Postprint version of the paper:
Palacios, J.J., González-Rodríguez I., Vela, C.R., Puente J.
Satisfying flexible due dates in fuzzy job shop by means of hybrid evolutionary algorithms. Integrated Computer-Aided Engineering, vol. 26, no. 1, pp. 65-84, 2019.
doi: 10.3233/ICA-180583

Undefined 1 (2018) 1–5
IOS Press

Satisfying flexible due dates in fuzzy job shop by means of hybrid evolutionary algorithms [1]

Juan José Palacios [a], and Inés González-Rodríguez [b] and Camino R. Vela [a*] and Jorge Puente [a]
[a] *Department of Computer Science, Campus de Viesques s/n, Gijón, 33204, University of Oviedo, (Spain)*
E-mail: {palaciosjuan,crvela,puente}@uniovi.es
[b] *Department of Mathematics, Statistics and Computing, Los Castros s/n, Santander, 39005 University of Cantabria, (Spain)*
E-mail: ines.gonzalez@unican.es

Abstract. This paper tackles the job shop scheduling problem with fuzzy sets modelling uncertain durations and flexible due dates. The objective is to achieve high-service level by maximising due-date satisfaction, considering two different overall satisfaction measures as objective functions. We show how these functions model different attitudes in the framework of fuzzy multicriteria decision making and we define a measure of solution robustness based on an existing a-posteriori semantics of fuzzy schedules to further assess the quality of the obtained solutions. As solving method, we improve a memetic algorithm from the literature by incorporating a new heuristic mechanism to guide the search through plateaus of the fitness landscape. We assess the performance of the resulting algorithm with an extensive experimental study, including a parametric analysis, and a study of the algorithm's components and synergy between them. We provide results on a set of existing and new benchmark instances for fuzzy job shop with flexible due dates that show the competitiveness of our method.

Keywords: Job Shop Scheduling, Fuzzy processing times, Flexible due dates, Memetic Algorithm, Robustness

1. Introduction

Scheduling is a decision-making process that tackles the allocation of resources to tasks over given time periods with the goal of optimising one or more objectives. This kind of problems is pervasive in a growing number of domains, including engineering, management science or distributed and parallel computing, to mention but a few [2,63,86]. Within scheduling, one of the most relevant problems is the job shop, a combinatorial optimisation problem where pre-defined sequences of tasks are organised into jobs, so each job must visit several resources or machines in a predetermined route. Different processing restrictions and constraints (e.g. job release or due dates or machine setup times) as well as different objective functions (many of them dependent on the completion times of the jobs) yield different variants of the problem. It has many practical applications (e.g. wafer fabs in the semiconductor industry often function as job shops). It also poses a challenge to the research community due to its complexity [63].

The most common objective in the literature is to find a schedule with minimum makespan (which is the time when the last job is finished), but due-date fulfilment has also occupied researchers and is receiving increasing attention in recent years [40,63]. Indeed, on-time fulfilment appears to be of prime importance in modern pull-oriented supply chain systems and keeping job due dates is a prerequisite for serving customers within the promised delivery time and avoiding out-of-stocks costs. Hence, the use of due-date satisfaction measures in production scheduling helps the companies to increase their logistic service and create competitive advantage.

[1] This research has been supported by the Spanish Government under research grant TIN2016-79190-R.

Classical objectives related to due-date satisfaction are total weighted tardiness or maximum lateness, which are typically associated with customer satisfaction and service level in make-to-order environments. These approaches fail however to account for the case — often encountered in practice — where due-date constraints are flexible, for instance, if preferences of customers are incorporated [7]. Traditional approaches also assume that scheduling problems are static and certain: all activities and their durations are precisely known in advance and do not change as the solution is being executed. Still, in many real-world applications design variables are subject to perturbations or changes, causing optimal solutions to the original problem to be of little or no use in practice [6,35].

With the goal of narrowing the gap between theory and applications, some researchers have started to take into consideration uncertainty and flexibility. To this end, fuzzy sets provide an interesting framework since they allow both for tackling uncertainty and accounting for preferences, as well as addressing solution robustness [22,38].

Whether in its more classical form or taking into account uncertainty or flexibility, many scheduling problems, including the job shop, belong to the class of hard combinatorial optimisation problems. For this reason, researchers often resort to metaheuristic search methods to solve them. In particular, hybrid metaheuristic approaches such as memetic algorithms have proved very successful in tackling complex problems of combinatorial nature [16,49,71,84].

In the following, the fuzzy job shop scheduling problem, or FJSP in short, is addressed. This is a job shop problem with uncertain durations and flexible due dates modelled as fuzzy numbers and fuzzy thresholds. The goal is to maximise the overall due-date satisfaction, which translates into two different objective functions depending on how this "overall" satisfaction is defined. The proposal builds upon the contribution presented at IWINAC'2017 Conference [61], where a hybrid algorithm for the FJSP was introduced. This preliminary work is enhanced in several ways. First, we reformulate the objective functions modelling overall due-date satisfaction in the context of multicriteria aggregation and decision making, shedding more light into their meaning and the different requirements they model. Second, we introduce the concept of solution robustness and we define new measures of robustness and surrogate robustness regarding due-date satisfaction are given, which will allow to assess the quality of the fuzzy schedule at the moment of its practical use. Third, we improve the hybrid algorithm in two different ways. We consider a more varied set of genetic operators and we follow a robust design method in the experiments to select the most competitive configuration, ensuring an adequate balance between intensification and exploration. Also, based on the original reformulation of the objective functions, we propose a novel heuristic technique to improve the algorithm by effectively guiding the search across plateaus in the objective space for one of the objective functions. Finally, the experimental evaluation is greatly extended with a parametric analysis, separate assessment of each component of the hybrid algorithm and the synergy between them, convergence analysis and solution robustness evaluation. Experiments are run on more challenging job shop instances by incorporating due dates to them and show that the enhanced algorithm not only outperforms the state-of-the-art, but also provides robust solutions and is more capable of navigating plateaus in the fitness landscape. As a result, we provide a new benchmark together with competitive results for due-date satisfaction which should serve as future reference for researchers working in this area.

1.1. Related work

A considerable effort has been made to extend classical scheduling models and algorithms in order to capture and deal with complex constraints and features present in real-life environments. Notable examples of this are scheduling problems considering resource setup times [5], presence of human operators [48], flexibility in machines [19] or features typical of construction projects such as precedence relationships, multiple crew-strategies, and time-cost trade-off [70].

The most frequent objectives related to due-date satisfaction in deterministic problems are maximum lateness and total weighted tardiness (cf. [14,63] and references therein). For the latter, exact and heuristic search methods exist in the literature [62,72] and the state-of-the-art is formed by different metaheuristics incorporating local search: the Genetic Local Search from [28] which combines a genetic algorithm with an iterated local search, the Hybrid Shifting Bottleneck with Tabu Search approach proposed in [17], the Hybrid Genetic Algorithm with Tabu Search from [31], and the Extended GRASP algorithm presented in [14].

Many of these successful methods belong to the class of memetic algorithms, whose structure is characterized by an evolutionary framework incorporating

local search components [49,50]. Indeed, memetic algorithms have proved successful in solving complex combinatorial optimisation problems, such as job shop scheduling problem, thanks to the synergy obtained from combining the global search diversification of the evolutionary framework with the intensification provided by the local search components. Examples of successful applications of evolutionary algorithms to scheduling and related complex optimisation problems abound. Among recent contributions, we find genetic algorithms for solving a multiobjective elevator scheduling problem [3] and a time-constrained project scheduling problem [41]. Distributed embodied evolution is used to obtain solutions in real-time to distributed engineering problems with collectively non-separable spaces such as the dynamic fleet size and mix vehicle routing problem with time windows [64] and a quantum inspired evolutionary algorithm is applied to real-world optimisation problems in [81]. Evolutionary strategies have also proved effective in solving complex multiobjetive optimisation problem [66, 78].

In a complementary approach to modelling real-world characteristics, fuzzy sets were first incorporated to scheduling in the turn of the 1980s [18]. Since then, fuzzy sets have been widely used in scheduling: using fuzzy priority rules or fuzzy decision making with linguistic qualifiers [65], modelling soft constraints and uncertain durations [22,46] or even as a means of improving solution robustness, a much-desired property in real-life applications [38,77]. In particular, fuzzy sets have been used to model gradual satisfaction of flexible due dates since the 1990s, sometimes in combination with uncertain processing times [23,33,68].

In [47] we find one of the first attempts of using fuzzy sets to model uncertain times in a job shop problem, while [23] also uses fuzzy sets to model flexible constraints. After these seminal papers, many contributions for different variants of fuzzy job shop have appeared in the literature, as can be seen in the recent reviews on fuzzy job shop [1], fuzzy shop scheduling [9] and benchmarks for fuzzy job shop [59]. The simulated annealing algorithm from [30] and the genetic algorithm from [69] constitute two landmarks in the application of metaheuristic search to FJSP. Currently, some of the most competitive methods for fuzzy makespan minimisation in job shop problems are the genetic algorithm from [44] and the memetic algorithm from [59].

For problems combining fuzzy durations and fuzzy due dates, the term *agreement index* was coined in [69] to refer to the degree to which a job's fuzzy completion time satisfies the flexible due-date, and a genetic algorithm was proposed to maximise the minimum agreement index across all jobs. Maximising the minimum agreement index is also the objective of a random-key genetic algorithm in [44], a scatter search method in [27], a hybrid discrete imperialist competition algorithm in [79] and a memetic algorithm in [61]. This memetic algorithm is also applied to maximise the average minimum index, which is also the objective of the co-evolutive method from [82], here for a fuzzy job shop with multi-process routes, and of the multiobjective genetic algorithm from [29], which also attempts to minimise the number of tardy jobs. Maximisation of average or minimum agreement index is also one of the objectives of several multiobjective approaches together with makespan minimisation: based on fuzzy decision making using genetic algorithms [33,68], based on lexicographical goal programming also with a genetic algorithm [32] or Pareto-front approximation using a genetic algorithm [54], Pareto archive particle swarm optimisation [43] or a memetic algorithm for multi-process routes [76]. For a classical benchmark from [68], the most competitive methods are the random-key genetic algorithm from [44] and the memetic algorithm from [61], while [32] reports minimum and average agreement index values for another set of instances. In fact, these methods obtain almost full due-date satisfaction, suggesting there is no room for improvement in this area. However, in [61] new results on harder instances make it clear that the problem of due-date satisfaction is far from being solved.

The remainder of the paper is organised as follows. Section 2 gives the necessary background on the fuzzy job shop problem under consideration. Section 3 provides a new insight into the two objective functions usually considered in the literature by situating them in the more general framework of fuzzy multicriteria aggregation and decision making and develops the concept of solution robustness regarding due dates and measures thereof. Then, Section 4 enhances the hybrid algorithm from [61], combining a genetic algorithm with local search, to solve the problem. Different genetic operators are considered and, based on the framework of fuzzy multicriteria aggregation, a new heuristic strategy to guide the search through plateaus in the fitness landscape is proposed for the case of the most demanding objective function. Finally, Section 5

presents a thorough experimental evaluation of the proposed method, introducing a new more challenging benchmark based on existing instances for makespan minimisation, and Section 6 presents the main conclusions and proposals for future work.

2. The Fuzzy Job Shop Problem

The classical *job shop scheduling problem*, also denoted *JSP*, consists of a set of jobs $J = \{J_1, \ldots, J_n\}$ that have to be processed on a set of physical resources or machines $M = \{M_1, \ldots, M_m\}$. Each job J_i consists of m_i tasks $\{\theta_{i1}, \ldots, \theta_{im_i}\}$ that must be sequentially executed in this order. Additionally, each task θ_{ij} must be processed on a specific machine from the set of available ones M exclusively and without preemption for its whole processing time $p_{\theta_{ij}}$. A *feasible schedule* is an allocation of starting times $st_{\theta_{ij}}$ for all tasks such that all precedence and resource constraints hold, that is:

$$st_{\theta_{ij}} + p_{\theta_{ij}} \leq st_{\theta_{i(j+1)}} \tag{1}$$

for $1 \leq j \leq m_i - 1, 1 \leq i \leq n$ and

$$st_{\theta_{ij}} + p_{\theta_{ij}} \leq st_{\theta_{kl}} \text{ or } st_{\theta_{kl}} + p_{\theta_{kl}} \leq st_{\theta_{ij}}, \tag{2}$$

for every pair of tasks θ_{ij} and θ_{kl} requiring the same machine.

Each job J_i may have a due date d_i by which it is desirable that the job be completed, so if $c_{\theta_{ij}} = st_{\theta_{ij}} + p_{\theta_{ij}}$ denotes the completion time of a task θ_{ij} according to a given schedule and $c_i = c_{\theta_{im_i}}$ denotes the job's completion time, $c_i \leq d_i$. In this case, a feasible solution is *optimal* if it obtains maximal due-date satisfaction.

2.1. Fuzzy Durations and Flexible Due Dates

In real-life applications, it is difficult, if not impossible, to foresee in advance the exact time it will take to process a task. It is reasonable however to have some knowledge (albeit uncertain) about the duration, possibly based on previous experience. The crudest representation of such uncertain knowledge would be a human-originated confidence interval and, if some values appear to be more plausible than others, then a natural extension is a fuzzy interval or fuzzy number. The simplest model is a *triangular fuzzy number* or *TFN*, denoted $\widehat{a} = (a^1, a^2, a^3)$, given by an interval $[a^1, a^3]$ of possible values and a modal value $a^2 \in [a^1, a^3]$, so its membership function takes the following triangular shape:

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

Triangular fuzzy numbers (or, more generally, fuzzy intervals) are widely used in scheduling as a model for uncertain processing times [1,22,59].

The job shop essentially requires two operations on fuzzy numbers, the sum and the maximum. These are usually defined by extending the corresponding operations on real numbers. The resulting addition is pretty straightforward, so for any pair of TFNs $\widehat{a}$ and $\widehat{b}$ we have:

$$\widehat{a} + \widehat{b} = (a^1 + b^1, a^2 + b^2, a^3 + b^3). \tag{4}$$

Unfortunately, computing the extended maximum is not that simple and the set of TFNs is not even closed under this operation. Hence, it is common in the fuzzy scheduling literature to approximate the maximum of two TFNs as

$$\max(\widehat{a}, \widehat{b}) \approx (\max\{a^1, b^1\}, \max\{a^2, b^2\}, \max\{a^3, b^3\}). \tag{5}$$

Besides its extended use, several arguments can be given in favour of this approximation (cf. [59]).

Fuzzy sets can also be used to model flexible due dates. Consider the case where there is a preferred delivery date d^1, but some delay may be allowed until a later date d^2. Satisfying the due date constraint thus becomes a matter of degree, our degree of satisfaction that the job is finished on a certain date. A fuzzy set $\widetilde{d} = (d^1, d^2)$ can be used to model such gradual satisfaction level with a decreasing membership function:

$$\mu_{\widetilde{d}}(x) = \begin{cases} 1 & : x \leq d^1 \\ \frac{x-d^2}{d^1-d^2} & : d^1 < x \leq d^2 \\ 0 & : d^2 < x \end{cases} \tag{6}$$

This expresses a flexible threshold "less than", representing the satisfaction level $sat(t) = \mu_{\widetilde{d}}(t)$ for the ending date t of the job [22].

When the job's completion time is no longer a real number t but a TFN $\widehat{c}$, the degree to which $\widehat{c}$ satisfies the due-date constraint $\widetilde{d}$, $\widehat{c} \leq \widetilde{d}$, may be measured using the *agreement index* [69]:

$$AI(\widehat{c}, \widetilde{d}) = \frac{area(\widetilde{d} \cap \widehat{c})}{area(\widehat{c})} \quad (7)$$

where $area(\widetilde{d} \cap \widehat{c})$ and $area(\widehat{c})$ denote the areas under the membership functions of $(\widetilde{d} \cap \widehat{c})$ and $\widehat{c}$ respectively. This essentially measures the degree to which $\widehat{c}$ is contained in $\widetilde{d}$ following the standard definition of degree of subsethood. $AI(\widehat{c}, \widetilde{d})$ ranges between 0, when the due date is not satisfied at all, and 1, when the due date is fully satisfied.

2.2. Fuzzy Schedules

To determine a solution for a fuzzy JSP, it is necessary to establish partial task processing orders on all machines. A schedule s (that is, the starting times of all tasks) may be easily computed based on these orders. For every task x with processing time $\widehat{p}_x$, let PM_x and SM_x denote the tasks preceding and succeeding x in the established machine sequence, and let PJ_x and SJ_x denote respectively the predecessor and successor tasks of x in the job sequence. The starting time $\widehat{st}_x$ of x is a TFN given by:

$$\widehat{st}_x = \max(\widehat{st}_{PJ_x} + \widehat{p}_{PJ_x}, \widehat{st}_{PM_x} + \widehat{p}_{PM_x}). \quad (8)$$

The resulting schedule s is fuzzy in the sense that the starting and completion times of all tasks are fuzzy intervals, interpreted as possibility distributions on the values that the times may take. However, notice that the task processing ordering on every machine that determines the schedule is deterministic; there is no uncertainty regarding the order in which tasks are to be processed.

Once a schedule s is built, the agreement index $AI_i(\widehat{c}_i, \widetilde{d}_i)$ as defined in (9), denoted $AI_i(s)$ or AI_i for short, measures to what degree is each job's flexible due date $\widetilde{d}_i$ satisfied in this schedule, $i = 1, \ldots, n$.

The overall value of due-date satisfaction for the schedule s can be obtained by aggregating the individual $AI_i(s)$ values for $i = 1, \ldots, n$. Two main approaches for aggregation can be found in the fuzzy scheduling literature the minimum agreement index AI_{min} and the average agreement index AI_{avg}

$$AI_{min} = \min_{i=1,\ldots,n} AI_i(s), \quad (9)$$

$$AI_{avg} = \frac{1}{n} \sum_{i=1,\ldots,n} AI_i(s). \quad (10)$$

The resulting job shop problem, with fuzzy processing times and fuzzy due dates, and where the objective is to maximise the aggregated agreement index AI_{agg} (where AI_{agg} can be AI_{avg} or AI_{min}) can be denoted $J|\widehat{p}_i, \widetilde{d}_i|AI_{agg}$ according to the three-field notation from [34].

3. Flexible Due-Date Satisfaction Under Uncertainty

3.1. Aggregated due-date satisfaction

Both AI_{min} and AI_{avg} are particular cases of fuzzy set aggregation operators. In general, these operators provide a mathematical setting for the integration of subjective categories represented by membership functions (the individual due-date satisfaction degrees in our case) and have proved useful in the field of multifactorial evaluation for the construction of multiple-criterion aggregation functions [21,25]. Remarkably, little or no connection has been made to date between AI_{min} or AI_{avg} and aggregation operators in the more general framework of fuzzy multicriteria decision making.

From the viewpoint of multifactorial evaluation, the flexible due dates $\widetilde{d}_i$ can be seen as fuzzy goals G_i, so for a given schedule s the agreement index $AI_i(s)$ represents the grade of compatibility $\mu_{G_i}(s)$ between s and the due-date goal. In this setting, the compatibility of the schedule with the overall objective of due-date satisfaction D can be approximated by some suitable aggregation of fuzzy sets G_i with membership functions $AI_i(s)$, $i = 1, \ldots, n$. To this end, there must exist a mapping $f_{agg} : [0,1]^n \to [0,1]$ such that, for every schedule s, $\mu_D(s) = f_{agg}(AI_1(s), \ldots, AI_n(s))$ provides the overall degree of due-date satisfaction. Additionally, this mapping should comply with the following requirements [25]:

$R1$: $f_{agg}(0, \ldots, 0) = 0$ and $f_{agg}(1, \ldots, 1) = 1$

$R2$: $\forall (r_i, t_i) \in [0,1]^2$, if $r_i \geq t_i, i = 1, \ldots, n$, then $f_{agg}(r_1, \ldots, r_n) \geq f_{agg}(t_1, \ldots, t_n)$.

Under these requirements, if it is possible to find a schedule that fully satisfies individual job due dates then at least one of these maximal schedules has full membership in D. If the aggregation mapping f_{agg} is strictly monotonic (as is the arithmetic means), then D contains only maximal elements. Actually, $R2$ expresses consistency with Pareto optimality.

In general, when an overall decision must be made, three fundamental attitudes in front of several goals $G_1, \ldots, G_n$ can be identified. The first attitude is the conjunctive one, requiring the simultaneous satisfaction of goals, so no compensation exists between goals and the global compatibility of a schedule with the overall objective is bounded from above by its compatibility with the least achieved goal. The second attitude is the disjunctive one and expresses the redundancy of goals, corresponding to the situation where it is desired that any criterion be satisfied regardless of the remaining individual criteria, so the best partial evaluation of goals is a lower bound of the compatibility of a schedule with the overall objective. Finally, there is the compromise attitude, expressing a tradeoff between goals, so the overall evaluation lies between extreme grades of partial compatibility. The minimum is a triangular norm (the only idempotent one) modelling intersection or conjunction of fuzzy sets, hence corresponding to the conjunctive attitude, and it constitutes the most traditional approach to aggregating fuzzy goals, as proposed by Bellman and Zadeh [10]. On the other hand, the arithmetic means corresponds to the compromise attitude, in line with the utility theory assumption that there is always a tradeoff between goals. It is bounded from below by the minimum and from above by the maximum.

In some cases, f_{agg} can be written as a convex combination of individual fuzzy goals

$$f_{agg}(AI_1, \ldots, AI_n) = \sum_{i=1}^{n} w_i AI_i(s), \qquad (11)$$

where $\sum_{i=1}^{n} w_i = 1$ and $0 \leq w_i \leq 1$ assesses the relative importance of satisfying the ith due date. Each weight w_i can be interpreted as the probability that G_i is the relevant goal, so f_{agg} is the probability of the fuzzy event "s achieves the relevant goals". In particular, the arithmetic means AI_{avg} is obtained with a uniform distribution on the relevance of due-date goals $w_i = 1/n$, while the minimum AI_{min} is obtained when $w_i = 1$ if $i = \arg\min AI_i(s)$ and $w_i = 0$ otherwise, representing that the only relevant goal is the least achieved.

Both AI_{min} and AI_{avg} can also be seen as two particular cases of Ordered Weighted Averaging aggregations or OWAs given by

$$f_{agg}(AI_1, \ldots, AI_n) = W_1 b_1 + \cdots + W_n b_n \quad (12)$$

where b_i is the ith largest element in the collection $AI_1, \ldots, AI_n$ and the weights are such that $W_i \in [0, 1]$ and $\sum_{i=1}^{n} W_i = 1$. AI_{min} is obtained when $W_n = 1$ and $W_i = 0$ for all $i \neq n$ while AI_{avg} is obtained with $W_i = 1/n$ for all $i = 1, \ldots, n$. A characteristic feature of this kind of aggregation is that weights are associated to a particular ordered position rather than a particular element. This corresponds to the underlying philosophy of equal importance of criteria, which translates into symmetry with respect to these criteria.

Depending on the weight values, OWAs can range from conjunctive attitude to the disjunctive one. AI_{min}, corresponding to the conjunctive case, provides a lower bound on any the aggregation using OWAs. Its degree of "andness", as defined in [83], is 1 and the dual degree of "orness" is 0, supporting its interpretation as the situation where there is no satisfaction until all the criteria are satisfied. Meanwhile, AI_{avg} has degrees of "orness" and "andness" equal to 0.5, being situated exactly halfway between the conjunctive and the disjunctive attitude. When interpreted in terms of degree of satisfaction obtained by the decision maker, it corresponds to a linear increase in the proportion of criteria that are satisfied. Finally, it is possible to measure the dispersion of the weight vector using entropy; for any weight vector such that $W_i = 1$ for some i (including the minimum), there is zero dispersion, while for the average, with $W_i = 1/n$, $i = 1, \ldots, n$, the dispersion reaches its maximum, $\ln n$. The dispersion can be related to the concept of Shannon information, so the more disperse the weight vector is, the more information about the individual criteria is used in the aggregation.

In summary, both AI_{min} and AI_{avg} provide measures of overall due-date satisfaction within the framework of fuzzy multicriteria decision making, but they model very different attitudes and present very different behaviours, with $AI_{min} \leq AI_{avg}$ in all cases.

3.2. Robust schedules

A fuzzy schedule does not provide exact starting times for each task. Instead, it gives a fuzzy interval of possible values for each starting time, provided

that tasks are executed in the order determined by the schedule. In fact, it is impossible to predict what the exact time-schedule will be, because it depends on the realisation of the task's durations, which is not known yet. This idea is the basis for a semantics for fuzzy schedules from [33] by which solutions to the fuzzy job shop should be understood as a-priori solutions, also called baseline or predictive schedules in the literature [35]. These solutions are found when the duration of tasks is not exactly known and the set of all possible scenarios must be taken into account. Only after tasks are executed according to the ordering provided by the fuzzy schedule is their real duration known and a real (executed) schedule is obtained, the a-posteriori solution with deterministic times. Clearly, it is desirable that a fuzzy solution yields reasonably good executed schedules at the moment of its practical use, in clear relation with the concept of schedule robustness.

Roughly speaking, a schedule is said to be *robust* if it minimises the effect of executional uncertainties on its primary performance measure [6]. This straightforward definition may, however, be subject to many different interpretations when it comes to specifying robustness measures [67].

For the problem at hand, uncertainties apply to task processing times, the performance measure is the overall due-date satisfaction and the effect of executional uncertainties is understood relative to all possible scenarios.

In the context of stochastic uncertainty, a *robust predictive schedule* is proposed as a predictive schedule such that the quality of the eventually executed one is close to the quality of the predictive schedule [12]. This yields a definition of ϵ-robustness which essentially gives an upper bound for the relative error made by the predictive schedule's performance with respect to the actual performance of the executed schedule. The definition of ϵ-robustness has already been adapted to different scheduling problems where uncertainty is represented with fuzzy sets and the performance was measured in terms of makespan [56–58]. It is thus tempting to extend this measure directly to the problem at hand. However, important differences between the performance measures should not be overlooked. While makespan values are always positive and very variable, flexible due-date satisfaction values may be null and are always in the interval $[0, 1]$. Therefore, in the context of makespan it makes more sense to evaluate the goodness of a prediction in terms of relative errors rather than absolute errors, but in the context of flexible due-date satisfaction it seems instead more sensible to simply consider absolute errors (which will always be bounded by 1).

Given the above, in this work a definition of robustness measure is proposed which, although keeping in line with the spirit of ϵ-robustness, is not a straightforward translation of this concept and instead takes into account the peculiarities of evaluating schedule performance in terms of flexible due-date satisfaction.

Definition 1. Let $sat_{agg} = f_{agg}(\mu_{\widetilde{d}_1}(c_1), \ldots, \mu_{\widetilde{d}_n}(c_n))$ denote the aggregated due-date satisfaction value of the eventually executed schedule, where c_i denotes the (deterministic) completion time of job $i = 1, \ldots, n$, in the executed schedule and the membership value $\mu_{\widetilde{d}_i}(c_i)$ gives the degree of satisfaction of the job's flexible due date $\widetilde{d}_i$ in that schedule. Then, the predictive schedule with (predictive) overall due-date satisfaction $AI_{agg} = f_{agg}(AI_1, \ldots, AI_n)$ is δ-*robust* for a given $\delta \in [0, 1]$ if it holds that:

$$|AI_{agg} - sat_{agg}| \leq \delta \tag{13}$$

That is, the error of the estimation made by the predictive schedule (i.e. the aggregated agreement index) regarding overall due-date satisfaction is bounded by δ.

The rationale behind this definition is that a predictive schedule is robust if the quality of the eventually executed schedule is close to the quality of the predictive schedule in any of the possible scenarios for task durations. Obviously, since δ provides an upper bound for the prediction error, the smaller δ is, the better.

It is worth noticing that the approach to robustness taken here is different from the better-known approach from combinatorial optimisation, based on min-max or min-max regret criteria, which aims at constructing solutions having the best possible performance in the worst case [4]. The study of such criteria is motivated by practical applications where an anticipation of the worst case is crucial and there already exist proposals to translate it to the fuzzy framework [38,75]. However, the min-max approach may be deemed as too conservative in some cases where the worst case is not that critical and an overall acceptable performance, with a solution that behaves "well" or "not too bad" in all the scenarios, is preferred [37]. It is in these situations where an approach such as the one proposed here might be more adequate. Also, more classical approaches measure robustness as a deviation from the optimal solution on each possible case, thus assuming that such optimal solution can indeed be found.

Unfortunately, this is a somewhat unrealistic assumption when dealing with complex problems such as job shop (even in its deterministic version). The approach adopted in this work takes the alternative stance of measuring deviations between the prediction and the real execution (be it optimal or not).

The above definition of δ-robustness requires a real execution of the problem which may not always be available. In fact, in the literature it is very common to use synthetic problems for which no real execution exists. In this case, a Monte-Carlo simulation can be used to provide a surrogate of the δ-robustness measure. Specifically, given a fuzzy instance, a sample of K possible realisations (also called scenarios) of that instance is generated by assigning an exact duration to each task, providing K deterministic instances where the δ-robustness of the solution can be evaluated.

Definition 2. For a given fuzzy job shop problem instance and K scenarios (deterministic instances) obtained after a Monte-Carlo simulation, let $sat_{agg,k}$ denote the overall due-date satisfaction obtained by executing tasks according to the ordering provided by a predictive schedule in the k-th scenario, $k = 1, \ldots, K$. Then, the *surrogate δ-robustness* of the predictive schedule, denoted $\overline{\delta}$, is calculated as:

$$\overline{\delta} = \frac{1}{K} \sum_{k=1}^{K} |AI_{agg} - sat_{agg,k}|. \quad (14)$$

A crucial factor in this method is the way in which task crisp durations are obtained. This is done by simulating crisp durations for tasks following a probability distribution that is consistent with the possibility distribution μ_A defined by each fuzzy duration A. Following [58], two alternative approaches are followed to obtain, from each TFN (a possibility distribution on the task durations) a probability distribution that can be used for the Monte-Carlo simulation.

The first approach consists in taking the uniform probability distribution that is bounded by the support of the TFN; the resulting simulation will be referred to as "Scenario I". This transformation is motivated by several results from the literature (see [8,24]) that justify the use of TFNs as fuzzy counterparts to uniform probability distributions and model-free approximations of probability distributions with bounded support.

The second method consists in taking the probability distribution obtained from each fuzzy duration after applying the pignistic transformation obtained by considering cuts as uniformly distributed probabilities [26]. This is the probability one would obtain from the membership function of a fuzzy duration applying a generalised version of the Insufficient Reason Principle by Laplace. This distribution is much more "focused" on the modal value, in the sense that it gives high probability to values close to the mode and very low probability to values at the extreme of the support interval. The simulation that results from this transformation will be referred to as "Scenario II"

4. A hybrid algorithm to maximise AI_{agg}

Memetic algorithms, a kind of hybrid algorithms combining genetic algorithms with local search methods, have proved to be very powerful in different optimisation problems [36,49]. The reason is their ability to integrate the intensification provided by the local search with the diversification provided by the population-based algorithm. In particular, some state-of-the-art methods for different variants of fuzzy job shop are hybrids of this kind [55,59]. This motivated the proposal in [61] of a memetic algorithm for AI_{agg} maximisation in FJSP, which combined a genetic component with local search. In the following, this promising memetic algorithm is enhanced by considering several genetic operators as well as incorporating a novel heuristic strategy to guide the search through plateaus in the fitness landscape when $AI_{agg} = AI_{min}$.

4.1. Genetic Component

For the genetic component of the algorithm, solutions are codified into chromosomes as permutations with repetitions [13]. Each permutation represents a feasible task processing order by identifying each operation θ_{ij} with j-th occurrence of index i in the permutation. For example, in a problem with three jobs and three machines, sequence (1,3,2,2,3,1,1,3,2) represents the task ordering $(\theta_{11}, \theta_{31}, \theta_{21}, \theta_{22}, \theta_{32}, \theta_{12}, \theta_{13}, \theta_{33}, \theta_{23})$. For fitness evaluation, chromosomes are decoded into schedules using an insertion schedule generation scheme, or SGS in short, as proposed in [60] and the resulting AI_{agg} is taken as fitness value.

The algorithm starts from a random population. It then iterates until $maxIter$ consecutive iterations pass without any improvement in the best solution found so far. At each iteration a new generation is built from the previous one by applying the genetic operators of selection, recombination and replacement.

In the selection phase all chromosomes are randomly paired, and then each pair is mated to obtain two offspring by applying crossover and mutation with a certain probability. Two individuals are then selected from each pair of parents and their two offspring to pass onto the next generation using tournament. If the two selected individuals have the same fitness value, the replacement strategy may discard one of them and select the next best individual instead, as a mechanism to preserve diversity.

For recombination, several classical operators are considered: Job-Order Crossover (JOX) [53], Generalised Order Crossover (GOX) [13] and Generalised Partially-Mapped Crossover (GPMX) [15] and Order-Based Mutation (OBM), Insertion Mutation (IM) and Simple Inversion Mutation (SIM) [42]. In GOX and GPMX a substring is randomly chosen from the first parent and then the offspring is generated by initially copying the second parent in the offspring and then deleting the genes in the substring taking into account the positions where they occur. GOX then inserts the substring at the position where the first gene of the substring had occurred (before deletion) in the second parent. Alternatively, GPMX inserts the substring at the position where it occurred in the first parent. JOX is more job oriented; it starts by randomly choosing the jobs whose loci are to be preserved, which are then copied from the first parent onto the offspring preserving their loci and the offspring is completed with the remaining jobs preserving their relative order in the second parent. In all cases, a second offspring may be obtained by reversing the parents' roles.

Regarding mutation operators, OBM (also known as Swap Mutation operator or Exchange Mutation operator) exchanges two positions of the chromosome at random, IM (also known as Position-Based Mutation) inserts a randomly chosen gene into another random position and SIM reverses a random substring in the chromosome.

4.2. Local Search Component

The local search component consists in a simple hill climbing procedure using one of the neighbourhood structures $\mathcal{N}_{AI_{avg}}$ or $\mathcal{N}_{AI_{min}}$ from [61], depending on the objective function considered. This results in quite a fast local search procedure that is applied with probability p_L to every individual that is being evaluated by the genetic algorithm and unconditionally to the best individual in the population (introducing elistism), unless $p_L = 0$, in which case no local search is applied.

Require: A *FJSP* instance
Ensure: A schedule
  Generate random population P with size $popSize$
  Evaluate all $p_i \in P$ using an insertion SGS
  if $p_L > 0$ **then**
    Apply Local Search to best individual $p_{best} \in P$
    Update genotype of p_{best} (lamarckism)
  for $i = 1, \ldots, popSize$ **do**
    Apply Local Search to p_i with probability p_L
    Update genotype of p_i (lamarckism)
  $best \leftarrow$ Best solution $p_{best} \in P$
  $iter \leftarrow 0$
  while $iter < maxIter$ **do**
    Generate P' by applying Selection operator
    $i \leftarrow 1$
    for $i < popSize$ **do**
      Get $rand \sim U(0, 1)$
      if $rand < p_{cross}$ **then**
        $(off_1, off_2) \leftarrow$ Apply crossover to (p'_i, p'_{i+1})
      else
        $(off_1, off_2) \leftarrow (p'_i, p'_{i+1})$
      Apply mutation to off_1 with probability p_{mutate}
      Apply mutation to off_2 with probability p_{mutate}
      Evaluate off_1 and off_2 using an insertion SGS;
      Apply Local Search to off_1 with probability p_L
      Update genotype of off_1 (lamarckism)
      Apply Local Search to off_2 with probability p_L
      Update genotype of off_2 (lamarckism)
      $(p'_i, p'_{i+1}) \leftarrow$ Apply replacement operator in $(p'_i, p'_{i+1}, off_1, off_2)$
      $i \leftarrow i + 2$
    if $p_L > 0$ **then**
      Apply Local Search to best individual $p'_{best} \in P'$
      Update genotype of p'_{best} (lamarckism)
    if Best solution $p'_{best} \in P'$ is better than $best$ **then**
      $best \leftarrow p'_{best} \in P'$
      $iter \leftarrow 0$
    else
      $iter \leftarrow iter + 1$
    $P \leftarrow P'$
  return The schedule obtained from $best$ using an insertion SGS

Algorithm 1: The Memetic Algorithm

The pseudocode for the resulting memetic algorithm is given in Algorithm 1

4.3. Heuristic tie-breaking for AI_{min}

The experimental results using AI_{min} as fitness value reported in [58] show that plateaus of constant zero fitness are very frequent in hard problem instances, making it very difficult to guide the algorithm to promising areas of the search space. This suggests

that the memetic algorithm needs to be improved to efficiently traverse large neutral plateaus caused by the conjunctive nature of the objective function described in Section 3.1, thus addressing one of the design issues for competent memetic algorithms [39].

As highlighted in [74], a possible strategy to break the plateaus of a landscape and guide the search toward better regions consists in having a secondary criterion that allows to discriminate individuals that have the same value for the fitness function and is correlated with the main objective of the problem.

This approach is taken in [20] to avoid misleading the search to local optima in deceptive problems: a measure of the "potential" of individuals is proposed as an additional criterion for selecting them for reproduction. Similarly, in [11] a secondary measure of the "improvability" of individuals is used to break ties in tournament selection in order to solve bin packing problems, where equal integer fitness values often occur. A secondary fitness is also used to break ties in tournament selection in [45] and [80] in the context of genetic programming to guide the search through fitness plateaus.

Building on these works, in the following it is proposed that the MA optimising AI_{min} incorporates a tie-breaking mechanism to guide the search through large neutral plateaus. Notice however that the selective pressure of the memetic algorithm presented herein resides not in the selection operator, but in the replacement one. Therefore, ties will be broken in the replacement phase by considering additional measures of the overall due-date satisfaction provided by the solutions.

For any schedule s let $f^i_{agg}(AI_1(s), \ldots, AI_n(s))$, or $f^i_{agg}(s)$ in short, denote the OWA aggregation given by $W_{n-i+1} = 1$ and $W_j = 0$ for all $j \neq n - i + 1$, $i = 1, \ldots, n$. f^i_{agg} gives full weight to the i-th smallest AI value, corresponding to the case where the relevant due date is the i-th least satisfied and no satisfaction is obtained until all due dates except the $i - 1$ worst ones are satisfied. Clearly, $f^1_{agg}(s) = AI_{min}(s)$ and, for every possible value of i, the dispersion of the weighting vector is zero, so the same information about individual criteria is used in all cases. Then, to break ties in tournament between two chromosomes representing two schedules, the two corresponding vectors of aggregated values $(f^1_{agg}, f^2_{agg}, \ldots, f^n_{agg})$ are compared in a lexicographic way.

It is important to notice that the procedure does not assign a new fitness value, but instead uses a modified tournament replacement operator to consider additional criteria in order to break ties and guide the search. The rationale behind this strategy is that, when two solutions are equally bad with respect to the due date that is satisfied the least, the solution achieving more satisfaction in the second-worst due date might be preferred. If ties persist, i.e., solutions are still indistinguishable from the point of view of the next-to-last due-date satisfaction, then the schedule with better performance in the third-worst due date is preferred, and so on. Equivalently, if two schedules are equally bad regarding the least-satisfied due date (or i least satisfied due dates), then they may be compared using the minimum aggregator AI_{min} for the reduced vector of AI values obtained after removing the smallest one (or i smallest ones). Thus, the modified tournament replacement compares solutions following the tie-breaking method in Algorithm 2.

Require: Two solutions s_1, s_2 for the FJSP
Ensure: The preferred solution
  $A \leftarrow (AI_1(s_1), AI_2(s_1), \ldots, AI_n(s_1))$
  $B \leftarrow (AI_1(s_2), AI_2(s_2), \ldots, AI_n(s_2))$
  Sort A in increasing order
  Sort B in increasing order
  for $i = 1, \ldots, n$ **do**
    if $A_i > B_i$ **then**
      return s_1
    else
      if $B_i > A_i$ **then**
        return s_2
  return s_1 *//Both solutions are equal*

Algorithm 2: Heuristic tie-breaking

Alternative strategies to handle plateaus and mostly flat fitness landscapes can be found in the literature. For instance, in [51] and [52] we find two complex evolutionary algorithms that avoid getting trapped in a local optimum based on measuring the population diversity and, based on this, dinamically adjust the algorithm parameters and activate/deactivate the use of local search strategies to guide the search. For the experimental evaluation, we have adapted the Adaptive Multimeme Algorithm (AMmA) from [51] to our problem. We have plugged the genetic operators and the only local search from our memetic algorithm into AMmA while keeping the variable population size and crossover and mutation probabilities as well as the application of the local search dependent on the diversity of the population. We consider the two different diversity measures extending them so they are zero in the case where all the individuals in the population have null fitness (and the original expressions of diversity

yield an indeterminate). This results in two different versions of AMmA adapted to our problem, denoted AMmA_1 and AMmA_2, using respectively the diversity measures from [51] and [52].

5. Experimental Evaluation

The memetic algorithm or the MA proposed in Section 4 builds on the memetic algorithm from [61], by enhancing it with a variety of genetic operators, allowing for different intensities in the hybridisation with local search and incorporating a heuristic mechanism to navigate through plateaus when the objective is to maximise AI_{min}. In consequence, the experimental evaluation of the MA presented herein also builds on the results reported in [61] and is based on three premises.

First, the preliminary version of the MA from [61] already compared favourably with the methods which, up to that moment, conformed the state of the art in the literature on fuzzy job shop with flexible due dates. Hence, no further comparison with these already outperformed methods is needed.

Second, among the FJSP instances most widely used in the literature up to now, (S6.1-3 and S10.1-3 from [68], S6.4 and S10.4 from [69] and Lei01 and Lei02 from [44]) only the two largest ones, Lei01 and Lei02, seem to offer enough room for improvement to continue serving as benchmarks. Additionally, instances La21, La24, La25, La27, La38, La40, ABZ7, ABZ8 and ABZ9, obtained from classical JSP instances by fuzzifying processing times and adding due dates as explained in [54] have proved hard to solve. Therefore, it seems reasonable to start the empirical evaluation of the MA on the eleven challenging instances.

Finally, the results obtained by the preliminary version of the MA suggest that the due dates generated for the fuzzified instances are, in general, very tight or strict, in the sense that they are difficult to satisfy. This is likely to be related to the method used to generate due-date values, based on a lower bound of the expected job's completion time obtained by relaxing resource constraints. Depending on the ratio between the number of jobs and the number of machines, it may result in a loose lower bound and, in consequence, the due date turns out to be excessively strict or even impossible to satisfy. This, together with the conjunctive behaviour of AI_{min}, results in problem instances for which it is very unlikely to obtain any overall due-date satisfaction in terms of AI_{min}. For this reason, it seems reasonable to first evaluate MA's performance using AI_{avg} (which allows for compensation among individual due-date satisfaction degrees) and, afterwards, tackle the particularly difficult case of using AI_{min}.

All results reported in the sequel correspond to a C++ implementation of the MA running on a PC with Xeon processor at 2,2Ghz and 24 Gb RAM with Linux (SL 6.0.1).

5.1. Parametric and convergence analysis

In a first set of experiments, a parametric analysis is conducted to find the best configuration of operators and parameters for the MA. Specifically, the three crossover and mutation operators described in Section 4.1 are considered, together with typical values for crossover and mutation probability, and two different replacement strategies as follows:

- Replacement strategy: allow repeated individuals (AR) or not (NR)
- Crossover operator: JOX, GOX, GPMX
- Crossover probability: 0.8, 0.9, 1.0 .
- Mutation operator: IM, OBM, SIM
- Mutation probability: 0.05, 0.10, 0.20

Population size is taken to be 100, the local search probability p_L is 1 and the stopping criterion is set to $maxIter = 25$ iterations without any improvement.

Perhaps two of the most extended approaches to parameter tuning are a sequential strategy or an experimental design approach, such as the Taguchi method [74]. Arguments can be given in favour and against both of them. The sequential strategy consists in tuning one parameter at a time, starting from a base set-up and determining each optimal parameter value empirically. This method is widely used, but is heavily dependent on the order in which parameters are tuned. The Taguchi or robust design method sorts the parameters according to their relevance in the final performance of the algorithm and at the same time gives the "best" parameter configuration based on statistical principles but it is also subject to criticisms. These two methods are somehow complementary: the Taguchi method can provide the base set-up for the sequential strategy as well as an order for the sequential parameter tuning.

For the Taguchi method, given that there are four parameters with 3 different levels and another parameter with two levels, the orthogonal array L18 is used. The

Table 1
Delta values and set-up obtained after the parametric study

Parameter	Delta value	Base set-up	Final set-up
Replacement	0.136	AR	NR
Crossover	1.077	JOX	JOX
Crossover Prob.	0.101	0.80	0.90
Mutation	0.115	OBM	OBM
Mutation Prob.	0.082	0.10	0.10

obtained "Delta values" in the second column of Table 1 indicate that the most relevant parameter for the algorithm is the crossover operator, followed by the replacement strategy, the mutation operator and finally the crossover and mutation probabilities. The "Base set-up" column shows the configuration that results from the Taguchi method. Starting from this configuration, parameters are then sequentially tuned in the order given by the Taguchi analysis, yielding the final configuration in the last column of the table.

The performance of a memetic algorithm can also be affected by the local search, since this component may in some cases provide excessive intensification, negatively affecting the diversity in the population and leading to premature convergence. In order to fix a value for the probability p_L of applying local search, a series of convergence studies are carried out. With the configuration that results from the parameter tuning process, the pressure of the local search in the MA is varied with the probability p_L of applying local search taking values in $\{0, 0.25, 0.5, 0.75, 1\}$. Figure 1 shows the results obtained on instance La21, which are representative of the overall behaviour of the hybrid algorithm across all instances.

The main difference in convergence curves lies in having or not local search. For the cases where local search is applied, regardless of the probability, fitness values and convergence patterns are quite similar, with slightly better fitness values when local search is applied to every individual in the population ($p_L = 1$). This indicates that local search does not produce excessive intensification nor causes premature convergence when applied with any probability. Therefore, the remaining experimental results will be conducted with $p_L = 1$, which ensures the maximum level of intensification without leading the search to local optima.

Finally, in order to assess the behaviour of the MA with different population sizes, another convergence study is carried out by running the algorithm with varying population sizes of 100, 200 and 500 with a time limit of 300 seconds, which is more than twice the time it took to converge in the previous experiments. Figure 2 shows the average AI_{avg} fitness evolution on instance La21 for all population sizes, with the algorithm presenting a very similar behaviour in the remaining instances. Only the first 125 seconds of runtime are plotted in the figure, so the first iterations can be better appreciated and because there is barely any evolution after that point in all cases. As could be expected, smaller populations converge faster than larger ones, since they contribute with less diversity to balance the intensification effect of the local search. However, this does not translate into premature convergence, with very similar average fitness values for all population sizes once convergence is attained. There seems to be no gain in having larger populations (with the consequent increase in computational cost, especially if local search is to be applied to a big portion of the population), so population size is set to 100.

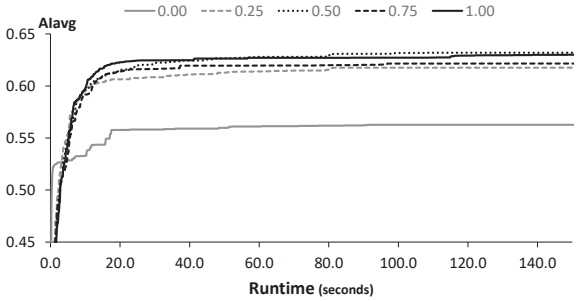

Fig. 1. Evolution of the MA with different local search pressure on instance La21

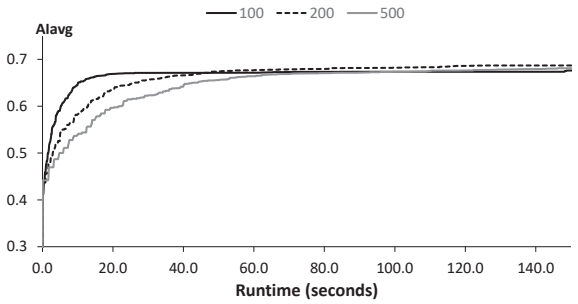

Fig. 2. Evolution of the MA with different population sizes on instance La21

5.2. Performance of the MA

Once a configuration has been decided for the MA, its performance can be assessed on the same set of

Table 2
Results obtained by the MA maximising AI_{avg}

Instance	Best AI_{avg}	Average AI_{avg}	Runtime (s)
ABZ7	0.661	0.645 (0.010)	137.0
ABZ8	0.688	0.661 (0.015)	124.6
ABZ9	0.709	0.666 (0.021)	167.2
La21	0.650	0.619 (0.013)	23.9
La24	0.685	0.643 (0.018)	22.7
La25	0.673	0.654 (0.010)	22.3
La27	0.510	0.469 (0.022)	71.3
La29	0.547	0.516 (0.023)	62.2
La38	0.845	0.830 (0.012)	50.4
La40	0.875	0.866 (0.012)	58.9
Lei01	1.000	0.999 (0.000)	21.9
Lei02	1.000	0.999 (0.000)	19.1

Table 3
Surrogate $\overline{\delta}$-robustness for AI_{avg}

Instance	Scenario I	Scenario II
ABZ7	0.1100 (0.0132)	0.1040 (0.0128)
ABZ8	0.0893 (0.0189)	0.0848 (0.0183)
ABZ9	0.1097 (0.0165)	0.1031 (0.0154)
La21	0.0917 (0.0158)	0.0925 (0.0160)
La24	0.0998 (0.0250)	0.1002 (0.0256)
La25	0.1109 (0.0124)	0.1111 (0.0125)
La27	0.0555 (0.0155)	0.0566 (0.0162)
La29	0.0736 (0.0176)	0.0734 (0.0176)
La38	0.1085 (0.0098)	0.1063 (0.0090)
La40	0.1176 (0.0166)	0.1169 (0.0154)
Lei01	0.0011 (0.0006)	0.0009 (0.0005)
Lei02	0.0095 (0.0023)	0.0012 (0.0005)

instances. A summary of the results is shown in Table 2, where each row corresponds to a problem instance. The second and third columns contain the best and average values of AI_{avg} obtained across 30 runs of the algorithm. Standard deviation values are also given between brackets next to the average fitness. The last column reports average runtime in seconds. It can be seen that that for instances Lei01 and Lei02 the MA is on average very close to full overall due-date satisfaction and the best solution is actually optimal. For the remaining instances, the optimum is unknown, making it impossible to know if it has been reached or how far are the solutions from it. However, overall due date satisfaction is always greater than 0.5, suggesting that the algorithm provides good solutions where due dates are satisfied to an acceptable extent. Also, small standard deviation values indicate that the algorithm is stable in the sense that it always finds solutions of similar quality.

It can be appreciated that the degree of overall due-date satisfaction varies significantly across the different instances, ranging from more than 0.82 on instances La38 and La40 to values in the vicinity of 0.5 on instances La27 and La29. This might be related to the method used in [54] to generate due-date values and with the shape of the instances as explained above. For instances La27 and La29, with twice as many jobs as tasks per job, due dates are very tight and there is little room to schedule tasks in every job in such a way that precedence and resource constraints hold and due dates are met. On the other hand, for "square" instances La38 and La40, with the same number of jobs and tasks per job, due dates turn out less strict and tasks can be scheduled so as to satisfy due dates to a large extent.

Further evaluation can be provided in terms of solution robustness. For each problem instance, the 30 runs of the MA provide a total of 30 solutions or predictive schedules. For each of these predictive schedules the surrogate $\overline{\delta}$-robustness is computed based on $K = 1000$ simulated realisations, both for Scenario I and Scenario II as explained in Section 3.2. Table 3 shows, for each problem instance the average value of the $\overline{\delta}$-robustness across the 30 schedules together with the standard deviation (in brackets). For approximately half of the instances, the error of the predicted due-date satisfaction with respect to the satisfaction obtained after execution is below 0.1 and in all cases it does not go beyond 0.12. That is, the estimate of due-date satisfaction provided by the fuzzy schedule is quite close to the real one. This suggests that the fuzzy schedules provided by the the MA are quite reliable as predictive solutions.

A final set of experiments is conducted to assess to what extent does each component of the MA contribute to its overall performance. It also allows to check if there is an expected synergy effect between the intensification provided by the local search (LS) and the diversification provided by the genetic algorithm (GA). To this end, the GA and LS are run independently. For a fairer comparison, LS is run as a multi-start local search with as many restarts as the average number of evaluations performed by the MA on each instance. Analogously, the GA is run with the same configuration as the MA except for the stop-

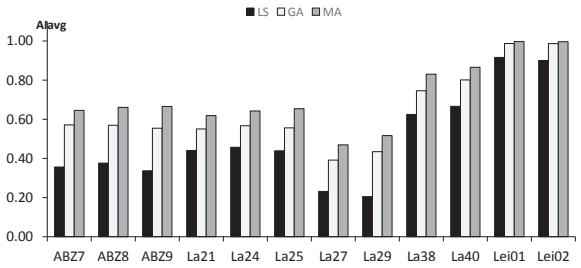

Fig. 3. Performance of the different components of MA with respect to AI_{avg}.

ping criterion, which is changed to leaving GA run for as long as MA takes to converge. The comparison is shown in Figure 3. The multi-start LS starting from random solutions obtains the worst results, not only in performance, but also in runtime which is 62% longer than for MA. GA, although performing much better than LS, is not as good as MA. Indeed, a synergy effect can be appreciated when combining both strategies, with MA obtaining much better results in the same running time than GA. This shows that MA benefits from the interaction between the exploration of GA and the intensification of LS.

5.3. A more challenging benchmark

The review of existing benchmarks for fuzzy job shop in [59] concludes that most of the existing benchmark instances are not especially hard in terms of makespan minimisation and proposes a new more challenging benchmark based on the well-known Ta benchmark for the classical JSP from [73] containing 20 instances with as many tasks per job as the number of machines, $n_i = m$ for $i = 1, \ldots, n$: Ta21-Ta30, with $n = 20$ jobs, and $m = 20$ machines, and Ta41-50, with $n = 30$ jobs and $m = 20$ machines. The fuzzy version is obtained after transforming the original task processing times into TFNs; however, no due dates are given either for the original instances, or for their fuzzy versions. In the following, we enlarge the benchmark by fuzzifying the whole Ta benchmark composed of 80 instances, each with a number of tasks varying from 225 ($n = 15, m = 15$) to 2000 ($n = 100$, $m = 20$). Out of these, instancesTa01-50 are considered to be harder than the remaining ones since the optimal solution has so far been found only for 17 of those 50 instances. In particular, instances Ta21-30 and Ta41-50 are considered to be "the most difficult JSP benchmark problems" [85] and to date the best-known solutions have not been proved to be optimal yet. Flexible due dates are incorporated to each instance, following a method that generates the due-date values from a lower bound of each job's expected completion time, as explained in [54]. The result is a new challenging benchmark for fuzzy job shop with flexible due dates.

Table 4

Results obtained by MA maximising AI_{avg} on the new benchmark instances Ta21-50

Instance	Best AI_{avg}	Average AI_{avg}	Runtime (s)
Ta21	0.790	0.751 (0.023)	356.4
Ta22	0.824	0.797 (0.012)	313.9
Ta23	0.853	0.827 (0.021)	347.5
Ta24	0.832	0.818 (0.008)	228.5
Ta25	0.802	0.749 (0.026)	329.5
Ta26	0.817	0.775 (0.021)	311.1
Ta27	0.834	0.811 (0.017)	307.8
Ta28	0.838	0.815 (0.014)	354.2
Ta29	0.826	0.803 (0.013)	284.8
Ta30	0.849	0.820 (0.014)	310.0
Ta41	0.528	0.495 (0.022)	1442.5
Ta42	0.540	0.502 (0.020)	1331.3
Ta43	0.520	0.475 (0.023)	1496.6
Ta44	0.508	0.474 (0.023)	1414.6
Ta45	0.507	0.473 (0.021)	1363.7
Ta46	0.490	0.457 (0.029)	1439.1
Ta47	0.496	0.464 (0.032)	1512.3
Ta48	0.539	0.488 (0.039)	1231.0
Ta49	0.481	0.435 (0.027)	1239.5
Ta50	0.508	0.453 (0.028)	1344.7

Table 4 reports the results obtained by MA on the 20 most challenging instances of this new benchmark when optimising $AI_{agg} = AI_{avg}$. The values in the last column, corresponding to the time taken by the algorithm to converge (up to 1512 seconds for Ta47) are indicative of the instances difficulty.

As was the case with the previous benchmark, the method used to generate due dates results in very tight due dates for those problem instances where the number of jobs is larger than the number of tasks in each job (Ta41-50), making it very difficult to achieve high levels of due date satisfaction. This explains the fact that for instances Ta21-30 the average overall due-date satisfaction degree is always between 0.74 and 0.83, while for instances Ta41-50 it is never greater than 0.51.

Table 5

Surrogate $\overline{\delta}$-robustness obtained for AI_{avg} on the new benchmark instances Ta21-50

Instance	Scenario I	Scenario II
Tai21	0.1150 (0.0211)	0.1154 (0.0210)
Tai22	0.1133 (0.0209)	0.1129 (0.0212)
Tai23	0.1170 (0.0160)	0.1149 (0.0163)
Tai24	0.0947 (0.0196)	0.0938 (0.0195)
Tai25	0.1221 (0.0160)	0.1221 (0.0156)
Tai26	0.1179 (0.0191)	0.1166 (0.0187)
Tai27	0.1405 (0.0189)	0.1383 (0.0192)
Tai28	0.1379 (0.0146)	0.1394 (0.0140)
Tai29	0.1407 (0.0214)	0.1390 (0.0224)
Tai30	0.1140 (0.0123)	0.1136 (0.0122)
Tai41	0.0910 (0.0103)	0.0900 (0.0102)
Tai42	0.0739 (0.0102)	0.0721 (0.0101)
Tai43	0.0760 (0.0124)	0.0741 (0.0124)
Tai44	0.0778 (0.0141)	0.0761 (0.0142)
Tai45	0.0709 (0.0143)	0.0704 (0.0145)
Tai46	0.0659 (0.0116)	0.0647 (0.0118)
Tai47	0.0829 (0.0143)	0.0825 (0.0143)
Tai48	0.0815 (0.0160)	0.0806 (0.0163)
Tai49	0.0673 (0.0106)	0.0666 (0.0110)
Tai50	0.0768 (0.0084)	0.0749 (0.0085)

Additional assessment is provided by computing surrogate δ-robustness values for $K = 1000$ simulated realisations. The results, in Table 5, are in line those for the previous instances. The error made by the fuzzy schedules when predicting the degree of due-date satisfaction of executed schedules is low, making the solutions provided by the MA fairly reliable.

Figure 4 illustrates the relationship between the obtained results and the structure of the problem instances. It depicts the average AI_{avg} value obtained on the 80 Ta instances grouped by size and ordered in ascending order according to the ratio n/m between the number of jobs and the number of machines. The actual ratio value can be seen above each bar. We can see how the tightness of the due dates increases with the ratio and, in consequence, the value of AI_{avg} decreases, becoming nearly null for problems of size $n = 100$ and $m = 20$. Figure 5 shows running times for the algorithm depending on the problem size $n \times m$. The labels indicate where does each group of problems lie. We can see a polynomial increase in CPU time with respect to the problem size. We have left out the group Ta71-80 which, despite being the group of

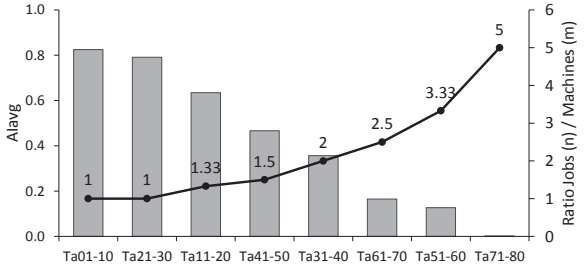

Fig. 4. Average AI_{avg} values for Ta instances grouped by size.

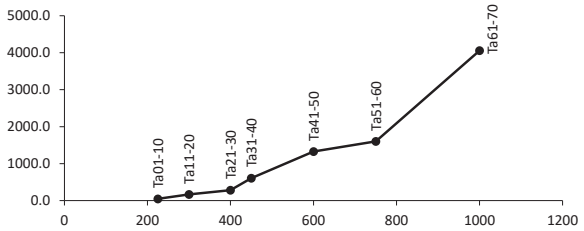

Fig. 5. Average runtime depending on the instance size.

largest instances (2000 tasks), yields a runtime of approximately 520 seconds (similar to problems of size 450). This is due to the stopping criterion of 25 iterations without improvement: being very complex, the algorithm has a quick convergence, even if it is to bad values, as shown in Figure 4. More detailed results are given in the Appendix A.

5.4. Maximising AI_{min}

Finally, we obtain results of MA incorporating the tie-breaking heuristic to maximise $AI_{agg} = AI_{min}$, denoted tbMA hereafter, compared with AMmA_1 and AMmA_2.

First, the challenging problem instances from [61] are considered. These instances can be classified into three groups. A first group of "easy" instances includes Lei01 and Lei02, where all three algorithms obtain very good solutions. Almost full due-date satisfaction is achieved on average and the best solution of MA for Lei02 is actually optimum. There is no significant difference between using MA or tbMA, nor between tbMA and any of AMmA_1 and AMmA_2. In fact, since due dates are not that strict, there is no null plateau in the fitness landscape that calls for the use of such mechanism. There is a second group of extremely hard instances, ABZ7–9 and La21–29, where none of the algorithms can find any solution with positive AI_{min} value regardless of whether the tie-breaking or diversity preserving mechanism is used in tbMA and

AMmA$_1$ or AMmA$_1$ respectively. As already argued, the due dates generated for these instances seem to be extremely tight and it may even be the case that at least one of them cannot be satisfied by any feasible solution, which, combined with the conjunctive nature of the AI_{min} aggregator, makes the resulting optimisation problem very difficult, if not impossible, to solve. There is however a third, more interesting group, with the "square" instances La38 and La40. The results obtained with AI_{avg} already suggest that here due dates are not unreasonably tight but MA without the tie-breaking mechanism cannot find any solution with $AI_{min} \neq 0$ for La38, nor can either AMmA$_1$ or AMmA$_2$. Interestingly, the heuristic tie-breaking mechanism boosts the performance of MAtb from 0 due-date satisfaction to 0.370 on average and 0.432 in the best case. For La40 results are somehow better, as shown by the boxplots in Figure 6 corresponding to the AI_{avg} values obtained by MA, AMmA$_1$, AMmA$_2$ and tbMA in 30 runs of each algorithm. MA obtains an average AI_{min} value of 0.494, but the standard deviation is 0.2 and the worst solution in the 30 runs actually yields a null AI_{min}. The same occurs with AMmA$_1$ and AMmA$_2$. However, when the tie-breaking mechanism is incorporated, tbMA is always able to find solutions with positive overall due-date satisfaction. Furthermore, AI_{min} values improve 19.2% on average, with standard deviation dropping to 0.022. Therefore, on these instances the tie-breaking mechanism not only improves MA's performance, but also adds to its stability. Additionaly, since the boxplots overlap, statistical tests have been conducted for the 30 runs of each method for pairwise comparisons. Since data do not correspond to a normal distribution and samples are unpaired, we have used a Wilcoxon rank-sum-test with significance level 0.05. The results indicate that there are significant differences between MA and AMmA$_1$ and MA and AMmA$_2$ in favour of MA; no significant difference exists between AMmA$_1$ and MA and AMmA$_2$ and tbMA is significantly better than the three other methods.

A similar behaviour can be seen on the more challenging benchmark Ta. In the set of "rectangular" instances Ta41–50, with more jobs than tasks per job, AI_{min} values are always null. This, together with worse overall due date satisfaction values when AI_{avg} is considered in Table 4 suggest (as was the case with the previous benchmark instances) that the generated due dates are excessively strict to obtain any overall due-date satisfaction with a conjunctive aggregator such as AI_{min}. On the other hand, for the set

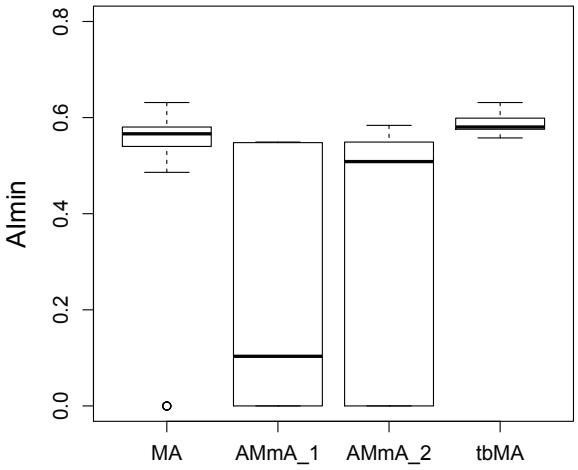

Fig. 6. AI_{min} values obtained on instance La40

of "square" instances Ta21–30 there is a clear benefit from using the tie-breaking heuristic to guide the search. When this mechanism is not used, MA is unable to find any solution with $AI_{min} > 0$ in any of the instances. However, when it is incorporated to MA, AI_{min} values are clearly improved in all instances as shown in Table 6.

Besides Ta21–30, the new benchmark only has another set of square instances, Ta01-10. These are the only instances where some solution is found without null AI_{min}; the due dates prove too tight in the remaining rectangular instances. Figure 7 corresponds to the results on this square set for the four methods: the bars correspond to average AI_{min} values with a line going up to the best found value. The benefit of using the tie-breaking heuristic is clear: not only does tbMA outperform MA as well as AMmA$_1$ and AMmA$_2$, it also provides greater stability, with the shortest distance between the average and best AI_{min} values. This contrasts with the variability of the other methods, especially on instance Ta04 for all three methods and on instance Ta02 for MA. Notice as well that the other three methods only obtain AI_{min} values greater than 0.1 on Ta03, Ta08 (where all methods reach the same solution in the best case) and Ta10.

Out of completeness, we have conducted another set of statistical tests for pairwise comparisons for all four methods on Ta01-10. Again we have discarded normality and run Wilcoxon tests with significance level 0.05. These show that there are significant differences between MA and both AMmA$_1$ and AMmA$_2$ in favour of these on instances Ta03 and Ta08. No significant

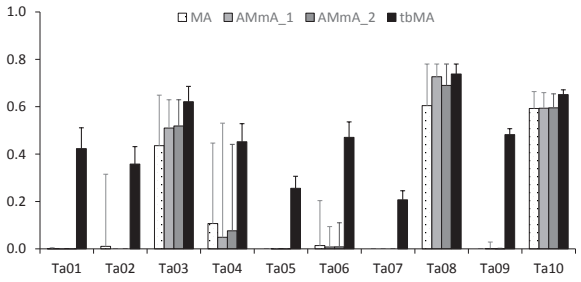

Fig. 7. Average and best AI_{min} values on instances Ta01–10

Table 6

Results obtained by MA maximising AI_{min} on the new square benchmark instances Ta21-30

Instance	Best AI_{min}	Average AI_{min}	Runtime (s)
Ta21	0.478	0.139 (0.180)	48.7
Ta22	0.568	0.485 (0.050)	87.1
Ta23	0.694	0.607 (0.093)	89.7
Ta24	0.450	0.332 (0.138)	75.9
Ta25	0.360	0.188 (0.148)	89.6
Ta26	0.599	0.524 (0.045)	99.1
Ta27	0.656	0.607 (0.023)	83.3
Ta28	0.602	0.553 (0.039)	85.0
Ta29	0.631	0.547 (0.043)	75.3
Ta30	0.662	0.563 (0.081)	90.0

difference exists between both AMmA_1 and AMmA_2, while btMA is always significantly better.

To better understand the difference in behaviour between the heuristic tie-breading method and the alternative strategy to escape plateaus from AMmA_1 and AMmA_2, we have generated 10000 random solutions for the ABZ and La benchmarks. After measuring the Kendall tau rank distance, adapted to permutations with repetitions, to ensure that they correspond to distant areas of the search space, we conclude that it is not possible to find promising areas where search can be intensified. For this reason, the AMmA method, with good behaviour in other problems, is not appropriate for the fuzzy job shop problem with AI_{min} maximisation at hand. On the other hand, the tie-breaking strategy motivated by the interpretation of AI_{min} in the framework of fuzzy multicriteria decision making has helped to find an alternative means of navigating almost flat landscapes.

6. Conclusions

The fuzzy job shop scheduling problem or FJSP, is a variant of the job shop problem with uncertain durations and flexible due dates modelled as fuzzy numbers and fuzzy thresholds. It is an interesting problem with the potential of narrowing the gap between scheduling theory and practical applications in different fields of engineering. Its complexity calls for the use of metaheuristic search techniques which provide good solutions in a reasonable time.

Here, the concept of solution robustness for FJSP has been introduced and a new measure of robustness has been defined. Additionally, the two most common objective functions from the literature have been placed in a more general framework of fuzzy multicriteria decision making, shedding more light into their different behaviours. This has later been used to devise a heuristic mechanism to guide the solving method through flat plateaus of null fitness. This solving method is based on a memetic algorithm from [61], combining a genetic algorithm with local search, which has been enhanced by considering a wider range of genetic operators, allowing for different intensities in the hybridisation of the genetic and local search components and by incorporating the heuristic tie-breaking mechanism. A thorough experimental evaluation has shown the potential of the memetic algorithm, which not only outperforms the state-of-the-art, but also provides robust solutions. A new more challenging benchmark has been proposed by incorporating flexible due dates to existing instances and results obtained with MA using both objective functions, hopefully providing a reference for future research on more powerful methods to solve this problem. The resulting test bed is available on the internet[1], in order to facilitate experiment reproducibility and encourage research competition.

As future work, it would be interesting to devise a method to generate due dates that takes into account the "shape" of the problem instances, so for those instances with a ratio $n/m > 1$, the due dates generated by the new method are not excessively strict, making it reasonable to use AI_{min} as objective function to measure overall due-date satisfaction. Also, it would be interesting to provide formal definitions for the neighbourhood structures used in the local search and study their theoretical properties. Among others, this study

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

might help to design more sophisticated local-search components, such as tabu search, for the memetic algorithm.

References

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

[2] H. Adeli and A. Karim. *Construction scheduling, Cost Optimization and Management*. CRC Press, 2003.

[3] S. Ahn, S. Lee, and H. Bahn. A smart elevator scheduler that considers dynamic changes of energy cost and user traffic. *Integrated Computer-Aided Engineering*, 24:187–202, 2017.

[4] H. Aissi, C. Bazgan, and D. Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197:427–438, 2009.

[5] A. Allahverdi. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2):345–378, 2015.

[6] H. Aytung, M. A. Lawley, K. McKay, M. Shantha, and R. Uzsoy. Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, 161:86–110, 2005.

[7] R. Barták, M. A. Salido, and F. Rossi. New trends in constraint satisfaction, planning, and scheduling: a survey. *The Knowledge Engineering Review*, 25(3):249?279, 2010.

[8] C. Baudrit and D. Dubois. Practical representations of incomplete probabilistic knowledge. *Computational Statistics & Data Analysis*, 51:86–108, 2006.

[9] J. Behnamian. Survey on fuzzy shop scheduling. *Fuzzy Optimization and Decision Making*, 15:331–366, 2016.

[10] R. E. Bellman and L. A. Zadeh. Decision-making in a fuzzy environment. *Management Science*, 17(4):141–164, 1970.

[11] J. Benjamin and B. A. Julstrom. Breaking ties with secondary fitness in a genetic algorithm for the bin packing problem. In *Proceedings of 12th annual conference on Genetic and Evolutionary Computation. GECCO'2010*, pages 657–664. ACM, 2010.

[12] J. Bidot, T. Vidal, and P. Laboire. A theoretic and practical framework for scheduling in stochastic environment. *Journal of Scheduling*, 12:315–344, 2009.

[13] C. Bierwirth. A generalized permutation approach to jobshop scheduling with genetic algorithms. *OR Spectrum*, 17:87–92, 1995.

[14] C. Bierwirth and J. Kuhpfahl. Extended GRASP for the job shop scheduling problem with total weighted tardiness objective. *European Journal of Operational Research*, 261:835–848, 2017.

[15] C. Bierwirth, D. C. Mattfeld, and H. Kopfer. On permutation representations for scheduling problems. In *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, pages 310–318, London, UK, 1996. Springer-Verlag.

[16] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.

[17] K. Bülbül. A hybrid shifting bottleneck-tabu serach heuristic for the job shop total weighted tardiness problem. *Computers & Operations Research*, 38:967–783, 2011.

[18] S. Chanas and J. Kamburowski. The use of fuzzy variables in pert. *Fuzzy Sets and Systems*, 5(1):11–19, 1981.

[19] I. A. Chaudhry and A. A. Khan. A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3):551–591, 2016.

[20] Y. Chen, J. Hu, K. Hirasawa, and S. Yu. Solving deceptive problems using a genetic algorithm with reserve selection. In *IEEE Congress on Evolutionary Computation, 2008. CEC 2008.*, 2008.

[21] D. Dubois. The role of fuzzy sets in decision sciences: Old techniques and new directions. *Fuzzy Sets and Systems*, 184:3–28, 2011.

[22] 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.

[23] D. Dubois, H. Fargier, and H. Prade. Fuzzy constraints in job-shop scheduling. *Journal of Intelligent Manufacturing*, 6:215–234, 1995.

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

[25] D. Dubois and H. Prade. A review of fuzzy set aggregation connectives. *Information Sciences*, 36:85–121, 1985.

[26] D. Dubois, H. Prade, and S. Sandri. On possibility/probability transformations. In *Fuzzy Logic*, volume 12 of *Theory and Decision Library*, pages 103–112. Kluwer Academic, 1993.

[27] O. Engin, M. K. Yilmaz, C. Kahraman, M. E. Baysal, and A. Sarucan. A scatter search method for fuzzy job shop scheduling problem with availability constraints. In *Proceedings of the World Congress on Engineering 2011 (WCE 2011)*, volume II, pages 1144–1148, London (U.K.), 2011. Newswood Limited.

[28] I. Essafi, Y. Mati, and S. Dauzère-Pérès. A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. *Computers & Operations Research*, 35:2599–2616, 2008.

[29] C. Fayad and S. Petrovic. A fuzzy genetic algorithm for real-world job-shop scheduling. *Innovations in Applied Artificial Intelligence, Lecture Notes in Computer Science*, 3533:524–533, 2005.

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

[31] M. A. González, I. González-Rodríguez, C. Vela, and R. Varela. An efficient hybrid evolutionary algorithm for scheduling with setup times and weighted tardiness minimization. *Soft Computing*, 16:2097–2113, 2012.

[32] I. González Rodríguez, J. Puente, and C. R. Vela. A multiobjective approach to fuzzy job shop problem using genetic algorithms. *CAEPIA 2007, Lecture Notes in Artificial Intelligence*, 4788:80–89, 2007.

[33] 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.

[34] R. Graham, E. Lawler, J. Lenstra, and A. Rinnooy Kan. Optimization and approximation in deterministic sequencing and

scheduling: a survey. *Annals of Discrete Mathematics*, 4:287–326, 1979.

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

[36] L. Jia, Y. Wang, and L. Fan. Multiobjective bilevel optimization for production-distribution planning problems using hybrid genetic algorithm. *Integrated Computer-Aided Engineering*, 21:77–90, 2014.

[37] R. Kalaï, C. Lamboray, and D. Vanderpooten. Lexicographic α-robustness: An alternative to min-max criteria. *European Journal of Operational Research*, 220:722–728, 2012.

[38] A. Kasperski and M. Kule. Choosing robust solutions in discrete optimization problems with fuzzy costs. *Fuzzy Sets and Systems*, 160:667–682, 2009.

[39] N. Krasnogor and J. Smith. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488, 2005.

[40] J. Kuhpfahl and C. Bierwirth. A study on local search neighbourhoods for the job shop scheduling problem with total weighted tardiness objective. *Computers & Operations Research*, 261:44–57, 2016.

[41] C. Kyriklidis and G. Dounias. Evolutionary computation for resource leveling optimization in project management. *Integrated Computer-Aided Engineering*, 23:173–184, 2016.

[42] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999.

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

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

[45] S. Luke and L. Panait. Lexicographic parsimony pressure. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, GECCO'02, pages 829–836, 2002.

[46] M. Masmoudi and A. Haït. Project scheduling under uncertainty using fuzzy modelling and solving techniques. *Engineering Applications of Artificial Intelligence*, 26:135–149, 2013.

[47] C. S. McCahon. Using PERT as an approximation of fuzzy project-network analysis. *IEEE Transactions on Engineering Management*, 40:146–153, 1993.

[48] R. Mencía, M. Sierra, C. Mencía, and R. Varela. Genetic algorithms for the scheduling problem with arbitrary precedence relations and skilled operators. *Integrated Computer-Aided Engineering*, 23:269–285, 2016.

[49] F. Neri and C. Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.

[50] F. Neri, C. Cotta, and P. Moscato, editors. *Handbook of Memetic Algorithms*. Studies in Computational Intelligence. Springer, 2012.

[51] F. Neri, J. Toivanen, G. L. Cascella, and Y.-S. Ong. An adaptive multimeme algorithm for designing hiv multidrug therapies. *IEEE Transactions on Computational Biology and Bioinformatics*, 4(2):264–278, 2007.

[52] F. Neri, J. Toivanen, and R. A. Mäkinen. An adaptive evolutionary algorithm with intelligent mutation local searchers for designing multidrug therapies for hiv. *Applied Intelligence*, 27:219–235, 2007.

[53] I. Ono, M. Yamamura, and S. Kobayashi. A genetic algorithm for job-shop scheduling problems using job-based order crossover. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 547–552. IEEE, 1996.

[54] J. J. Palacios and B. Derbel. On maintaining diversity in MOEA/D: Application to a biobjective combinatorial FJSP. In *GECCO '15 Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 719–726. ACM, July 2015.

[55] 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.

[56] 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.

[57] J. J. Palacios, I. González-Rodríguez, C. R. Vela, and J. Puente. Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop. *Fuzzy Sets and Systems*, 278:81–97, 2015.

[58] J. J. Palacios, I. González-Rodríguez, C. R. Vela, and J. Puente. Robust multiobjective optimisation for fuzzy job shop problems. *Applied Soft Computing*, 56:604–616, 2017.

[59] J. J. Palacios, J. Puente, C. R. Vela, and I. González-Rodríguez. Benchmarks for fuzzy job shop problems. *Information Sciences*, 329:736–752, 2016.

[60] J. J. Palacios, C. R. Vela, I. González-Rodríguez, and J. Puente. Schedule generation schemes for job shop problems with fuzziness. In T. Schaub, G. Friedrich, and B. O'Sullivan, editors, *Proceedings of ECAI 2014*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 687–692. IOS Press, 2014.

[61] J. J. Palacios, C. R. Vela, I. González-Rodríguez, and J. Puente. A memetic algorithm for due-date satisfaction in fuzzy job shop scheduling. In *IWINAC2017 International Work-Conference on the Interplay Between Natural and Artificial Computation. LNCS 10337*, pages 135–145. Springer, June 2017.

[62] M. Pinedo and M. Singer. A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. *Nav. Res. Logist*, 46(1):1–17, 1999.

[63] M. L. Pinedo. *Scheduling. Theory, Algorithms, and Systems.* Springer, fifth edition, 2016.

[64] A. Prieto, F. Bellas, P. Trueba, and R. Duro. Real-time optimization of dynamic problems through distributed embodied evolution. *Integrated Computer-Aided Engineering*, 23:237–253, 2016.

[65] S. Rokni and A. Fayek. A multi-criteria optimization framework for industrial shop scheduling using fuzzy set theory. *Integrated Computer-Aided Engineering 17 (2010) 175–196*, 17:175–196, 2010.

[66] S. Rostami and F. Neri. Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm. *Integrated Computer-Aided Engineering*, 23:3137–329, 2016.

[67] B. Roy. Robustness in operational research and decision aiding: A multi-faceted issue. *European Journal of Operational Research*, 200:629–638, 2010.

[68] 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.

[69] 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.

[70] A. Senouci and H. Adeli. Resource scheduling using neural dynamics model of Adeli and Park. *Journal of Construction Engineering and Management*, 127(1):28–34, 2001.

[71] N. H. Siddique and H. Adeli. Nature inspired computing: An overview and some future directions. *Cognitive Computation*, 7(6):706–714, 2015.

[72] M. Singer and M. Pinedo. A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. *IIE Transactions*, 30:109–118, 1998.

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

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

[75] 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.

[76] C. Wang, N. Tian, Z. Ji, and Y. Wang. Multi-objective fuzzy flexible job shop scheduling using memetic algorithm. *Journal of Statistical Computation and Simulation*, 87(14):2828–2846, 2017.

[77] J. Wang. A fuzzy robust scheduling approach for product development projects. *European Journal of Operational Research*, 152:180–194, 2004.

[78] Q. Wang, H.-L. Liu, J. Yuan, and L. Chen. Optimizing the energy-spectrum efficiency of cellular systems by evolutionary multi-objective algorithm. *Integrated Computer-Aided Engineering*, In press, 2018.

[79] S. Wang, Aorigele, G. Liu, and S. Gao. A hybrid discrite imperialist competition algorithm for fuzzy job-shop scheduling problems. *IEEE Access*, 4:9320–9331, 2016.

[80] C. H. Westerberg and J. Levine. Optimising plans using genetic programming. In *Proceedings of the Sixth European Conference on Planning. ECP-01*, pages 272–274. AAAI, 2001.

[81] J. Wright and I. Jordanov. Quantum inspired evolutionary algorithms with improved rotation gates for real-coded synthetic and real world optimization problems. *Integrated Computer-Aided Engineering*, 24:2013–223, 2017.

[82] Z. Xiang, B. Zhenqiang, W. Guijun, and P. Quanke. Optimization of fuzzy job-shop scheduling with multi-process routes and its co-evolutionary algorithm. In *Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference on*, volume 1, pages 866–870. IEEE, March 2011.

[83] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, 1988.

[84] Z. Yang, M. Emmerich, T. Bäck, and J. Kok. Multi-objective inventory routing with uncertain demand using population-based metaheuristics. *Integrated Computer-Aided Engineering*, 23(3):205–220, 2016.

[85] 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.

[86] C. Zhu, L. Xu, and E. D. Goodman. Generalization of pareto-optimality for many-objective evolutionary optimization. *IEEE T. Evol. Comput.*, 20(2):299–315, 2016.

Appendix

A. Detailed Results

Tables 7 and 8 report detailed AI_{avg} results for those Ta instances not included in Table 4. It includes the best AI_{avg} value across all runs, together with the average and standard deviation and running times in seconds.

Table 7

Results obtained by MA maximising AI_{avg} on instances Ta01-Ta20 and Ta31-Ta40

Instance	Best AI_{avg}	Average AI_{avg}	Runtime (s)
Ta01	0.817	0.782 (0.018)	19.7
Ta02	0.832	0.805 (0.013)	16.7
Ta03	0.887	0.861 (0.024)	16.6
Ta04	0.840	0.830 (0.007)	17.0
Ta05	0.786	0.752 (0.012)	17.0
Ta06	0.876	0.840 (0.019)	17.0
Ta07	0.792	0.781 (0.009)	16.3
Ta08	0.898	0.885 (0.011)	15.4
Ta09	0.848	0.823 (0.011)	18.1
Ta10	0.901	0.892 (0.008)	15.7
Ta11	0.633	0.587 (0.025)	61.0
Ta12	0.660	0.624 (0.021)	59.2
Ta13	0.699	0.659 (0.015)	60.8
Ta14	0.732	0.664 (0.026)	48.7
Ta15	0.680	0.632 (0.024)	64.8
Ta16	0.712	0.675 (0.023)	60.4
Ta17	0.690	0.657 (0.022)	51.5
Ta18	0.667	0.618 (0.028)	76.3
Ta19	0.649	0.602 (0.023)	64.4
Ta20	0.651	0.627 (0.016)	49.6
Ta31	0.401	0.348 (0.023)	220.8
Ta32	0.393	0.355 (0.023)	191.6
Ta33	0.389	0.359 (0.017)	226.6
Ta34	0.414	0.361 (0.025)	234.8
Ta35	0.425	0.358 (0.027)	201.3
Ta36	0.420	0.374 (0.020)	217.8
Ta37	0.406	0.375 (0.017)	214.7
Ta38	0.390	0.352 (0.018)	235.1
Ta39	0.371	0.337 (0.022)	201.1
Ta40	0.388	0.345 (0.026)	217.5

Table 8

Results obtained by MA maximising AI_{avg} on the instances Ta51-Ta80

Instance	Best AI_{avg}	Average AI_{avg}	Runtime (s)
Ta51	0.175	0.129 (0.019)	547.8
Ta52	0.160	0.123 (0.017)	445.0
Ta53	0.160	0.137 (0.015)	610.7
Ta54	0.160	0.114 (0.022)	395.9
Ta55	0.173	0.130 (0.021)	551.1
Ta56	0.158	0.126 (0.019)	557.6
Ta57	0.159	0.122 (0.021)	557.7
Ta58	0.176	0.126 (0.019)	556.7
Ta59	0.160	0.135 (0.015)	624.2
Ta60	0.157	0.126 (0.013)	880.0
Ta61	0.198	0.162 (0.017)	1343.1
Ta62	0.195	0.169 (0.013)	1415.7
Ta63	0.205	0.166 (0.015)	1595.0
Ta64	0.183	0.162 (0.014)	1241.0
Ta65	0.177	0.157 (0.014)	1523.8
Ta66	0.198	0.167 (0.012)	1623.8
Ta67	0.195	0.160 (0.017)	1519.2
Ta68	0.207	0.171 (0.020)	1470.3
Ta69	0.216	0.175 (0.018)	1461.9
Ta70	0.186	0.161 (0.019)	1293.2
Ta71	0.010	0.001 (0.003)	112.6
Ta72	0.010	0.001 (0.003)	124.7
Ta73	0.010	0.003 (0.005)	303.7
Ta74	0.010	0.002 (0.004)	219.9
Ta75	0.020	0.002 (0.005)	202.9
Ta76	0.010	0.002 (0.004)	248.3
Ta77	0.010	0.001 (0.003)	138.4
Ta78	0.010	0.001 (0.003)	220.8
Ta79	0.010	0.000 (0.002)	95.0
Ta80	0.010	0.001 (0.003)	184.5