Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Mathematical models and benchmarking for the fuzzy job shop scheduling problem☆

Sezin Afsar [a], Camino R. Vela [a], Juan José Palacios [a], Inés González-Rodríguez [b,*]

[a] Department of Computing, University of Oviedo, Campus de Gijón, 33204, Gijón, Spain
[b] Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria, Av. Los Castros s/n, 39005, Santander, Spain

ARTICLE INFO

Dataset link: Benchmarks for Fuzzy Job Shop Problem (Reference data)

Keywords:
Scheduling
Job shop
Fuzzy numbers
Mathematical models
Benchmark
Metaheuristics

ABSTRACT

The fuzzy job shop scheduling problem with makespan minimisation has received considerable attention over the last decade. Different sets of benchmark instances have been made available, and many metaheuristic solutions and corresponding upper bounds of the optimal makespan have been given for these instances in different publications. However, unlike the deterministic case, very little work has been invested in proposing and solving mathematical models for the fuzzy problem. This has resulted both in a lack of a good characterisation of the hardness of existing benchmark instances and in the absence of reliable lower and upper bounds for the makespan. In consequence, it is difficult, if not impossible to properly assess and compare new proposals of exact or approximate solving methods, thus hindering progress in this field. In this work we intend to fill this gap by proposing and solving two mathematical models, a mixed integer linear programming model and a constraint programming model. A thorough analysis on the scalability of solving these mathematical models with commercial solvers is carried out. A state-of-the-art metaheuristic algorithm from the literature is also used as reference point for a better understanding of the results. Using solvers of different nature allows us to improve known upper and lower bounds for all existing instances, and certify optimality for many of them for the first time. It also enables us to structurally characterise the instances' hardness beyond their size.

1. Introduction

Scheduling aims at finding the optimal arrangement of tasks by allocating limited resources. Due to its complex nature and its high relevance to many sectors such as industry, building renovation, welfare, education, etc., it remains a significant research field (de Fátima Morais, dal Molin Ribeiro, Gomes da Silva, Cocco Mariani, & dos Santos Coelho, 2022; Lunardi, Birgin, Laborie, Ronconi, & Voos, 2020; Pinedo, 2016).

One of the most studied scheduling problems is the *job shop problem* (*JSP*) and the most common objective is to minimise the time span required to complete the project, known as makespan. Despite the numerous contributions in the literature, the JSP with makespan minimisation still poses a challenge with its numerous variants and complexity (Xiong, Shi, Ren, & Hu, 2022). Although the JSP is NP-hard in the strong sense, recent advancements in off-the-shelf solvers have made it possible to solve mathematical models of the problem for small to medium size instances or, at least, to obtain good-quality lower and upper bounds for the optimal makespan (Ku & Beck, 2016). Also, the extension of classical constraint programming (CP) to mathematical concepts such as intervals and functions have made it easier to model the JSP while providing an interesting problem structure for an automatic search algorithm (Laborie, 2018; Laborie, Rogerie, Shaw, & Vilím, 2018; Novas, 2019). For larger size instances that remain unsolved with off-the-shelf solvers, metaheuristic techniques are often employed to find good solutions within reasonable computational effort (Dokeroglu, Sevinc, Kucukyilmaz, & Cosar, 2019; Gendreau & Potvin, 2019). Usually the evaluation of such methods in terms of performance and effectiveness is carried out using a variety of available benchmarks such as the instances from Adams, Balas, and Zawack (1988), Applegate and Cook (1991), Fisher and Thomson (1963), Lawrence (1984) or Taillard (1993).

The interest in the job shop stems not only from its complexity but also from its applications. Indeed, it serves as a model for real-world problems arising in many manufacturing and service industries. A wide

☆ Funding: This work was supported by the Spanish Government [grant number PID2019-106263RB-I00] and [grant number TED2021-131938B-I00].
* Corresponding author.
E-mail addresses: afsarsezin@uniovi.es (S. Afsar), crvela@uniovi.es (C.R. Vela), palaciosjuan@uniovi.es (J.J. Palacios), gonzalezri@unican.es (I. González-Rodríguez).

https://doi.org/10.1016/j.cie.2023.109454
Received 20 July 2022; Received in revised form 13 March 2023; Accepted 15 July 2023
Available online 25 July 2023

list of applications of the job shop scheduling in its multiple variants can be found in the survey from Xiong et al. (2022). Additionally, real-life applications of the flexible variant of the job shop are reviewed in Xie, Gao, Peng, Li, and Li (2019).

Flexibility, together with crane transportation and energy objectives, are some of the characteristics that are currently being explored to increase the applicability of the job shop problem (Du, Li, Li, & Duan, 2022; Li, Du et al., 2022). Another feature to help narrow the gap between academic and real-life settings is uncertainty, since the lack of complete information is an inevitable part of project management (Hazır & Ulusoy, 2020; Verderame, Elia, Li, & Floudas, 2010). In particular, activity durations tend to deviate from the initial estimates, either due to inaccurate estimation or to unexpected events faced during execution. In consequence, activity durations constitute one of the four major sources of uncertainty in the work content, which could broadly be defined as the effort required for completing all the project activities (Hazır & Ulusoy, 2020).

While stochastic scheduling is the most common framework for modelling uncertainty, it may not always be a feasible or efficient approach. A first reason is that stochastic modelling requires a large amount of high-quality data, which may be difficult to obtain or even unavailable in some cases. Also, working with complex probability distributions can be computationally demanding and time-consuming. Finally, stochastic models may not always provide the flexibility needed to capture a wide range of uncertainty types, including vagueness and ambiguity. Fuzzy sets within the framework of possibility theory provide an alternative model which is computationally less demanding while allowing for greater flexibility in representing uncertainty (Dubois, Fargier, & Fortemps, 2010). Moreover, fuzzy numbers can be obtained based on expert opinions or subjective estimates, making them a suitable alternative when data are limited or unavailable.

The fuzzy approach to uncertainty in scheduling can be traced back to Prade (1979), where fuzzy numbers are used to represent uncertain processing times in a real problem concerned with building a course schedule for a semester in a French "Grand École". A probabilistic representation is discarded because statistical information is very poor or even non-existent; fuzzy numbers are instead elicited based on which values are impossible, not completely impossible, possible and very possible given the available knowledge of the problem. A similar approach, based on the proposal by Rommelfanger (1990), is used in Hapke, Jaszkiewicz, and Slowinski (1994) to elicit fuzzy numbers representing uncertain activity durations in a real software project scheduling problem. The authors argue the difficulty of foreseeing durations of particular activities and the inadequacy of an stochastic approach unless the project being scheduled is very similar to previous ones, which is not usually the case given the particularities of each project and the rapid evolution of techniques, methodologies and tools in software development. However, it appears that software project managers are able in practice to estimate approximate durations of particular activities, taking into account their experience, good knowledge about new techniques and skills of particular software engineers, naturally leading to fuzzy numbers. In Petrovic, Fayad, Petrovic, Burke, and Kendall (2008), triangular fuzzy numbers are used to model processing times with uncertainty stemming from staff-operated machines in a real-life scheduling problem from a printing company; the processing time of each operation is obtained by estimating the lower and upper bounds of the processing times as well as a modal or most likely value, interpreted as pessimistic, moderate and optimistic processing times. Uncertain durations appearing in a real-life scheduling problem derived from motorway construction in Greece are also modelled as triangular fuzzy numbers derived from the deterministic contractual durations based on the experience of the site engineers in Maravas and Pantouvakis (2012). In general, Dubois et al. (2010) advocate the use of fuzzy numbers for ill-known processing times when the available information is in terms of more or less plausible values, so fuzzy numbers can be viewed as nested confidence intervals with varying plausibility levels. In particular, triangular fuzzy numbers, with an interval of possible values and a most-likely duration, have become a popular approach to modelling uncertain processing times in scheduling.

Detailed reviews of the literature and benchmarks for fuzzy shop scheduling problems up to 2016 can be found in Abdullah and Abdolrazzagh-Nezhad (2014), Behnamian (2016) and Palacios, Puente, Vela, and González-Rodríguez (2016). Another review focused on methods for population initialisation in metaheuristics for the fuzzy job shop is presented in Shukor, Shaheed, and Abdullah (2018). More recently, we encounter multiobjective fuzzy job shop problems optimising makespan together with due-date satisfaction, robustness or energy consumption (Afsar, Palacios, Puente, Vela, & González-Rodríguez, 2022; Palacios, González-Rodríguez, Vela, & Puente, 2017; Wang, Gao, & Pedrycz, 2022; Wang, Tian, Ji, & Wang, 2017) although in the case of Wang et al. (2022) they use LR fuzzy numbers instead of the more extended triangular fuzzy numbers. Due-date satisfaction is the single objective in Palacios, González-Rodríguez, Vela, and Puente (2019) and Vela, Afsar, Palacios, González-Rodríguez, and Puente (2020) using triangular fuzzy numbers and in Gao, Wang, and Pedrycz (2020) using LR fuzzy numbers, while energy consumption alone or combined with due-date satisfaction in fuzzy job shops with triangular fuzzy numbers is addressed in González-Rodríguez, Puente, Palacios, and Vela (2020) and Zhao et al. (2019). In all cases, the resulting job shop is solved using metaheuristics. Several metaheuristics have also been proposed for the flexible variant of the fuzzy job shop, minimising makespan on its own or in a multiobjective setting together with total workload or energy consumption (Li, Gong and Lu, 2022; Li, Liu, Li, & Zheng, 2021; Lin, Zhu, & Wang, 2019).

When it comes to classical JSP and other deterministic scheduling problems, mixed integer programming is one of the most common initial approaches, followed by constraint programming in the recent years (Ku & Beck, 2016). Surprisingly, in the case of fuzzy JSP, these approaches are scarce. MILP models for the fuzzy job shop are proposed in Vela et al. (2020) for a due-date satisfaction objective function and in Zhao et al. (2019) for a total energy objective function, in this last case only to describe but not to solve the problem. Regarding other problems in the family of fuzzy job shop, a MILP formulation for a flexible bi-objective problem minimising makespan and non-processing energy can be found in Afsar et al. (2022), where the commercial solver is shown to be unable to tackle any but the smallest instances. A mathematical formulation for a problem minimising a linear combination of makespan and total energy consumption, incorporating flexibility and crane transportation, and with uncertain durations are modelled as type-2 interval fuzzy numbers is provided in Li et al. (2021), here only to give an accurate description of the problem, which is later solved with an improved artificial immune system.

Metaheuristics constitute the majority, if not the entirety, of the solving methods proposed in the literature for the fuzzy job shop with makespan minimisation. Due to their nature, these methods always provide an upper bound of the optimal solution, with no guarantee of optimality. Assessing their performance is not a trivial issue and in fact there is no general consensus on the performance assessment criteria. A means of evaluating the performance of metaheuristic methods is to measure relative errors with respect to a lower bound. Obviously, such relative measures are only representative if the lower bound is reasonably tight, as highlighted in Palacios et al. (2016). Otherwise, fair and meaningful comparisons between different proposals and reliable performance assessments are not possible. Furthermore, the gap between the upper and lower bounds provides some insight into the hardness of an instance. Ultimately, only when lower and upper bounds take the same value, optimality can be certified. Having accurate lower bounds is thus a critical issue.

Indeed, conclusions on the relevance of new methods based on their results on non-challenging instances might be unrealistic. For deterministic job shop some authors have concluded that the hardness of an instance does not depend exclusively on the size, but also on other indicators such as the ratio between the number of jobs and the number of machines (Strassl & Musliu, 2022; Streeter & Smith, 2006). To our knowledge, there is no in-depth study on the hardness of the available fuzzy JSP instances exists. The only exception is Palacios et al. (2016), where instances are classified into challenging and non-challenging ones based on the empirical results obtained with different metaheuristic methods. Even then, no attempt is made to somehow characterise their hardness.

In this work, we take the stance that commercial solvers using mixed integer programming or constraint programming models may provide accurate lower bounds. Additionally, the availability of different models and associated solving methods should provide a framework to characterise the hardness of benchmark instances and the challenge they pose.

The contributions of this paper are the following:

- We propose for two mathematical models for the fuzzy JSP: a mixed integer linear programming model and a constraint programming one. These models make it possible to apply the commercial solvers CPLEX and CP Optimizer to the existing problem instances.
- We perform an exhaustive experimental study on a varied set of instances, including those from the literature as well as some new ones that complement the existing benchmarks. In total, we consider 149 fuzzy JSP instances with sizes ranging between 36 and 2000 tasks. All instances are tackled using the two commercial solvers for the mathematical models together with a state-of-the-art metaheuristic.
- We evaluate the scalability of the mathematical models and the associated commercial solvers for fuzzy JSP based on the experimental results.
- Also based on the combined results from all three solving methods we structurally characterise the instances' hardness by identifying significant indicators, hinting that size should not be the only factor taken into account to identify the most challenging instances.
- We provide updated and more accurate upper and lower bounds for the optimal makespan values of all 149 instances and certify, for the first time in the literature, the optimality of solutions in 83 of them.

All these contributions fill an existing gap in the field and facilitate scientific progress by providing a solid basis for future research in solving methods for the fuzzy JSP with makespan minimisation.

The remaining of this paper is organised as follows. After introducing the background on the job shop problem and fuzzy durations in Section 2, Section 3 presents the three models for the problem: the new mixed integer linear programming and constraint programming models together with a disjunctive graph model from the literature. Solution methods for these models are described in detail in Section 4. Next, the existing benchmarks are reviewed in Section 5. Section 6 presents a thorough experimentation and, based on this, analyses the hardness of the existing instances, finds the limitations of the exact solvers and encourages the proposal of a new test-bed. Finally, conclusions and future prospects are given in Section 7.

2. Background on the fuzzy job shop scheduling problem

In the *job shop scheduling problem* there is a set J of jobs that need to be processed on a set M of machines or resources. Each job $j \in J$ consists of a sequence O_j of tasks or operations. Each task in O_j needs to be processed on a specific machine m from a subset $M_j \subseteq M$ with no recirculation, meaning that no two tasks in the same job can require the same machine. The processing of the task from O_j in machine m takes a fixed time p_{jm}. Preemption is not allowed and machines can only handle one task at a time. Therefore, when tasks are scheduled two types of constraints need to be taken into account.

To start with, *precedence constraints* ensure that the tasks in job j are executed in the order given by the sequence O_j. For every task in O_j, if $m \in M_j$ is the machine where it has to be executed, s_{jm} denotes its starting time and, $succ_j(m)$ denotes the machine where the following task according to O_j has to be processed, then:

$$s_{jm} + p_{jm} \leq s_{j\,succ_j(m)} \tag{1}$$

where $s_{j\,succ_j(m)}$ is taken to be ∞ if m is the machine corresponding to the last task in the sequence O_j.

Capacity constraints in turn guarantee that a machine performs at most one task at a time during its whole processing time without being interrupted. That is, for every pair of jobs $i, j \in J$, $i \neq j$ with tasks requiring the same machine m, the execution of these tasks cannot overlap:

$$s_{jm} + p_{jm} \leq s_{im} \vee s_{im} + p_{im} \leq s_{jm} \tag{2}$$

An assignment of starting times to all tasks that is feasible with respect to the aforementioned constraints is a *schedule*. In this work, we tackle the most common objective in the literature, minimising makespan (denoted as $C_{\max}$) which is defined as the time elapsed between the start and end of the processing of all tasks, in other words, the completion time of the last task to be processed, $C_{\max} = \max_{j \in J, m \in M_j}(s_{jm} + p_{jm})$.

The *fuzzy job shop problem* (*fJSP*) is an extension of the JSP where task processing times (and sometimes job due dates) are modelled using fuzzy numbers (Abdullah & Abdolrazzagh-Nezhad, 2014; Behnamian, 2016). In the following, we describe a fuzzy job shop where uncertain task durations are represented by triangular fuzzy numbers, which is the most common approach in the literature.

2.1. Uncertain processing times

In the classical job shop problem, it is assumed that the processing times of operations are known in advance and do not change. However, in real life this is often not the case. For instance, durations of tasks such as subcontracted activities in manufacturing, debugging in software development or tasks performed by more or less experienced workers are usually not deterministic. Despite this uncertainty, an expert might have some knowledge about the duration of the task based on past experience. One of the simplest ways of representing uncertain task durations would be defining an experience-based confidence interval. If different values in the interval are more plausible than others, it can be naturally extended to a fuzzy number (Dubois et al., 2010).

The model most commonly used in the literature on fuzzy scheduling is a *triangular fuzzy number* (*TFN*), consisting of an interval $[a^1, a^3]$ of possible values and a modal value a^2 in it. A TFN $\hat{a} = (a^1, a^2, a^3)$, is a fuzzy number having the following membership function:

$$\mu_{\hat{a}}(x) = \begin{cases} \frac{x-a^1}{a^2-a^1} & : a^1 \leq x \leq a^2 \\ \frac{x-a^3}{a^2-a^3} & : a^2 < x \leq a^3 \\ 0 & : x < a^1 \text{ or } a^3 < x \end{cases} \tag{3}$$

Since a natural ordering does not exist between TFNs, various ranking methods have been studied in the literature for comparison. Many of these are based on the *expected value* of a fuzzy number $\hat{a}$, which is generally defined as $E[\hat{a}] = \frac{(a^1+2a^2+a^3)}{4}$ for a TFN. Since $E[\hat{a}]$ is a real value, it is often used to establish an ordering on the set of TFNs, denoted as $\leq_E$ (see Palacios, González-Rodríguez, Vela, and Puente (2015) for details). To handle uncertain durations in the fJSP, two arithmetic operations are needed: the *sum* and the *maximum*. By extending the corresponding operations on real numbers using the *Extension Principle* the *sum* of two TFNs $\hat{a}, \hat{b}$ is another TFN given by $\hat{a} + \hat{b} = (a^1 + b^1, a^2 + b^2, a^3 + b^3)$.

The *maximum* of two TFNs is not that simple to calculate. Besides, the result of the operation may not be a TFN although it is a fuzzy number. Therefore, various methods are proposed in the literature that approximate the result by a TFN and two of these are explained below.

The first method, which is used since the early works (Fortemps, 1997; Sakawa & Kubota, 2000) to the most recent (Gao et al., 2020; Palacios et al., 2016) among others, is based on a component-by-component operation so, for every $\hat{a}$, $\hat{b}$ TFNs:

$$\max(\hat{a},\hat{b}) \approx \max{}_I(\hat{a},\hat{b}) = (\max(a^1,b^1), \max(a^2,b^2), \max(a^3,b^3)) \tag{4}$$

In the second method, the ranking $\leq_E$ based on expected value is extended so $\max(\hat{a},\hat{b}) \approx \max_R(\hat{a},\hat{b})$ where $\max_R(\hat{a},\hat{b}) = \hat{a}$ if $E[\hat{a}] > E[\hat{b}]$ or else if $E[\hat{a}] = E[\hat{b}]$ and $(a^3 - a^1) > (b^3 - b^1)$ or else if $E[\hat{a}] = E[\hat{b}]$, $(a^3-a^1) = (b^3-b^1)$ and $a^2 > b^2$. Otherwise, $\max_R(\hat{a},\hat{b}) = \hat{b}$. This approach has been used in other papers such as Lei (2010a), Li, Gong et al. (2022) and Zheng, Li, and Lei (2011).

In Lei (2010a), the author claims that the approximation error of this criterion (max_R) is similar or smaller than that of Sakawa criterion (max_I). On the other hand, several arguments based on a theoretical background are given in Palacios et al. (2016) that support the use of max_I. In brief, the authors argue that max_I verifies that the resulting TFN preserves the modal and support values, despite having the potential of artificially increasing the actual maximum. This is not guaranteed with the approximated maximum max_R. In the framework of scheduling problems, this means that using max_R may lead to a makespan value that is not a natural coverage of all possible realisations of a schedule. That is, for a given task processing order, if the duration of each task takes one of its possible values (i.e., a value in the support of the corresponding TFN), then it is not guaranteed that the resulting crisp makespan lies in the support of the fuzzy makespan calculated with max_R, contrary to what happens with the fuzzy makespan value calculated with max_I. Therefore, in this work we adopt the use of the max_I approximation.

Let us underline the fact that for any two TFNs $\hat{a}$ and $\hat{b}$, if $\hat{m}_I = \max_I(\hat{a},\hat{b})$ and $\hat{m}_R = \max_R(\hat{a},\hat{b})$, the inequality $m^i_R \leq m^i_I$ holds for $i = 1,2,3$ and, hence, $\hat{m}_R \leq_E \hat{m}_I$ and $\hat{m}_R \leq_R \hat{m}_I$.

2.2. Schedule Robustness

When dealing with uncertainty, solution robustness is an important factor to consider. A solution to a fuzzy scheduling problem is what is referred to as predictive schedule: it gives a fuzzy interval $\hat{C}_{\max}$ of possible values for the makespan. It is only when tasks are executed in a real scenario according to the ordering provided by the predictive schedule that we can find the real makespan value $C^{ex}_{\max}$. The predictive schedule is thus robust if the executed makespan does not deviate much from the prediction. In fuzzy scheduling with makespan minimisation, a widely-used metric of robustness is the ϵ-robustness (Palacios, González-Rodríguez, Vela, & Puente, 2014). A predictive schedule is considered to be ϵ-robust for a given ϵ if:

$$\frac{|C^{ex}_{\max} - E[\hat{C}_{\max}]|}{E[\hat{C}_{\max}]} \leq \epsilon \tag{5}$$

Clearly, solutions are more robust for smaller ϵ values. When a real execution of the schedule is not possible (for example, for synthetic benchmark instances), a Monte-Carlo simulation can be used to generate a sample of K possible scenarios of the instance by assigning an exact duration to each task. Several results in the literature motivate the use of TFNs as fuzzy counterparts to uniform probability distributions (Dubois, Foulloy, Mauris, & Prade, 2004), so exact durations for each task can be generated using a uniform distribution in the support of the corresponding TFN. Let $C^k_{\max}$, denote the makespan obtained by executing the predictive schedule on the kth scenario. Then, the average ϵ-robustness $\bar{\epsilon}$ is calculated as:

$$\bar{\epsilon} = \frac{1}{K}\sum_{k=1}^{K} \frac{|C^k_{\max} - E[\hat{C}_{\max}]|}{E[\hat{C}_{\max}]} \tag{6}$$

3. Models for the fuzzy job shop scheduling problem

Both mixed integer and constraint programming models abound in the literature on scheduling problems (e.g. Laborie et al., 2018; Lunardi et al., 2020). However, a recent survey on fuzzy scheduling (Behnamian, 2016) shows that, in this subfield, the majority of solving methods fall in the category of metaheuristic search, amounting to more than 70% of all the existing proposals. In this work, we intend to take into account all three approaches (mixed integer programming, constraint programming and metaheuristics) simultaneously. In doing so, we hope to be able to profit from the different strengths of each approach and related techniques and assess when is each one more adequate.

First, we propose a mixed integer linear programming (MILP) and a constraint programming (CP) model. As far as we know, these are the first proposals of this kind for the fuzzy JSP. Next, we describe a disjunctive graph model that serves as the basis for developing the neighbourhood structures used in the most successful metaheuristic methods.

3.1. MILP model

Mixed integer linear programming models provide a precise definition of scheduling problems (Ku & Beck, 2016; Lunardi et al., 2020). In addition, many papers in the literature show the significance of MILP models and the associated methods as a baseline for comparisons (Laborie, 2018), either on their own or in the core of other algorithms for scheduling problems (Basán, Cóccola, García del Valle, & Méndez, 2019; Ku & Beck, 2016).

Here, we propose a MILP formulation for the fuzzy JSP with makespan minimisation. The parameters and variables of the model are given in Tables 1 and 2, respectively. As mentioned before, in fJSP the processing time $\hat{p}_{jm}$ of a job j on machine m is modelled as a TFN with 3 components and each of its components is represented as p^l_{jm}, $l \in \{1,2,3\}$. Since processing times of the jobs are TFNs, starting times of their operations and the makespan are also TFNs, denoted $\hat{s}_{jm}$ and $\hat{C}_{\max}$, respectively. The variables x_{ijm} are binary and, in a feasible solution, they would take value 1 if the task of job i immediately precedes the task of job j on machine m (that is, if they are scheduled back-to-back), and 0 otherwise.

$$\min E[\hat{C}_{\max}]$$

s.t.

$$C^l_{\max} \geq s^l_{j\mu_j} + p^l_{j\mu_j} \qquad \forall j \in J, l \in \{1,2,3\} \tag{7}$$

$$s^l_{jsucc_j(m)} \geq s^l_{jm} + p^l_{jm} \qquad \forall j \in J, m \in M_j, l \in \{1,2,3\} \tag{8}$$

$$s^l_{jm} \geq s^l_{im} + p^l_{im} - W \cdot (1 - x_{ijm}) \qquad \forall i,j \in J, m \in M_j \cap M_i, \; l \in \{1,2,3\} \tag{9}$$

$$\sum_{\substack{i,j \in J \\ i \neq j}} x_{ijm} = n_m - 1 \qquad \forall m \in M_j \cap M_i \tag{10}$$

$$\sum_{\substack{i \in J \\ i \neq j}} x_{ijm} \leq 1 \qquad \forall j \in J, m \in M_j \cap M_i \tag{11}$$

$$\sum_{\substack{j \in J \\ j \neq i}} x_{ijm} \leq 1 \qquad \forall i \in J, m \in M_j \cap M_i \tag{12}$$

$$0 \leq s^1_{jm} \leq s^2_{jm} \leq s^3_{jm} \qquad \forall j \in J, m \in M_j \tag{13}$$

$$x_{ijm} \in \{0,1\} \qquad \forall i,j \in J, m \in M_j \cap M_i \tag{14}$$

Constraint (7) defines the makespan by stating that $\hat{C}_{\max}$ is greater than or equal to the completion time of every job. Constraint (8) enforces the right task order within jobs. Note that x_{ijm} is defined if job j and job i are both processed on machine m, i.e., if $m \in M_j \cap M_i$, so according to constraint (9), there is no overlap in the processing of two consecutive tasks in a machine. Constraint (10) ensures that exactly

Table 1
Parameters of MILP and CP models.

J	Set of all jobs
M	Set of all machines
$M_j \subseteq M$	Set of machines that process job j
$l \in \{1,2,3\}$	TFN components
$\hat{p}_{jm}$	Processing time of job j on machine m (TFM), $j \in J$, $m \in M_j$
p^l_{jm}	lth component of $\hat{p}_{jm}$
n_m	Total number of jobs processed by machine m
$succ_j(m)$	The machine that succeeds machine m in the processing order of job j
μ_j	The final machine that processes job j
W	A large number

Table 2
Variables of MILP model.

$\hat{s}_{jm}$	Starting time (TFN) for the processing of job j on machine m, $m \in M_j$, $j \in J$
s^l_{jm}	lth component of $\hat{s}_{jm}$, $l \in \{1,2,3\}$
$\hat{C}_{\max}$	Makespan (TFN)
$C^l_{\max}$	lth component of makespan, $l \in \{1,2,3\}$
x_{ijm}	Binary, takes value 1 if job i immediately precedes job j on machine m and is 0 otherwise

n_m tasks are processed in each machine, while constraints (11) and (12) ensure that each task has at most one immediate predecessor and one immediate successor in its machine. Constraint (13) guarantees that the three components s^1_x, s^2_x, s^3_x of the starting time of each task x actually define a TFN $\hat{s}_x = (s^1_x, s^2_x, s^3_x)$. Lastly, constraint (14) defines the domain of variables x.

To our knowledge, this is the first MILP model for fuzzy JSP in the literature. Also, it is not a trivial extension of existing MILP models for JSP. For instance, in Ku and Beck (2016) the authors distinguish four types of MILP models. One of them, the so-called time-indexed model, is also used in Laborie (2018). Our model is different since it is the only one where decision variables model the relative order between jobs in each machine. This choice is motivated by the fact that each fuzzy starting time is represented by three variables s^i_{jm}, $i = 1,2,3$ and it should not be possible that each of these variables yields a different processing order in the machines. Also, constraints such as (13) are specific of the fuzzy setting.

3.2. CP model

Recent advances in CP and commercial CP solvers show that it is a promising approach when used alone or combined with many other exact and heuristic optimisation methods for scheduling problems (Laborie, 2018; Laborie et al., 2018; Lunardi et al., 2020). In Laborie (2018), the authors define a modelling paradigm that extends classical CP to integer variables with intervals and functions which is of particular interest for this work. Overall, constraint programming allows us to model scheduling problems in a flexible way. To model the properties of TFNs, some changes are introduced in the classical CP model for JSP, where every task is modelled as an interval that starts at the task's starting time and whose length is the task's processing time. Specifically, we propose to have three variables representing the TFN components of the starting time of job j on machine m are defined: s^1_{jm}, s^2_{jm} and s^3_{jm}. To impose the component-wise ordering of the TFNs, the constraint `same − sequence` from Laborie et al. (2018) is used. Then, every task can be presented as an interval $[s^l_{jm}, s^l_{jm} + p^l_{jm}]$ where $l \in \{1,2,3\}$. This idea is very similar to that proposed in Laborie et al. (2018) for stochastic scheduling.

It is also possible to model the restricted capacity of a machine with a `unary− resource` constraint which ensures that two tasks cannot be performed at the same time on the same machine (Vilím, Barták, & Čepek, 2004). In other words, in the crisp case, a machine m can be defined as a set of sequence variables r_m where the intervals that define each variable cannot overlap. Similarly to starting times, in the context of fuzzy scheduling, we need to define each machine as three machines that follow the same processing order, so each machine is represented with three sequence variables instead of one: r^1_m, r^2_m and r^3_m. Note that r^l_m contains all intervals $[s^l_{jm}, s^l_{jm} + p^l_{jm}]$ where $l \in \{1,2,3\}$. It is clear that the components of all tasks should be scheduled in the same order in r^1_m, r^2_m and r^3_m. For a machine m that is required for processing job j and j' $(j \neq j')$, either

$$s^l_{jm} + p^l_{jm} \le s^l_{j'm}, \forall l \in \{1,2,3\},$$

or

$$s^l_{j'm} + p^l_{j'm} \le s^l_{jm}, \forall l \in \{1,2,3\}$$

holds.

Taking all of this in consideration, in the following we present a CP model for the fJSP with expected makespan minimisation. The parameters and variables are given in Tables 1 and 3, respectively.

Table 3
Variables of CP model.

$\hat{s}_{jm}$	Starting time (TFN) of the task of job j on machine m, $m \in M_j$, $j \in J$
s^l_{jm}	lth component of $\hat{s}_{jm}$, $l \in \{1,2,3\}$
$\hat{C}_{\max}$	Makespan (TFN)
$C^l_{\max}$	lth component of makespan, $l \in \{1,2,3\}$
r^l_m	The lth component of set of tasks (intervals) of machine m

$$\min E[\hat{C}_{\max}]$$

s.t.

$$C^l_{\max} = \max\{s^l_{j\mu_j} + p^l_{j\mu_j}, \forall j \in J\} \quad \forall l \in \{1,2,3\} \tag{15}$$

$$\texttt{unary} - \texttt{resource}(r^l_m) \quad \forall m \in M, \forall l \in \{1,2,3\} \tag{16}$$

$$\texttt{same} - \texttt{sequence}(r^1_m, r^2_m, r^3_m) \quad \forall m \in M \tag{17}$$

$$s^l_{j\,succ_j(m)} \ge s^l_{jm} + p^l_{jm} \quad \forall j \in J, m \in M_j, \forall l \in \{1,2,3\} \tag{18}$$

$$s^l_{jm} \le s^{l+1}_{jm} \quad \forall j \in J, m \in M_j, \forall l \in \{1,2\} \tag{19}$$

Since the makespan is a TFN, the maximum completion time of each fuzzy component is calculated separately in constraint (15). Constraint (16) ensures that none of the operations requiring machine m overlap for any fuzzy component. Constraint (17) imposes a unique processing order on the problem, that is, tasks are scheduled in the same order for every fuzzy component, which keeps the integrity of the problem. Constraint (18) guarantees that job precedence constraints hold. Lastly, constraint (19) ensures that each starting time is a proper TFN.

3.3. Disjunctive graph model

A classical model for JSP with makespan minimisation is the well-known *disjunctive graph model G* where tasks are represented by nodes, and constraints by arcs (Pinedo, 2016). The arcs describing the precedence relations are *conjunctive*, whereas the ones that stand for capacity constraints are *disjunctive*. The cost of every arc is the processing time of its source node, which is a TFN in our problem.

By *selecting* a disjunctive arc from each pair such that the resulting subgraph is acyclic, we obtain a *solution graph*. This is often denoted as $G(\sigma)$, where σ is the corresponding task processing order. A solution to the problem (i.e. a feasible schedule) can be easily computed by propagating constraints in $G(\sigma)$ using the sum and the maximum of TFNs.

This idea is extended to the case of fuzzy durations in González Rodríguez, Vela, Puente, and Varela (2008). Three *parallel solution* *graphs* $G^i(\sigma)$, $i = 1, 2, 3$, are defined to decompose $G(\sigma)$, maintaining its topology whilst using the ith defining point of the TFN $\hat{p_x}$ as the cost of an arc $(x, y) \in G^i(\sigma)$. This decomposition allows to improve the search ability of local search strategies, while keeping the integrity of the fuzzy problem by not letting it turn into three independent crisp problems in any case. This idea serves the same purpose as constraint (17) of the CP model, which ensures that the task ordering is the same in the three machines, thus maintaining the integrity of the problem. Since weights in each parallel graph are deterministic, a critical path in $G^i(\sigma)$ is the longest path from node *start* to node *end*. The set of critical paths in $G(\sigma)$ is the union of critical paths in $G^i(\sigma)$, $i = 1, 2, 3$. Similarly, the set of critical nodes, arcs and blocks is the union of those sets in each $G^i(\sigma)$. This model offers some desirable properties that have allowed to design some of the most successful heuristic solving methods for fJSP, namely:

- The reversal of a critical arc in any solution graph produces a feasible processing order (González Rodríguez et al., 2008).
- The reversal of a non-critical arc can never improve the makespan (González Rodríguez et al., 2008).
- The evaluation of the makespan after the reversal of a critical arc can be done very efficiently using head and tail propagation in the graph (González Rodríguez, Vela, Hernández-Arauzo, & Puente, 2009).
- The use of heads and tails in the graph allows for a fast estimation (constant computational cost) of the makespan of the solution that results from reversing a critical arc. This estimation is also a lower bound of the makespan (González Rodríguez et al., 2009).
- The reversal of an arc that is not the first or the last arc of a critical block, can never improve the makespan (González Rodríguez et al., 2009).

4. Solving methods

IBM ILOG CPLEX and its CP Optimizer tool are commercial solvers widely used to solve MILP and CP models, respectively (IBM, 2020). Here, we run CPLEX and CP Optimizer version 12.9 with a time limit of six hours (i.e. 21 600 s.) and using single thread to search for optimal solutions. In both cases, if the solver finds the optimum before the time limit, the execution stops and the run time is reported. Otherwise, if a feasible sub-optimal solution has been obtained at the end of the run, the corresponding lower and upper bounds for the optimum are reported.

For the disjunctive graph model, a metaheuristic method is employed. The review of published metaheuristic methods in Palacios et al. (2016) concluded that a memetic algorithm (MA) was the method offering the best results for makespan minimisation across a wide set of benchmark instances, compared with other seven state-of-the-art solving methods from the literature: an ant bee colony algorithm, three different genetic algorithms, two particle-swarm optimisation algorithms and a swarm-based neighbourhood search. For the comparisons, $\max_R$ and $\leq_R$ were used as fuzzy maximum approximation and ranking method respectively, since most of the state-of-the-art algorithms reported results based on this arithmetic. A variant of MA substituting the original local search with tabu search using a more sophisticated neighbourhood structure was proposed in Palacios, Puente, González-Rodríguez, and Vela (2013) under the name HTS. The experimental results, using $\max_I$ and $\leq_E$, showed that HTS outperformed MA, thus becoming the most competitive method so far. Since then, the methods proposed to minimise makespan for the fuzzy job shop either tackle multiobjective problems or use fuzzy numbers other than TFNs, making it impossible to establish any comparisons. Also, the instances used in these works to obtain experimental results are solved to optimality by HTS, as will be shown in Section 6. Therefore, we use HTS as a solver for the fJSP based on the disjunctive graph model. For each of its parameters (population size, stopping criteria for the local search and genetic algorithm, crossover and mutation probabilities etc.), the values that ensure a proper convergence are chosen. Due to its stochastic nature, it is run 30 times for each instance.

5. Available benchmarks

In this section, the benchmarks for fuzzy job shop scheduling from the literature are summarised along with key information for meaningful comparisons. A complete review of the benchmark can be found in Palacios et al. (2016), including best-known upper bounds until then and the fuzzy arithmetic used in each case. In addition, all the instances are available as supplementary material.

They can be classified in two groups: the first group of instances generated from scratch that we will call "original fuzzy instances" and second one of "fuzzified instances" built from benchmarks for the deterministic JSP by fuzzifying task durations.

The first group contains the set of instances proposed by Sakawa et al. in Sakawa and Kubota (2000), Sakawa and Mori (1999), which is extensively used for fuzzy job shop (denoted S6.1, S6.2, S6.3, S6.4 and S10.1, S10.2, S10.3, S10.4 depending of their size 6×6 or 10×10), and three more original instances: Lei01, Lei02 of size 15×10 from Lei (2010b), and instance LP01 of size 16×16 from Li and Pan (2013).

The fuzzified instances have been built from the following well-known benchmarks: FT from Fisher and Thomson (1963), La from Lawrence (1984), ABZ from Adams et al. (1988), ORB from Applegate and Cook (1991) and Ta from Taillard (1993). Four different methods have been mainly used in the literature to generate fuzzy versions of JSP instances. the method originally proposed by Fortemps in Fortemps (1997) for six-point fuzzy numbers (denoted F) was adapted to obtain TFNs in González Rodríguez et al. (2008) and Palacios et al. (2016); another method (G) was proposed by Ghrayeb in Ghrayeb (2003), and later used too in Niu, Jiao, and Gu (2008); a third one (Z) was proposed by Zheng et al. in Zheng et al. (2011); and the fourth one (S) was proposed by Song et al. in Song, Zhu, Yin, and Fuming (2006). Two more methods were introduced in Tsujimura, Gen, and Kubota (1995) (T) and Lin (2002) (L), but they were only applied to fuzzify two and three instances respectively. In the following, we refer to each fuzzified instance using the name of the original crisp instance subscripted with the fuzzification method, in accordance with Palacios et al. (2016).

Table 4 summarises all available fJSP instances. The first two columns contain the names of the instances and their respective sizes represented by $|J| \times |M|$ where $|J|$ is the number of jobs and $|M|$ is the number of machines. In the case of fuzzified instances, extra columns are included showing the different fuzzification methods applied to generate the fuzzy durations.

It is worth mentioning that other instances have been used in other papers. However, either these instances are not available or they correspond to different variants of the problem. Therefore, they are not relevant for our study.

The first set of lower bounds for the expected makespan of all instances is provided in Palacios et al. (2016). The second source of lower bounds are optimal solutions (or lower bounds thereof) for associated deterministic instances. In the case of fuzzified instances obtained in such a way that every TFN is symmetric and its modal value is the original duration, the optimal solution of the original deterministic instance (or any lower bound thereof) provides a lower bound for the expected makespan of the fuzzy solution (Fortemps, 1997). Such optimal solutions or lower bounds of the associated deterministic instances (well-known benchmarks for JSP) can be found and keep being updated in the literature[1] (van Hoorn, 2018). In the case of original instances or instances fuzzified with non-symmetric TFNs, thanks to the linearity of the expected value, the optimal solution (or a

[1] An up-to-date record of these bounds can be found in the repository http://optimizizer.com/jobshop.php.

Table 4
Existing instances for fJSP.

Instance	Size	Fuzz.Method						Instance	Size	Fuzz.Method					
S6.1–4	6 × 6	*NA*						La$_X$01	10 × 5			G	L		
S10.1–4	10 × 10	*NA*						La$_X$02	10 × 5					S	
Lei01,02	15 × 10	*NA*						La$_X$03,05	10 × 5			G			
LP01	16 × 16	*NA*						La$_X$06	15 × 5				L		
								La$_X$07,09	15 × 5			G			
FT$_X$06	6 × 6	F	T	G	L			La$_X$11	20 × 5	F					
FT$_X$10	10 × 10	F		G		S		La$_X$12–14	20 × 5	F		G			
FT$_X$20	20 × 5	F	T	G				La$_X$19	10 × 10					S	
								La$_X$20	10 × 10						Z
ORB$_X$1–5	10 × 10						Z	La$_X$21	15 × 10	F				S	Z
								La$_X$22	15 × 10						Z
ABZ$_X$5,6	10 × 10			G			Z	La$_X$24,25	15 × 10	F				S	
ABZ$_X$7–9	20 × 15	F						La$_X$36,37,39	15 × 15					S	
								La$_X$27,29	20 × 10	F				S	
Ta$_X$21–30	20 × 20	F						La$_X$38,40	15 × 15	F				S	
Ta$_X$41–50	30 × 20	F													

lower bound) of the deterministic problem where durations correspond to the expected value of the TFNs provides a lower bound for the fuzzy solution. Bounds for these associated deterministic instances can then be obtained using the commercial solver IBM ILOG CPLEX (IBM, 2020).

6. Experimental results

In this section, we present the results obtained on the existing fuzzy JSP instances with the MILP and CP solvers considering together with the HTS algorithm. For the sake of clarity, the results will be presented with instances organised in three groups (small, medium and large size) according to the total number of tasks.

With the help of commercial solvers, a first aim of this study is to identify the point before which we can rely on mathematical models and after which metaheuristics become the dominant option. In the process, we also intend to update the best-known lower and upper bounds for the expected value of the optimal makespan of each instance.

In particular, for the lower bound, we will consider the lower bounds from the literature and those obtained from the deterministic problems as mentioned in Section 5, together with those obtained in this experimental study, so the maximum of these values will be referred to as *best lower bound*. The expected makespan values of the best solutions found using the different solvers mentioned in Section 4 constitute the upper bounds. The complete list of lower bounds from the literature, as well as all the newly-found bounds are disclosed in Appendix A. When comparing the upper bounds obtained by the different solvers in the following subsections, the Relative Error (RE) with respect to the best known lower bound is used:

$$RE = \frac{E[\widehat{UB}] - E[\widehat{LB}]}{E[\widehat{LB}]} \tag{20}$$

The third goal of this study is to assess the hardness of the existing instances, identifying those that represent a challenge for future exact or approximate solving methods. Finally, we will also propose a new benchmark for the fJSP, built from well-known instances for the deterministic problem, and provide a first set of upper and lower bounds for the expected makespan as reference for future research.

All the solving methods have been run on the same machine (a PC with Intel Xeon Gold 6132 processor at 2.6 GHz and 128 Gb RAM with Linux CentOS v.6.9) using the same programming language (C++) and compiler. More detailed results of each method on every instance are openly available online.[2]

[2] Repository section at https://www.di.uniovi.es/iscop.

6.1. Small-size instances

Table 5 contains the average results on the instances with less than 100 tasks with instances in the same family and with the same dimension grouped together. The number and dimension of instances in each group are shown in the second and third columns, respectively. Columns 4, 5 and 6 report the results of the MILP solver Since it cannot always find the optimal solution within the time limit of 6 h, column 4 (# feas.) indicates the number of instances where at least one feasible solution is found. Column 5 (RE) gives the average relative error with respect to the *best lower bound* for the instances that are not solved to optimality. Column 6 reports the average run time for the instances that are optimally solved within the time limit, with the dash sign representing that all instances of this group consume the time limit without reaching or proving optimality. Finally, columns 7 and 8 give the average run times of CP model and HTS algorithm in seconds, respectively. Since both methods find optimal solutions in all instances, their RE values are not presented.

It can be observed that the instances of the first two families do not present a challenge for any of the methods since they are optimally solved in very short times by all three methods independently on the fuzzification method. In the last families, MILP reaches the time limit before proving optimality. The low (even zero) RE values mean that it actually finds the optimal solution in many cases, but it cannot prove its optimality. The CP model is solved optimally in a very short time in comparison to MILP, and HTS finds (but does not certify) the optimal solution in every run. Hence, there is a remarkable difference between MILP and the other two methods (i.e. HTS and CP).

Although these instances could be used in the future for method validation and verification, due to the fact that they can be optimally solved in less than a minute by CP and in less than a fraction of a second by HTS, it is safe to say that they are too trivial to be considered to evaluate the performance of a new method.

6.2. Medium-size instances

In Table 6, the average results of instances from 100 to 200 tasks (excluded) are given. As before, the name of the instance family, the number of instances in it and the dimensions are given in the first three columns. For the MILP, the number of instances for which a feasible solution is found, and the average RE for these instances with respect to the *best lower bound* are presented in columns 4 and 5, respectively. Since it hits the time limit for all the instances, its run time is omitted in the table. For the last two families, MILP cannot find a feasible solution within the time limit, therefore the RE is represented as a dash. On the contrary, the CP model is solved optimally within the time limit in every case, therefore only its run time is reported. Lastly, HTS also finds the optimal solution of each instance in at least one run, but not

Table 5
Average results of instances with less than 100 tasks.

Family	# inst.	Size	MILP			CP	HTS
			# feas.	RE	Time (s)	Time (s)	Time (s)
S6.1–4	4	6 × 6	4	0.00%	37.4	0.1	0.2
FT_X06	4	6 × 6	4	0.00%	45.2	0.1	0.2
La_X01–03,05	5	10 × 5	5	1.04%	–	21.2	0.3
La_X06,07,09	3	15 × 5	3	0.00%	–	38.0	0.3

Table 6
Average results of instances from 100 to 200 tasks.

Family	# inst.	Size	MILP		CP	HTS	
			# feas.	RE	Time (s)	RE	Time (s)
ABZ_X5,6	4	10 × 10	4	3.98%	612.5	0.05%	0.9
FT_X10	3	10 × 10	3	14.27%	938.8	0.38%	1.4
La_X19,20	2	10 × 10	2	1.68%	745.5	0.16%	0.9
ORB_X1–5	5	10 × 10	5	10.40%	727.6	0.26%	1.4
S10.1–4	4	10 × 10	4	5.32%	76.5	0.17%	1.0
FT_X20	3	20 × 5	2	19.49%	236.3	0.11%	1.8
La_X11–14	7	20 × 5	3	0.00%	92.1	0.00%	0.4
La_X21–25	8	15 × 10	0	–	4868.0	0.60%	2.6
Lei01,02	2	15 × 10	0	–	5105.5	0.63%	3.4

in all of them. Columns 7 and 8 report the average RE and run time of the HTS method are given.

In opposition to the small-size instances, in this set MILP spends 6 h without finding the optimum, and sometimes even fails to obtain a feasible solution. However, CP can solve all instances optimally within much shorter time. These results imply that the modelling approach has a big impact and despite MILP being the most popular method in the field, CP has a significant potential as far as the exact methods go. On another note, HTS takes remarkably short time to solve all medium-size instances. However, it does not reach the optimal solution in all its runs as its average RE is not zero. Despite that, now that we know the optimal expected makespan values for all these instances, we can confirm that HTS finds high-quality suboptimal solutions in very short time.

It is noteworthy that even among the same size instances, run times of CP and HTS vary greatly such as for FT_X10 and S10.1–4. Moreover, instances with the same number of tasks but different structure such as La_X19,20 and La_X11–14 have remarkable differences in CP and HTS performances in terms of run times and RE values despite both being from the same family. Even for the 7 La_X11–14 instances, MILP obtains optimal solutions for 3 of them but does not even find a feasible solution for the other 4. These results suggest that there are factors influencing the hardness of an instance other than the total number of tasks. This is in line with the results from Applegate and Cook (1991), where the computational hardness of JSP instances with the same size and structure vary remarkably.

Regarding the use of these instances as future benchmarks, instances S10.1–4 do not pose much of a challenge for CP and therefore do not seem to be of special interest. On the other hand, the remaining 10 × 10 families can be considered interesting since CP needs longer time to solve them and HTS is not finding the optimal solution in all its runs. However, one could argue that families FT_X10 and ORB_X1–5 are more challenging due to the HTS average RE. In terms of the 100-task non-square instance set, FT_X20 family is slightly more challenging than La_X11–14. Lastly, both families with size 15 × 10 could be taken into account for a comparative study in the future since MILP cannot find any feasible solutions for neither of them and CP and HTS take considerably longer time to solve them.

6.3. Large-size instances

In Table 7, the average results of instances with 200 tasks or more are given. Since MILP cannot find a feasible solution for any of the instances within the time limit, it is excluded from this table. Column 4 gives the number of instances optimally solved by CP along with their average run times in column 6. In the last two families, none of instances are solved optimally within 6 h, so their average run times are represented with a dash. The average RE values of the instances that do not terminate within the time limit are given in column 5. Finally, the average RE values with respect to *best lower bound* and average run times of HTS are presented in the last columns.

In this table, run times are noticeably longer than in the previous ones, showing that the problem takes longer time to solve as the number of tasks increases. Among 15 instances with 200 to 300 tasks, only 8 of them can be solved optimally by CP within the time limit. The results of HTS point in the same direction: RE values and run times increase compared to small or medium-size instances. As expected, the number of tasks has a clear impact on the hardness of an instance. However, we can see once more that it is not the only factor, since HTS obtains better RE for La_X36–40 than for La_X27,29. We shall discuss this further in Section 6.6. For future research, all instances in this set seem adequate for testing solving methods. Nonetheless, it can be observed that LP01 and ABZ_X7–9 instances are the most challenging ones, since CP cannot find any optimal solutions and the average RE and run times of HTS are higher, hence they are the most suitable for comparative studies.

6.4. Taillard instances

Instance size is often viewed as a measure of difficulty (Shukor et al., 2018). However, results from previous subsections already suggest that size on its own may not be so decisive. To shed more light on this, we propose a thorough analysis on fuzzy versions of the whole Ta benchmark (Taillard, 1993), with instance sizes ranging from 225 tasks to 2000 tasks. In addition to Ta_F21–30 and Ta_F41–50 from Palacios et al. (2016), we use the same fuzzification method to obtain fuzzy versions of the remaining instances in the benchmark set: Ta01–10 (15 × 15), Ta11–20 (20 × 15), Ta31–40 (30 × 15), Ta51–60 (50 × 15), Ta61–70 (50 × 20), and Ta71–80 (100 × 20). As explained earlier, with this fuzzification method the best-known lower bounds for the original deterministic instances are also lower bounds for the expected makespan. All the new instances are made available online as supplementary data.

In Table 8, the average results of all the Ta instances are presented. Similarly to the large-size instances, MILP cannot find any feasible solution within the time limit in any case, hence its results are excluded. The number of instances solved optimally by CP are given in column 4. For some instances, CP finds a feasible solution with the objective function value equal to the *best lower bound* despite not terminating. We

Table 7
Average results of instances with more than 200 tasks.

Family	# inst.	Size	CP			HTS	
			# opt.	RE	Time (s)	RE	Time (s)
La_X27,29	4	20 × 10	3	1.18%	6196.0	2.10%	5.6
La_X36–40	7	15 × 15	5	0.47%	1841.2	1.00%	4.9
LP01	1	16 × 16	0	3.63%	–	3.30%	10.7
ABZ_X7–9	3	20 × 15	0	3.66%	–	4.20%	10.5

Table 8
Average results of Taillard instances.

Family	# inst.	Size	CP				HTS	
			# opt.	# cert.	RE	Time (s)	RE	Time (s)
Ta_X01–10	10	15 × 15	8	0	0.79%	6612.2	1.24%	5.2
Ta_X11–20	10	20 × 15	2	0	3.22%	5084.4	3.27%	11.2
Ta_X21–30	10	20 × 20	0	0	6.14%	–	5.16%	16.9
Ta_X31–40	10	30 × 15	1	0	2.48%	2366.8	4.15%	26.6
Ta_X41–50	10	30 × 20	0	0	6.97%	–	9.49%	45.4
Ta_X51–60	10	50 × 15	0	4	0.09%	–	0.49%	55.6
Ta_X61–70	10	50 × 20	0	2	0.30%	–	3.13%	129.4
Ta_X71–80	10	100 × 20	0	6	0.04%	–	0.38%	273.5

Table 9
Summary of updated bounds.

	#inst.	#opt.	*#new LB*
Small	16	16	10
Medium	38	38	35
Large	15	6	5
Taillard	80	23	9
Total	149	83	59

call this kind of solutions as *certified* and the number of such solutions is presented in column 5. The average RE of the instances that are not optimally solved can be found in column 6 and the average run times of the ones that finished before the time limit are given in column 7. In the last two columns, the average RE and run times of HTS are presented.

As above, as the number of tasks increase, there are fewer instances where the CP solver terminates before hitting the time limit. Likewise, HTS takes longer time to run. However, for Ta_X51–60, Ta_X61–70 and Ta_X71–80 instances, we can observe that CP reaches the optimal solution in 12 out of 30 instances and has quite small RE values for the rest. This is particularly interesting when we consider that only 1 out of 30 instances from Ta_X21–30, Ta_X31–40 and Ta_X41–50 is optimally solved despite having much smaller sizes. We can observe a similar behaviour in the RE values of HTS: the larger instances have surprisingly smaller RE values.

In terms of number of optimal or certified solutions and average REs, Ta_X21–30 and Ta_X41–50 appear to be the most difficult instances for both methods, which makes them the most challenging benchmarks for future research. However, all of the instances on this set pose a challenge for future solving methods.

6.5. Impact of considering uncertainty

When working with uncertainty, it is common to question whether it is worth considering the uncertainty during the optimisation process. Doing so may increase the complexity of the solving methods, but it may yield more robust solutions as well, since it considers all available information. Following the methodology used for other fuzzy scheduling problems (i.e. Palacios et al., 2014, 2015), we solve the already available instances from Section 5 without considering uncertainty. For each fuzzy instance, we create a crisp one where the processing time of each task is the expected value of the original TFN, and solve it using HTS. To compare the predictive schedules obtained after solving the fuzzy instances and the crisp ones, we use the $\bar{\epsilon}$-robustness measure

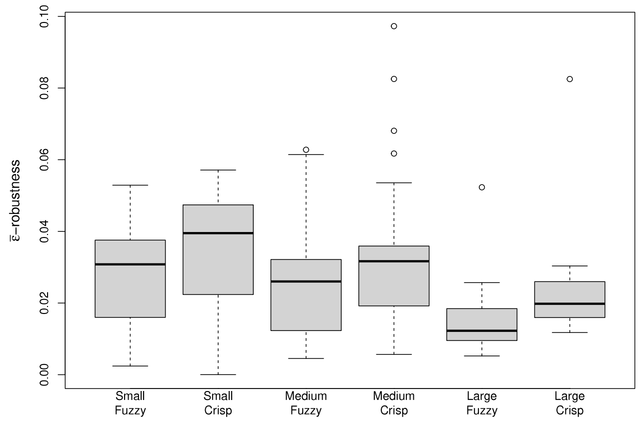

Fig. 1. $\bar{\epsilon}$ values obtained by solving the small, medium and large sized fJSP instances and their associated deterministic counterparts.

introduced in Section 2 generating $K = 1000$ scenarios. Fig. 1 shows the $\bar{\epsilon}$ values obtained for the instances grouped by size according to the previous sections. The figure shows that for all sizes, the schedules obtained by solving the fuzzy instances are more robust than those obtained by solving the deterministic ones. Out of the 69 tested instances, solving the fuzzy version yields more robust solutions in 67 of them (97%). A Wilcoxon test for paired samples reveals that for each size, the differences are significant between the two approaches. These results align with those of Palacios et al. (2014, 2015), showing that modelling uncertainty leads to more robust solutions. Moreover, 44 of the tested instances are solved to optimality both in the fuzzy and crisp versions, showing that the optimal solution of the fuzzy problem is also more robust than the optimal solution of its associated crisp problem.

6.6. Summary

Results have shown that, as expected from an NP-hard problem, instance size is the main indicator of the hardness of an instance. The commercial solver has not been able to find feasible solutions for the MILP model for any instance with 150 tasks or more, but it is capable of finding the optimal solution to the smallest instances, of size 6 × 6. For instances between those sizes, MILP never terminates within the time limit, despite reaching the optimal solutions. This indicates that commercial solvers for the MILP model are not sufficient for medium

Table A.10
Best known bounds for each instance.

Inst.	Size	$E[\widehat{LB}]$			Best $\widehat{UB}$		
		old	*new*	Met.	$\widehat{UB}$	$E[\widehat{UB}]$	Met.
S6.1	6 × 6	**79.75**	**79.75**	MC	**(56 80 103)**	**79.75**	MCH
S6.2		69.25	**70.25**	MC	**(51 70 90)**	**70.25**	MCH
S6.3		66.00	**66.25**	MC	**(51 65 84)**	**66.25**	MCH
S6.4		**36.00**	**36.00**	MC	**(27 36 45)**	**36.00**	MCH
S10.1	10 × 10	128.50	**128.75**	C	**(96 129 161)**	**128.75**	CH
S10.2		122.50	**123.75**	C	**(92 120 163)**	**123.75**	CH
S10.3		**115.00**	**115.00**	C	**(85 116 143)**	**115.00**	CH
S10.4		45.75	**46.50**	C	**(28 47 64)**	**46.50**	CH
Lei01	15 × 10	197.25	**198.75**	C	**(136 200 259)**	**198.75**	C
Lei02		163.00	**165.00**	C	**(118 164 214)**	**165.00**	CH
LP01	16 × 16	**186.00**	155.75	C	(144 187 246)	191.00	H
FT$_F$06	6 × 6	**55.00**	**55.00**	MC	**(54 55 56)**	**55.00**	MCH
FT$_T$06		**55.25**	**55.25**	MC	**(42 55 69)**	**55.25**	MCH
FT$_G$06		**55.75**	**55.75**	MC	**(52 55 61)**	**55.75**	MCH
FT$_L$06		**56.25**	**56.25**	MC	**(42 55 73)**	**56.25**	MCH
FT$_F$10	10 × 10	930.00	**932.75**	C	**(882 930 989)**	**932.75**	CH
FT$_G$10		938.50	**945.75**	C	**(865 930 1058)**	**945.75**	CH
FT$_S$10		935.25	**937.75**	C	**(844 930 1047)**	**937.75**	CH
FT$_F$20	20 × 5	1165.00	**1165.50**	C	**(1094 1165 1238)**	**1165.50**	CH
FT$_T$20		1164.25	**1164.50**	C	**(1112 1165 1216)**	**1164.50**	CH
FT$_G$20		1189.75	**1190.00**	C	**(1074 1165 1356)**	**1190.00**	CH
ORB$_Z$01	10 × 10	1073.50	**1074.00**	C	**(961 1060 1215)**	**1074.00**	CH
ORB$_Z$02		**896.00**	**896.00**	C	**(784 889 1022)**	**896.00**	CH
ORB$_Z$03		**1015.25**	**1015.25**	C	**(900 1005 1151)**	**1015.25**	CH
ORB$_Z$04		1014.50	**1015.00**	C	**(894 1006 1154)**	**1015.00**	CH
ORB$_Z$05		897.25	**898.00**	C	**(800 887 1018)**	**898.00**	CH
ABZ$_G$5	10 × 10	1247.00	**1251.00**	C	**(1126 1236 1406)**	**1251.00**	C
ABZ$_Z$5		1248.75	**1250.75**	C	**(1111 1239 1414)**	**1250.75**	C
ABZ$_G$6		950.25	**957.50**	C	**(876 943 1068)**	**957.50**	C
ABZ$_Z$6		952.50	**954.25**	C	**(842 945 1085)**	**954.25**	C
ABZ$_F$7	20 × 15	**656.00**	638.00	C	(628 660 703)	662.75	C
ABZ$_F$8		**648.00**	538.50	C	(643 675 718)	677.75	H
ABZ$_F$9		**678.00**	552.50	C	(653 687 731)	689.50	H
La$_G$01	10 × 5	–	**674.00**	MC	**(625 666 739)**	**674.00**	CH
La$_L$01		–	**673.75**	MC	**(501 666 862)**	**673.75**	CH
La$_S$02		–	**656.00**	MC	**(601 655 713)**	**656.00**	CH
La$_G$03		–	**612.75**	MC	**(549 597 708)**	**612.75**	CH
La$_G$05		–	**599.75**	MC	**(548 593 665)**	**599.75**	CH
La$_L$06	15 × 5	–	**926.25**	MC	**(667 926 1186)**	**926.25**	CH
La$_G$07		–	**901.00**	MC	**(821 890 1003)**	**901.00**	CH
La$_G$09		–	**959.00**	MC	**(869 951 1065)**	**959.00**	CH
La$_F$11	20 × 5	–	**1222.00**	C	**(1164 1222 1280)**	**1222.00**	CH
La$_F$12		–	**1039.00**	C	**(975 1039 1103)**	**1039.00**	CH
La$_G$12		–	**1062.50**	C	**(968 1039 1204)**	**1062.50**	CH
La$_F$13		–	**1150.00**	C	**(1072 1150 1228)**	**1150.00**	CH
La$_G$13		–	**1174.00**	C	**(1070 1150 1326)**	**1174.00**	CH
La$_F$14		–	**1292.00**	C	**(1203 1292 1381)**	**1292.00**	CH
La$_G$14		–	**1325.75**	C	**(1197 1292 1522)**	**1325.75**	CH
La$_S$19	10 × 10	843.25	**846.00**	C	**(752 842 948)**	**846.00**	CH
La$_Z$20		912.50	**914.00**	C	**(816 902 1036)**	**914.00**	CH
La$_F$21	15 × 10	1046.00	**1049.25**	C	**(977 1046 1128)**	**1049.25**	C
La$_S$21		1044.75	**1050.50**	C	**(943 1046 1167)**	**1050.50**	CH
La$_Z$21		1056.50	**1057.75**	C	**(943 1046 1196)**	**1057.75**	C
La$_Z$22		937.00	**938.00**	C	**(833 927 1065)**	**938.00**	C
La$_F$24		935.00	**938.75**	C	**(871 937 1010)**	**938.75**	C
La$_S$24		938.25	**942.25**	C	**(840 938 1053)**	**942.25**	C
La$_F$25		977.00	**978.50**	C	**(913 978 1045)**	**978.50**	CH
La$_S$25		985.75	**986.75**	C	**(882 977 1111)**	**986.75**	C

(*continued on next page*)

to large-size instances, and the use of more advanced and tailored exact solving methods is recommended.

Size is not the only factor affecting the solving methods. For example MILP has found the optimal solutions to all instances of size 15 × 5 (without certifying it), but it could not find it for the instances of size 10 × 5. Similarly, in the set of instances with 100 tasks MILP finds feasible solutions for all instances of size 10 × 10, but the RE values are far from being similar.

Regarding the CP model, the solver can find solutions to larger-size instances within a reasonable time limit. In fact, it can reach optimality on all instances with less than 200 tasks within the time limit. Furthermore, it finds better bounds than the MILP model in all instances and, when both methods can reach optimality, CP is faster.

Table A.10 (*continued*).

Inst.	Size	$E[\widehat{LB}]$			Best $\widehat{UB}$		
		old	*new*	Met.	$\widehat{UB}$	$E[\widehat{UB}]$	Met.
La_F27	20 × 10	**1235.00**	**1235.00**	C	**(1154 1235 1316)**	**1235.00**	C
La_S27		**1233.00**	**1233.00**	C	(1132 1242 1365)	1245.25	H
La_F29		**1152.00**	935.50	C	(1081 1154 1238)	1156.75	C
La_S29		**1156.50**	935.00	C	(1058 1165 1315)	1175.75	H
La_S36	15 × 15	1277.00	**1281.50**	C	**(1160 1275 1416)**	**1281.50**	C
La_S37		1398.25	**1405.00**	C	**(1262 1399 1560)**	**1405.00**	C
La_F38		**1196.00**	1025.50	C	(1128 1196 1284)	1201.00	C
La_S38		**1204.50**	1026.00	C	(1091 1205 1342)	1210.75	CH
La_S39		1234.00	**1241.25**	C	**(1112 1233 1387)**	**1241.25**	C
La_F40		1222.00	**1226.75**	C	**(1146 1224 1313)**	**1226.75**	C
La_S40		1226.50	**1233.00**	C	**(1119 1228 1357)**	**1233.00**	CH

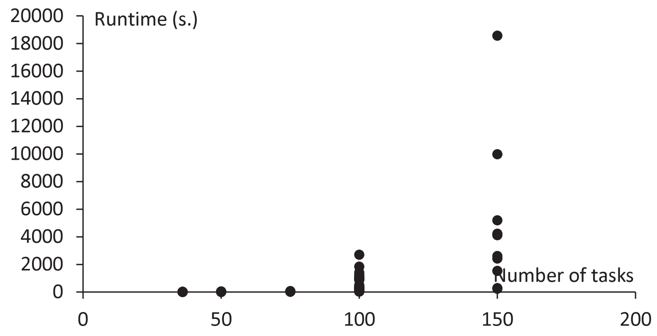

Fig. 2. Average run time taken by CP to prove optimality on instances grouped by size.

Fig. 2 shows the time elapsed to solve CP on the instance sizes that could be solved to optimality sorted by size. On average, the behaviour is as before: the larger the instance, the longer it takes to prove optimality. However, we can see a high variance among instances of the same size, indicating once more that size is not the only factor affecting difficulty.

In fact, the ratio between the number of machines and the number of jobs ($r = |M|/|J|$) seems to be a contributing factor for instance difficulty. Consider for example the instances FT_X10 and FT_X20, from the same test-bed and with the same number of tasks, but with ratios $r = 1$ and $r = 0.25$ respectively. The CP model is optimally solved in all cases, but it takes four times longer on average for the three fuzzy instances obtained from FT10 compared to the three instances obtained from FT20. HTS can also find the optimal solution in all cases, but the RE of an average solution is 0.38% on the FT10 instances compared to 0.11% on the FT20. This is even more obvious when looking at fuzzy instances from La19,20 and La11–14. This pattern is clearer when studying the Ta family in terms of RE values for CP and HTS. The impact of the ratio r also seems to increase with instance size: instances Ta_Z71–80 with 2000 tasks and $r = 0.2$ present smaller RE average values than instances with fewer tasks but higher ratio. The same happens with instances Ta_Z21–30 ($r = 0.5$). Regarding the different fuzzification methods, a more detailed study has not shown significant differences between the results on instances made with different fuzzification methods, thus this does not appear to be an influencing factor.

In summary, the MILP model is always outperformed by the CP model (in this sense results are completely aligned with the deterministic counterpart (Laborie, 2018)). Thanks to these models, solvers can find and certify optimality for small and medium-size instances. We can also experimentally confirm that the run time depends not only on the instance size, but also on other factors, in particular, the ratio r. Up to medium size, the state-of-the-art metaheuristic can also find the optimal solution in at least one of its runs, if not all, in much shorter time. However, due to its nature it is not capable of certifying optimality. For instances with more than 200 tasks, the lower and upper bounds that CP reports are not equal, thus it cannot prove optimality. On the other hand, HTS provides similar upper bounds in much less time, so metaheuristics can be seen as a better alternative for large-size instances.

During the experimental study, many bounds for the fJSP have been updated. Table 9 summarises the number of new bounds and optimal solutions found per instance set. Instance sets, number of instances per set, number of optimal solutions reached, and number of newly found lower bounds are presented. In total, optimal solutions have been certified for 83 instances, confirming 24 of the existing lower bounds for the expected makespan to be optimal and reporting 59 new ones. Among these, 40 of them improve the previously known LB values and 19 are reported for the first time. Detailed data for each instance (former best LB, new LB, best feasible makespan TFN and best expected UB, along with the methods that allow us to compute these bounds) are provided in Appendix A.

7. Conclusions

In this paper we have addressed the fuzzy JSP with makespan minimisation. We have proposed two mathematical models for this problem: a MILP and a CP model. Both models are solved with off-the-shelf solvers. A disjunctive graph model from the literature is also considered and a competitive metaheuristic (HTS) from the literature is used as solving method.

Combining all three approaches, we have carried out a critical evaluation of all available test-beds. In total, 149 fuzzy instances ranging from size 6 × 6 to 100 × 20, have been covered. Our study has shown the potential of exact solvers for the fuzzy job shop problem, solving to optimality the majority of the instances from the literature within a reasonable amount of time. Besides providing optimal solutions, we have also improved the former best LB and reported many new LB for the first time. The limitations of each method have been studied, showing that CP outperforms MILP in small to medium-size instances, but it starts failing to prove optimality when instances have 200 tasks or more. After that point, meta-heuristics provide similar results in less time. Numerous upper bounds for the expected makespan have been also found based on the best feasible solution for those instances where the optimum has not been found yet. We have also proposed 60 new fuzzy instances of varied sizes to increase the variety of benchmark instances for future research.

Additionally, a thorough analysis of the difficulty of the existing benchmark instances has shown that, as expected, larger instances tend to be more difficult. But size is not the only factor to take into account. In particular the ratio between the number of machines and number of jobs also seems to be a relevant factor. Based on the computational analysis, we have indicated those instances that appear to be interesting as benchmarks for future research on fJSP. We have made all our results publicly available to the research community, together with LB and UB values for the expected makespan as well as the makespan of the best feasible solution, for future reference.

Table A.11
Best known bounds for fuzzified Taillard instances.

Inst.	Size	$E[\widehat{LB}]$			Best $\widehat{UB}$		
		old	*new*	Met.	$\widehat{UB}$	$E[\widehat{UB}]$	Met.
Ta$_F$01	15 × 15	1231.00	**1238.75**	C	**(1149, 1231, 1344)**	**1238.75**	CH
Ta$_F$02		1244.00	**1246.00**	C	**(1152, 1245, 1342)**	**1246.00**	CH
Ta$_F$03		1218.00	**1223.00**	C	**(1147, 1219, 1307)**	**1223.00**	C
Ta$_F$04		1175.00	**1177.00**	C	**(1083, 1175, 1275)**	**1177.00**	CH
Ta$_F$05		**1224.00**	1030.00	C	(1158, 1224, 1319)	1231.25	C
Ta$_F$06		**1238.00**	1011.25	C	(1156, 1241, 1341)	1244.75	H
Ta$_F$07		1227.00	**1231.25**	C	**(1147, 1228, 1322)**	**1231.25**	CH
Ta$_F$08		1217.00	**1224.25**	C	**(1134, 1218, 1327)**	**1224.25**	CH
Ta$_F$09		1274.00	**1280.50**	C	**(1181, 1274, 1393)**	**1280.50**	C
Ta$_F$10		1241.00	**1243.75**	C	**(1163, 1243, 1326)**	**1243.75**	CH
Ta$_F$11	20 × 15	**1357.00**	1118.50	C	(1289, 1378, 1492)	1384.25	H
Ta$_F$12		**1367.00**	1188.50	C	(1279, 1377, 1488)	1380.25	C
Ta$_F$13		**1342.00**	1094.50	C	(1269, 1358, 1478)	1365.75	H
Ta$_F$14		**1345.00**	**1345.00**	C	**(1248, 1345, 1442)**	**1345.00**	C
Ta$_F$15		**1339.00**	1071.50	C	(1274, 1366, 1483)	1372.25	H
Ta$_F$16		**1360.00**	1109.50	C	(1276, 1368, 1475)	1371.75	H
Ta$_F$17		1462.00	**1468.25**	C	**(1352, 1464, 1593)**	**1468.25**	C
Ta$_F$18		**1377.00**	1134.50	C	(1331, 1433, 1538)	1433.75	H
Ta$_F$19		**1332.00**	1090.75	C	(1251, 1349, 1468)	1354.25	H
Ta$_F$20		**1348.00**	1135.00	C	(1257, 1366, 1492)	1370.25	H
Ta$_F$21	20 × 20	**1642.00**	1372.50	C	(1561, 1669, 1804)	1675.75	H
Ta$_F$22		**1561.00**	1336.50	C	(1521, 1627, 1764)	1634.75	H
Ta$_F$23		**1518.00**	1314.25	C	(1493, 1578, 1692)	1585.25	H
Ta$_F$24		**1644.00**	1351.50	C	(1552, 1665, 1798)	1670.00	H
Ta$_F$25		**1558.00**	1323.50	C	(1530, 1633, 1756)	1638.00	H
Ta$_F$26		**1591.00**	1355.50	C	(1582, 1680, 1795)	1684.25	H
Ta$_F$27		**1652.00**	1438.50	C	(1585, 1694, 1828)	1700.25	H
Ta$_F$28		**1603.00**	1399.00	C	(1514, 1628, 1748)	1629.50	H
Ta$_F$29		**1583.00**	1357.75	C	(1541, 1645, 1766)	1649.25	H
Ta$_F$30		**1528.00**	1316.00	C	(1505, 1607, 1718)	1609.25	H
Ta$_F$31	30 × 15	**1764.00**	1379.50	C	(1669, 1783, 1905)	1785.00	H
Ta$_F$32		**1774.00**	1378.50	C	(1690, 1824, 1979)	1829.25	C
Ta$_F$33		**1788.00**	1424.50	C	(1724, 1849, 1992)	1853.50	H
Ta$_F$34		**1828.00**	1441.50	C	(1709, 1866, 2023)	1866.00	C
Ta$_F$35		**2007.00**	**2007.00**	C	**(1847, 2007, 2167)**	**2007.00**	CH
Ta$_F$36		**1819.00**	1436.00	C	(1720, 1847, 1987)	1850.25	C
Ta$_F$37		**1771.00**	1423.50	C	(1647, 1796, 1967)	1801.50	C
Ta$_F$38		**1673.00**	1321.25	C	(1575, 1692, 1847)	1701.50	C
Ta$_F$39		**1795.00**	1385.00	C	(1709, 1815, 1945)	1821.00	H
Ta$_F$40		**1651.00**	1314.00	C	(1600, 1714, 1858)	1721.50	C
Ta$_F$41	30 × 20	**1906.00**	1564.00	C	(1958, 2090, 2222)	2090.00	C
Ta$_F$42		**1884.00**	1563.50	C	(1883, 2019, 2173)	2023.50	H
Ta$_F$43		**1809.00**	1548.00	C	(1809, 1949, 2090)	1949.25	C
Ta$_F$44		**1948.00**	1575.00	C	(1933, 2044, 2173)	2048.50	C
Ta$_F$45		**1997.00**	1650.00	C	(1928, 2036, 2170)	2042.50	C
Ta$_F$46		**1957.00**	1626.00	C	(1949, 2066, 2215)	2074.00	C
Ta$_F$47		**1807.00**	1576.00	C	(1828, 1973, 2120)	1973.50	C
Ta$_F$48		**1912.00**	1641.00	C	(1881, 2012, 2160)	2016.25	C
Ta$_F$49		**1931.00**	1576.00	C	(1897, 2030, 2177)	2033.50	C
Ta$_F$50		**1833.00**	1528.50	C	(1894, 2019, 2158)	2022.50	H
Ta$_F$51	50 × 15	**2760.00**	1867.50	C	**(2549, 2760, 2971)**	**2760.00**	CH
Ta$_F$52		**2756.00**	1881.50	C	**(2581, 2756, 2931)**	**2756.00**	CH
Ta$_F$53		**2717.00**	1830.00	C	(2521, 2717, 2915)	2717.50	CH
Ta$_F$54		**2839.00**	2002.00	C	**(2605, 2839, 3073)**	**2839.00**	CH
Ta$_F$55		**2679.00**	1809.50	C	(2497, 2679, 2880)	2683.75	C
Ta$_F$56		**2781.00**	1853.00	C	(2574, 2781, 2994)	2782.50	CH
Ta$_F$57		**2943.00**	2040.00	C	**(2719, 2943, 3167)**	**2943.00**	CH
Ta$_F$58		**2885.00**	1963.50	C	(2682, 2885, 3128)	2895.00	CH
Ta$_F$59		**2655.00**	1809.00	C	(2469, 2655, 2865)	2661.00	CH
Ta$_F$60		**2723.00**	1898.50	C	(2525, 2723, 2927)	2724.50	CH
Ta$_F$61	50 × 20	**2868.00**	2078.00	C	(2691, 2868, 3072)	2874.75	C
Ta$_F$62		**2869.00**	2124.25	C	(2729, 2904, 3085)	2905.50	C
Ta$_F$63		**2755.00**	2022.00	C	(2529, 2755, 2987)	2756.50	C
Ta$_F$64		**2702.00**	1973.50	C	**(2495, 2702, 2909)**	**2702.00**	C
Ta$_F$65		**2725.00**	2002.00	C	(2559, 2725, 2948)	2739.25	C
Ta$_F$66		**2845.00**	2103.50	C	(2625, 2845, 3094)	2852.25	C
Ta$_F$67		**2825.00**	2057.50	C	(2606, 2826, 3083)	2835.25	C
Ta$_F$68		**2784.00**	2075.00	C	(2609, 2784, 2971)	2787.00	C
Ta$_F$69		**3071.00**	2247.50	C	**(2850, 3071, 3292)**	**3071.00**	CH
Ta$_F$70		**2995.00**	2126.25	C	(2792, 2995, 3219)	3000.25	C

(*continued on next page*)

Table A.11 (*continued*).

Inst.	Size	$E[\widehat{LB}]$			Best $\widehat{UB}$		
		old	*new*	Met.	$\widehat{UB}$	$E[\widehat{UB}]$	Met.
Ta_F71	100 × 20	**5464.00**	3406.50	C	**(5089, 5464, 5839)**	**5464.00**	CH
Ta_F72		**5181.00**	3228.00	C	**(4822, 5181, 5540)**	**5181.00**	CH
Ta_F73		**5568.00**	3469.25	C	**(5195, 5568, 5941)**	**5568.00**	CH
Ta_F74		**5339.00**	3320.25	C	(4950, 5339, 5739)	5341.75	CH
Ta_F75		**5392.00**	3369.50	C	(4959, 5392, 5830)	5393.25	C
Ta_F76		**5342.00**	3310.25	C	(5022, 5342, 5736)	5360.50	CH
Ta_F77		**5436.00**	3340.00	C	**(5050, 5436, 5822)**	**5436.00**	CH
Ta_F78		**5394.00**	3347.00	C	**(4980, 5394, 5808)**	**5394.00**	CH
Ta_F79		**5358.00**	3330.00	C	(4974, 5358, 5744)	5358.50	CH
Ta_F80		**5183.00**	3208.75	C	**(4833, 5183, 5533)**	**5183.00**	CH

CRediT authorship contribution statement

Sezin Afsar: Conceptualization, Methodology, Writing – review & editing, Software, Formal analysis, Investigation. **Camino R. Vela:** Supervision, Writing – original draft, Conceptualization, Methodology. **Juan José Palacios:** Conceptualization, Methodology, Writing – review & editing, Formal analysis, Investigation, Visualization. **Inés González-Rodríguez:** Conceptualization, Writing – review & editing, Methodology, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This work was supported by the Spanish Government [grant number PID2019-106263RB-I00] and [grant number TED2021-131938B-I00].

Data availability

We have shared the data as supplementary material with a link to the repository

Benchmarks for Fuzzy Job Shop Problem (Reference data) (IScOP Research Group).

Appendix A. Detailed solutions

Tables A.10 and A.11 contain the best bounds found during this work grouped by instance family and size. For each group, the size is given next to the first instance's name. Column $E[\widehat{LB}]_{old}$ reports the best known lower bound, either from the literature, or obtained by applying the method explained in Section 5 for instances with symmetric TFN values. A dash represents that no previous LB is available. Column $E[\widehat{LB}]_{new}$ contains the best lower bound found in this work, together with the method(s) that reached that value. The best lower bound for each instance is highlighted in bold. Regarding the upper bounds, column $Best\widehat{UB}$ reports the best found upper bound together with its expected value and the method(s) that obtained it. If the UB is proven to be optimal, it is highlighted in bold. For the sake of simplicity, we use letters M, C andH to refer to MILP, CP and HTS respectively.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cie.2023.109454.

References

Abdullah, S., & Abdolrazzagh-Nezhad, M. (2014). Fuzzy job-shop scheduling problems: A review. *Information Sciences*, *278*, 380–407. http://dx.doi.org/10.1016/j.ins.2014.03.060.

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Managament Science*, *34*, 391–401. http://dx.doi.org/10.1287/mnsc.34.3.391.

Afsar, S., Palacios, J. J., Puente, J., Vela, C. R., & González-Rodríguez, I. (2022). Multi-objective enhanced memetic algorithm for green job shop scheduling with uncertain times. *Swarm and Evolutionary Computation*, [ISSN: 2210-6502] *68*, Article 101016. http://dx.doi.org/10.1016/j.swevo.2021.101016.

Applegate, D., & Cook, W. (1991). A computational study of the job-shop scheduling problem. *ORSA Journal of Computing*, *3*, 149–156. http://dx.doi.org/10.1287/ijoc.3.2.149.

Basán, N., Cóccola, M. E., García del Valle, A., & Méndez, C. A. (2019). An effective MILP-based decomposition algorithm for the scheduling and redesign of flexible job-shop plants. *Chemical Engineering Transactions*, *74*, 613–618. http://dx.doi.org/10.3303/CET1974103.

Behnamian, J. (2016). Survey on fuzzy shop scheduling. *Fuzzy Optimization and Decision Making*, *15*, 331–366. http://dx.doi.org/10.1007/s10700-015-9225-5.

de Fátima Morais, M., dal Molin Ribeiro, M., Gomes da Silva, R., Cocco Mariani, V., & dos Santos Coelho, L. (2022). Discrete differential evolution metaheuristics for permutation flow shop scheduling problems. *Computers & Industrial Engineering*, *166*, Article 107956. http://dx.doi.org/10.1016/j.cie.2022.107956.

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., & Cosar, A. (2019). A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, *137*, Article 106040. http://dx.doi.org/10.1016/j.cie.2019.106040.

Du, Y., Li, J., Li, C., & Duan, P. (2022). A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15. http://dx.doi.org/10.1109/TNNLS.2022.3208942, (in press).

Dubois, D., Fargier, H., & Fortemps, P. (2010). Scheduling under flexible constraints and uncertain data: the fuzzy approach. In *Production scheduling* (pp. 301–332). Wiley, http://dx.doi.org/10.1002/9780470611050.ch11, chapter 11.

Dubois, D., Foulloy, L., Mauris, G., & Prade, H. (2004). Probability-possibility transformations, triangular fuzzy sets and probabilistic inequalities. *Reliable Computing*, *10*, 273–297.

Fisher, H., & Thomson, G. L. (1963). Probabilistic learning combinations of local job-shop scheduling rules. In J. F. Muth, & G. L. Thomson (Eds.), *Industrial scheduling* (pp. 225–251). Prentice Hall.

Fortemps, P. (1997). Jobshop scheduling with imprecise durations: a fuzzy approach. *IEEE Transactions on Fuzzy Systems*, *7*, 557–569. http://dx.doi.org/10.1109/91.649907.

Gao, D., Wang, G.-G., & Pedrycz, W. (2020). Solving fuzzy job-shop scheduling problem using de algorithm improved by a selection mechanism. *IEEE Transactions on Fuzzy Systems*, *28*(12), 3265–3275. http://dx.doi.org/10.1109/TFUZZ.2020.3003506.

Gendreau, M., & Potvin, J.-Y. (Eds.), (2019). *International series in operations research & management science: volume 272*, *Handbook of metaheuristics* (3rd ed.). Springer, http://dx.doi.org/10.1007/978-3-319-91086-4, ISBN 978-3-319-91085-7, 978-3-319-91086-4.

Ghrayeb, O. A. (2003). A bi-criteria optimization: minimizing the integral value and spread of the fuzzy makespan of job shop scheduling problems. *Applied Soft Computing*, *2*(3), 197–210. http://dx.doi.org/10.1016/S1568-4946(02)00069-8.

González-Rodríguez, I., Puente, J., Palacios, J. J., & Vela, C. R. (2020). Multi-objective evolutionary algorithm for solving energy-aware fuzzy job shop problems. *Soft Computing*, *24*, 16291–16302. http://dx.doi.org/10.1007/s00500-020-04940-6.

González Rodríguez, I., Vela, C. R., Hernández-Arauzo, A., & Puente, J. (2009). Improved local search for job shop scheduling with uncertain durations. In *Proceedings of the nineteenth international conference on automated planning and scheduling (ICAPS-2009)* (pp. 154–161). Thesaloniki: AAAI Press, ISBN: 9781577354062.

González Rodríguez, I., Vela, C. R., Puente, J., & Varela, R. (2008). A new local search for the job shop problem with uncertain durations. In *Proceedings of the eighteenth international conference on automated planning and scheduling (ICAPS-2008)* (pp. 124–131). Sidney: AAAI Press, ISBN: 9781577353867.

Hapke, M., Jaszkiewicz, A., & Slowinski, R. (1994). Fuzzy project scheduling system for software development. *Fuzzy Sets and Systems*, *67*(1), 101–117. http://dx.doi.org/10.1016/0165-0114(94)90211-9.

Hazır, O., & Ulusoy, G. (2020). A classification and review of approaches and methods for modeling uncertainty in projects. *International Journal of Production Economics*, *223*, Article 107522. http://dx.doi.org/10.1016/j.ijpe.2019.107522.

IBM (2020). IBM IBM CPLEX optimizer. URL https://www.ibm.com/analytics/cplex-optimizer.

Ku, W.-Y., & Beck, J. C. (2016). Mixed integer programming models for job shop scheduling: A computational analysis. *Computers & Operations Research*, *73*, 165–173. http://dx.doi.org/10.1016/j.cor.2016.04.006.

Laborie, P. (2018). An update on the comparison of MIP, CP and hybrid approaches for mixed resource allocation and scheduling. In W.-J. van Hoeve (Ed.), *Integration of constraint programming, artificial intelligence, and operations research* (pp. 403–411). Springer, http://dx.doi.org/10.1007/978-3-319-93031-2_29.

Laborie, P., Rogerie, J., Shaw, P., & Vilím, P. (2018). IBM ILOG CP optimizer for scheduling. *Constraints*, *23*(2), 210–250. http://dx.doi.org/10.1007/s10601-018-9281-x.

Lawrence, S. (1984). *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement)*: *Technical report*, Graduate School of Industrial Administration, Carnegie Mellon University.

Lei, D. (2010a). Fuzzy job shop scheduling problem with availability constraints. *Computers & Industrial Engineering*, *58*, 610–617.

Lei, D. (2010b). Solving fuzzy job shop scheduling problems using random key genetic algorithm. *International Journal of Advanced Manufacturing Technologies*, *49*, 253–262. http://dx.doi.org/10.1007/s00170-009-2379-y.

Li, J.-Q., Du, Y., Gao, K.-Z., Duan, P.-Y., Gong, D.-W., Pan, Q.-K., et al. (2022). A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem. *IEEE Transactions on Automation Science and Engineering*, *19*(3), 2153–2170. http://dx.doi.org/10.1109/TASE.2021.3062979.

Li, R., Gong, W., & Lu, C. (2022). Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time. *Computers & Industrial Engineering*, *168*, Article 108099. http://dx.doi.org/10.1016/j.cie.2022.108099.

Li, J.-Q., Liu, Z.-m., Li, C., & Zheng, Z.-x. (2021). Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem. *IEEE Transactions on Fuzzy Systems*, *29*(11), 3234–3248. http://dx.doi.org/10.1109/TFUZZ.2020.3016225.

Li, J.-Q., & Pan, Y.-X. (2013). A hybrid discrete particle swarm optimization algorithm for solving fuzzy job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, *66*, 583–596. http://dx.doi.org/10.1007/s00170-012-4337-3.

Lin, F.-T. (2002). Fuzzy job-shop scheduling based on ranking level $(\lambda, 1)$ interval-valued fuzzy numbers. *IEEE Transactions on Fuzzy Systems*, *10*(4), 510–522. http://dx.doi.org/10.1109/TFUZZ.2002.800659.

Lin, J., Zhu, L., & Wang, Z.-J. (2019). A hybrid multi-verse optimization for the fuzzy flexible job-shop scheduling problem. *Computers & Industrial Engineering*, *127*, 1089–1100. http://dx.doi.org/10.1016/j.cie.2018.11.046.

Lunardi, W. T., Birgin, E. G., Laborie, P., Ronconi, D. P., & Voos, H. (2020). Mixed integer linear programming and constraint programming models for the online printing shop scheduling problem. *Computers & Operations Research*, *123*, http://dx.doi.org/10.1016/j.cor.2020.105020.

Maravas, A., & Pantouvakis, J.-P. (2012). Project cash flow analysis in the presence of uncertainty in activity duration and cost. *International Journal of Project Management*, *30*(3), 374–384. http://dx.doi.org/10.1016/j.ijproman.2011.08.005.

Niu, Q., Jiao, B., & Gu, X. (2008). Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time. *Applied Mathematics and Computation*, *205*, 148–158. http://dx.doi.org/10.1016/j.amc.2008.05.086.

Novas, J. M. (2019). Production scheduling and lot streaming at flexible job-shops environments using constraint programming. *Computers & Industrial Engineering*, *136*, 252–264. http://dx.doi.org/10.1016/j.cie.2019.07.011.

Palacios, J. J., González-Rodríguez, I., Vela, C. R., & Puente, J. (2014). Robust swarm optimisation for fuzzy open shop scheduling. *Natural Computing*, *13*(2), 145–156.

Palacios, J. J., González-Rodríguez, I., Vela, C. R., & Puente, J. (2015). Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop. *Fuzzy Sets and Systems*, [ISSN: 0165-0114] *278*, 81–97. http://dx.doi.org/10.1016/j.fss.2014.12.003.

Palacios, J. J., González-Rodríguez, I., Vela, C. R., & Puente, J. (2017). Robust multiobjective optimisation for fuzzy job shop problems. *Applied Soft Computing*, *56*, 604–616. http://dx.doi.org/10.1016/j.asoc.2016.07.004.

Palacios, J. J., González-Rodríguez, I., Vela, C. R., & Puente, J. (2019). Satisfying flexible due dates in fuzzy job shop by means of hybrid evolutionary algorithms. *Integrated Computer-Aided Engineering*, *26*, 65–84. http://dx.doi.org/10.3233/ICA-180583.

Palacios, J. J., Puente, J., González-Rodríguez, I., & Vela, C. R. (2013). Hybrid tabu search for fuzzy job shop. In J. M. Ferrández Vicente, J. R. Álvarez Sánchez, F. Paz López, & F. Toledo Moreo (Eds.), *Lecture notes in computer science*: *Vol. 7930*, *Natural and artificial models in computation and biology* (pp. 376–385). Springer, http://dx.doi.org/10.1007/978-3-642-38637-4_39.

Palacios, J. J., Puente, J., Vela, C. R., & González-Rodríguez, I. (2016). Benchmarks for fuzzy job shop problems. *Information Sciences*, *329*, 736–752. http://dx.doi.org/10.1016/j.ins.2015.09.042.

Petrovic, S., Fayad, S., Petrovic, D., Burke, E., & Kendall, G. (2008). Fuzzy job shop scheduling with lot-sizing. *Annals of Operations Research*, *159*, 275–292. http://dx.doi.org/10.1007/s10479-007-0287-9.

Pinedo, M. L. (2016). *Theory, algorithms, and systems*, *Scheduling* (5th ed.). Springer, http://dx.doi.org/10.1007/978-1-4614-2361-4.

Prade, H. (1979). Using fuzzy set theory in a scheduling problem: a case study. *Fuzzy Sets and Systems*, *2*(2), 153–165. http://dx.doi.org/10.1016/0165-0114(79)90022-8.

Rommelfanger, H. (1990). FULPAL — an interactive method for solving (multiobjective) fuzzy linear programming problems. In R. Slowinski, & J. Teghem (Eds.), *Stochastic versus fuzzy approaches to multiobjective mathematical programming under uncertainty* (pp. 279–299). Springer, http://dx.doi.org/10.1007/978-94-009-2111-5_14.

Sakawa, M., & Kubota, R. (2000). Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms. *European Journal of Operational Research*, *120*, 393–407. http://dx.doi.org/10.1016/S0377-2217(99)00094-6.

Sakawa, M., & Mori, T. (1999). An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate. *Computers & Industrial Engineering*, *36*, 325–341. http://dx.doi.org/10.1016/S0360-8352(99)00135-7.

Shukor, S. A., Shaheed, I. M., & Abdullah, S. (2018). Population initialisation methods for fuzzy job-shop scheduling problems: Issues and future trends. *International Journal on Advanced Science Engineering Information Technology*, *4*(2), 1820–1828. http://dx.doi.org/10.18517/ijaseit.8.4-2.6808.

Song, X., Zhu, Y., Yin, C., & Fuming, L. (2006). A hybrid strategy based on ant colony and taboo search algorithms for fuzzy job shop scheduling. In *Proceedings of the 8th world congress on intelligent control and automation* (pp. 7362–7365). http://dx.doi.org/10.1109/WCICA.2006.1714516.

Strassl, S., & Musliu, N. (2022). Instance space analysis and algorithm selection for the job shop scheduling problem. *Computers & Operations Research*, *141*, Article 105661. http://dx.doi.org/10.1016/j.cor.2021.105661.

Streeter, M., & Smith, S. (2006). How the landscape of random job shop scheduling instances depends on the ratio of jobs to machines. *Journal of Artificial Intelligence Research*, *26*, 247–287. http://dx.doi.org/10.1613/jair.2013.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*, 278–285. http://dx.doi.org/10.1016/0377-2217(93)90182-M.

Tsujimura, Y., Gen, M., & Kubota, E. (1995). Solving job-shop scheduling problem with fuzzy processing time using genetic algorithm. *Journal of Japan Society for Fuzzy Theory and Systems*, *7*, 1073–1083. http://dx.doi.org/10.3156/jfuzzy.7.5_1073.

van Hoorn, J. J. (2018). The current state of bounds on benchmark instances of the job-shop scheduling problem. *Journal of Scheduling*, *21*(1), 127–128. http://dx.doi.org/10.1007/s10951-017-0547-8.

Vela, C. R., Afsar, S., Palacios, J. J., González-Rodríguez, I., & Puente, J. (2020). Evolutionary tabu search for flexible due-date satisfaction in fuzzy job shop scheduling. *Computers & Operations Research*, *119*, Article 104931. http://dx.doi.org/10.1016/j.cor.2020.104931.

Verderame, P. M., Elia, J. A., Li, J., & Floudas, C. A. (2010). Planning and scheduling under uncertainty: A review across multiple sectors. *Industrial and Engineering Chemistry Research*, *49*(9), 3993–4017. http://dx.doi.org/10.1021/ie902009k.

Vilím, P., Barták, R., & Čepek, O. (2004). Unary resource constraint with optional activities. In *International conference on principles and practice of constraint programming* (pp. 62–76). Springer, http://dx.doi.org/10.1007/978-3-540-30201-8_8.

Wang, G.-G., Gao, D., & Pedrycz, W. (2022). Solving multiobjective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm. *IEEE Transactions on Industrial Informatics*, *18*(12), 8519–8528. http://dx.doi.org/10.1109/TII.2022.3165636.

Wang, C., Tian, N., Ji, Z., & Wang, Y. (2017). Multi-objective fuzzy flexible job shop scheduling using memetic algorithm. *Journal of Statistical Computation and Simulation*, *87*(14), 2828–2846. http://dx.doi.org/10.1080/00949655.2017.1344846.

Xie, J., Gao, L., Peng, K., Li, X., & Li, H. (2019). Review on flexible job shop scheduling. *IET Collaborative Intelligent Manufacturing*, *1*(3), 67–77. http://dx.doi.org/10.1049/iet-cim.2018.0009.

Xiong, H., Shi, S., Ren, D., & Hu, J. (2022). A survey of job shop scheduling problem: The types and models. *Computers & Operations Research*, *142*, Article 105731. http://dx.doi.org/10.1016/j.cor.2022.105731.

Zhao, J., Peng, S., Li, T., Lv, S., Li, M., & Zhang, H. (2019). Energy-aware fuzzy job-shop scheduling for engine remanufacturing at the multi-machine level. *Frontiers of Mechanical Engineering*, *14*, 474–488. http://dx.doi.org/10.1007/s11465-019-0560-z.

Zheng, Y., Li, Y., & Lei, D. (2011). Swarm-based neighbourhood search for fuzzy job shop scheduling. *International Journal of Innovative Computing and Applications*, *3*(3), 144–151. http://dx.doi.org/10.1504/IJICA.2011.041915.