6,623 out of 6,623) of instances, solves 16.50% (1,093 out of 6,623) and 11.13% (737 out of 6,623) of instances to optimality, and achieve a weighted grand average optimality gap of 46.00% and 27.91% on their solved instances, respectively. The optimality performance of CP Optimizer is either better or the same as CPLEX 20.1 in all *scheduling* problems. Note that PMSP has only assignment variables and it does not have sequencing variables. Since the number of instances for PMSP is large (1400) and the optimality rate of CPLEX 20.1 is much higher than that of the CP Optimizer (64.93% versus 4.6%), it inflates the grand optimality rate of CPLEX compared to that of the CP Optimizer—844 more instances are solved to optimality by CPLEX 20.1 on PMSP instances. It is interesting to know that the optimality rate of the CP Optimizer is even better than that of the Gurobi 9.1.2 (excluding PMSP). If we exclude the optimality rate of the PMSP from our analysis, the CP Optimizer 20.1 is significantly better than CPLEX 20.1 in terms of optimality rate in that the former finds 672 optima, whereas the latter finds 249 optima. It is noteworthy that in addition to superior performance in previous performance measures, CP Optimizer 20.1 achieves an 18% ($46\% - 28\% = 18\%$) lower average optimality gap while it finds feasible solutions for additional more difficult 1,735 instances of the problem—26.45% higher feasibility rate than CPLEX 20.1. *In view of the provided analysis and the grand scheme of things, we can conclude that CP Optimizer 20.1 is better suited for solving our scheduling problems than CPLEX 20.1.*

We now provide a comparative detailed analysis between CPLEX and CP Optimizer. Not only does CP Optimizer 20.1 solve 100% of instances to feasibility, but also it provides new optima for FSP, N-FSP, and H-FSP, which could not be achieved by CPLEX 20.1. In terms of the average optimality gap, CP Optimizer is vastly superior to CPLEX. Specifically, the performance of the CP Optimizer is drastically better than that of the CPLEX on H-FSP in that CP Optimizer and CPLEX achieve average optimality gaps of 3.46% and 76.92% on 100% and 68.5% of solved instances of the problem, respectively—at least 22 times lower average optimality gap by CP Optimizer. On comparably solved instances of the problem by both solvers, one can allude to the substantial performance difference between the CP Optimizer and CPLEX on FSP in which the average optimality gap is 60.71% and 3.23%, respectively. The only strength of the CPLEX is in achieving a lower optimality gap and higher feasibility and optimality rate for PMSP instances. The difference in the average optimality gap of CPLEX and CP Optimizer on the PMSP instance is around 12%. We note that this gap is still acceptable from a practical standpoint especially when we know a fraction of this gap is due to the weaker bound in the CP Optimizer. This analysis provides us with a peice of compelling evidence that CP Optimizer must be employed as the solver when scheduling problems have either sequencing or assignment/sequencing variables; otherwise, for pure assignment problems, CPLEX is a better choice.

[17] You can find the best-known solutions for each problem instance that we have retrieved from the literature in the following link: http://soa.iti.es/files/Results_22_9_2021.xlsx.

Table 8: Comparison of MIP performances on CPLEX and CP Optimizer 20.10. Percentages reported under the gap column show the average optimality gap of solved instances and percentages included in parentheses () under the gap column show the average optimality gaps of MIP and CP solvers on similarly solved instances; **bold**: Best performance under each category; We break ties by giving priority to feasibility; That is, on each problem type, we use bold for the MIP solver that has the highest feasibility rate even if its average optimality gap is higher; **underlined numbers**: One MIP solver outperforms the other MIP solver by at least 10% on any performance measure.

Problem	#	MIP solved via `CPLEX 20.1`			CP solved via `CP Optimizer 20.1`		
		Status (%)		Gap (%)	Status (%)		Gap (%)
		Feasible	Optimal		Feasible	Optimal	
FSP	360	28.9	0.0	64.62 (60.71)	**100.0**	**12.5**	**10.74 (3.23)**
N-FSP	360	50.8	0.0	76.92 (69.12)	**100.0**	**9.0**	**13.85 (4.69)**
NW-FSP	360	71.7	**0.0**	84.56 (91.19)	**100.0**	**0.0**	**50.57 (39.75)**
TCT-FSP	360	34.2	**0.0**	64.22 (59.33)	**100.0**	**0.0**	**46.55 (20.44)**
TT-FSP	540	32.2	**0.0**	92.42 (91.19)	**100.0**	**0.0**	**91.06 (79.94)**
SDST-FSP	480	91.0	**0.0**	84.34 (81.97)	**100.0**	**1.5**	**29.64 (27.89)**
D-FSP	600	73.8	**1.5**	**42.27 (41.11)**	**100.0**	**4.0**	48.91 (39.28)
H-FSP	1440	68.5	0.0	76.92 (62.53)	**100.0**	**11.3**	**3.46 (3.37)**
F-JSP	289	97.6	11.42	42.32 (38.09)	**100.0**	**38.4**	**27.62 (22.68)**
JSP	242	**100.0**	5.79	41.14 (41.14)	**100.0**	**50.8**	**3.83 (3.83)**
OSP	192	**100.0**	66.67	8.93 (8.93)	**100.0**	**98.4**	**0.03 (0.03)**
PMSP	1400	**100.0**	**64.93**	**0.22 (0.22)**	**100.0**	4.6	12.52 (12.52)
W. G. Total	6,623	4,824	**1,093**	46.00 (35.78)	**6,623**	737	**27.91 (23.00)**
(%)		72.83	**16.50**		**100.0**	11.13	

We compare the quality of integer solutions of different models solved via different models+solvers using previous performance measures along with the popular RPD performance measure (Table 9). RPD allows for the comparison of the quality of integer solutions of each model+solver on each instance against other models+solvers as well as existing best solutions in the literature. The calculation of RPD is tricky because some of the models+solvers may not be able to obtain integer solutions for all instances of a problem. As such, we consider two types of RPDs: (i) RPD over *all* solved instances of a problem, denoted by RPD_1 and (ii) RPD over *comparably* solved instances across different models+solvers, denoted by RPD_2. Since the value of RPD may substantially be impacted by a single poor solution of any model,[18] we also report the percentage of best solutions found by each model+solver on each problem type. The general findings from Table 9 are similar to the ones obtained in the previous two tables with the difference that CP Optimizer is the best option for our scheduling problems even in terms of achieving high-quality integer solutions and finding the highest number of best solutions. Other than PMSP, the CP Optimizer finds highest best solutions and lowest RPD on all and comparably solved instances of the problem. *The interesting observation is that the quality of integer solutions of the CP Optimizer on PMSP is not as large as its optimality gap. That is, the CP optimality gap on PMSP is 12.52%, whereas its RPD is 2.3%, more than 10% lower than its average optimality gap. This observation reveals that if the CP community improves the bounding mechanism within the CP Optimizer 20.1, it can be used as a standalone solver for all types of optimization problems, and not only for scheduling problems. The improvement of the*

[18]Consider a model+solver that has found 9 optimal solutions for 10 instances of a problem type. If the objective function value of the 10th instance is 100 and the optimal solution is 10, the RPD of this solution is 900%, leading to an average RPD of 90% (900%/100) for that model+solver. This performance appears to be worse than a model+solver that has achieved a hypothetical RPD of 85% for each instance, resulting in an average RPD of 85%. To avoid providing misleading insights, we report both measures.

Table 9: Comparison of CP and MIP solvers in terms of *percentage* of feasibility (Feas.), optimality (Opt.), best solutions found (Best), and RPD = $(\frac{\text{Int. sol.}-\text{Best Int. Sol.}}{\text{Best Int. Sol.}}) \times 100$. We use two RPD measures to ascertain the quality of integer solutions. RPD_1 is calculated over *all* solved instances by a model and RPD_2 is calculated over *comparably solved* instances across different solvers; **Bold**: Best performance under each category; We indicate by bold the RPD_1 of those problems whose feasibility percentage is higher even if their value of RPD_1 is higher; Best integer solutions found from the literature can be found in http://soa.iti.es/files/Results_22_9_2021.xlsx; due to lack of best integer solutions, RPDs of NW-FSP and TCT-FSP are calculated best on the solutions of the models studied in this paper.

Problem	#	CP Optimizer 20.1					CPLEX 20.1					Gurobi 9.1.2				
		(%)					(%)					(%)				
		Feas.	Opt.	Best	RPD_1	RPD_2	Feas.	Opt.	Best	RPD_1	RPD_2	Feas.	Opt.	Best	RPD_1	RPD_2
FSP	360	**100**	**12.5**	**13.9**	**7.5**	**0.7**	28.9	0.0	0.6	8.4	4.9	23.3	0.0	0.28	5.2	5.2
N-FSP	360	**100**	**9.4**	**12.8**	**7.7**	**1.6**	50.8	0.0	0.00	17.3	14.0	24.2	0.0	0.00	76.3	76.3
NW-FSP	360	**100**	**0.0**	**2.5**	**1.4**	**2.2**	50.8	0.0	0.8	23.24	18.51	43.3	0.0	0.8	33.3	23.82
TCT-FSP	360	**100**	**0.0**	**1.7**	**1.6**	**2.1**	34.2	0.0	0.6	9.4	7.1	21.7	0.0	0.28	7.3	7.3
TT-FSP	540	**100**	**0.0**	**12.6**	**24.3**	**13.3**	32.2	0.0	0.00	32.5	31.3	35.5	0.0	0.56	35.2	33.1
SDST-FSP	480	**100**	**1.5**	**1.5**	**5.2**	**4.0**	91.0	0.0	0.00	13.5	11.7	74.8	0.0	0.21	12.6	12.6
D-FSP	600	**100**	**4.0**	**12.0**	**4.2**	**2.0**	73.8	1.5	7.00	14.6	11.1	75.5	0.0	5.5	31.4	8.8
H-FSP	1440	**100**	**11.3**	**11.9**	**40.4**	**3.2**	68.5	0.0	0.00	141.3	46.6	32.5	0.0	0.00	33.9	33.9
JSP	242	**100**	**50.8**	**52.5**	**1.6**	**1.6**	**100**	5.8	21.1	60.0	60.0	91.0	22.2	21.5	13.0	13.0
F-JSP	289	**100**	**38.4**	**63.3**	**1.1**	**0.8**	97.6	11.4	24.9	245.6	186.2	**100**	12.0	27.3	71.6	70.1
OSP	192	**100**	**98.4**	**99.0**	**0.0**	**0.0**	**100**	66.7	88.5	0.5	0.5	**100**	78.1	90.1	0.6	0.6
PMSP	1400	**100**	4.6	22.0	2.3	2.3	**100**	**64.9**	94.8	-0.11	-0.11	**100**	58.7	**98.3**	**-0.14**	**-0.14**

bounding mechanism also enhances the fathoming/pruning/propagation mechanism within the CP Optimizer culminating in solutions whose optimality can be verified much more swiftly. It is through this analysis that we recommend CP Optimizer 20.1 as the best candidate for our select scheduling problems. The performance of the MIP+CPLEX and MIP+Gurobi is highly variable on different scheduling problem types. As such, we cannot strongly conclude that one of these software is superior to another in terms of RPD; as such, care must be taken when choosing either CPLEX or Gurobi for different scheduling problems. However, we continue the remainder of our analysis based on the performance of the MIP+CPLEX due to its ability to achieve better feasibility and optimality rates.

4.3. Detailed performance evaluation

4.3.1. Impact of problem characteristics: flow shop (FSP), nowait-FSP (NW-FSP), and sequence-dependent setup times FSP (SDST-FSP)

We report and analyze the detailed average optimality gap, RPD, and the number of feasible and optimal solutions found by the best MIP and CP model on different benchmarks of similar problem types in terms of sequencing decisions (Table 10). Due to extreme performance variability, it is difficult to fairly compare these models in terms of the average optimality gap and RPD as any comparison will do injustice to the CP models. Despite this performance difference, we continue our analysis and show that the CP model is able to find integer feasible and optimal solutions for the FSP in 100% (120 out of 120) and 37.5% of problem instances (45 out of 120) for the TFS instances, respectively. These percentages are 75.8% (91 out of 120) and 0% (0 out of 120) for the MIP+CPLEX model, respectively. While capable of solving all instances of the TFS benchmark, the CP model yields an average optimality gap of 3.6%. The CP model maintains its superior performance for the VFR benchmark that consists of much larger instances, while the performance of the MIP+CPLEX deteriorates to a level of non-performance. Specifically, the CP model finds

Table 10: Comparison of average results for flow shop problems: Problem characteristics. **Bold**: Best performance under each category. We break ties based on feasibility rate; The RPD values reported for NW-FSP on VFR benchmark have been calculated based only on the best integer solutions of the three models studied in this paper. RPDs have been calculated over all solved instances of the problem.

Problem	Dataset	#	CP Optimizer 20.1					CPLEX 20.1				
			(%)				Time	(%)				Time
			Feas.	Opt.	Gap	RPD		Feas.	Opt.	Gap	RPD	
FSP	TFS	120	**100**	**37.5**	**3.6**	**1.8**	**2,413**	75.8	0.0	62.3	6.3	3,600
	VFR	240	**100**	**0.0**	**14.3**	**10.3**	**3,600**	5.4	0.0	79.2	23.8	3,600
NW-FSP	TFS	120	**100**	**0.0**	**33.7**	**4.1**	**3,600**	100.0	0.0	75.4	17.9	3,600
	VFR	240	**100**	**0.0**	**59.0**	**0.0**	**3,600**	57.5	0.0	92.5	23.2	3,600
SDST-FSP	RMA	480	**100**	**1.4**	**29.6**	**5.2**	**3,569**	91.0	0.0	84.3	13.5	3,600

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

integer feasible solutions for 100% (240 out of 240) of the instances in the VFR benchmark, whereas the MIP+CPLEX model finds integer feasible solutions for only 5.4% (13 out of 240) of the problem instances. The average optimality gap of these 13 solved instances is at least five times higher (79.2% versus 14.3%) than that of the CP model that solves all the difficult VFR instances. In terms of RPD, the CP model is superior to the MIP model on both instances, achieving an average RPD of 1.8% and 10.3% on TFS and VFR benchmarks, respectively. These percentages for the MIP model are 6.3% and 23.3%. The substantially superior performance of the CP model (in terms of average feasibility and optimality rates, optimality gap, and RPD) renders the CP model as the 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 the off-the-shelf exact techniques.

In addition to the FSP, we illustrate the performance of the MIP+CPLEX and CP models on the NW-FSP and SDST-FSP problems in Table 10. As stated earlier, the only difference between the FSP and the NW-FSP is the restriction that enforces no idle time must exist between the completion and starting times of two operations of a job on any two successive machines on the shop floor. The enforcement of such a restriction results in significantly worse performance measures for both models, especially for the CP model. The general finding from comparing the performance of the MIP and CP models for the NW-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 MIP model. The difference, however, is that the CP average optimality gap in the NW-FSP substantially increases (compared to its performance in the FSP). The CP average optimality gaps increase at least 9 times (from 3.6% to 33.7%) and 4 times (from 14.3% to 59.0%) for the TFS and VFR benchmarks, respectively. Another notable difference is that the MIP model can solve all instances of the TFS benchmarks. The CP model achieves an average RPD of 4.1% and 0.0% for TFS and VFR instances, respectively, leaving no room for the MIP models (including MIP+Gurobi) to compete, specifically on the VFR benchmark, which is considerably more difficult than the TFS for the NW-FSP. Despite MIP's absolute worse performance concerning the CP model on NW-FSP, an interesting observation is that when we make a transition from FSP to NW-FSP, the feasibility rate of the MIP model improves significantly on both benchmarks. From this analysis, we conclude strongly that the CP model is the best of-the-shelf exact technique for solving the NW-FSP.

For the SDST-FSP, we use the RMA benchmark, which is based on TFS instances. Again, the CP model clearly outperforms the MIP model in terms of solvability, average optimality gap, and average RPD. The CP and MIP models solve 100% and 91% of instances, yielding average optimality gaps of 29.6% and 84.3%, respectively. The average RPDs of the CP and MIP models are 5.2% and 13.5%, respectively. An interesting observation is that the quality of the integer solutions produced

Table 11: Comparison of average results for flow shop problems: Problem characteristics. **Bold**: Best performance under each category. We break ties based on feasibility rate; The RPD values reported for TCT-FSP on VFR benchmark have been calculated based only on the best integer solutions of the three models studied in this paper. RPDs have been calculated over all solved instances of the problem.

Objective	Dataset	#	CP Optimizer 20.1					CPLEX 20.1				
			(%)				Time	(%)				Time
			Feas.	Opt.	Gap	RPD		Feas.	Opt.	Gap	RPD	
FSP	TFS	120	**100**	**37.5**	**3.6**	**1.8**	**2,413**	75.8	0.0	62.3	6.3	3,600
	VFR	240	**100**	**0.0**	**14.3**	**10.3**	**3,600**	5.4	0.0	79.2	23.8	3,600
TCT-FSP	TFS	120	**100**	0.0	**24.0**	**4.7**	3,600	79.2	0.0	63.3	7.3	3,600
	VFR	240	**100**	0.0	**57.8**	**0.0**	3,600	11.7	0.0	67.4	13.0	3,600
TT-FSP	VRM	540	100	11.9	**80.3**	**24.3**	3,191	32.2	0.0	92.4	32.5	3,600

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

by the MIP moves closer to that of the CP—the average RPD of the MIP 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 26.0% when compared to the FSP ($29.6\% - 3.6\%$). Despite this complexity, the performance of the CP model at 29.6% optimality gap is much more acceptable than that of the MIP model at 84.3%. Again, we can conclude that the CP model is superior to the MIP model and constitutes the best off-the-shelf exact technique for the SDST-FSP.

4.3.2. Objective function impact: Makespan, Total Completion Time, and Total Tardiness

The results of the MIP 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 11. 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 for both models. The CP solves 100% of instances of both the TFS and VFR benchmarks, whereas these percentages are 79.2% and 11.7% for the MIP model, respectively. The average optimality gap in the CP model is 24.0% and 57.8% 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 associated with solving the TCT-FSP, even though TCT is considered as a completion-time related objective function similar to makespan. As for the quality of integer solutions, the CP model obtains an average RPD of 4.7% and 0.0% over 100% of solved instances of TFS and VFR, respectively, whereas the MIP model yields an average RPD of 7.3% and 13.0%, respectively, on a fraction of solved instances. We see that a change in the objective function significantly deteriorates the performance of both models, but it does not change the outcome: the CP model is superior to MIP models. The conclusion that CP usually performs better on makespan-type objectives versus summation-type objectives (e.g., TCT) has been known for a long time in the CP community, but the fact that the CP model is even superior to CPLEX on the most popular FSP benchmarks is a new finding. Precisely, we seek to highlight and quantify the performance improvement (higher feasibility rate, lower gap, and lower RPD) that CP offers to the scheduling community through this analysis.

We now address the impact of the TT objective function on the performance of the CP and MIP 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 MIP models achieve average gaps of 80.3% and 92.4% respectively, and yield average RPD values of 24.3% and 32.5%

with respect to the best-known integer solutions in the literature respectively. The rate of solvability of the MIP model with the TT objective function is significantly worse than that of the FSP with the makespan objective function. This is in contrast to the robust performance of the CP model that irrespective of the problem type achieves 100% solvability on all problem instances. 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 reason for the strong performance of CP when tardiness is zero stems from the fact that CP relies on constraint propagation and the propagation of sum expressions like $\sum_{j \in \mathcal{J}} \max\{\text{EndOf}(Task_{j|I|}) - D_j, 0\}$ propagates weakly unless the upper bound of the sum is 0 as in this case, it directly propagates $\text{EndOf}_j <= D_j$. When the upper bound is strictly larger than 0, the slack on the variables gets "diluted" in all the different terms "j" and propagation becomes weaker. 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 24.3% 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 24.3% 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 MIP model even on summation-type objective functions that are the Achilles' heel of CP models. This finding will culminate possibly in deploying CP models for solving scheduling problems with summation-type objective functions by many scholars who use MIP as a go-to modeling technology.

4.3.3. Problem decision dimensions: Sequencing and assignment

We proceed to an analysis of the performance of the CP and MIP 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 12 depicts the average gap and RPD for these models in different scheduling problems: D-FSP, H-FSP, and F-JSP. The general finding from comparing the performance of the MIP 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). This finding however does not apply to the H-FSP in which the CP's average optimality gap is 3.5% due to the way we have formulated the CP model. We later explain this particular case.

We observe an interesting change in the performance of the MIP+CPLEX on both TFS and VFR benchmarks of the N-FSP in that the rate of feasibility in the MIP model is significantly improved from 75.8% and 5.4% to 92.5% and 30.0% on TFS and VFR benchmarks, respectively. We did not expect to see such an improvement in the feasibility rate of the MIP model because optimizing sequencing decisions in N-FSP is more difficult than those in FSP. Despite this drastic feasibility rate improvement in the MIP+CPLEX on N-FSP, the CP leaves no room for second-guessing as to which solver must be used for solving the N-FSP because it finds feasible solutions for 100% of instances and obtains average optimality gaps of 5.2% and 18.2% on TFS and VFR instances, respectively.

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

Table 12: Average results for scheduling problems with different problem dimensions.

Problems	Dataset	#	CP Optimizer 20.1 (%) Feas.	Opt.	Gap	RPD	Time	CPLEX 20.1 (%) Feas.	Opt.	Gap	RPD	Time
OSP	TOS	60	**100**	**100**	**0.0**	**0.0**	**1.0**	**100**	68.3	27.8	1.5	2,025
	GP	80	**100**	**100**	**0.0**	**0.0**	**3.6**	**100**	**100**	**0.00**	**0.00**	144
	BHJ	52	**100**	**98**	**0.1**	**0.2**	**302.8**	**100**	94.2	0.8	1.5	570
JSP	TJS	80	**100**	**51.3**	**2.9**	**0.9**	**1,911**	**100**	1.3	44.9	158.7	3,502
	DUM	80	**100**	**22.5**	**6.9**	**3.34**	**2,936**	**100**	0.0	51.0	18.9	3,600
	LA	40	**100**	**100**	**0.0**	**0.0**	**110**	100	17.5	22.3	0.36	2,604
	Others	42	**100**	45.0	3.3	1.1	1,693	100	14.0	33.1	7.1	2,495
FSP	TFS	120	**100**	**37.5**	**3.6**	**1.8**	**2,413**	75.8	0.0	62.3	6.3	3,600
	VFR	240	**100**	**0.0**	**14.3**	**10.3**	**3,600**	5.4	0.0	79.2	23.8	3,600
N-FSP	TFS	120	**100**	**28.3**	**5.2**	**2.6**	**2,670**	92.5	0.0	73.8	14.9	3,600
	VFR	240	**100**	**0.0**	**18.2**	**10.2**	**3,600**	30.0	0.0	81.8	20.8	3,600
D-FSP	NR	600	**100**	**4.2**	**48.9**	**4.2**	**3,502**	73.8	2.0	42.3	14.6	3,542
H-FSP	PRA	1440	**100**	**11.3**	**3.5**	**40.4**	**3,309**	68.5	0.0	76.9	141.3	3,600
F-JSP	Brand.	10	**100**	**50.0**	**7.4**	**29.9**	**1,607**	**100**	0.0	32.4	6.09	3,243
	data-la	129	**100**	**51.2**	**16.1**	**2.7**	**1,875**	**100**	15.5	26.5	3.9	2,867
	D-Peres	18	**100**	**30.5**	**24.2**	**30.5**	**3,206**	85.6	0.0	48.9	62.9	3,600
	BV	96	**100**	**2.1**	**55.9**	**3.2**	**3,526**	92.7	0.0	81.0	759.0	3,600
	Others	36	**100**	**100**	**0.0**	**0.0**	**38.0**	**100**	36.0	2.6	0.2	**12.6**
PMSP	FR	1400	**100**	**4.6**	**12.5**	**2.3**	**3,446**	100	63.4	0.2	-0.11	1,431

average computation times of 1, 3.6 and 302.8 seconds for the TOS, GP and BHJ benchmarks, respectively. Comparatively, the MIP solves 128 out of 192 instances to optimality with average CPU times of 2,025, 144, and 570 seconds for TOS, GP, and BHJ, respectively. For the JSP, the average optimality gap of the CP model across all 242 instances is just 2.9% and that of the MIP model is 44.9%. The CP optimally solves 123 instances, whereas the MIP only solves 14 smaller instances. The results for the JSP are somewhat expected as CP solvers have been finely tuned to perform robustly on a wide variety of large-scale scheduling problems, and JSP, in particular (Laborie et al., 2018).

The D-FSP is the first scheduling problem in Table 12 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 38.17% (= 48.91% − 10.74%—average gap of FSP in Table **??**). 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 surprising finding for D-FSP is that the average optimality gap of CP is almost 5 times higher than its average optimality gap on the FSP; however, its average RPD is 4.2% compared to that of the FSP at 7.47%. The average RPD of 4.2% indicates that the CP model is competitive with a plethora of ad-hoc metaheuristic algorithms that have been designed for D-FSP. If we assume the best solutions from the literature are near-optimal, we can conclude that a large fraction of the optimality gap for the D-FSP is associated with its poor dual bound (likely caused by constraints (39) and (40)).

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 unexpectedly decreases from 48.91% to 3.5%. We initially observed expectedly that the CP performs well only on problems with sequencing decisions and its performance gets deteriorated when it solves scheduling

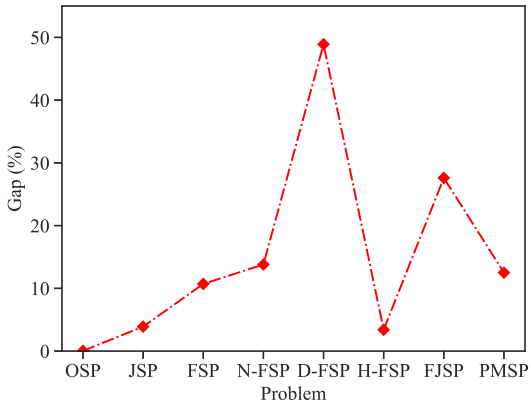


Figure 2: Average gap of the CP in the selected scheduling problems

problems requiring joint optimization of assignment and sequencing variables. We initially used "Alternative" global constraints (similar to the ones used in F-JSPs) to optimize the assignment part of the H-FSP along with "NoOverlap" global constraints to avoid overlapping of operations within each stage of H-FSP. Such formulation yielded poor performance due to the high number of assignment variables. To achieve better performance for the CP model for the H-FSP, we exploited the special structure within the H-FSP to circumvent the complexity associated with the optimization of assignment decisions and developed a model that is more computationally efficient than some of the pure scheduling problems. We used the global constraint "Pulse" that is widely used in cumulative scheduling problems (Laborie et al., 2018). The new CP formulation based on "Pulse" limits the cardinality of simultaneous jobs being processed at any stage of H-FSP to the number of *identical* machines in each stage. The use of global constraints "Pulse" are precluded when machines at each stage are non-identical. For joint assignment+sequencing problems that cannot be effectively reduced to a cumulative scheduling problem (e.g., D-FSP and F-JSP), we use "Alternative" and "NoOverlap" to combine assignment and sequencing, which culminates in weaker CP performance. The unexpected superior performance of this new CP model on H-FSP using function "Pulse" inspires hope that resemblant superior performances *might* be attainable for D-FSP and F-JSP that require joint optimization of assignment and sequencing.

F-JSP is another scheduling problem that combines assignment and sequencing decisions. The difference, though, is that the processing route of jobs on the shop floor is non-linear, unlike D-FSP and H-FSP, leading to more complex sequencing decisions. Our finding shows that there is not a single benchmark in which the MIP model outperforms the CP model. The CP model with the feasibility rate of 100% on all instances achieves much lower average optimality gaps and RPDs. The most difficult instance of F-JSP for both CP and MIP models is the new benchmark of BV Naderi and Roshanaei (2021) in which the average gap is 55.9% and 81% for the CP and MIP models, respectively; however, the RPD of the CP for this problem instance is 3.2%, whereas that of the MIP model is 759%. This finding shows the reliability of the CP model in finding feasible solutions and the quality associated with these solutions. We can thus strongly conclude that the CP model is the best off-the-shelf exact technique for solving the F-JSPs.[19]

So far we have discussed scheduling problems that are either pure sequencing or problems that

[19]The best global exact technique for solving F-JSP belongs to the CP-LBBD$_{\text{CP}}$ of Naderi and Roshanaei (2021) that achieves an average optimality gap of 0.57% for the BV benchmark.

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 12). The PMSP is the only scheduling problem in which the MIP model is superior to the CP model. While both models find feasible solutions for 100% of instances, the MIP and CP models solve 63.4% and 4.6% of instances to optimality, respectively. In addition to solving more instances to optimality, the MIP model has an average optimality gap of just 0.2%, which is much lower than that of the CP with 12.5%. This indicates a clear superiority of the MIP 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. The CP yields an average RPD of 2.3% and the MIP yields an average RPD of -0.11% (this negative number indicates that the solution provided by the MIP 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 MIP models are much better for solving the PMSP.

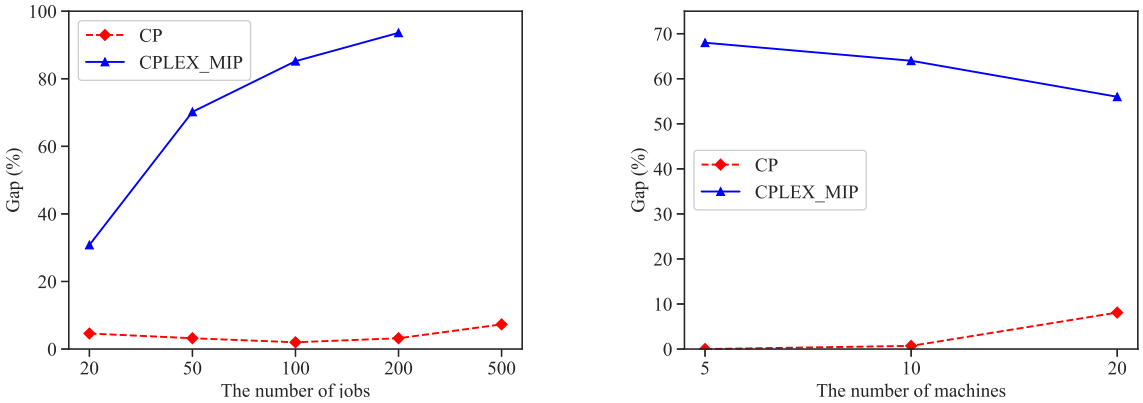
4.3.4. Impact of problem size on performance

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 MIP 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 MIP 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. MIP models fail to find feasible solutions for problem instances exceeding 100 jobs, but the CP model can find integer feasible solutions for all instances (up to 500 jobs). It is remarkable that the CP remains robust concerning 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 MIP 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 MIP'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.

4.3.5. Bound analysis

Up to this point we have employed the RPD and the optimality gap as performance measures. RPD measures the relative distance with respect to the best-known solutions from the literature and the optimality gap measures the absolute distance between the objective function value of the integer solution with that of the LP relaxation of each model on each instance. We analyze the origin of the optimality gap for both models.[20] There are origins for an optimality gap, relative distance of upper/lower bounds from the optimal. The MIP model obtains the optimality gap

[20]Note that we are not concerned with how bounds are calculated within the CP Optimizer. For the black box analysis of gaps in CP and MIP models, it suffices to know that there is a built-in mechanism within the CP Optimizer that takes care of the bound calculation. See, for instance, the seminal work of Hooker and Yan (2002) on this matter that calculates bounds via the relaxations of global constraints.



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

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

for sizes of up to $n = 100$. Figure 4 shows the optimality gap, RPDs of both upper and lower bounds. Clearly, the large optimality gap of the MIP models comes from its weak lower bound as a consequence of disjunctive constraints of sequencing (i.e., big-M constraints). These disjunctive constraints would lead to a poor linear relaxation and a weak lower bound.

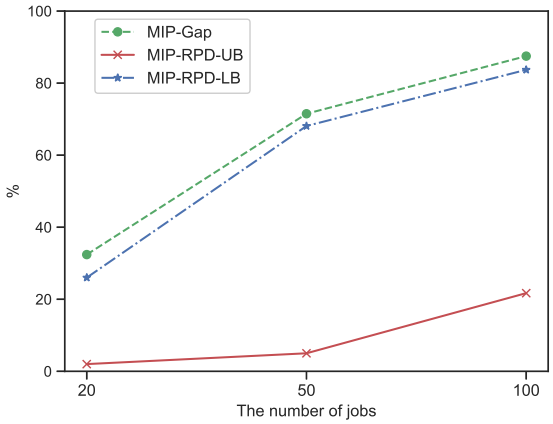


Figure 4: Permutation flow shop: MIP performance.

Remark. Although the major portion of the large optimality gap of the MIP model is because of the weak dual bound (i.e., poor linear relaxation of disjunctive constraints), its primal solution (i.e., integer solution) is still inferior for sequencing problems and is highly impacted by the number of jobs.

5. Conclusions and future research

It is a common practice in the scheduling literature to compare algorithms (and also other MIPs) against one another. MIP models are often used as a baseline and their mostly poor solutions are 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 12.8, which can provide optimality gaps for its integer feasible solutions. We took advantage of these characteristics and conducted a comparative evaluation of the performance of MIP 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, flexible job shops, parallel machine scheduling, job shop scheduling and open shop scheduling. These problems range diversely 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 to have the best possible bounds and feasible solutions for comparison.

For each of the shop scheduling problems stated earlier, we have presented a MIP 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 to be able to calculate the RPD. We reviewed all the papers published in the past 20 years that used these standard benchmarks 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. Having tested 19,869 (3*6,623) experiments, we obtained many improved integer solutions against which researchers can compare the performance of their algorithms. We have also provided insights on which scheduling problems, models yield large average optimality gaps but at the same time low RPDs with respect to best-known integer solutions in the literature.

After conducting our computational campaign on around 6,623 problem instances gathered from the extant literature, we have shown the following:

- CPLEX 20.1 is strongly superior to Gurobi 9.1.2 in terms of feasibility and optimality rates. CPLEX 20.1 and Gurobi 9.1.2 obtained 72.83% and 16.50% and 59.91% and 16.21% feasibility and optimality rates, respectively. In view of our analysis, we strongly recommend CPLEX, as a superior mathematical programming solver, for solving our scheduling MIP problems if researchers intend to use MIP technology as their modeling choice.

- CP Optimizer 20.1 is massively superior to the CPLEX 20.1 in terms of feasibility rate, optimality rate, optimality gap, and RPD. In the grand scheme of the considered performance measures, CP Optimizer 20.1 strongly outperforms both CPLEX 20.1 and Gurobi 9.1.2. We

thus recommend CP Optimizer 20.1 as the best off-the-shelf exact technique for solving our scheduling problems.

- The no-wait or sequence-dependent setup time variants significantly degrade the performance of CP, while the impact on MIP is just the opposite. For example, the no-wait variant results in marginal performance improvements for MIP models. Despite this change, CP is still a better option.

- 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 4.5 times greater than that of the CP model with makespan. This suggests a large variability in the expected results concerning 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. MIP models are superior when the scheduling problems only involve assignment decisions.

- The performance of MIP models is very sensitive to an increase in the number of jobs while the performance of the CP models 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 a 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 MIP models.

- We compared the performance of MIP and CP models on disjunctive *shop* scheduling problems using default settings of solvers. We specified earlier that we do not seek to compare nuances in MIP and CP models. MIP solvers, however, are notorious for the multitude of settings one can change to affect their behavior. Such settings can have a dramatic effect on their ability to zero in on solutions and prove optimality. Usage of primal and dual heuristics, feasibility pumps, cutting planes, primal/dual balance, branching strategies, integrality focus are just a handful of the dozens of parameters that can dramatically affect the MIP engine. Comparing MIP and CP performances with optimized hyperparameters is one interesting research direction. To the best of our knowledge, we used the best MIP models for comparison purposes, but one might argue that other existing CP models could be brought into computational comparisons. We must acknowledge that we are satisfied with the quality of the CP models we presented for our scheduling problems as they helped us spread the message regarding the efficacy of newly offered capabilities of CP Optimizer 20.1 in scheduling problems—the possibility of calculating bounds for found integer solutions. We however acknowledge the need for conducting a computational comparison among different CP models solved via different CP solvers, similar to the ones that we performed to ascertain which mathematical programming solver is more efficient for solving our problems. For instance, a future study can include detailed comparisons between existing JSP models and the one that we proposed in this study or it can be general comparisons among all scheduling models. Another future study can include comparative evaluations between CP and MIP models on *cumulative* scheduling problems in that a machine can process more than one operation at a time. Last but not least, many new decomposition methods can be proposed to solve our scheduling problems. It would be interesting to compare the performance of these decompositions with our CP models, similar to the one performed in Naderi and Roshanaei (2021) for the F-JSP. Recently, it has been shown that effective decomposition approaches can be developed when CP and

MIP models are hybridized (see Roshanaei et al., 2020; Booth et al., 2016; Naderi et al., 2021). Future studies could include multiple factories, each of which includes 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 MIP 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.

We offer our highest thankfulness to the review team for their in-depth comments and critical views on various aspects of our select problems. In particular, we would like to express our deepest gratitude to the first reviewer who provided us with efficient alternatives for PMSP and H-FSP CP models.

Appendix A. Alternative CP models for H-FJSP

These are the alternative CP models for the H-FSP and PMSP.

Appendix A.1. Alternative CP models for H-FSP

$$\begin{aligned}
&\text{minimize} && C_{\max}, && && (\text{CP}_{\text{H-FSP}})\\
&\text{subject to} && \text{Constraints (14) and (18)}, && &&\\
& && Task^*_{jik} = \text{IntervalVar}(P_{ji}, \text{Optional}) && j \in \mathcal{J}, i \in \mathcal{I}, k \in \mathcal{K}_i, && (A.1)\\
& && \text{Alternative}(Task_{ji}, Task^*_{jik} : k \in \mathcal{K}_i) && \forall j \in \mathcal{J}, i \in \mathcal{I}, && (A.2)\\
& && \text{NoOverlap}(Task^*_{jik} : j \in \mathcal{J}) && \forall i \in \mathcal{I}, k \in \mathcal{K}_i. && (A.3)
\end{aligned}$$

(A.1) defines an optional interval variable for each operation on each machine. Constraint (A.2) selects one interval variable for each operation that determines the assignment.

Appendix A.2. Alternative CP models for PMSP

$$\begin{aligned}
&\text{minimize} && C_{\max}, && && (\text{CP}_{\text{PMSP}})\\
&\text{subject to} && Task^*_{ji} = \text{IntervalVar}(P_{ji}, \text{Optional}) && j \in \mathcal{J}, i \in \mathcal{M}, && (A.4)\\
& && \text{Alternative}(Task_j, Task^*_{ji} : i \in \mathcal{M}) && \forall j \in \mathcal{J} && (A.5)\\
& && \text{NoOverlap}(Task^*_{ji} : j \in \mathcal{J}) && \forall i \in \mathcal{M}, && (A.6)\\
& && C_{\max} = \max_{(j)} (\text{EndOf}(Task_j)). && && (A.7)
\end{aligned}$$

Constraint (A.4) defines an optional interval variable for each operation on each machine. Constraint (A.5) selects one interval variable for each job that determines the assignment.

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.

Brandimarte, P. 1993. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research* **41**(3) 157–183.

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.

Dauzère-Pérès, S., J. Paulli. 1997. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research* **70** 281–306.

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.

Fontaine, Daniel, Laurent Michel, Pascal Van Hentenryck. 2016. Parallel composition of scheduling solvers. Claude-Guy Quimper, ed., *Integration of AI and OR Techniques in Constraint Programming*. Springer International Publishing, Cham, 159–169.

Gedik, R, D. Kalathia, G. Egilmez, E. 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.

Hooker, J. N., H. Yan. 2002. A relaxation of the cumulative constraint. Pascal Van Hentenryck, ed., *Principles and Practice of Constraint Programming - CP 2002*. Springer Berlin Heidelberg, Berlin, Heidelberg, 686–691.

Hurink, J., B. Jurisch, M. Thole. 1994. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum* **15**(4) 205–215.

IBM. 2016. Constraint programming modeling for python v2.22 documentation. URL https://ibmdecisionoptimization.github.io/docplex-doc/cp/docplex.cp.modeler.py.html.

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, P., D. Godard. 2007. Self-adapting large neighborhood search: Application to single-mode scheduling problems. *Proc. MISTA* URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1084.1196&rep=rep1&type=pdf.

Laborie, Philippe. 2018. An update on the comparison of mip, cp and hybrid approaches for mixed resource allocation and scheduling. Willem-Jan van Hoeve, ed., *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer International Publishing, Cham, 403–411.

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., V. Roshanaei. 2021. Critical-path-search logic-based benders decomposition approaches for flexible job shop scheduling. *INFORMS Journal on Optimization* URL https://pubsonline.informs.org/doi/10.1287/ijoo.2021.0056.

Naderi, B., V. Roshanaei, C. Luong, M. Aleman, D. Urbach. 2021. Increasing surgical capacity with additional resources: Generalized operating room planning and scheduling. *Accepted for publication in Production and Operations Management* .

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., Ahmed Azab, H. ElMaraghy. 2013. Mathematical modelling and a meta-heuristic for flexible job shop scheduling. *International Journal of Production Research* **51**(20) 6247–6274.

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.

Roshanaei, V., B. Naderi. 2021. Solving integrated operating room planning and scheduling: Logic-based benders decomposition versus branch-price-and-cut. *European Journal of Operational Research* **293**(1) 65–78.

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.

Tran, Tony T., Arthur Araujo, J. Christopher Beck. 2016. Decomposition methods for the parallel machine scheduling problem with setups. *INFORMS Journal on Computing* **28**(1) 83–95. doi: 10.1287/ijoc.2015.0666. URL https://doi.org/10.1287/ijoc.2015.0666.

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.

Vilim, P., P. Laborie, P. Shaw. 2015. Failure-directed search for constraint-based scheduling. Michel L. (eds), ed., *Integration of AI and OR Techniques in Constraint Programming*. Springer, 437–453. URL https://link.springer.com/chapter/10.1007/978-3-319-18008-3_30.

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.