Benchmarks for Fuzzy Job Shop Problems

Juan José Palacios[1], Jorge Puente[1], Camino R. Vela[1], Inés González-Rodríguez[b,*]

[a]Department of Computing, University of Oviedo, Campus de Gijón, 33204, Gijón, (Spain)
[b]Dept. of Mathematics, Statistics and Computing, University of Cantabria, Av. Los Castros s/n, 39011, Santander (Spain), Tel: +34 942202201

Abstract

The fuzzy job shop scheduling problem with makespan minimisation is a problem with a significant presence in the scientific literature. However, a common meaningful comparison base is missing for such problem. This work intends to fill the gap in this domain by reviewing existing benchmarks as well as proposing new benchmark problems. First, we shall survey the existing test beds for the fuzzy job shop, analysing whether they are sufficiently varied and, most importantly, whether there is room for improvement on these instances — an essential requirement if the instances are to be useful for the scientific community in order to compare and develop new solving strategies. In the light of this analysis, we shall propose a new family of more challenging benchmark problems and provide lower bounds for the expected makespan of each instance as well as reference makespan values obtained with a memetic algorithm from the literature. The resulting benchmark will be made available so as to facilitate experiment reproducibility and encourage research competition.

Keywords: Fuzzy sets, Scheduling, Job Shop, Benchmark, Metaheuristics

1. Introduction

Scheduling is with no doubt a research field of great importance, involving complex combinatorial constraint-satisfaction and optimisation problems and with relevant applications in industry, finance, welfare, education, etc [53, 61]. In particular, the job shop problem in its numerous variants is a model for many real problems which has posed and still poses a challenge to the research community, due to its complexity. As is the case with many hard optimisation problems, it is usual to resort to metaheuristic techniques that find approximate

*Corresponding author
Email addresses: palaciosjuan@uniovi.es (Juan José Palacios), puente@uniovi.es (Jorge Puente), crvela@uniovi.es (Camino R. Vela), gonzalezri@unican.es (Inés González-Rodríguez)

good solutions [7, 59]. Traditionally, such techniques are empirically evaluated on several benchmarks of common use which are available to the researchers, such as the instances from [2, 3, 17, 34] or [58]. Results and comparisons on these benchmark instances allow to validate the quality of different proposals and to advance in the quest for better solving methods.

To enhance the applicability of scheduling, part of the research is devoted to modelling and handling the uncertainty pervading real-world situations [28]. Probability theory is the most extended approach to scheduling with uncertainty. However, it may prove difficult to use in practice due to the quantity and quality of information needed to elicit probability distributions as well as to the computational complexity of working with these distributions. An alternative and increasingly popular approach is to use fuzzy sets in the setting of possibility theory [16, 19, 66], since they provide an acceptable trade-off between expressivity and computational difficulty. There are in fact numerous research papers where uncertain durations are represented as fuzzy numbers, mostly triangular fuzzy numbers, among others, for single machine scheduling [10],[32], parallel machine scheduling [4, 49], flow shop scheduling [9, 31], open shop scheduling [46], job shop scheduling [18, 24, 35, 45, 52, 54, 55, 60, 71] and [41], and for flexible job shop scheduling in [40, 48, 64, 65] to mention but a few. Indeed, the recent review of metaheuristic algorithms to solve fuzzy job shop problems in [1] highlights the relevance of this topic. Additionally, fuzzy numbers and, in general, fuzzy intervals, can be linked with traditional intervals [20], which constitute an alternative approach to modelling uncertain durations [21].

Unlike the deterministic case, no common test-bed is available for the fuzzy job shop that allows for fair and meaningful comparisons and assessment of different proposals. Moreover, only a portion of the instances used in the literature for experimental results are available to the research community. This is an important issue since, as Beasley puts it in his introduction of the OR-Library "If a standard set of test problems is available, then algorithms can be compared on a more realistic basis (with regard to their performance on *exactly* the same set of test problems) than would otherwise be so"[5]. This paper attempts to contribute to filling this gap. To do that, we start by reviewing and studying the level of difficulty of the available instances with regard to the most widely used objective function, the makespan. This will allow us to identify those that are easy and already solvable with the existing methods, making them unsuitable for the development of new more powerful metaheuristics. We shall then argue the necessity of providing a family of more challenging instances. To this end, we shall fuzzify a well-known benchmark and obtain preliminary results and lower bounds. The contribution of this paper is thus to identify a benchmark with the most challenging instances and, consequently, provide a solid basis for future research on the fuzzy job shop scheduling problem.

In the sequel, after introducing the necessary background on job shop and fuzzy durations in Section 2, Section 3 includes a review of the existing papers on fuzzy job shop and the instances used therein to obtain experimental results. Section 4 is devoted to introduce a common framework that will be used in

Section 5 to analyse and evaluate the difficulty of the available instances. Then, in Section 6 we shall propose a new test-bed and provide first results for future reference. Finally, in Section 7 we summarise the main conclusions.

2. Background on the Fuzzy Job Shop Scheduling Problem

The *job shop scheduling problem*, or *JSP* in short, consists in scheduling a set of n jobs $J_1, \ldots, J_n$ to be processed on a set of m physical resources or machines $M_1, \ldots, M_m$ subject to a set of constraints. There are *precedence constraints*, so each job J_i, $i = 1, \ldots, n$, consists of m tasks $\{\theta_{i1}, \ldots, \theta_{im}\}$ to be sequentially scheduled. There are also *capacity constraints*, whereby each task θ_{ij} requires the exclusive use of a different machine for its whole processing time without preemption, i.e. tasks must be processed without interruption. A solution to this problem is a *schedule*–an allocation of starting times for all tasks – which is *feasible*, in the sense that all constraints hold, and is also optimal according to some criterion. Here, we shall consider the objective of minimising the makespan C_{max}, that is, the time lag from the start of the first operation until the end of the last one, as it has been the objective function most used by researchers.

An extension of the JSP is the *fuzzy job shop problem* or *FJSP*, where task durations are taken to be triangular fuzzy numbers, as explained below.

2.1. Uncertain processing times

In real-life applications, it is often the case that the exact time it takes to process a task is not known in advance. Consider for instance subcontracted activities in manufacturing environment, debugging tasks in software engineering or activities performed by more or less skilled workers. However, based on previous experience, an expert may have some knowledge (albeit uncertain) about the duration. The crudest representation for uncertain processing times would be a human-originated confidence interval. If some values appear to be more plausible than others, a natural extension is a fuzzy interval or fuzzy number (cf. [13, 15]).

A fuzzy quantity Q is a fuzzy set on the reals $\mathbb{R}$ with membership function $\mu_Q : \mathbb{R} \to [0, 1]$. The *$\alpha$-cuts* of a fuzzy quantity are given by $Q_\alpha = \{r \in \mathbb{R} : \mu_Q(r) \geq \alpha\}, \alpha \in (0, 1]$, and its *support* is defined as $Q_0 = \{r \in \mathbb{R} : \mu_Q(r) > 0\}$. A *fuzzy interval* is a fuzzy quantity whose α-cuts are intervals (bounded or not) and a *fuzzy number* M is a fuzzy quantity with compact support and unique modal value whose α-cuts are closed intervals, denoted $M_\alpha = [\underline{m}_\alpha, \overline{m}_\alpha]$.

The simplest model is a *triangular fuzzy number* or *TFN*, using an interval $[a^1, a^3]$ of possible values and a modal value a^2 in it, so a TFN A, denoted $A = (a^1, a^2, a^3)$, has a membership function given by:

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

As we shall see, TFNs are widely used in the literature on fuzzy scheduling.

2.1.1. Ordering of TFNs

There is no natural relation of total order in the set of TFNs. Hence, in order to compare different TFNs, several ranking methods have been proposed in the literature [6, 8].

The membership function μ_N of a fuzzy number N can be seen as a possibility distribution on the real numbers; this allows to define the *expected value* of a fuzzy quantity [27], given for a TFN A by

$$E[A] = \frac{1}{4}(a^1 + 2a^2 + a^3). \tag{2}$$

It induces a total ordering $\leq_E$ on the set of fuzzy intervals [18], where for any two fuzzy intervals M, N $M \leq_E N$ if and only if $E[M] \leq E[N]$. The expected value coincides with the *neutral scalar substitute* of a fuzzy interval and can also be obtained as the centre of gravity of its *mean value* or using the *area compensation* method [15]. Additionally, $\leq_E$ coincides with several other ranking methods from the literature as highlighted in [47]. It is also possible to establish a relation with classical interval comparison in the light of imprecise probabilities, with $\leq_E$ coming down to using Hurwicz criterion on upper and lower expectations derived from the fuzzy number [12]. This provides us with an interpretation for comparisons based on $\leq_E$ as those corresponding to a decision maker who keeps an equilibrium between pessimism and optimism.

Related to this is a ranking method widely used in the fuzzy scheduling literature following the seminal papers of Sakawa et al. [55, 56]. For any TFN N, we define three defuzzification indices: $c_1(N) = E[N]$, $c_2(N) = n^2$ and $c_3(N) = n^3 - n^1$, so $N <_R M$ if $c_1(N) < c_1(M)$ or else if $c_1(N) = c_1(M)$ and $c_2(N) < c_2(M)$ or else if $c_1(M) = c_1(N)$, $c_2(N) = c_2(M)$ and $c_3(N) < c_3(M)$.

2.1.2. Arithmetic

For the job shop, we essentially need two operations on fuzzy numbers, the sum and the maximum. These are obtained by extending the corresponding operations on real numbers using the *Extension Principle* [14]: for any two fuzzy numbers M and N and any bivariate function f on the reals,

$$\forall r \in \mathbb{R}, \mu_{f(M,N)}(r) = \sup\{\min(\mu_M(r_1), \mu_N(r_2)) : f(r_1, r_2) = r\} \tag{3}$$

if $f^{-1}(r) \neq \emptyset$, being equal to 0 otherwise.

According to this, the sum of two TFNs M, N is another TFN given by the following equation [44]:

$$M + N = (m^1 + n^1, m^2 + n^2, m^3 + n^3) \tag{4}$$

Unfortunately, computing the maximum of two TFNs is not that simple and can result cumbersome. Also the result, although guaranteed to be a fuzzy number, may not be a TFN. In practice, for the sake of simplicity and tractability of numerical calculations, it is usual to approximate the maximum by a TFN which is relatively easy to compute. Two methods can be found in the literature on fuzzy scheduling.

The most common approach is to approximate the maximum by the TFN that results from evaluating this operation on the three defining points of each TFN, that is, for every M, N TFNs:

$$\max(M,N) \approx \max_I(M,N) = (\max(m^1,n^1), \max(m^2,n^2), \max(m^3,n^3)) \quad (5)$$

This approximation has been widely used in the scheduling literature, among others, in [9, 11, 18, 24, 26, 33, 35, 45, 51, 55], or [63].

Some arguments can be given to support this approximation. First, for any two fuzzy numbers M and N, if f is a bivariate continuous isotonic function, then $F = f(M,N)$ is another fuzzy number such that

$$\forall \alpha \in [0,1], F_\alpha = [f(\underline{m}_\alpha, \underline{n}_\alpha), f(\overline{m}_\alpha, \overline{n}_\alpha)]. \quad (6)$$

Computing $f(M,N)$ is then equivalent to computing f on every α-cut. In particular, the maximum is a continuous isotonic function, so it can be calculated by evaluating two maxima of real numbers for every value $\alpha \in [0,1]$. It seems then natural to approximate the maximum by the TFN that results from using linear interpolation, evaluating equation (6) only for certain values of α (this is proposed for 6-point fuzzy numbers in [18]). Given that the defining values (m^1, m^2, m^3) of a TFN N are such that $M_0 = [m^1, m^3]$ and $M_1 = [m^2, m^2]$, the approximated maximum as in (5) corresponds to such an interpolation for $\alpha = 0$ and $\alpha = 1$. Secondly, if $F = \max(M,N)$ denotes the maximum of two TFNs M and N and $G = \max_I(M,N)$ the approximated value by interpolation, then $F = G$ if M and N do not overlap and, in any case, it holds that

$$\forall \alpha \in [0,1], \quad \underline{f}_\alpha \leq \underline{g}_\alpha, \overline{f}_\alpha \leq \overline{g}_\alpha. \quad (7)$$

The approximated maximum G is thus a TFN which artificially increases the value of the actual maximum F, but maintaining the support and modal value, that is, $F_0 = G_0$ and $F_1 = G_1$. This approximation can be trivially extended to the case of more than two TFNs.

More recently, it has been proposed in [37] to approximate the maximum of two TFNs M and N by means of the above ranking method, so $\max(M,N) \approx \max_R(M,N)$ where $\max_R = M$ if $N <_R M$ and $\max_R(M,N) = N$ otherwise. It is not guaranteed that the approximated maximum maintains the support nor the modal value. However, in [37] and [38] some examples are considered which lead the author to conclude that "the approximate max obtained by the new criterion approaches the real max better than that obtained from " $\max_I$. Since then, this alternative approximation for the maximum has been adopted among others in [40, 70, 71] and [64].

It is important to notice that, for any two TFNs N_1 and N_2, if $M_I = \max_I(N_1,N_2)$ is the maximum approximated by interpolation and $M_R = \max_R(N_1,N_2)$ is the maximum approximated by the ranking method, it is always the case that $m_R^i \leq m_I^i$ for $i = 1,2,3$ and, hence, $M_R \leq_E M_I$ and $M_R \leq_R M_I$.

3. Review of Existing Benchmarks

In this section we review the existing test-beds for the fuzzy job shop scheduling problem, including information on how the instances have been generated and whether they are openly available to the research community. A first review of benchmark datasets for fuzzy job shop is already included in [1]; here we elaborate on the information given in that survey and complement it with data that allows for fair and rigorous comparisons of different proposals from the literature. We shall analyse the results reported for those instances, taking into account both the objective function considered in each case as well as the fuzzy arithmetic used (maximum approximation and order relation), this being a key issue for meaningful comparisons. The size of each instance will be given using the notation $n \times m$, meaning that it consists of n jobs and m machines.

The instances used so far in the literature can be divided in two big groups, depending on whether they have been generated from scratch (called "original instances" hereafter) or by fuzzifying well-known benchmark instances from crisp job shop.

3.1. Original Fuzzy Instances

In this group of test beds we find what is perhaps the most widely used set of instances for fuzzy job shop, those proposed by Sakawa et al. in [56] and [55]. The first paper provides the data for an instance of size 6×6 and an instance of size 10×10, while the latter provides three more instances for each size. In both cases, the original instances also include data regarding job due dates since the objective functions are at least partially concerned with due-date satisfaction. In the following, we shall denote the 6×6 and 10×10 instances from [55] as S6.1,S6.2,S6.3 and S10.1,S10.2,S10.3 respectively, and the instances from [56] as S6.4 and S10.4.

This set of 8 problems was originally proposed to test genetic algorithms and has been later used by several authors to evaluate different metaheuristics designed for a great variety of objective functions: genetic algorithms with different codifications in [23, 24], and [36]; particle swarm optimisation [35]; memetic algorithm (MA1) combining a GA with local search [25]; simulated annealing [63]; differential evolution [29]; and hybrid discrete particle swarm optimisation (HDPSO) in [41]. Results for makespan minimisation can be found in [23], where the authors propose a GA using $<_E$ and $\max_I$ to obtain results on the instances from [55], and in [50] where the authors propose a shifting bottleneck hybridised with a GA using $<_R$ and $\max_I$. More detailed results for all 8 instances are obtained in [36] with a random-key based GA, denoted RKGA using $<_R$ and $\max_R$. Finally, in [41] the HDPSO method is compared with RKGA and a GA from [55] using $<_R$ and $\max_R$.

Both in [36] and [41] the authors argue the need of larger instances in order to verify the ability of their proposals to improve the state-of-the-art. In [36] the author proposes two instances of size 15×10 while in [41] an instance of size 16×16 is proposed, in both cases accompanied by the corresponding

results. In the following, we will refer to these instances as Lei01,Lei02 and LP01 respectively.

3.2. Fuzzified Instances

A second group of instances is obtained by fuzzifying task durations of well-known benchmarks for crisp JSP. Specifically, the instances considered herein have been taken from the following test beds: FT from [17], La from [34], ABZ from [2], and ORB from [3]. To our knowledge, this approach was first adopted by Fortemps in [18] to generate fuzzy versions of the well-known FT06 and La11–14. Although the fuzzification method was originally intended to generate symmetric 6-point fuzzy numbers, it can be easily adapted to generate symmetric TFNs. This is actually done in [23], where durations for FT06 and La11–14 but also for FT10 and FT20, La24 and ABZ7 are fuzzified as TFNs. The authors use the proposed problems in [23] and also in [25] in order to compare a GA proposed therein with the SA from [18] in terms of $E[C_{max}]$ and to evaluate new memetic approaches. The same authors argue the need of harder instances in a later work [26], so using the same fuzzification method, they extend the test bed by adding 8 new fuzzy instances of famous crisp benchmark problems: La21, La25, La27, La29, La38 and La40, and ABZ8, ABZ9. Results for these fuzzified instances are obtained in [26] and [54] using $\max_I$ and $<_E$. In the sequel, we shall denote these instances with the original name subscripted with F, in reference to the author of the fuzzification method.

In the paper of Tsujimura et al. [62] two more instances of size 6×6 and 20×5 respectively, built from FT06 and FT20, are explicitly given in the paper, although the fuzzification method used is not made explicit. So far, the instances have been only used to evaluate the GA proposed in this paper (that we shall call TGA) for makespan minimisation using $<_R$ and $\max_R$ as well as an alternative ranking method and associated maximum. Only graphical results are provided, without accompanying numerical results. We shall denote these instances as $\mathsf{FT}_\mathsf{T}06$ and $\mathsf{FT}_\mathsf{T}20$.

A new fuzzification method is proposed by Ghrayeb [22] to transform durations into non-symmetric TFNs, and FT06, La12, La13 and La14 are fuzzyfied accordingly (the resulting problem instances will be denoted with the original name followed by the subscript G) in order to test a bi-criteria genetic algorithm. Unfortunately, only $\mathsf{FT}_\mathsf{G}06$ is made available. For all four instances the paper reports the best makespan obtained with GA (denoted GGA hereafter) using $\max_I$ and $<_E$.

The same fuzzification method is used in [45] for FT06, FT10, FT20 as well as La01, La03, La05, La07, La09 and ABZ5, ABZ6. The author reports results on all instances for a PSO algorithm combined with genetic operators (GPSO) using $\max_I$ together with an alternative ranking method; only the ranking index of the resulting makespan is given, making comparisons unsuitable on these instances.

A similar fuzzification method is used by Lin in [42] to build ten fuzzy instances from each of the crisp instances FT06, La01, La06. These fuzzy instances (denoted hereafter with the original name followed by the subscript L) are used to illustrate the difference between two proposals made in the paper (namely,

using normal vs. non-normal TFNs to model uncertainty) and then solved using a classical genetic algorithm from the literature. The solutions reported by the author correspond to a different arithmetic for TFNs (based on a ranking using the so-called signed distance) and makespan values are already defuzzified, thus making comparisons impossible.

A different method for fuzzifying task durations as non-symmetric TFNs is proposed by Song et al. in [57] to generate fuzzy instances of FT10, La02, La19, La21, La24, La25, La27, La29 and La36–40. The authors use the resulting instances to evaluate an ant colony algorithm hybridised with tabu search (TSANT) to optimise due-date satisfaction; no information is reported about the arithmetic for TFNs used and only the average agreement index, a measure related to due-date satisfaction, is reported, making comparisons with their method unsuitable on these instances. The resulting problem instances will be denoted with the original name followed by the subscript S.

Finally, in the work of Zheng et al. [70], a third fuzzification method for generating non-symmetric TFNs is proposed and used to fuzzify instances ORB1–5 as well as La20–22 and ABZ5,ABZ6. The resulting instances are denoted with the original name followed by the subscript Z hereafter. The information given in [70] about the experimental results is quite exhaustive: it reports the results obtained with the swarm based neighbourhood search (SNS) method proposed in that paper, but also provides results on the same instances for the RKGA and their own implementation of the GPSO, all of them using $\max_R$ and $<_R$. Similar results on the same instances for GPSO, RKGA and a new ant bee colony (ABC) algorithm are given in [71]. Notice that these are the instances referred to as "Lei's" in [1], since the same original instances and fuzzification method is used by one of the authors in [39]; however, the results reported in this late paper correspond to a variant of FJSP considering the additional constraint of preventive maintenance, together with the corresponding additional data for the benchmark instances.

Table 1: Existing instances for FJSP

Instance	Size	Prop.	Best C_{max} Result		Avail.
S6.1	6×6	[55]	HDPSO [41]	$(\max_R, <_R)$	Y
			RKGA [36]	$(\max_R, <_R)$	
			MA1 [25]	$(\max_I, <_E)$	
S6.2–3	6×6	[55]	HDPSO [41]	$(\max_R, <_R)$	Y
			RKGA [36]	$(\max_R, <_R)$	
S6.4	6×6	[56]	HDPSO [41]	$(\max_R, <_R)$	Y
			RKGA [36]	$(\max_R, <_R)$	
S10.1	10×10	[55]	HDPSO [41]	$(\max_R, <_R)$	Y
			RKGA [36]	$(\max_R, <_R)$	
S10.2	10×10	[55]	HDPSO [41]	$(\max_R, <_R)$	Y

Continued on next page

Table 1 – continued from previous page

Instance	Size	Prop.	Best C_{max} Result		Avail.
S10.3	10×10	[55]	HDPSO [41]	$(\max_R, <_R)$	Y
			RKGA [36]	$(\max_R, <_R)$	
			MA1 [25]	$(\max_I, <_E)$	
S10.4	10×10	[56]	RKGA [36]	$(\max_R, <_R)$	Y
Lei01,02	15×10	[36]	RKGA [36]	$(\max_R, <_R)$	Y
LP01	16×16	[41]	HDPSO [41]	$(\max_R, <_R)$	Y
FT_T06	6×6	[62]	TGA [62]	$(\max_R, <_R)$	Y
FT_L06	6×6	[42]	–	$--$	N
FT_G06	6×6	[22]	GGA [22]	$(\max_I, <_E)$	Y
FT_F06	6×6	[23]	MA1 [25]	$(\max_I, <_E)$	Y
FT_G10	10×10	[45]	–		N
FT_F10	10×10	[23]	MA [54]	$(\max_I, <_E)$	Y
FT_S10	10×10	[57]	–		N
FT_T20	20×5	[62]	TGA [62]	$(\max_R, <_R)$	Y
FT_G20	20×5	[45]	–		N
FT_F20	20×5	[23]	MA [54]	$(\max_I, <_E)$	Y
La_L01	10×5	[42]	–	$--$	N
La_S02	10×5	[57]	–		N
$La_G01,03,05$	10×5	[45]	–		N
La_L06	15×5	[42]	–	$--$	N
$La_G07,09$	15×5	[45]	–		N
La_G12-14	20×5	[22]	GGA [22]	$(\max_I, <_E)$	N
La_F11-14	20×5	[23]	MA1 [25]	$(\max_I, <_E)$	Y
La_S19	10×10	[57]	–		N
La_Z20	10×10	[70]	ABC [71]	$(\max_R, <_R)$	N
			GPSO [70]	$(\max_R, <_R)$	
La_F21	15×10	[26]	MA [54]	$(\max_I, <_E)$	Y
La_Z21	15×10	[70]	SNS [70]	$(\max_R, <_R)$	N
La_Z22	15×10	[70]	ABC [71]	$(\max_R, <_R)$	N
La_F24	15×10	[23]	MA [54]	$(\max_I, <_E)$	Y
La_F25	15×10	[26]	MA [54]	$(\max_I, <_E)$	Y
$La_S21,24,25$	15×10	[57]	–		N
$La_F27,29$	20×10	[26]	MA [54]	$(\max_I, <_E)$	Y
$La_S27,29$	20×10	[57]	–		N
$La_F38,40$	15×15	[26]	MA [54]	$(\max_I, <_E)$	Y
La_S36–40	15×15	[57]	–		N
ORB_Z1–4	10×10	[70]	ABC [71]	$(\max_R, <_R)$	N
ORB_Z5	10×10	[70]	ABC [71]	$(\max_R, <_R)$	N
			SNS [70]	$(\max_R, <_R)$	
			GPSO [70]	$(\max_R, <_R)$	

Continued on next page

Table 1 – continued from previous page

Instance	Size	Prop.	Best C_{max} Result		Avail.
$\mathsf{ABZ_G5,6}$	10×10	[45]	–		N
$\mathsf{ABZ_Z5}$	10×10	[70]	ABC [71]	$(\max_R, <_R)$	N
$\mathsf{ABZ_Z6}$	10×10	[70]	SNS[70]	$(\max_R, <_R)$	N
			ABC [71]	$(\max_R, <_R)$	
$\mathsf{ABZ_F7}$	20×15	[23]	MA [54]	$(\max_I, <_E)$	Y
$\mathsf{ABZ_F8,9}$	20×15	[26]	MA [54]	$(\max_I, <_E)$	Y

A summary of the relevant FJSP instances can be found in Table 1. The three first columns contain the name of the instances, as described above, their size and the paper where they were first proposed. The fourth column (Best C_{max} Result) indicates (if they exist) the algorithms obtaining the best makespan results so far and the paper where these results are reported, together with the name of the method and the fuzzy arithmetic used (maximum approximation, ordering criterion). Notice that for some algorithms (e.g. RKGA, GPSO) different results are reported in different papers; when this is the case, we have opted for the most favourable results. Finally, the last column (Avail.) indicates whether the resulting instances are openly available to the research community.

Finally, not all instances reviewed in [1] are included in Table 1 and in the posterior analysis. This is either because the instances are not available (directly or via a fuzzification method) as those from [33] or [50], or because they do not correspond to the standard FJSP as presented in Section 2, but to variants thereof. This is the case with the instances from [60], [68], [65] or [67]. We have opted for excluding these variants of FJSP from our study, as doing otherwise would make it far too lengthy.

4. Framework for Further Analysis

We now introduce a common framework that will be used in the following sections to analyse and evaluate the difficulty of the available instances. The analysis will be based on what seems to be the state-of-the-art algorithms for makespan minimisation: RKGA, SNS, HDPSO, ABC, GPSO, GGA and MA. Remember that only graphical results are provided for TGA, making it unsuitable for comparisons. For the latter, notice that MA using $(\max_I, <_E)$ has been shown (cf. [54],[26]) to outperform MA1. Furthermore, MA1 itself improved a permutation-based GA from [23] which compared favourably with a GA similar to that from [55].

A summary of the most relevant methods can be seen in Table 2. It contains all methods which have obtained best-known results for some of the instances in Table 1; most of these methods will be considered in the following sections. For each row, the first column contains the acronym of the method, the second column, its full name and the third and fourth column report respectively the

Table 2: Acronyms of most relevant methods for FJSP

Acronym	Method	Proposed in	Best result
ABC	Ant Bee Colony	[71]	[71]
GGA	Ghrayeb's Genetic Algorithm	[22]	[22]
GPSO	Genetic-operator Particle Swarm Optimisation	[45]	[70],[71]
HDPSO	Hybrid Discrete Particle Swarm Optimisation	[41]	[41]
MA	Memetic Algorithm	[54]	[54]
RKGA	Random-Key Genetic Algorithm	[36]	[36],[71]
SNS	Swarm-based Neighbourhood Search	[70]	[70]
TGA	Tsujimura's Genetic Algorithm	[62]	[62]

references to the papers where the method was first proposed and where best results have been reported.

For the sake of meaningful comparisons, we need to adopt a single operational approach to fuzzy durations. We have opted for using $\max_R$ and $<_R$, given that all but two of the state-of-the-art algorithms report results based on this arithmetic. For all these algorithms, we shall use the data available in the literature (notice that not all instances have been solved with every method). Additionally, we will provide new results on every instance using $\max_R$ and $<_R$ obtained with the memetic algorithm MA and a new proposed GRASP algorithm.

4.1. Parameters for GRASP and MA

In addition to the state-of-the-art methods, we will also use a simple greedy randomised adaptive search procedure (GRASP) based on the neighbourhood structure described in [54]. The reason is to avoid possible objections to the experimental analysis based on the relative complexity of the MA— combining a genetic algorithm with local search— compared to lighter metaheuristics such as genetic algorithms. This GRASP algorithm will help analyse the limitations of the existing benchmarks in Section 5.

The parameter values for the MA are chosen so as to evaluate approximately the same number of individuals as the remaining state-of-the-art methods. In particular, in [70] and [71] we find results obtained with RKGA, GPSO, SNS and ABC using the following configuration: RKGA, population size 100 and 600/1000 generations; GPSO, population size 20 and maximum number of generations 600/1000; SNS, swarm size 100 and maximum number of generations 600; ABC, swarm size 100 and maximum number of cycles 500. We also find results for RKGA in [36] using population size 100 and 200/300 generations for instances of size 6×6 and 10×10 respectively. Preliminary experiments suggest that this stopping criterion for MA becomes excessive for most of the instances, since convergence is generally achieved earlier and the marginal improvement in

quality solution for longer runs does not justify the added computational effort, especially as the problem size increases. Indeed, for each instance we have run MA 10 times on a population with 100 individuals, recording the point when the gradient in the convergence curve becomes very small. The obtained results suggest that convergence is achieved with 125 generations for problems with 100 tasks or less, 160 generations for 150 to 256 tasks and 300 generations for 300 tasks. This will be the configuration used hereafter.

Regarding GRASP, it is run to generate $n_i \times n_g$ solutions for each instance, being n_i and n_g the number of individuals and generations respectively used by the MA on the same instance. We believe this is as close as it can get to having identical running conditions: although both MA and GRASP use a hill-climbing strategy which performs an a-priori unknown number of iterations, neighbour evaluations do not compute a full schedule from scratch, but instead use an optimised partial evaluation; it is also impossible to predict in GAs how many chromosomes will pass onto the next generation and will not be re-evaluated.

An alternative approach would be to use CPU times as a basis for comparison. However, this information is not always available for the state-of-the-art algorithms; additionally, there are many variables that can affect the outcome of computerized stochastic optimizers, from the CPU architecture, CPU speed (MHz), bus speed, amount of memory, etc to operating system, computer language and compiler version. It has also been known that speed "adjustments" as those from The Standard Performance Evaluation Corporation (SPEC, `http://www.spec.org`) do not take into account all involved factors. Keeping this in mind and for the sake of completeness, Table 3 contains a summary of the known running times for each method under consideration —when this information is available—, as well as for GRASP and MA on instances grouped by size, having ascertained that there is great similarity in running conditions for instances of the same size. It is worth to remember the different configurations used in the three referenced papers as it also influences the CPU times; notice, for example the big differences in CPU times used in 10×10 instances between RKGA in [36] and in [71]. We can observe that MA takes in average 78% of the time of GRASP. This difference is due to the fact that MA does not re-evaluate identical individuals in consecutive generations together with the lower cost of the local search in the last iterations, where individuals are closer to the final solution.

4.2. Non-available fuzzified instances

As it can be appreciated in Table 1, results for many algorithms in the literature have been obtained on fuzzified instances that are not openly available. When this is the case, in order to decide on their potential as future benchmarks, we shall generate new instances from the original crisp ones following the corresponding fuzzification method, so GRASP and MA are run on these new instances. In consequence, we cannot guarantee that all the methods considered in this study obtain results on exactly the same instance, we can only guarantee that the instances have been generated from the same original crisp instance following the same method.

Table 3: CPU times on fuzzy JSP instances grouped by sizes

Size	CPU (sec)						
	RKGA[1]	RKGA[2]	GPSO[2]	SNS[3]	ABC[2]	GRASP[4]	MA[4]
6×6	–	–	–	–	–	0.65	0.42
10×10	7.56	15.30	16.55	8.64	11.98	2.43	1.78
20×5	–	–	–	–	–	2.82	2.37
15×10	–	30.65	31.90	15.95	20.85	5.73	4.35
20×10	–	–	–	–	–	8.70	7.17
15×15	–	–	–	–	–	9.52	7.47
16×16	–	–	–	–	–	14.10	9.03
20×15	–	–	–	–	–	29.12	23.59

[1]: Results given in [36] using Visual C++ 7.0 on a Pentium IV 2.0GHz PC
[2]: Results given in [71] using Visual C++ 6.0 on a PC 2GB RAM 2.5GHz CPU
[3]: Results given in [70] using Visual C++ 6.0 on a PC 2GB RAM 2.5GHz CPU
[4]: GRASP and MA are implemented in C++ on a Xeon 2.2GHz

As detailed in Section 3, there are essentially four fuzzification methods: the one proposed by Fortemps [18], adapted to TFNs in [23], the one proposed by Ghrayeb [22], the one proposed by Zheng [70] and the one proposed in Song [57]. For all four methods, each fuzzy duration is generated so the most likely value of the TFN is the original crisp duration, while the lower and upper bounds of the support are taken as random values in some interval. Differences reside on how these bounds are generated; the definition of the interval from which they are taken is different in each case, the values can be integer [18, 70] or real [22] and the resulting TFNs may be symmetric [23] or not [22, 57, 70]. We refer the interested reader to the original references for further detail.

It should be noticed nonetheless that the random nature of the fuzzification methods results in differences in the fuzzy durations and, hence, in oscillations in the makespan values obtained for the fuzzified instances. This must be kept in mind when presenting new results on these instances and making comparisons between different algorithms.

4.3. Lower Bounds for Fuzzy Instances

It is common in the literature to assess the performances of a method or compare several methods not in terms of the makespan itself, which constitutes an absolute performance measure, but in terms of relative measures such as the relative error w.r.t. a lower bound LB, defined in our case as:

$$RE = \frac{\mid E[C_{max}] - LB \mid}{LB} \tag{8}$$

However, in order to use such relative quality measures some lower bound must be available and, furthermore, such lower bound should be as tight as possible.

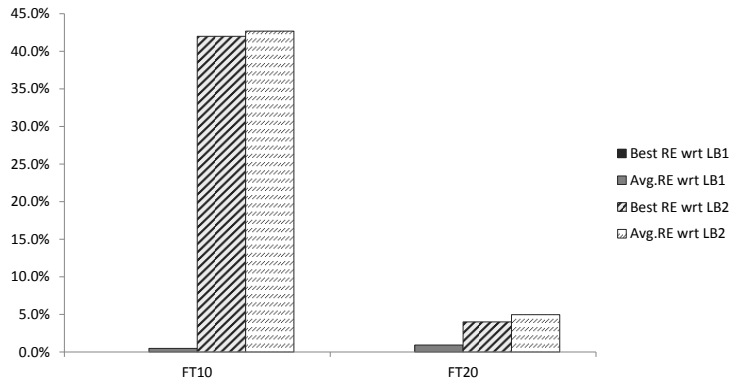

Figure 1: Comparison of relative errors w.r.t different Lower Bounds

In the case of fuzzy instances, it can be proved that a lower bound for the expected makespan of the fuzzy instance is given by an optimal solution (or any lower bound) of the *associated expected crisp problem*, that is, the problem where durations are the expected value of the corresponding fuzzy ones. This holds both when $(\leq_R, \max_R)$ or $(\leq_E, \max_I)$ is used. Let us denote this lower bound as LB_1. In the case of symmetric instances, the associated expected crisp problem is actually the original JSP instance, so the optimal solution of the crisp instance (or any lower bound) provides a lower bound for the expected makespan of the fuzzy solution, as already stated in [18]. In the case of non-symmetric instances, we propose to obtain the lower bound using the IBM ILOG CPLEX CP Optimizer software [30] to find the optimal solution (when possible) of the associated expected crisp problem.

A simpler way to compute lower bounds, proposed in [58], can be easily generalised to provide a lower bound, denoted LB_2, for the expected makespan of any FJSP instance. However, LB_2 strongly depends on the correlation between the number of jobs and the number of machines, with tighter lower bound values for problems where the number of jobs and machines differ and worse lower bound values for square problems. Consider for instance the lower bounds obtained for the symmetric fuzzy versions of $\mathsf{FT10}$ and $\mathsf{FT20}$: $LB_1 = 930$ vs. $LB_2 = 655$ for $\mathsf{FT_F10}$ and $LB_1 = 1165$ compared to $LB_2 = 1120$ for $\mathsf{FT_F20}$

The importance of the tightness of the lower bound is illustrated in Figure 1. For the same instances, $\mathsf{FT_F10}$ and $\mathsf{FT_F20}$, it shows RE values w.r.t. both LB_1 and LB_2 for the best and average solutions obtained with MA. The null RE w.r.t. LB_1 shows that the best solution found for both instances is in fact optimal, while the small average relative error suggests that the quality of all solutions is quite similar (and hence close to optimality). However, the values of RE w.r.t. LB_2 may be quite misleading: while it is approximately only 4% for $\mathsf{FT_F20}$ it soars up to 42% for $\mathsf{FT_F10}$, which may (wrongly) lead us to conclude that the behaviour of MA on $\mathsf{FT_F10}$ is actually an order of magnitude worse than for $\mathsf{FT_F20}$. For this reason, in the following we shall always use LB_1 to

compute RE values, refering to it as LB for the sake of simplicity.

5. Analysis of Existing Instances

In this section we shall analyse the potential of the existing instances as future benchmarks. As in Table 1, we will group the instances in families and, within each family, according to their size. For each group, a table will report for each instance and solving method considered the best and average RE values, together with the best makespan found, its expected value and the average expected makespan. To improve readability, the best average expected makespan value obtained across all methods for each instance appears in bold.

5.1. Analysis of Original Fuzzy Instances

We start the analysis of existing benchmarks with the so-called original fuzzy instances: S6.1–4, S10.1–4, Lei01,02 and LP01. Together with the results available for HDPSO in [41] and RKGA in [36] we will provide new results obtained with 30 runs of GRASP and MA.

5.1.1. Instances S6.1–4

The best solution found on each instance is always the same for all four methods, namely:

$$\begin{aligned}
\mathsf{S6.1} &: C_{max} = (56, 80, 103) \\
\mathsf{S6.2} &: C_{max} = (51, 70, 86) \\
\mathsf{S6.3} &: C_{max} = (50, 65, 84) \\
\mathsf{S6.4} &: C_{max} = (29, 36, 43)
\end{aligned}$$

Furthermore, the average expected makespan across all executions always coincides with the LB of each instance, with the exception of RKGA ,which obtains an average expected makespan of 70.45 for S6.2 while the best expected value obtained is equal to the LB=69.25.

The fact that all four methods (HDPSO, RKGA, GRASP and MA) obtain the optimal solution in all runs for every problem (except for the minor variation for RKGA in S6.2) indicates that these instances offer no room for improvement and hence are not adequate for future comparisons.

5.1.2. Instances S10.1–4

Results for problems S10.1–4 of size 10×10 can be seen in Table 4. It should be noted that the data for HDPSO on instances S10.2 and S10.4 are those published in the Erratum to [41].

We can see that GRASP obtains very similar results to RKGA and HDPSO; in average, GRASP is slightly better than RKGA in three instances and slightly worse than HDPSO in two instances, but overall differences in expected makespan are not significant, with an improvement of at most 2.7% and a worsening of at most 0.9%. Results are also very similar for the best makespan.

Table 4: Results on instances S10.1–4 (10×10)

Problem	LB	Method	Best C_{max}	$E[C_{max}]$		RE	
				Best	Avg.	Best	Avg.
S10.1	128.50	HDPSO	(96, 129, 160)	128.50	128.93	0.00	0.33
		RKGA	(96, 129, 160)	128.50	129.78	0.00	0.99
		GRASP	(96, 129, 160)	128.50	130.01	0.00	1.17
		MA	(96, 129, 160)	128.50	**128.50**	0.00	0.00
S10.2	122.50	HDPSO	(92, 119, 160)	122.50	125.90	0.00	2.78
		RKGA	(89, 123, 158)	123.25	127.25	0.61	3.88
		GRASP	(92, 122, 161)	124.25	126.53	1.43	3.29
		MA	(86, 125, 155)	122.75	**124.75**	0.20	1.84
S10.3	115.00	HDPSO	(85, 116, 143)	115.00	115.15	0.00	0.13
		RKGA	(85, 116, 143)	115.00	116.25	0.00	1.09
		GRASP	(85, 116, 143)	115.00	**115.00**	0.00	0.00
		MA	(85, 116, 143)	115.00	**115.00**	0.00	0.00
S10.4	45.75	HDPSO	(27, 47, 62)	45.75	46.28	0.00	1.15
		RKGA	(28, 47, 62)	46.00	47.13	0.55	3.01
		GRASP	(27, 47, 62)	45.75	45.88	0.00	0.29
		MA	(26, 46, 65)	45.75	**45.75**	0.00	0.00

Regarding MA, it obtains slightly better results and, in three of the four instances it achieves the optimal solution in all runs, while for the remaining instance S10.2—which is optimally solved by HDPSO— the variation between the best and the average is less than 1.7%. All instances are solved to optimality by at least one method (in the case of S10.1 and S10.3 by all methods). It is therefore reasonable to conclude that instances S10.1–4 do not have significant room for improvement.

5.1.3. Instances Lei01,02 and LP01

Finally, we analyse the larger instances proposed "to verify the capability to solve large-scale problem" in [36] of size 15×10 and [41] of size 16×16. Table 5 summarises the available data for RKGA and HDPSO as well as the results obtained with GRASP and MA. We can see that GRASP improves the average $E[C_{max}]$ obtained by RKGA in instances Lei01,02; unfortunately, the average value for HDPSO in LP01 is not available. GRASP also slightly improves the best solution for Lei01 and LP01. Regarding MA, it clearly outperforms the best and average expected makespan for all instances, with an improvement over 6% in the best solutions found for LP01 and over 4.5% in the average solutions obtained on Lei01 and Lei02.

The small RE values obtained by MA suggest that these instances offer little room for improvement. On the other hand, unlike previous test-beds, none of the methods reach the LB, being quite far from it in average. This indicates

Table 5: Results on Larger Instances Lei01,02 and LP01

Problem	LB	Method	Best C_{max}	$E[C_{max}]$		RE	
				Best	Avg.	Best	Avg.
Lei01	197.25	RKGA	(142, 207, 271)	206.75	210.45	4.82	6.69
		GRASP	(144, 207, 268)	206.50	208.85	4.69	5.88
		MA	(136, 200, 258)	198.50	**199.34**	0.63	1.06
Lei02	163.00	RKGA	(118, 170, 223)	170.25	175.60	4.45	7.73
		GRASP	(119, 171, 227)	172.00	175.11	5.52	7.43
		MA	(116, 162, 213)	163.25	**167.48**	0.15	2.74
LP01	186.00	HDPSO	(151, 202, 247)	200.50	–	7.80	–
		GRASP	(141, 194, 254)	195.75	201.48	5.24	8.32
		MA	(138, 190, 233)	187.75	**188.87**	0.94	1.54

that, despite the scarce room for improvement, the instances remain potentially unsolved. In consequence, we believe that they should be considered when evaluating future new algorithms.

5.2. Analysis of Fuzzified Instances

In this section we analyse the instances obtained after fuzzifying well-known benchmarks for crisp JSP. In the experimental study we will consider any fuzzy instance from Table 1 for which results with at least one method from RKGA, SNS, GPSO, ABC or MA are available. We now proceed to analyse the different fuzzified instances, grouped by families.

5.2.1. Instances FT

The FT instances, proposed by Fisher and Thomson [17], are three instances of size 6×6 (FT06), 10×10 (FT10) and 20×5 (FT20).

Instance FT06 has been fuzzified in the literature in three different ways, yielding four different fuzzy instances $\mathsf{FT_T}06$, $\mathsf{FT_F}06$, $\mathsf{FT_L}06$ and $\mathsf{FT_G}06$, which have been used in a few contributions. As it was the case with the S6.1–4 family, in these small instances both MA and GRASP reach the optimal solution in all runs, making it clear that these fuzzy instances obtained from FT06 are easy to solve and not complex enough to serve as future benchmark.

Table 6 contains results of GRASP and MA on the remaining fuzzy instances in this family. The only possible comparison with other methods from the literature consists in a qualitative comparison between GRASP and MA results for $\mathsf{FT_T}20$ and the convergence charts in [62]; in this case, it is easily seen that both methods outperform the GA given in that paper. In general, the relative improvement of MA with respect to GRASP in terms of the average expected makespan — $\mathsf{FT_F}10$ and $\mathsf{FT_G}10$ (5.2%), $\mathsf{FT_S}10$ (4.5%), $\mathsf{FT_T}20$ (8.7%), $\mathsf{FT_F}20$ (9.1%) and for $\mathsf{FT_G}20$ (9.9%)— suggests that the algorithms have a similar behaviour on every fuzzified version of each instance. This similar behaviour

Table 6: Results on Fuzzified FT Instances

Problem	LB	Method	Best C_{max}	$E[C_{max}]$		RE	
				Best	Avg.	Best	Avg.
FT_F10	930.00	GRASP	(899, 962, 1025)	962.00	986.13	3.44	6.04
		MA	(871, 930, 989)	930.00	**934.53**	0.00	0.49
FT_G10	938.50	GRASP	(888, 967, 1073)	973.75	997.23	3.76	6.26
		MA	(855, 930, 1039)	938.50	**948.06**	0.00	1.02
FT_S10	935.25	GRASP	(876, 959, 1078)	968.00	988.15	3.50	5.66
		MA	(844, 930, 1037)	935.25	**943.49**	0.00	0.88
FT_T20	1164.25	GRASP	(1184, 1242, 1291)	1239.75	1285.17	6.48	10.39
		MA	(1112, 1165, 1213)	1164.25	**1173.37**	0.00	0.78
FT_F20	1165.00	GRASP	(1178, 1258, 1338)	1258.00	1293.50	7.98	11.03
		MA	(1094, 1165, 1236)	1165.00	**1175.73**	0.00	0.92
FT_G20	1189.75	GRASP	(1172, 1260, 1402)	1273.50	1310.23	7.04	10.13
		MA	(1073, 1165, 1356)	1189.75	**1201.97**	0.00	1.03

between symmetric and non-symmetric instances can be further confirmed by the high similarity between the corresponding RE values, both for MA and GRASP algorithms.

Regarding the hardness of the instances in this family, MA already finds the optimal solution for all of them in at least one run, with average RE values also small (ranging from 0.5% to 1%). This indicates that there is scarce room for improvement on these instances, not encouraging research competition.

5.2.2. Instances La

The La family is a set of 40 instances proposed by Lawrence [34]; seven of these instances, La21, La24, La25, La27, La29, La38 and La40 form part of the set of ten problem instances of crisp JSP considered hard to solve [3]. As mentioned in Section 3, fuzzy versions of several of the La instances have been considered in the literature, in some cases using more than one fuzzification method.

Regarding the smallest instances, La_L01, La_S02, La_L06 and La_G0i (i=1, 3, 5, 7, 9), they are always optimally solved by MA, and sometimes even by GRASP. For instances La_F11–14 we observe in [25] the same behaviour as for FT06 above: MA obtains practically the same makespan value in all runs and the expected makespan is identical or very close to the lower bound. Also in [22] the proposed GA obtains solutions for La_G12–14 for which the most likely value coincides with the optimal solution of the original crisp instance. Furthermore, both MA and GRASP algorithms reach the optimal solution in every run. In consequence, it is reasonable to perform a deep analysis only for the larger instances: from La_F21 in the case of symmetric instances and from La_S19 in the case of non-symmetric ones.

Table 7 shows the results on these remaining La instances. For La$_Z$20–22 we reproduce the values provided in [70] for SNS together with the values provided in [71] for ABC, RKGA and GPSO, and the new values obtained with GRASP and MA. For the remaining instances in this family, the table includes new makespan and RE values obtained with GRASP and MA.

Table 7: Results on La Instances

Problem	LB	Method	Best C_{max}	$E[C_{max}]$		RE	
				Best	Avg.	Best	Avg.
La$_S$19	843.25	GRASP	(752, 850, 943)	848.75	863.91	0.65	2.45
		MA	(751, 837, 948)	843.25	**843.80**	0.00	0.07
La$_Z$20	912.50	RKGA	(801, 912, 1038)	915.75	932.25	0.36	2.16
		GPSO	(796, 900, 1025)	905.25	938.88	-0.79	2.89
		SNS	(790, 897, 1010)	898.50	915.18	-1.53	0.29
		ABC	(790, 891, 1008)	895.00	**913.50**	-1.92	0.11
		GRASP	(817, 904, 1040)	916.25	923.68	0.41	1.23
		MA	(817, 904, 1040)	916.25	916.25	0.41	0.41
La$_F$21	1046.00	GRASP	(995, 1095, 1195)	1095.00	1116.53	4.68	6.74
		MA	(981, 1053, 1125)	1053.00	**1054.80**	0.67	0.84
La$_S$21	1044.75	GRASP	(997, 1092, 1218)	1099.75	1117.15	5.26	6.93
		MA	(947, 1052, 1156)	1051.75	**1053.73**	0.67	0.86
La$_Z$21	1056.50	RKGA	(963, 1078, 1244)	1090.75	1126.98	3.24	6.67
		GPSO	(998, 1116, 1268)	1124.50	1145.63	6.44	8.44
		SNS	(950, 1075, 1241)	1085.25	1118.98	2.72	5.91
		ABC	(977, 1094, 1234)	1099.75	1095.20	4.09	3.66
		GRASP	(991, 1104, 1266)	1116.25	1127.75	5.66	6.74
		MA	(951, 1053, 1201)	1064.50	**1067.62**	0.76	1.05
La$_Z$22	937.00	RKGA	(859, 956, 1087)	964.50	984.23	2.93	5.04
		GPSO	(876, 983, 1138)	995.00	1018.25	6.19	8.67
		SNS	(844, 954, 1093)	961.25	985.58	2.59	5.18
		ABC	(840, 951, 1082)	956.00	980.88	2.03	4.68
		GRASP	(865, 972, 1113)	980.50	1003.42	4.64	7.09
		MA	(830, 927, 1065)	937.25	**943.20**	0.03	0.66
La$_F$24	935.00	GRASP	(908, 987, 1066)	987.00	1000.10	5.56	6.96
		MA	(865, 941, 1017)	941.00	**946.23**	0.64	1.20
La$_S$24	938.25	GRASP	(885, 990, 1090)	998.75	1001.26	5.38	6.72
		MA	(830, 951, 1156)	943.25	**947.35**	0.53	0.97
La$_F$25	977.00	GRASP	(959, 1027, 1095)	1027.00	1043.13	5.12	6.77
		MA	(897, 977, 1057)	977.00	**983.70**	0.00	0.69
La$_S$25	985.75	GRASP	(925, 1022, 1177)	1036.50	1051.94	5.15	6.71
		MA	(882, 977, 1109)	986.25	**991.08**	0.05	0.54
La$_F$27	1235.00	GRASP	(1254, 1326, 1398)	1326.00	1335.20	7.37	8.11
		MA	(1173, 1256, 1339)	1256.00	**1263.33**	1.70	2.29

Continued on next page

Table 7 – continued from previous page

Problem	LB	Method	Best C_{max}	$E[C_{max}]$		RE	
				Best	Avg.	Best	Avg.
La$_S$27	1233.00	GRASP	(1179, 1294, 1444)	1302.75	1336.01	5.66	8.35
		MA	(1148, 1251, 1368)	1254.50	**1261.56**	1.74	2.32
La$_F$29	1152.00	GRASP	(1171, 1262, 1353)	1262.00	1283.47	9.55	11.41
		MA	(1099, 1185, 1261)	1185.00	**1197.47**	2.86	3.95
La$_S$29	1156.50	GRASP	(1152, 1260, 1437)	1277.25	1290.76	10.44	11.61
		MA	(1064, 1173, 1336)	1186.50	**1200.35**	2.59	3.79
La$_S$36	1277.00	GRASP	(1188, 1315, 1497)	1328.75	1350.92	4.05	5.79
		MA	(1158, 1278, 1426)	1285.00	**1296.03**	0.63	1.49
La$_S$37	1398.25	GRASP	(1333, 1471, 1636)	1477.75	1513.23	5.69	8.22
		MA	(1273, 1419, 1569)	1420.00	**1432.99**	1.56	2.48
La$_F$38	1196.00	GRASP	(1210, 1295, 1380)	1295.00	1321.93	8.28	10.53
		MA	(1131, 1214, 1297)	1214.00	**1224.80**	1.51	2.41
La$_S$38	1204.50	GRASP	(1182, 1310, 1428)	1307.50	1327.54	8.55	10.22
		MA	(1110, 1219, 1345)	1223.25	**1234.55**	1.56	2.49
La$_S$39	1234.00	GRASP	(1172, 1302, 1474)	1312.50	1334.80	6.36	8.17
		MA	(1087, 1240, 1390)	1239.25	**1244.41**	0.43	0.84
La$_F$40	1222.00	GRASP	(1201, 1291, 1381)	1291.00	1314.00	5.65	7.53
		MA	(1144, 1233, 1322)	1233.00	**1239.80**	0.90	1.46
La$_S$40	1226.50	GRASP	(1178, 1274, 1410)	1248.00	1313.93	4.69	7.13
		MA	(1117, 1234, 1334)	1229.75	**1241.79**	0.26	1.25

The RE values for instance La$_Z$20 highlight the importance of publishing the exact data for each instance instead of only making known the fuzzification method used, due to the stochastic nature or the latter. Indeed, negative RE values in Table 7 indicate that the expected values of the fuzzy durations generated for this review are greater than those used in [71]. This advocates the need of making all information regarding benchmark instances openly available to the research community, to allow for fair and rigourous comparisons.

The results obtained for La$_F$21, La$_S$21 and La$_Z$21, as well as those obtained for the rest of instances having symmetric and non-symmetric fuzzy versions, allow for some additional comparisons between the symmetric and non-symmetric fuzzy versions of the same original instance. In particular, we can see that the behaviour of GRASP and MA on each pair of corresponding instances is very similar when measured in terms of RE values. Also, the results clearly show that GRASP is not competitive on these instances, while MA reveals itself as the best method. In particular, for La$_Z$21 and La$_Z$22 MA is respectively 84% and 89% better than the other methods in terms of RE. In general, MA reduces the average RE w.r.t. GRASP more than 75% using considerably less running time. More importantly, the relatively small RE values obtained with MA suggest that, perhaps with the exception of La$_F$29, La instances do not really

stand as a major challenge. Having said this, all fuzzy versions of instances La21,24,36,39,40 show a complexity similar to that of the instances Lei01,02 and LP01, where the optimal solutions have not been reached but average RE values are relatively small. Fuzzy versions of La27,37,40 appear to be slightly harder, being also unsolved and with average RE values around 2.5%. Therefore we consider that these instances may still be considered open and useful to assess future solving methods, specially the instances built from La29.

5.2.3. Instances ORB

The original benchmark ORB from [3] consists of ten instances of size 10×10; it is the first five of these instances that have been fuzzified in [70] to evaluate different metaheuristics for the FJSP.

Table 8 contains the results on these five instances reported in [70] for SNS together with the values provided in [71] for RKGA, GPSO and ABC, as well as new values obtained with GRASP and MA. According to these results, MA is the method with best average performance, achieving the best $E[C_{max}]$ on all five instances (together with ABC for $\text{ORB}_\text{Z}2$). In fact, the best expected makespan coincides with LB in all cases. Not only does MA reach the optimal solution in all instances, but also the average RE values are considerably small, less than 1% in average. This leads us to conclude that there is scarce room for improvement in these problems.

5.2.4. Instances ABZ

The five ABZ instances for JSP, first proposed by Adams, Balas, and Zawack [2], can be divided in two groups: instances ABZ5,6 of size 10×10 and the larger and more difficult instances ABZ7–9 of size 20×15 — in fact, the last two instances are still open problems, since the optimal solution is to date unknown. As already mentioned, fuzzy versions ABZ_F7–9 of the larger instances first appear in [23, 26] while the smaller ones are fuzzified in [70] to obtain ABZ_Z5,6 and in [22] to obtain ABZ_G5,6. Table 9 contains the results available in the literature for the small instances as well as results obtained with GRASP and MA on all five instances.

Table 9 again illustrates using the same fuzzification method can result in quite different instances (see the errors for ABZ_Z6). In any case, for the smallest instances we can observe that RE values are not only small but very similar for both GRASP and MA. In addition, the optimal solution is reached for each instance by either MA or GRASP. Regarding the largest instances, the behaviour on instance ABZ_F7 appears to be quite similar than that on La27,37,40, making it suitable for future comparisons. Even more interesting are instances ABZ_F8,9. Here, MA obtains much better RE values than GRASP, despite of which average RE values are greater than 7% for ABZ_F8,9, more than twice the RE values on previous instances. This shows the potential of the largest ABZ_F instances for future research.

5.3. Summary

As an overall conclusion from the analysis performed on all instances proposed in the literature, we see that not all of these instances are suitable for assessing future solving methods for FJSP. The existing instances that may offer enough room for improvement to serve as future benchmarks are the original fuzzy instances Lei01, Lei02 and LP01, the fuzzified instances from La21,24,36,39,40, La27,37,40 and ABZ_F7 (which are not yet solved to optimality even if they do not seem specially hard) and finally instances ABZ_F8,9, La_S29 and La_F29, this last group representing a real challenge.

The analysis has also illustrated the advantages of having an appropriate lower bound to calculate accurate RE values. Indeed, inaccurate lower bounds or errors measured w.r.t. best known solutions may (wrongly) lead us to think that an instance is far from being solved when this is not the case.

Notice that even though in this paper we have described a method to compute accurate lower bounds for fuzzy instances, the difficulty of some of these instances make it hard or even impossible to compute the lower bound in a reasonable amount of time. Take for example the fuzzy instances built from La29, where the IBM ILOG CPLEX CP Optimizer took more than 30 minutes to find the optimal solution. In fact, there could be deterministic instances for which the optimal solution cannot be found. We may say that the proposed method for finding lower bounds by solving the associated expected crisp problem is accurate but not scalable, and therefore not appropriate when dealing with harder instances. Here the symmetric instances offer a great advantage, since they can benefit from all the studies conducted through decades in classical JSP, both with exact methods and with algorithms that reach good lower bounds (cf [43]). In this case, good LB values can be obtained from standard repositories, as the OR library, thus allowing for fairer assessments when using symmetric fuzzy instances instead of non-symmetric ones. For these reasons, we propose to use the fuzzification method from[18] to generate new larger and harder fuzzy instances with accurate lower bounds.

6. New Instances

Results in Section 5 for the existing FJSP instances suggest that it is necessary to have more challenging problem instances in order to test the potential of future proposals to solving the FJSP.

Here we propose a new test bed based on the well-known Ta benchmark proposed in [58] for the JSP. The Ta benchmark is composed of 80 instances, each with a number of tasks varying from 225 to 2000. 50 of these instances are considered to be harder than the remaining ones, namely instances Ta01–50. In fact, the optimal solution has been found so far for only 17 of those 50 instances. Moreover, instances Ta21–30 (size 20×20) and Ta41–50 (size 30×20) are considered to be "the most difficult JSP benchmark problems" [69] and to date it has not been proved if the best-known solutions are indeed optimal or not.

We use the fuzzification method proposed in [18] to build fuzzy instances from the 20 hardest Ta instances, namely Ta21–30 and Ta41–50. As explained in the previous section, this fuzzification method allows to obtain accurate lower bounds for the expected makespan of the fuzzy instances. Here, the lower bounds for the resulting fuzzy instances are given by the best known lower bounds for the crisp instances[1]. The resulting test bed, together with all other instances used in this paper and detailed results of MA and GRASP, is available on the internet[2] and as supplementary electronic material to this work, in order to facilitate experiment reproducibility and encourage research competition.

We have run both MA and GRASP on this benchmark in order to analyse the difficulty of the problems and also to provide preliminary results for future reference. To find the MA's convergence point we have followed the method explained in Section 4, letting the algorithm evolve with a population of size 100 until it does not improve significatively. This method suggests that convergence is achieved with 300 generations for 20×20 instances and 425 generations for 30×20 instances, which takes an average runtime between 35 and 108 seconds in the case of the MA, and between 41 and 115 seconds for the GRASP evaluating the same number of solutions.

Tables 10 and 11 show respectively the results obtained with both algorithms for fuzzy instances $\mathsf{Ta_F}$21–30 and $\mathsf{Ta_F}$41–50, including the best makespan and best and average expected makespan values, together with the best and average RE values w.r.t the lower bound, which is also included next to the name of each instance.

For these two sets of instances, the behaviour of the algorithms is similar to their behaviour in previous test beds in the sense that MA reduces considerably the average RE obtained with GRASP, with an overall improvement over 50%. Notice however that average RE values for these instances are much larger than those obtained in the previously-analysed sets of instances, over 10% for MA and 20% for GRASP, being greater than these values in seven out of the 10 instances $\mathsf{Ta_F}$41–50. These RE values are only comparable to the RE values obtained on instances $\mathsf{ABZ_F}$7–9. This suggests that the new benchmark does indeed offer room for improvement, posing a challenge to researchers.

7. Conclusions

In this paper we have reviewed the state-of-the-art for FJSP, with a critical evaluation of the test beds commonly used to assess the performance of algorithms proposed for this problem.

A thorough analysis of the difficulty of existing benchmark instances has been carried out based on results reported in the literature for several solving methods, together with new results obtained with a memetic algorithm (MA)

[1] An up-to-date record of these bounds can be found in the benchmark repository `http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html`

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

from the literature. To avoid possible objections to the analysis based on the complexity of the MA, we have also proposed and used a GRASP algorithm, based on the neighbourhood structure from MA. Results have shown that most of the proposed instances either are already optimally solved or have solutions so close to the lower bound that no room for significant improvement is left. This suggests that these instances are not appropriate to assess future metaheuristic proposals. We have highlighted and made openly available to the research community those instances that are still challenging enough and in addition we have proposed a new more challenging benchmark composed of 20 fuzzy instances generated from the so-considered most difficult JSP problems. As future reference, we have provided preliminary results on these instances obtained with the GRASP and MA algorithms.

Acknowledgements

This research has been supported by the Spanish Government under Grants FEDER TIN2013-46511-C2-2-P and MTM2014-55262-P.

References

[1] S. Abdullah and M. Abdolrazzagh-Nezhad. Fuzzy job-shop scheduling problems: A review. *Information Sciences*, 278:380–407, 2014. doi: 10.1016/j.ins.2014.03.060.

[2] J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Managament Science*, 34:391–401, 1988.

[3] D. Applegate and W. Cook. A computational study of the job-shop scheduling problem. *ORSA Journal of Computing*, 3:149–156, 1991.

[4] S. Balin. Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation. *Information Sciences*, 181:3551–3569, 2011.

[5] J. E. Beasley. OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990. URL `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

[6] G. Bortolan and R. Degani. A review of some methods for ranking fuzzy subsets. In D. Dubois, H. Prade, and R. Yager, editors, *Readings in Fuzzy Sets for Intelligence Systems*, pages 149–158. Morgan Kaufmann, Amsterdam (NL), 1993.

[7] I. Boussaïd, J. Lepagnot, and P. Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013. doi: 10.1016/j.ins.2013.02.041.

[8] M. Brunelli and J. Mezei. How different are ranking methods for fuzzy numbers? A numerical study. *International Journal of Approximate Reasoning*, 54:627–639, 2013. doi: 10.1016/j.ijar.2013.01.009.

[9] G. Celano, A. Costa, and S. Fichera. An evolutionary algorithm for pure fuzzy flowshop scheduling problems. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 11:655–669, 2003.

[10] S. Chanas and A. Kasperski. On two single machine scheduling problems with fuzzy processing times and fuzzy due dates. *European Journal of Operational Research*, 147:281–296, 2003.

[11] S.-M. Chen and T.-H. Chang. Finding multiple possible critical paths using fuzzy PERT. *IEEE Transactions on Systems, Man, and Cybernetics–Part B:*, 31(6):930–937, 2001.

[12] S. Destercke and I. Couso. Ranking of fuzzy intervals seen through the imprecise probabilistic lens. *Fuzzy Sets and Systems*, In press, 2014. doi: 10.1016/j.fss.2014.12.009.

[13] D. Dubois and H. Prade. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenum Press, New York (USA), 1986.

[14] D. Dubois and H. Prade, editors. *Fundamentals of Fuzzy Sets*. The Handbooks of Fuzzy Sets. Kluwer Academic Publishers, Boston/London/Dordrecht, 2000.

[15] D. Dubois, H. Fargier, and P. Fortemps. Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. *European Journal of Operational Research*, 147:231–252, 2003.

[16] D. Dubois, H. Fargier, and P. Fortemps. Scheduling under flexible constraints and uncertain data: the fuzzy approach. In *Production Scheduling*, chapter 11, pages 301–332. Wiley, 2008.

[17] H. Fisher and G. L. Thomson. Probabilistic learning combinations of local job-shop scheduling rules. In J. F. Muth and G. L. Thomson, editors, *Industrial Scheduling*, pages 225–251. Prentice Hall, 1963.

[18] P. Fortemps. Jobshop scheduling with imprecise durations: a fuzzy approach. *IEEE Transactions of Fuzzy Systems*, 7:557–569, 1997.

[19] P. Fortemps. Editorial. Fuzzy sets in scheduling and planning. *European Journal of Operational Research*, 147:229–230, 2003.

[20] J. Fortin, D. Dubois, and H. Fargier. Gradual numbers and their application to fuzzy interval analysis. *IEEE Transactions on Fuzzy Systems*, 16(2):388–402, 2008. doi: 10.1109/TFUZZ.2006.890680.

[21] J. Fortin, P. Zielinski, D. Dubois, and H. Fargier. Criticality analysis of activity networks under interval uncertainty. *Journal of Scheduling*, 13(6): 609–627, 2010. doi: 10.1007/s10951-010-0163-3.

[22] O. A. Ghrayeb. 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, 2003.

[23] I. González Rodríguez, C. R. Vela, and J. Puente. A memetic approach to fuzzy job shop based on expectation model. In *Proceedings of IEEE International Conference on Fuzzy Systems, FUZZ-IEEE2007*, pages 692–697, London, 2007. IEEE.

[24] I. González Rodríguez, J. Puente, C. R. Vela, and R. Varela. Semantics of schedules for the fuzzy job shop problem. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 38(3):655–666, 2008.

[25] I. González Rodríguez, C. R. Vela, J. Puente, and R. Varela. 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)*, pages 124–131, Sidney, 2008. AAAI Press.

[26] I. González Rodríguez, C. R. Vela, A. Hernández-Arauzo, and J. Puente. Improved local search for job shop scheduling with uncertain durations. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS-2009)*, pages 154–161, Thesaloniki, 2009. AAAI Press.

[27] S. Heilpern. The expected value of a fuzzy number. *Fuzzy Sets and Systems*, 47:81–86, 1992.

[28] W. Herroelen and R. Leus. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165: 289–306, 2005.

[29] Y. Hu, M. Yin, and X. Li. A novel objective function for job-shop scheduling problem with fuzzy processing time and fuzzy due date using differential evolution algorithm. *International Journal of Advanced Manufactoring Technology*, 56:1125–1138, 2011.

[30] IBM. IBM CPLEX Optimizer, 2014. URL `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html`.

[31] H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, 67(3): 392–403, 1998.

[32] A. Kasperski. Some general properties of a fuzzy single machine scheduling problem. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 15(1):43–56, 2007.

[33] M. Kuroda and Z. Wang. Fuzzy job shop scheduling. *International Journal of Production Economics*, 44:45–51, 1996.

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

[35] D. Lei. Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems. *International Journal of Advanced Manufacturing Technology*, 37:157–165, 2008.

[36] D. Lei. Solving fuzzy job shop scheduling problems using random key genetic algorithm. *International Journal of Advanced Manufacturing Technologies*, 49:253–262, 2010.

[37] D. Lei. Fuzzy job shop scheduling problem with availability constraints. *Computers & Industrial Engineering*, 58:610–617, 2010.

[38] D. Lei. A genetic algorithm for flexible job shop scheduling with fuzzy processing time. *International Journal of Production Research*, 48(10): 2995–3013, 2010. doi: 10.1080/00207540902814348.

[39] D. Lei. Scheduling fuzzy job shop with preventive maintenance through swarm-based neighborhood search. *International Journal of Advanced Manufacturing Technology*, 61:1200–1208, 2011. doi: 10.1016/j.cie.2011.07.010.

[40] D. Lei. Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling. *Applied Soft Computing*, 12:2237–2245, 2012. doi: 10.1016/j.asoc.2012.03.025.

[41] J.-q. Li and Y.-x. Pan. A hybrid discrete particle swarm optimization algorithm for solving fuzzy job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 66:583–596, 2013. doi: 10.1007/s00170-012-4337-3.

[42] F.-T. Lin. Fuzzy job-shop scheduling based on ranking level $(\lambda, 1)$ interval-valued fuzzy numbers. *IEEE Transactions on Fuzzy Systems*, 10(4):510–522, 2002.

[43] C. Mencía, M. Sierra, and R. Varela. Intensified iterative deepening A* with application to job shop scheduling. *Journal of Intelligent Manufacturing*, 25 (6):1245–1255, 2014. doi: 10.1007/s10845-012-0726-6.

[44] H. T. Nguyen and E. A. Walker. *A First Course in Fuzzy Logic*. Chapman & Hall, second edition, 2000.

[45] Q. Niu, B. Jiao, and X. Gu. Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time. *Applied Mathematics and Computation*, 205:148–158, 2008.

[46] J. J. Palacios, I. González-Rodríguez, C. R. Vela, and J. Puente. Robust swarm optimisation for fuzzy open shop scheduling. *Natural Computing*, 13(2):145–156, 2014. doi: 10.1007/s11047-014-9413-1.

[47] J. J. Palacios, I. González-Rodríguez, C. R. Vela, and J. Puente. Co-evolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop. *Fuzzy Sets and Systems*, In press, 2014. doi: 10.1016/j.fss.2014.12.003.

[48] J. J. Palacios, M. A. González, C. R. Vela, I. González-Rodríguez, and J. Puente. Genetic tabu search for the fuzzy flexible job shop problem. *Computers & Operations Research*, 54:74–89, 2015. ISSN 0305-0548. doi: 10.1016/j.cor.2014.08.023.

[49] J. Peng and B. Liu. Parallel machine scheduling models with fuzzy processing times. *Information Sciences*, 166:49–66, 2004.

[50] S. Petrovic and C. Fayad. A fuzzy shifting bottleneck hybridised with genetic algorithm for real-world job shop scheduling. In *Mini-EURO Conference, Managing Uncertainty in Decision Support Models*, pages 1–6, 2004.

[51] S. Petrovic and X. Song. A new approach to two-machine flow shop problem with uncertain processing times. *Optimization and Engineering*, 7:329–342, 2006.

[52] S. Petrovic, S. Fayad, D. Petrovic, E. Burke, and G. Kendall. Fuzzy job shop scheduling with lot-sizing. *Annals of Operations Research*, 159:275–292, 2008.

[53] M. L. Pinedo. *Scheduling. Theory, Algorithms, and Systems.* Springer, third edition, 2008.

[54] J. Puente, C. R. Vela, and I. González-Rodríguez. Fast local search for fuzzy job shop scheduling. In *Proceedings of ECAI 2010*, pages 739–744. IOS Press, 2010.

[55] M. Sakawa and R. Kubota. 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, 2000.

[56] M. Sakawa and T. Mori. An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate. *Computers & Industrial Engineering*, 36:325–341, 1999.

[57] X. Song, Y. Zhu, C. Yin, and L. Fuming. 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*, pages 7362–7365, 2006.

[58] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64:278–285, 1993.

[59] E.-G. Talbi. *Metaheuristics. From Design to Implementation.* Wiley, 2009.

[60] R. Tavakkoli-Moghaddam, N. Safei, and M. Kah. Accessing feasible space in a generalized job shop scheduling problem with the fuzzy processing times: a fuzzy-neural approach. *Journal of the Operational Research Society*, 59: 431–442, 2008.

[61] C. K. Teoh, A. Wibovo, and M. S. Ngadiman. Review of state of the art for metaheuristic techniques in academic scheduling problems. *Artificial Intelligence Review*, 2013. doi: DOI10.1007/s10462-013-9399-6.

[62] Y. Tsujimura, M. Gen, and E. Kubota. Solving job-shop scheduling problem with fuzzy processing time using genetc algorithm. *Journal of Japan Society for Fuzzy Theory and Systems*, 7:1073–1083, 1995.

[63] B. Wang, Q. Li, X. Yang, and X. Wang. Robust and satisfactory job shop scheduling under fuzzy processing times and flexible due dates. In *Proc. of the 2010 IEEE International Conference on Automation and Logistics*, pages 575–580, 2010.

[64] L. Wang, G. Zhou, Y. Xu, and L. Min. A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem. *International Journal of Production Research*, 51(12):3593–3608, 2013.

[65] S. Wang, L. Wang, Y. Xu, and L. Min. An effective estimation of distribution algorithm for the flexible job-shop scheduling problem with fuzzy processing time. *International Journal of Production Research*, 51(12): 3779–3793, 2013.

[66] B. K. Wong and V. S. Lai. A survey of the application of fuzzy set theory in production and operations management: 1998–2009. *International Journal of Production Economics*, 129:157–168, 2011.

[67] Y. Xie, J. Xie, and J. Li. Fuzzy due dates job shop scheduling problem based on neural network. In *Advances in Neural Networks–ISNN 2005*, volume 3496 of *Lecture Notes in Computer Science*, pages 782–787. Springer, 2005. doi: 10.1007/11427391_125.

[68] Z. Xu, X. Gu, and B. Jiao. Research on job shop scheduling under uncertainty. *ACM*, pages 695–702, 2009.

[69] C. Y. Zhang, P. Li, Y. Rao, and Z. Guan. A very fast TS/SA algorithm for the job shop scheduling problem. *Computers & Operations Research*, 35: 282–294, 2008.

[70] Y. Zheng, Y. Li, and D. Lei. Swarm-based neighbourhood search for fuzzy job shop scheduling. *International Journal of Innovative Computing and Applications*, 3(3):144–151, 2011.

[71] Y.-L. Zheng and Y.-X. Li. Artificial bee colony algorithm for fuzzy job shop scheduling. *International Journal of Computer Applications in Technology*, 44 (2):124–129, 2012.

Table 8: Results on ORB Instances

Problem	LB	Method	Best C_{max}	$E[C_{max}]$		RE	
				Best	Avg.	Best	Avg.
ORB$_Z$1	1073.50	RKGA	(969, 1111, 1240)	1107.75	1138.65	3.19	6.07
		GPSO	(979, 1104, 1245)	1108.00	1141.60	3.21	6.34
		SNS	(958, 1086, 1222)	1088.00	1121.28	1.35	4.45
		ABC	(953, 1086, 1219)	1086.00	1118.65	1.16	4.21
		GRASP	(986, 1086, 1245)	1100.75	1148.81	2.54	7.02
		MA	(959, 1059, 1217)	1073.50	**1095.67**	0.00	2.07
ORB$_Z$2	896.00	RKGA	(795, 913, 1029)	912.50	925.28	1.84	3.27
		GPSO	(791, 909, 1031)	910.00	928.50	1.56	3.63
		SNS	(789, 892, 1013)	896.50	924.43	0.06	3.17
		ABC	(793, 894, 1003)	896.00	908.40	0.00	1.38
		GRASP	(799, 897, 1039)	908.00	920.15	1.34	2.70
		MA	(784, 889, 1022)	896.00	**896.93**	0.00	0.10
ORB$_Z$3	1015.25	RKGA	(919, 1032, 1175)	1039.50	1092.73	2.39	7.63
		GPSO	(934, 1064, 1205)	1066.75	1095.50	5.07	7.90
		SNS	(922, 1044, 1182)	1048.00	1080.33	3.23	6.41
		ABC	(916, 1033, 1171)	1038.25	1084.85	2.27	6.86
		GRASP	(942, 1051, 1203)	1061.75	1111.45	4.58	9.48
		MA	(900, 1005, 1151)	1015.25	**1026.51**	0.00	1.11
ORB$_Z$4	1014.50	RKGA	(907, 1034, 1177)	1038.00	1060.23	2.32	4.51
		GPSO	(921, 1030, 1170)	1037.75	1057.55	2.29	4.24
		SNS	(915, 1026, 1166)	1033.25	1052.68	1.85	3.76
		ABC	(909, 1018, 1157)	1025.50	1044.58	1.08	2.96
		GRASP	(908, 1018, 1160)	1026.00	1057.38	1.13	4.23
		MA	(894, 1005, 1154)	1014.50	**1022.69**	0.00	0.81
ORB$_Z$5	897.25	RKGA	(817, 913, 1032)	918.75	927.23	2.40	3.34
		GPSO	(813, 905, 1034)	914.25	924.85	1.89	3.08
		SNS	(813, 905, 1034)	914.25	918.70	1.89	2.39
		ABC	(813, 905, 1034)	914.25	916.70	1.89	2.17
		GRASP	(810, 903, 1039)	913.75	938.33	1.84	4.58
		MA	(793, 890, 1016)	897.25	**901.37**	0.00	0.46

Table 9: Results on ABZ Instances

Problem	LB	Method	Best C_{max}	$E[C_{max}]$		RE	
				Best	Avg.	Best	Avg.
ABZ_G5	1247.50	GRASP	(1127, 1242, 1379)	1247.50	1262.63	0.00	1.21
		MA	(1127, 1242, 1379)	1247.50	**1249.08**	0.00	0.13
ABZ_Z5	1248.75	RKGA	(1107, 1240, 1425)	1253.00	1269.13	0.34	1.63
		GPSO	(1103, 1249, 1427)	1257.00	1272.50	0.66	1.90
		SNS	(1108, 1242, 1418)	1252.50	1267.43	0.30	1.50
		ABC	(1107, 1245, 1403)	1250.00	1260.80	0.10	0.96
		GRASP	(1107, 1234, 1420)	1248.75	1263.13	0.00	1.15
		MA	(1108, 1239, 1413)	1249.75	**1251.86**	0.08	0.25
ABZ_G6	950.25	GRASP	(876, 942, 1041)	950.25	964.49	0.00	1.50
		MA	(876, 942, 1041)	950.25	**960.60**	0.00	1.09
ABZ_Z6	952.50	RKGA	(824, 948, 1074)	948.50	964.68	-0.42	1.28
		GPSO	(841, 943, 1070)	949.25	971.00	-0.34	1.94
		SNS	(831, 945, 1068)	947.25	961.43	-0.55	0.94
		ABC	(831, 945, 1068)	947.25	957.18	-0.55	0.49
		GRASP	(840, 945, 1080)	952.50	962.06	0.00	1.00
		MA	(844, 948, 1074)	953.50	**956.14**	0.10	0.38
ABZ_F7	656.00	GRASP	(676, 721, 766)	721.00	729.13	9.91	11.15
		MA	(627, 670, 713)	670.00	**679.47**	2.13	3.58
ABZ_F8	645.00	GRASP	(692, 731, 770)	731.00	746.07	13.33	15.67
		MA	(649, 683, 717)	683.00	**692.10**	5.89	7.30
ABZ_F9	661.00	GRASP	(718, 765, 812)	765.00	774.67	15.73	17.20
		MA	(660, 700, 740)	700.00	**707.97**	5.90	7.11

Table 10: Results on $\mathsf{Ta_F}$21–30 fuzzy instances

Problem	LB	Method	Best C_{max}	$E[C_{max}]$		RE	
				Best	Avg.	Best	Avg.
$\mathsf{Ta_F}$21	1573	GRASP	(1694, 1813, 1932)	1813.00	1850.00	15.26	17.61
		AM	(1547, 1679, 1811)	1679.00	1709.00	6.74	8.65
$\mathsf{Ta_F}$22	1542	GRASP	(1636, 1775, 1914)	1775.00	1808.37	15.11	17.27
		AM	(1485, 1632, 1779)	1632.00	1650.80	5.84	7.06
$\mathsf{Ta_F}$23	1474	GRASP	(1639, 1739, 1839)	1739.00	1758.80	17.98	19.32
		AM	(1506, 1600, 1694)	1600.00	1627.77	8.55	10.43
$\mathsf{Ta_F}$24	1606	GRASP	(1681, 1794, 1907)	1794.00	1820.80	11.71	13.37
		AM	(1555, 1676, 1797)	1676.00	1696.97	4.36	5.66
$\mathsf{Ta_F}$25	1518	GRASP	(1669, 1760, 1851)	1760.00	1781.80	15.94	17.38
		AM	(1570, 1655, 1740)	1655.00	1667.43	9.03	9.84
$\mathsf{Ta_F}$26	1558	GRASP	(1677, 1823, 1969)	1823.00	1851.30	17.01	18.83
		AM	(1561, 1686, 1811)	1686.00	1711.70	8.22	9.87
$\mathsf{Ta_F}$27	1617	GRASP	(1763, 1894, 2025)	1894.00	1926.73	17.13	19.15
		AM	(1602, 1716, 1830)	1716.00	1744.43	6.12	7.88
$\mathsf{Ta_F}$28	1591	GRASP	(1655, 1770, 1885)	1770.00	1803.60	11.25	13.36
		AM	(1497, 1629, 1761)	1629.00	1654.60	2.39	4.00
$\mathsf{Ta_F}$29	1525	GRASP	(1666, 1795, 1924)	1795.00	1821.00	17.70	19.41
		AM	(1545, 1647, 1749)	1647.00	1666.80	8.00	9.30
$\mathsf{Ta_F}$30	1485	GRASP	(1652, 1764, 1876)	1764.00	1791.77	18.79	20.66
		AM	(1512, 1631, 1750)	1631.00	1645.67	9.83	10.82

Table 11: Results on $\mathsf{Ta_F}$41–50 fuzzy instances

Problem	LB	Method	Best C_{max}	$E[C_{max}]$		RE	
				Best	Avg.	Best	Avg.
$\mathsf{Ta_F}$41	1874	GRASP	(2140, 2321, 2502)	2321.00	2377.87	23.72	26.75
		AM	(1955, 2143, 2331)	2143.00	2175.50	14.23	15.96
$\mathsf{Ta_F}$42	1867	GRASP	(2107, 2267, 2427)	2267.00	2309.90	21.42	23.72
		AM	(1893, 2058, 2223)	2058.00	2090.23	10.23	11.96
$\mathsf{Ta_F}$43	1809	GRASP	(2045, 2206, 2367)	2206.00	2233.20	21.95	23.45
		AM	(1866, 1985, 2104)	1985.00	2008.83	9.73	11.05
$\mathsf{Ta_F}$44	1927	GRASP	(2162, 2320, 2478)	2320.00	2350.27	20.39	21.97
		AM	(1947, 2090, 2233)	2090.00	2120.07	8.46	10.02
$\mathsf{Ta_F}$45	1997	GRASP	(2126, 2278, 2430)	2278.00	2306.73	14.07	15.51
		AM	(1942, 2076, 2210)	2076.00	2098.80	3.96	5.10
$\mathsf{Ta_F}$46	1940	GRASP	(2161, 2337, 2513)	2337.00	2377.07	20.46	22.53
		AM	(1978, 2133, 2288)	2133.00	2161.37	9.95	11.41
$\mathsf{Ta_F}$47	1789	GRASP	(2072, 2216, 2360)	2216.00	2242.17	23.87	25.33
		AM	(1893, 2011, 2129)	2011.00	2046.73	12.41	14.41
$\mathsf{Ta_F}$48	1912	GRASP	(2092, 2251, 2410)	2251.00	2279.73	17.73	19.23
		AM	(1922, 2059, 2196)	2059.00	2084.03	7.69	9.00
$\mathsf{Ta_F}$49	1915	GRASP	(2140, 2267, 2394)	2267.00	2297.80	18.38	19.99
		AM	(1904, 2038, 2172)	2038.00	2076.43	6.42	8.43
$\mathsf{Ta_F}$50	1807	GRASP	(2113, 2262, 2411)	2262.00	2294.67	25.18	26.99
		AM	(1940, 2053, 2166)	2053.00	2077.13	13.61	14.95