Co-evolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop

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

[a]Department of Computing, University of Oviedo, (Spain)
[b]Dept. of Mathematics, Statistics and Computing, University of Cantabria, (Spain)

Abstract

In this paper we tackle a variant of the flexible job shop scheduling problem with uncertain task durations modelled as fuzzy numbers. To minimise the schedule's makespan, we consider different ranking methods for fuzzy numbers. We then propose a cooperative co-evolutionary algorithm with two different populations evolving the two components of a solution: machine assignment and task relative order. Additionally, we incorporate a specific local search method for each population. The resulting hybrid algorithm is then evaluated on existing benchmark instances, comparing favourably with the state-of-the-art methods. The experimental results also serve to analyse the influence in the robustness of the resulting schedules of the chosen ranking method.

Keywords: flexible job shop scheduling, robustness, local search, co-evolutionary algorithm, ranking of fuzzy numbers, fuzzy processing times

1. Introduction

The importance of scheduling as a research topic is undeniable, both as a source of interesting complex combinatorial optimisation problems and as a field with multiple real applications in industry, finance, welfare, etc. In particular, shop problems in their multiple variants— for instance, incorporating flexibility or operators— can model many situations which naturally arise in manufacturing environments [1].

Fuzzy sets have contributed to enhancing the applicability of scheduling, helping to bridge the gap between classical techniques and real-world user needs. They have been used both for handling flexible constraints and uncertain data [2],[3],[4],[5]. They are also emerging as an interesting tool for improving

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

solution robustness, a much-desired property in real-life applications [6],[7],[8]. Incorporating fuzzy sets to scheduling is, however, far from trivial, and they usually require a reformulation of the problem under consideration and the development of new solving techniques. In that sense, one important issue (shared with many other applications of fuzzy sets) is how to rank different solutions when their quality is given as a fuzzy quantity.

In deterministic scheduling, the complexity of problems such as job shop means that practical approaches to solving them usually involve metaheuristic strategies [9]. Some attempts have been made to extend such methods, mostly evolutionary algorithms, to the case where uncertain durations are modelled via fuzzy intervals. In particular, the fuzzy flexible job shop problem is receiving an increasing attention, with proposals including a genetic algorithm [10], a hybrid artificial been colony algorithm [11], an estimation distribution algorithm [12], a swarm-based neighbourhood search algorithm [13] and a co-evolutionary algorithm [14].

Indeed, co-evolutionary algorithms [15] [16] are a special case of evolutionary algorithms which are proving to be very successful in solving complex problems [17],[18],[19]. They have been also applied for solving different scheduling problems: for example in [20] are used to solve the integrated problem of process planing and scheduling in job shop problems, in [21] for the stochastic job shop problem or in [14] for the fuzzy flexible job shop.

In the following we tackle the fuzzy flexible job shop problem, where uncertainty in task durations is modelled using fuzzy numbers. After introducing the problem, we consider different ranking methods to minimise the resulting fuzzy makespan and give a definition of schedule robustness based on average behaviour across all possible cases. We shall see how the problem naturally lends itself to coooperative co-evolution and we shall also propose neighbourhood structures for each population, so local search can be embedded in the co-evolutionary algorithm. The experimental results will illustrate the synergy between the co-evolution and the local search, as well as the competitiveness of our approach when compared to the state-of-the-art. The results will also allow for an empirical assessment of different ranking methods in terms of solution robustness.

2. The fuzzy flexible job shop scheduling problem

The *flexible job shop scheduling problem*, fJSP in short, consists in scheduling a set of jobs $J = \{J_1, \ldots, J_n\}$ on a set of physical machines $M = \{M_1, \ldots, M_m\}$, subject to a set of constraints. There are *precedence constraints*, so each job $J_i, i = 1, \ldots, n$ consists of a sequence of n_i tasks $\Theta_i = \{\theta_{i1}, \ldots, \theta_{in_i}\}$ that must be sequentially scheduled. There are also *capacity constraints*, whereby each task θ_{ij} requires the uninterrupted and exclusive use of one machine from a subset $R_{ij} \subset M$, so the task's processing time p_{ijk} depends on the machine $M_k \in R_{ij}$.

A *feasible schedule* or solution consists of an assignment to machines of all $N = \sum_{i=1}^{n} n_i$ tasks in the set $\Theta = \cup_{1 \leq i \leq n} \Theta_i$ together with an allocation

of starting times for each task such that all constraints hold. Alternatively, a solution can be represented as a feasible assignment of each task $\theta_{ij} \in \Theta$ to a machine $M_k \in R_{ij}$ and a task processing order for each machine in M. Indeed, given these two pieces of information, the starting time of θ_{ij}, denoted S_{ij}, is easily computed as the maximum between the completion times of the predecessor of θ_{ij} in its job and the predecessor of θ_{ij} in the machine M_k where it has been allocated, and the completion time is given by $C_{ij} = S_{ij} + p_{ijk}$. The objective is to find an *optimal* solution according to some criterion, in our case, minimise the *makespan*, which is the completion time of the last task to be executed, denoted $C_{max} = \max_{\theta_{ij} \in \Theta} C_{ij}$.

2.1. Uncertain processing times

In real-life applications, it is often the case that the exact processing time of tasks is not known in advance. However, based on previous experience, an expert may be able to estimate, for instance, an interval for the possible processing time or its most typical value. When there is little knowledge available, the crudest representation for uncertain processing times would be a human-originated confidence interval. If some values appear to be more plausible than others, a natural extension is a fuzzy interval or fuzzy number. The simplest model is a *triangular fuzzy number* or *TFN*, using an interval $[a^1, a^3]$ of possible values and a modal value a^2, and with the membership function taking a triangular shape as follows:

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

We shall denote such TFN as $A = (a^1, a^2, a^3)$. For $\alpha \in (0, 1]$, its α-cut $A_\alpha = \{x : \mu_A(x) \geq \alpha\}$ is a closed interval $[\underline{a}_\alpha, \overline{a}_\alpha]$; we shall abuse notation slightly and denote its support as A_0. TFNs are to date the most widely used model for uncertain durations in the fuzzy scheduling literature.

In the flexible job shop, we essentially need two operations on fuzzy numbers, the sum and the maximum. These are obtained by extending the corresponding operations on real numbers using the *Extension Principle*. In the case of the addition, it turns out that for any pair of TFNs A and B:

$$A + B = (a^1 + b^1, a^2 + b^2, a^3 + b^3) \tag{2}$$

Unfortunately, computing the maximum is not that straightforward and, most importantly, the set of TFNs is not closed under this operation. For the sake of simplicity and tractability of numerical calculations, a common approach is to approximate the maximum by the TFN that results from evaluating this operation on the three defining points of each TFN, that is, for every A, B TFNs:

$$\max(A, B) \approx \max_I(A, B) = (\max(a^1, b^1), \max(a^2, b^2), \max(a^3, b^3)) \tag{3}$$

This approximation has been widely used in the scheduling literature, among others, in [22, 23, 24, 25, 26] or [27].

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

$$F_\alpha = [f(\underline{a}_\alpha, \underline{b}_\alpha), f(\overline{a}_\alpha, \overline{b}_\alpha)]. \tag{4}$$

Computing $f(M, N)$ is then equivalent to computing f on every α-cut. In particular, the maximum is a continuous isotonic function, so it can be calculated by evaluating two maxima of real numbers for every value $\alpha \in [0, 1]$. If seems then natural to approximate the maximum by the TFN that results from using linear interpolation, evaluating equation (4) only for certain values of α (this is proposed for 6-point fuzzy numbers in [23]). Given that the defining values (a^1, a^2, a^3) of a TFN A are such that $A_0 = [a^1, a^3]$ and $A_1 = [a^2, a^2]$, the approximated maximum as in (3) corresponds to such an interpolation for $\alpha = 0$ and $\alpha = 1$. Secondly, if $F = \max(A, B)$ denotes the maximum of two TFNs A and B and $G = \max_I(A, B)$ the approximated value by interpolation, then $F = G$ if A and B do not overlap and, in any case, it holds that

$$\forall \alpha \in [0, 1], \quad \underline{f}_\alpha \leq \underline{g}_\alpha, \overline{f}_\alpha \leq \overline{g}_\alpha. \tag{5}$$

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

2.2. Ranking fuzzy makespan values

For a given schedule, the makespan C_{max}, the completion time of the last task to be executed, is obtained by performing addition and maximum operations on fuzzy durations and, hence, is a TFN. If several schedules are available, the "best" one would be the one with minimal makespan, which requires comparing fuzzy numbers.

In general, fuzzy scheduling problems involve ordering or ranking fuzzy numbers representing solution performance. However, no natural total order exists in the set of of fuzzy numbers and several ranking methods have been and keep being proposed in the literature (cf. [28, 29, 30]). Furthermore, quoting Brunelli and Mezei [29],

> It is impossible to give a final answer to the question on what ranking method is the best. Most of the time, choosing a method rather than another is a matter of preference or is context dependent.

We intend to consider some ranking methods for fuzzy numbers and their influence in the robustness of solutions of fuzzy flexible job shop scheduling problems.

Let $\mathcal{F}$ denote the set of fuzzy numbers. Ranking methods in $\mathcal{F}$ can be roughly divided in two types: those based on "defuzzification" and those based

on fuzzy binary relations. In the first case, a mapping $M : \mathcal{F} \to \mathbb{R}$ is defined which associates each fuzzy number X with a real number and then the natural ordering on the real line is used, most commonly, $X \leq_M Y$ iff $M(X) \leq M(Y)$. In the second case, a relation $M : \mathcal{F} \times \mathcal{F} \to [0, 1]$ is defined such that $M(X, Y)$ is the degree to which X is greater than Y and, consequently, if $M(X, Y) \geq M(Y, X)$, then $X \geq_M Y$.

In [31] it is proposed to summarise a fuzzy set X by the value:

$$E_\beta(X) = \int_0^1 (\beta \underline{x}_\alpha + (1 - \beta)\overline{x}_\alpha) d\alpha \tag{6}$$

where $\beta \in [0, 1]$ is a "pessimism" value. This proposal can also be found in [32]. Obviously, this value can be used in a ranking method of the first type.

For the special case of $\beta = \frac{1}{2}$, $E_{\frac{1}{2}}$ has been proposed by many authors, among others, as the the neutral scalar substitute of a fuzzy interval in [33], as the expected value of a fuzzy number in [34], using the area compensation method in [35], as the generative expected value induced by the evidence X [36], as the credibilistic expectation of a fuzzy variable [37] or as the middle point of a fuzzy number defined for ranking by distance minimisation [38]. It is also the centre of the mean value of a fuzzy number as defined in [39] and the expected value of the so-called pignistic probability distribution, which is found as the centroid of the set of probabilities dominated by the possibility measure associated with X, $\mathcal{P}(\Pi_X)$ (cf. [40]).

When TFNs are considered, the special case $\beta = 0$ coincides with the index A_α suggested by [41], which simply evaluates the fuzzy number based on the rightmost point of the α-cut for a given α, in this case, $\alpha = 0.5$. According to [29], this approach is the only one which satisfies all the reasonable properties proposed in [42, 43] for ordering fuzzy quantities.

2.2.1. Relationship with classical interval comparison and interpretation

In [44], the authors study the relationship between some well-known criteria for classical interval comparison and fuzzy ranking methods in the light of imprecise probabilities, extending some preliminary ideas which can already be found in [45]. In particular, they consider four interval comparisons:

- Weak ordering: $[\underline{x}, \overline{x}] \leq [\underline{y}, \overline{y}]$ iff $\underline{x} \leq \overline{y}$;
- Maximin: $[\underline{x}, \overline{x}] \leq [\underline{y}, \overline{y}]$ iff $\underline{x} \leq \underline{y}$;
- Maximax: $[\underline{x}, \overline{x}] \leq [\underline{y}, \overline{y}]$ iff $\overline{x} \leq \overline{y}$;
- Hurwicz: $[\underline{x}, \overline{x}] \leq_{H(\gamma)} [\underline{y}, \overline{y}]$ iff $\gamma\underline{x} + (1 - \gamma)\overline{x} \leq \gamma\underline{y} + (1 - \gamma)\overline{y}$.

Obviously, the Hurwicz comparison subsumes both the maximim ($\gamma = 1$) and the maximax ($\gamma = 0$). Interestingly, E_β comes down to using Hurwicz criterion on the expectation values with pessimism value β:

$$E_\beta(X) = \beta\underline{E}(X) + (1 - \beta)\overline{E}(X) \tag{7}$$

where $\underline{E}$ and $\overline{E}$ denote the upper and lower expectations derived from X [44, 45].

This provides us with a nice interpretation for comparing TFNs based on E_β:

- if $\beta = 0$, comparing TFNs based on E_0 would correspond to a pessimistic decision maker;
- if $\beta = 1$, comparing TFNs based on E_1 would correspond to an optimistic decision maker;
- if $\beta = \frac{1}{2}$, comparing TFNs based on $E_{\frac{1}{2}}$ would correspond to an in-between decision maker, with an equilibrium between pessimism and optimism;

2.2.2. Relationship with other fuzzy ranking methods

A recent numerical study in [29] suggests that several ranking methods represent very similar (referred to by the authors as *compatible*) points of view. In practice, this means that the ordering they induce in a sample of fuzzy numbers is strongly correlated. In particular, for TFNs, since the ranking based on $E_{\frac{1}{2}}$, is identical to Yager's ranking based on the neutral scalar substitute from [33] and the credibilistic mean from [37] and the ranking based on these two indices is grouped as compatible with seven more ranking methods (see [29] for further detail):

- $N^{0.5}$, the parametric method defined in [46] based on a fuzzy binary relation;
- CoM, the method based on the centre of maxima or mean of maxima [28];
- E_p, the method based on the possibilistic mean value [47];
- CH^1, the method based on a ranking index using the concepts of fuzzy maximising and minimising sets from [48];
- CoG, the method based on the centre of gravity of a fuzzy set [28];
- Med, the method based on the median of a fuzzy number [49];
- PD, the method based on the PD relation introduced in [50].

Interestingly, the latter method extends the weak ordering of intervals, with $PD(X, Y)$ corresponding to the upper probability of the event $X \geq Y$ under the monotonic dependence assumption [44].

As for $\beta = 0$, we have already noted that E_0 coincides with Adamo's index $A_{0.5}$. According to [29], the ranking based on this index is not strongly correlated to the one based on $E_{\frac{1}{2}}$ and is therefore to produce significantly different orderings in the set of TFNs.

In conclusion, if we consider three ranking possibilities, based on E_β with $\beta = 0, \frac{1}{2}, 1$, we are in fact modelling three different behaviours of the decision maker according to her level of pessimism. But it is also the case that, by considering

these three possibilities, we are taking into account many other ranking methods from the literature, either because they are based on a defuzzification index which coincides with some E_β for TFNs or because, according to [29], they yield very similar orderings to $E_{\frac{1}{2}}$ or E_0 (in the case of $A_{0.5}$).

2.3. Fuzzy Flexible Job Shop

When task durations in a flexible job shop problem are given as TFNs, the resulting problem is a *fuzzy flexible job shop problem*, FfJSP in short. The objective is to minimise the makespan C_{max} according to one of the ranking methods above.

Notice that a solution to the FfJSP is fuzzy in the sense that starting, processing and completion times of each task are fuzzy numbers, seen as possibility distributions on the actual values they may take. However, there is no uncertainty regarding the machine assignment nor the order in which tasks must be processed.

3. 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 machine and 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 [25] by which solutions to the fuzzy job shop should be understood as a-priori solutions, also called baseline or predictive schedules in the literature [51]. These solutions are found when the duration of tasks is not exactly known and a set of possible scenarios must be taken into account. When tasks are executed according to machine assignment and the ordering provided by the fuzzy schedule we shall know their real duration and, hence, obtain a real (executed) schedule, the a-posteriori solution with deterministic times.

Clearly, a fuzzy solution should yield reasonably good executed schedules in the moment of its practical use. Also, the estimates for starting and completion times and, in particular, for the makespan, should be reasonably accurate for each possible scenario of task durations. This leads us to the concept of solution robustness. As [52] puts it, "Intuitively, a solution can be considered as robust if it behaves "well" or "not too bad" in all the scenarios.". This is the idea underlying a definition of ϵ-robustness given in [53] for stochastic scheduling which can be adapted to the fuzzy flexible job shop as follows.

A predictive schedule is considered to be *robust* if the quality of the eventually executed schedule is close to the quality of the predictive schedule. In particular, a predictive schedule with objective value f^{pred} (a TFN) is ϵ-*robust* for a given ϵ if the objective value f^{exec} of the eventually executed schedule (a real value) is such that:

$$(1-\epsilon) \leq \frac{f^{exec}}{E_\beta(f^{pred})} \leq (1+\epsilon) \tag{8}$$

or, equivalently,

$$\frac{|f^{exec} - E_\beta(f^{pred})|}{E_\beta(f^{pred})} \leq \epsilon. \tag{9}$$

That is, the relative error of the estimation made by the predictive schedule is bounded by ϵ. Obviously, the smaller ϵ is, the better.

According to this interpretation, the robustness of a solution can only be measured once we have a real execution of the problem. However, it is very common in the literature to use synthetic problems instead of real ones, so no real execution is available. For those cases we propose to run a Monte-Carlo simulation to provide a surrogate of the ϵ-robustness measure. Given a fuzzy instance, we may generate a sample of K possible realisations of that instance by assigning an exact duration to each task, that is K deterministic instances in which we can evaluate the robustness of the solution. Now for each realisation $k = 1, \ldots, K$, let $C_{max,k}$ denote the makespan obtained by executing tasks according to the ordering and machine assignment provided by the predictive schedule. Then, the average ϵ-robustness of the predictive schedule, denoted $\bar{\epsilon}$, is calculated as:

$$\bar{\epsilon} = \frac{1}{K} \sum_{k=1}^{K} \frac{|C_{max,k} - E_\beta(C_{max,pred})|}{E_\beta(C_{max,pred})} \tag{10}$$

where $C_{max,pred}$ is the makespan estimated by the predictive schedule.

Notice that a crucial factor in this method is the way in which we simulate real durations for the tasks. This is actually done by generating real durations for tasks following a probability distribution that is consistent with the possibility distribution defined by each fuzzy duration. Originally, in [25] the authors use the renormalisation technique (dividing the membership function μ_M by its surface). However, this technique can be objected to; according to [40], it is arbitrary and the obtained probability may fail to belong to $\mathcal{P}(\mu_M)$, the set of probability measures dominated by μ_M. Here we will consider instead the probability distribution obtained from each fuzzy duration after applying the pignistic transformation obtained by considering cuts as uniformly distributed probabilities [54]. 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.

Our approach to robustness 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 [55]. The study of such criteria is motivated by practical applications where an anticipation of the worst case is crucial and has already been translated to the fuzzy framework [6], [7]. 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 is preferred. It is in these situations where an approach such as ϵ-robustness might be more adequate.

4. Cooperative co-evolutionary algorithm for the FfJSP

Co-evolutionary algorithms are advanced evolutionary techniques specially suited to solve complex problems which are decomposable. They handle two or more populations, each with its own coding schemes and recombination operators, that interact through evaluation. When all populations cooperate to build the problem solution, we talk about cooperative algorithms [15].

The nature of solutions to the FfJSP, with two separate components, suggests that cooperative co-evolution may be specially suited for this problem. The first subproblem we face when searching for a solution to the FfJSP is to assign the processing of each task θ_{ij} to a machine $M_k \in R_{ij}$. Once this has been done, we obtain a classical job shop problem where we need to establish the order in which tasks are to be processed in each machine. We propose to separately evolve those two components in a co-evolutionary framework, with a "machine assignment population" P^M in charge of evolving the machine assignment and a "task ordering population" P^T in charge of finding the processing order for tasks.

Algorithm 1 summarises the main steps of our co-evolutionary algorithm. It first builds a pool of initial solutions, which are then split to form the initial populations P_0^T and P_0^M. A variable *Best* will record the best full initial solution and will then be updated throughout the evolution so it always keeps record of the best solution found so far, thus incorporating elistism. After the initialisation phase, the algorithm iterates until a stopping criterion is met. At each iteration, the individuals of each population are paired and crossover and mutation operators are applied to each pair with probability $prob_c$ and $prob_m$ respectively; each individual is then evaluated using some partners from the other population in order to have a complete solution and, finally, a replacement strategy is applied.

In the following, we describe in more detail the algorithm's components.

4.1. Genotype coding and decoding

Every individual from population P^M encodes a machine assignment as a vector $\alpha = \{\alpha_1, \ldots, \alpha_N\}$; task θ_{ij} is associated to the element in position $p = j + \sum_{l=1}^{i-1} n_l$, so $\alpha_p \in R_{ij}$ represents the machine assigned to θ_{ij}. On the other hand, an individual in P^T encodes a topological order of tasks as a permutation with repetition $\pi = \{\pi_1, \ldots, \pi_N\}$ such that $\forall l, 1 \leq \pi_l \leq n$ and $|\{\pi_l : \pi_l = i\}| = n_i, \forall i = 1, \ldots, n$. This is a permutation of the set of tasks as proposed in [56] for the JSP, where each task is represented by its job number. For example, the topological order θ_{21}, θ_{11}, θ_{22}, θ_{31}, θ_{32}, θ_{12} is encoded as (2 1 2 3 3 1).

Notice that the encoding of each population is completely independent of the other population, unlike the co-evolutionary approach from [14] for the same problem. This independence allows both populations to evolve separately, interacting only at the evaluation phase. Indeed, to calculate a schedule we require a full solution, combining an individual α from P^M and an individual π from P^T. Then, a pair $(\alpha, \pi) \in P^M \times P^T$ will be decoded using the following insertion strategy. The task sequence is traversed in the order given by π and

Input A FfJSP instance
Output A solution
Generate a pool P of initial solutions.
$Best \leftarrow \arg\min_{S \in P}\{C_{max}(S)\}$;
Split P into populations P_0^T and P_0^M;
$i \leftarrow 1$;
while stop condition not satisfied **do**
//Evolve one iteration for population P_{i-1}^T
$P_i^T \leftarrow$ Paired individuals from P_{i-1}^T;
for each pair of individuals **do**
Apply crossover and mutation with probabilities $prob_c$ and $prob_m$;
Evaluate P_i^T using partners from P_{i-1}^M;*//Best is updated if necessary*
$P_i^T \leftarrow$ Apply 4:2 parent-children tournament between P_i^T and P_{i-1}^T;
//Evolve one iteration for population P_{i-1}^M
$P_i^M \leftarrow$ Paired individuals from P_{i-1}^M;
for each pair of individuals **do**
Apply crossover and mutation with probabilities $prob_c$ and $prob_m$;
Evaluate P_i^M using partners from P_{i-1}^M;*//Best is updated if necessary*
$P_i^M \leftarrow$ Apply 4:2 parent-children tournament between P_i^M and P_{i-1}^M;
//Apply elitism
Replace worst individual in P_i^T and in P_i^M with respective partial solutions from $Best$;
return $Best$

Algorithm 1: Main steps of the Co-Evolutionary algorithm

each task θ_{ij} is then scheduled in the machien M_k to which it is assigned by chromosome α. To assign a starting time to the task, it is necessary to compute a *feasible insertion interval*, that is a time interval $[t_{k,S}, t_{k,E}]$ in which machine M_k is idle and such that $t_{k,S} + p_{ijk} \leq t_{k,E}$ and $t_{k,S} \geq C_{i(j-1)}$ (if $j = 0$, $C_{i(j-1)}$ is taken to be 0); thus θ_{ij} can be processed within that time interval without violating precedence constraints. When $t_{k,S}, t_{k,E}$ and p_{ijk} are TFNs, we require that these inequalities hold in each of their three components (in accordance to the definition of maximum and addition). Then, the *earliest starting time* for operation θ_{ij} in machine M_k, denoted EST_{ijk}, is the smallest $t_{k,S}$ that can be found. We schedule operation θ_{ij} in machine M_k with starting time $S_{ij} = EST_{ijk}$.

4.2. Initial populations

The simplest way to generate both initial populations is to do it randomly. Alternatively, we propose a heuristic seeding method based on the insertion decoding algorithm. The idea is to use this algorithm as a production rule to generate a full schedule for the FfJSP and then encode its task ordering as an individual for P^T and its machine assignment as an individual for P^M.

The heuristic method is detailed in Algorithm 2. Let A denote the set of tasks that can be scheduled at a certain stage, initially the first task from each job. We iteratively select a random task $\theta_{ij} \in A$ and compute $C^\star = \min\{EST_{ijk} + p_{ijk} :$

Input A FfJSP instance
Output An individual for each population
$A \leftarrow \{o_{i1}, 1 \leq i \leq n\}$;
while $A \neq \emptyset$ **do**
$o_{ij} \leftarrow$ a task selected at random from A;
for each $k \in M_{ij}$ **do**
Compute EST_{ijk};
$C^* \leftarrow \min\{EST_{ijk} + p_{ijk}, k \in M_{ij}\}$;
$K \leftarrow \{k \in M_{ij}, EST_{ijk} + p_{ijk} = C^*\}$;
$k^* \leftarrow$ a machine selected at random from K;
Schedule the task θ_{ij} in machine k^* with $S_{ij} = EST_{ijk^*}$;
$A \leftarrow A - \{o_{ij}\}$
if $j < n_i$ **then**
$A \leftarrow A \cup \{\theta_{ij+1}\}$;
Split and encode the schedule.

Algorithm 2: The $FfInsertion$

$M_k \in R_{ij}\}$, the earliest possible completion time for θ_{ij} in all machines where it can be processed. A machine M_{k*} is then randomly selected from the set $K = \{M_k : EST_{ijk} + p_{ijk} = C^\star\}$ of machines where this earliest completion time can be achieved, so θ_{ij} is scheduled in M_{k*} with starting time EST_{ijk*}. θ_{ij} is removed from A and its successor in the job is added to A, provided it exists. The process finishes when A becomes empty, i.e., all tasks have been scheduled.

4.3. Recombination Operators

For the machine assignment population P^M, we use the one-point crossover: given two genotypes $\alpha^A = \{\alpha_1^A, \ldots, \alpha_N^A\}$ and $\alpha^B = \{\alpha_1^B, \ldots, \alpha_N^B\}$ the operator chooses a random point $p \in (1, N)$ and builds two offsprings α^C, α^D such that $\alpha_i^C = \alpha_i^A$ and $\alpha_i^D = \alpha_i^B$ for $i \leq p$ and $\alpha_i^C = \alpha_i^B$ and $\alpha_i^D = \alpha_i^A$ for $i > p$. A mutation strategy is also introduced, which takes a random gene α_q in the genotype associated to task θ_{ij} and changes its value to a random machine in R_{ij}.

In the case of P^T, individuals are combined using the JOX operator [57]. Given two genotypes π^A, π^B, JOX selects a random subset of jobs, copies their genes to one offspring in the same positions as in the first parent π^A, and fills the remaining genes from the second parent π^B so that they maintain their relative ordering. The second offspring is formed exchanging the role of the parents. The well-known insertion operator is used for mutation. A random gene π_p in the genotype is chosen and changes its position to a random one, while keeping the relative order of the other tasks.

In both populations, all individuals are grouped in pairs for mating. Acceptance is carried out using tournament in each group of parents and offsprings, selecting the best two individuals from this group of four to pass to the next generation. Additionally, we introduce elitism, so the *Best* solution is split and

the worst solutions from P^T and P^M are replaced by the task sequence and machine assignment of $Best$ respectively.

4.4. Cooperative partners for evaluation

It is at the time of evaluation that populations need to cooperate: any individual only encodes part of a solution and needs to be complemented by an individual from the other population, the so-called *cooperative partner*, to conform a full solution which can be evaluated, using the decoding method above. Based on [58], we use three cooperative partners to evaluate each individual. Assuming all individuals in both populations are arbitrarily ordered, an individual in position p from one population, has as cooperative partners from the other population the best individual in the previous generation, a random individual and the individual in the same position p. As we shall see in Section 5, the three full schedules built in this evaluation process may then improved using a local search strategy. Finally, the individual's fitness value is the best makespan of the three obtained schedules.

5. Local Search

Evolutionary algorithms are often hybridised with local search to benefit from the synergy between both methods, i.e., between the exploitation of the local search and the exploration of the evolutionary approach. Here, we propose to apply local search to each individual after its evaluation. This means applying local search three times for each individual, one for each of its cooperative partners. In order for the computational effort to not increase excessively, we implement the local search following a simple hill climbing strategy, which is one of the fastest ones. The best solution found after the three local search processes per individual is selected and the chromosome is updated accordingly, thus introducing lamarckism. Algorithm 3 illustrates the evaluation of chromosomes incorporating the selection of cooperative partners and the local search.

It is common in the literature to represent solutions to shop problems using acyclic graphs and define neighbourhood structures based on critical paths in these graphs. In this work we shall define different neighbourhood structures to be used on each population, taking into account their specific characteristics. We adapt the *solution graph* model from [59] to incorporate machine flexibility. A solution can be represented by an acyclic directed graph G with a node for each task of the problem, labelled with the machine to which it has been assigned, plus two nodes representing fictitious tasks *start* and *end* with null processing times. There are *conjunctive arcs* representing job precedence constraints (including arcs from node *start* to the first task of each job and arcs from the last task of each job to node *end*) and *disjunctive arcs* representing machine processing orders. Each arc is weighted with the processing time of the task at the source node (a TFN in our case) in the machine where it is to be processed.

The starting and completion times of each task can be found by propagating constraints in the graph, and the makespan will be the completion time of task

Input A FfJSP instance, P^T, P^M, *Best*
Output Fitness values for P^T's individuals and *Best* updated
  for each individual $I_p^T \in P^T$ **do**
    //Decoding with Cooperative partners
    $S_1 \leftarrow$ Decode(I_p^T, Best solution from P^M);
    $S_2 \leftarrow$ Decode(I_p^T, Random solution from P^M);
    $S_3 \leftarrow$ Decode(I_p^T, $I_p^M \in P^M$);
    //Intensification phase
    Apply Local search to S_1, S_2 and S_3;
    $S \leftarrow \arg\min_{j=1,2,3}\{C_{max}(S_j)\}$;
    $fitness(I_p^T) \leftarrow C_{max}(S)$;
    Update I_p^T chromosome with S;
    if $C_{max}(S) < C_{max}(Best)$ **then**
      $Best \leftarrow S$;
  return $fitness(P^T)$,*Best*

Algorithm 3: Evaluation Algorithm for P^T Population using cooperative partners of P^M

end (which may not coincide with the completion time of any job). In the crisp case, the makespan corresponds to the cost of a critical path, which is defined as the longest path in a solution graph from node *start* to node *end*. It is not trivial to extend concepts and algorithms related to criticality to the problem with uncertain durations (cf [23], [3]). Here we adopt the definition from [59], where it is proposed that a solution graph G be decomposed into three *parallel solution graphs* G^i, $i = 1, 2, 3$, with identical structure to G but where the cost of any arc is the i-th component of the TFN labelling that arc in G. The union of all critical paths in G^i $i = 1, 2, 3$ will be the set of critical paths in G and *critical nodes and arcs* will be those within a critical path. Finally, a *critical block* is a maximal subsequence of tasks of a critical path assigned to the same machine. The makespan of the schedule is not necessarily the cost of a critical path in G, but it holds that each component C^i_{max} is the cost of a critical path in the corresponding solution parallel graph G^i.

For population P^M, representing machine assignments, based on the work of [60] and [61] for other variants of fJSP, we build a neighbour by taking a critical task θ_{ij} and assigning it to a new random machine $M_k \in R_{ij}$. The resulting neighbours are always feasible, so no repair strategy is needed. The evaluation of neighbours and, hence, the cost of the local search procedure, is optimised by using makespan estimates in the line of [61], adapted to the fuzzy context.

Regarding population P^T, aimed at finding good task orderings, the local search assumes a fixed machine assignment (provided by the cooperative partner). This allows to use the structure for fuzzy job shop from [62], where a neighbour is built by reversing a critical arc at the extreme of a critical block; the motivation for this definition is that reversing critical arcs preserves feasibility and, additionally, reversing arcs inside critical blocks does not improve the makespan. Again, the evaluation of neighbours and consequent cost of the local

search procedure is optimised using makespan estimates, as proposed in [62].

6. Experimental study

The goal of this experimental study is twofold. The first objective is to evaluate the hybrid co-evolutionary algorithm proposed, analysing the contribution of each of its components and comparing its behaviour with the state-of-the-art methods. The second objective is to study the influence of the ranking method in the robustness of the solutions.

Experiments are made on the instances that are available in the literature for the FfJSP which are, to our knowledge, the four instances proposed in [10] (denoted 01–04), and the two instances proposed in [14] (denoted 05, 06). All instances have $m = 10$ machines. Instances 01–04 have 10 jobs each, with a total of 40 tasks for the first two instances 01, 02 and 50 tasks for instances 03 and 04, while instances 05 and 06 have 15 jobs and 80 tasks each. Despite the relatively low number of tasks, the difficulty of the benchmark is considerable. The reason is that all instances have full flexibility, meaning that every task can be performed in any machine with varying processing time, which significantly increases the size of the search space.

Our hybrid algorithm (denoted CELS hereafter) has been implemented in C++ on a PC with a Xeon E5520 processor and 24 Gb RAM. After some preliminary testing, the parameters have been set as follows: 50 individuals per population and 100 generations as stopping criterion, crossover probability equal to 0.90 and mutation probability equal to 0.05 for both populations.

6.1. Analysis of CELS's performance

In the experiments devoted to evaluate the proposed algorithm, to keep the experimentation within reasonable bounds and to be in line with the existing works for FfJSP in the literature, we will restrict ourselves to the ranking method based on $E_{\frac{1}{2}}$. Even though the ranking methods considered in this work have been used as such by other authors for solving the FfJSP, the state-of-the-art methods for this problem do use a ranking method based on $E_{\frac{1}{2}}$, which is combined with other defuzzification indices to break ties, as originally proposed by [24]. In the remaining of this section, we might refer to $E_{\frac{1}{2}}(C_{max})$ as *expected makespan* and, since no confusion is possible, for the sake of a simpler notation we will drop the subindex and simply write $E(C_{max})$. As a reference for the quality of a solution of a given instance of FfJSP, we will use a lower bound of the expected makespan given by $LB = E(\max_j\{\sum_{i=1}^{n} p^*_{ij}\})$ where $p^*_{ij} = \min\{p_{ijk}, k \in R_{ij}\}$.

A first set of experiments is devoted to analysing the different components of our algorithm. To evaluate the heuristic seeding we generate the initial pool of solutions using two different methods, the first one applying the heuristic introduced in Section 4.2 and the second one generating the solutions at random. We then evaluate the quality of the resulting chromosomes in terms of expected makespan. A summary of the results can be seen in Table 1. The first column

Inst	LB	pBKS	RP	HP	CCEA	LS	CELS
01	28.50	30.00	62.48	32.30	30.25	28.75	28.50
			(83.53)	(37.18)	(31.15)	(29.51)	(28.53)
02	45.00	45.25	81.98	47.80	45.75	45.25	45.25
			(107.39)	(54.53)	(46.60)	(45.38)	(45.25)
03	43.50	47.50	90.88	50.23	47.00	45.25	43.50
			(114.11)	(56.95)	(47.63)	(45.86)	(44.18)
04	33.50	37.75	76.25	39.25	37.25	36.00	34.25
			(95.22)	(44.85)	(38.28)	(36.51)	(35.08)
05	37.50	62.00	110.18	63.33	58.50	57.75	53.25
			(133.92)	(69.58)	(60.43)	(58.73)	(55.07)
06	40.25	63.75	103.89	61.63	57.00	55.75	52.75
			(125.26)	(68.04)	(58.50)	(57.41)	(53.93)
MRE		*Best*	*131.64*	*29.03*	*20.78*	*17.57*	*12.64*
		(Avg)	*(191.10)*	*(45.17)*	*(23.84)*	*(19.67)*	*(14.63)*

Table 1: Analysis of algorithm's components with best (average) expected makespan values obtained in each case.

corresponds to the instance id and for reference the second column reports the lower bound LB. The best-known solution so far (pBKS) is included in the third column, while the fourth and fifth columns report the best (average) expected makespan for both pools of initial solutions. We can observe a considerable gain in quality for the heuristic solutions: the average makespan mean relative error (MRE) w.r.t. LB is reduced in average 76% across all instances when HP is considered instead of RP.

We now evaluate the contribution of the cooperative co-evolutionary algorithm (CCEA) and the local search procedure (LS). To this end, CCEA is run with no local search for the same time taken by CELS. Additionally, since CELS uses two populations of size 50 and evolve for 100 generations, we evaluate LS by generating two populations of 500 individuals and applying LS to the resulting populations (three searches per chromosome, one per cooperative partner). The last three columns in Table 1 report the best (average) expected makespan values obtained with the three methods CCEA, LS and CELS. We can see that CCEA improves the best expected makespan 18% w.r.t. the heuristic initial population, which means that the heuristic seeding provides a good starting point for the CCEA in terms of quality but also in terms of diversity, allowing for a proper evolution of the populations. In fact, the results of CCEA are already quite competitive w.r.t. the previously best-known solutions, especially as the problem size increases. LS obtains even better results than CCEA, with a MRE equal to 17.57% in the best case and equal to 19.67% in average. More importantly, CELS, combining both CCEA and LS, improves the best and av-

erage expected makespan in every instance, with the exception of 02, where the best makespan is equal for CELS and LS. As an added value for CELS, the runtime of LS is in average 134% greater than the runtime of CELS. We conclude that there is a synergy effect between the two metaheuristics.

We now proceed to evaluate CELS compared with the state-of-the-art in the FfJSP. From the literature we gather that the most competitive approaches to FfJSP are the co-evolutionary genetic algorithm (CGA) from [14], the swarm-based neighbourhood search algorithm (SNSA) from [13], and the hybrid artificial bee bolony algorithm (hABC) from [11].

CGA is implemented in Microsoft Visual C++ 6.0 and run on a 512 RAM 1.7G CPU PC. The population size of CGA is 150 and the maximum generation is 1000. With this configuration the CPU times ranges from 8 and 11 seconds for every one of the 20 executions. SNSA is coded in Microsoft Visual C++ 6.0 and run on a 2.G RAM 2.2G CPU PC. The size of swarm is 100 and the number of iterations is limited to 500. With this configuration they report run times between 9 and 14 seconds, in average, in every one of the 20 runs. Finally, hABC is implemented in C++ and run on 2.83-GHz PC with 3.21-GB RAM. Parameters are set as follows: population size $2 \times n \times m$, steps for local search $n \times m$ and the number of trials after which a food source cannot be further improved (Limit) 20. For each instance, the algorithm is run 20 times and in average, every run takes between 11 and 15 seconds, but the two largest instances are not included in these results.

Table 2 shows the results of 30 runs of CELS on each instance compared to the methods above. For each method it includes the makespan of the best solution (with its expected value between brackets), the average expected makespan across all solutions found in several runs, the corresponding MRE values and the average runtime of a single run in seconds. The missing rows for instances 05 and 06 correspond to the cases when the original works do not report results on these instances. In bold we highlight the best solution from all methods, marked with "^{a}" when it improves the previous best-known solution and with ^{b} when the solution is optimal, given that the lower bound is reached. We see that CELS improves the best and average values in all cases except for instance 02, where it obtains the same best expected value as SNSA. For instances 01–04 (for which all algorithms provide results), CELS reduces the MRE more than 77% in average. For instance 05, the reduction w.r.t. CGA and SNSA exceeds 38%, and on instance 06 it obtains a 46% reduction w.r.t. SNSA. Notice that solutions for instances 01 and 03 are indeed optimal, as they coincide with the lower bound.

Overall, CELS establishes new best solutions for all instances except for 02, where it obtains the same expected makespan as SNSA. Regarding the average expected makespan, not only is CELS significantly better than the other methods, but it also improves the previously known best values.

6.2. Influence of Fuzziness and Ranking Methods in Robustness

In this subsection we propose to evaluate in terms of ϵ-robustness the behaviour of the predictive schedules obtained using different ranking methods

Instance	Algor.	$Best(C_{max})$	$AvgE$	MRE		Time
(LB)		$(E(Best(C_{max}))$)		Best	Avg	(s.)
01	CGA	21,29,41 (30.00)	33.18	5.26	16.40	8.3
(28.50)	SNSA	21,29,42 (30.25)	31.68	6.14	11.14	8.7
	hABC	19,30,43 (30.50)	32.15	7.02	12.81	9.9
	CELS	**21,28,37 (28.50)**^{a,b}	**28.53**	**0.00**	**0.09**	1.9
02	CGA	32,47,57 (45.75)	47.45	1.67	5.44	8.3
(45.00)	SNSA	**35,43,60 (45.25)**	47.05	**0.56**	4.56	8.9
	hABC	33,46,58 (45.75)	47.70	1.67	6.00	10.9
	CELS	**32,46,57 (45.25)**	**45.25**	**0.56**	**0.56**	2.3
03	CGA	34,47,63 (47.75)	51.00	9.77	17.24	10.7
(43.50)	SNSA	36,46,62 (47.50)	51.25	9.20	17.82	11.4
	hABC	33,47,64 (47.75)	50.70	9.77	16.55	14.8
	CELS	**31,43,57 (43.50)**^{a,b}	**44.18**	**0.00**	**1.55**	3.0
04	CGA	26,37,51 (37.75)	40.80	12.69	21.79	10.8
(33.50)	SNSA	26,39,53 (39.25)	41.45	17.16	23.73	11.5
	hABC	23,38,53 (38.00)	40.45	13.43	20.75	13.9
	CELS	**24,33,47 (34.25)**^{a}	**35.08**	**2.24**	**4.73**	2.7
05	CGA	42,62,82 (62.00)	65.95	65.33	75.87	23.9
(37.50)	SNSA	40,65,93 (65.75)	68.53	75.33	82.73	14.2
	CELS	**35,53,72 (53.25)**^{a}	**55.07**	**42.00**	**46.84**	6.7
06	SNSA	46,63,83 (63.75)	65.65	58.39	63.11	14.4
(40.25)	CELS	**35,52,72 (52.75)**^{a}	**53.93**	**31.06**	**34.00**	6.7

Table 2: Summary of results in FfJSP instances with best-known solutions in **bold**. ^{a} improves previous best known solution, ^{b} optimal solution is reached.

for fuzzy numbers. In our case, for each instance we obtain three predictive schedules, namely the schedules obtained after solving the fuzzy problem with CELS using each of the three fuzzy ranking methods from Section 2.2. The surrogate robustness of each predictive schedule is computed as explained in Section 3. The $\overline{\epsilon}$ value for each ranking method will be denoted $\overline{\epsilon}_0, \overline{\epsilon}_{\frac{1}{2}}$ and $\overline{\epsilon}_1$, corresponding respectively to $\beta = 0$, modelling the pessimistic decision maker, $\beta = \frac{1}{2}$, the compromising decision maker, and $\beta = 1$, the optimistic decision maker.

We run CELS 30 times on each instance for each ranking method, thus obtaining 30 predictive schedules. Each predictive schedule is then evaluated via a Monte-Carlo simulation with $K = 1000$, yielding a total of 30 $\overline{\epsilon}_\beta$ values for each ranking method ($\beta = 0, \frac{1}{2}, 1$). Figure 1 depicts for each instance, the average $\overline{\epsilon}_\beta$ value of each ranking method across the 30 predictive schedules, while the box plot in Figure 2 illustrates in more detail the distribution of ϵ-values obtained across the 30 predictive schedules for instance 06.

The predictive schedules obtained from the fuzzy problem using a compromising approach to rank fuzzy numbers appears to be the best option in terms of robustness (with smaller prediction error ϵ), being the pessimistic approach also quite good and clearly much better than the schedule obtained from the optimistic one. This behaviour has a natural explanation: in scheduling, every single delay in a critical task increases in the same quantity the makespan, whereas a shorter processing time of a critical task is likely to derive in this task being critical no more, thus having a small or non existing impact in the makespan which might be determined by a new critical path.

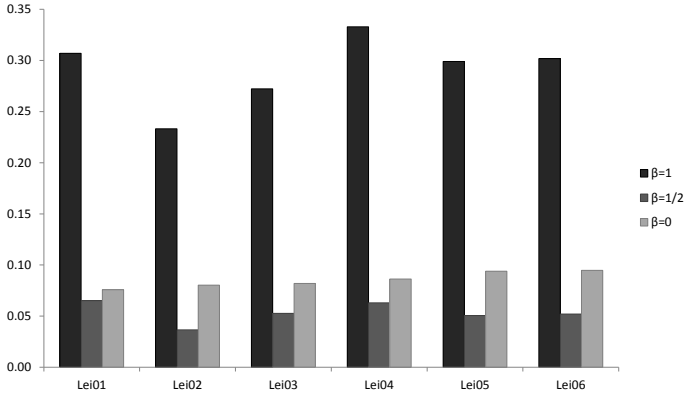

Figure 1: Mean $\overline{\epsilon}_\beta$ values for the different ranking methods

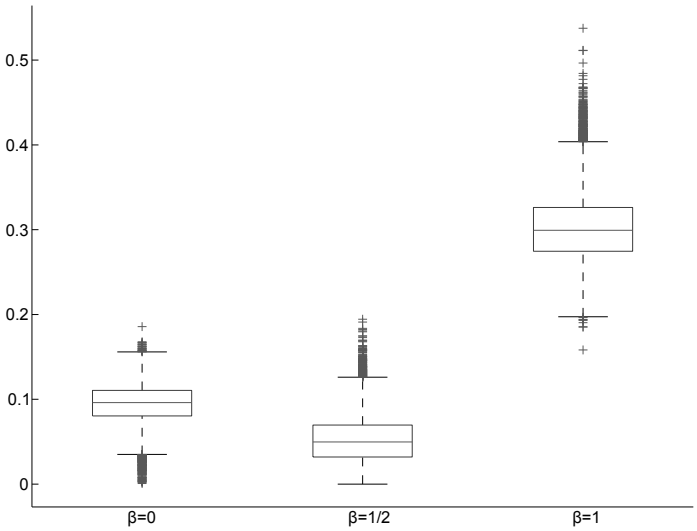

Figure 2: $\overline{\epsilon}_\beta$ values for the different ranking methods on instance 06

To enhance the conclusions of the experimental comparison based on ϵ-

robustness among solutions obtained with different raking methods, we have conducted some statistical analysis on instance 06, which seems to be the hardest one. We take for each value of β, the ϵ-robustness values $\overline{\epsilon}_\beta$ given by 30 runs of CELS. In a preliminary analysis, the Kolmogorov-Smirnov test rejected the hypotheses of normality so we have used non-parametric statistical techniques. Specifically, we have performed a Friedman test to rank the three different sets of data (corresponding to $\overline{\epsilon}_\beta$ for $\beta = 0, \frac{1}{2}, 1$) and deduce whether there are significant differences among the robustness of solutions for varying β. With a p-value ≈ 0, the mean rank values are 1.873 for $\beta = 0$, 1.127 for $\beta = \frac{1}{2}$ and 3.000 for $\beta = 1$, which show that the ranking method used for TFNs has a significant influence on the robustness of the solutions. Additionally, to have a pairwise comparison, we have made a Mann-Whitney-U test over every pair of samples: $\beta = 0$ vs. $\beta = \frac{1}{2}$, $\beta = 0$ vs. $\beta = 1$, and $\beta = \frac{1}{2}$ vs. $\beta = 1$ with p-value ≈ 0 in all cases. It is clear that ranking TFNs with expected value ($\beta = \frac{1}{2}$, corresponding to a decision maker who compromises between optimism and pessimism) gives the most robust solutions, whereas using a ranking with an optimistic interpretation seems to be the worst choice in scheduling problems.

A final set of experiments is conducted to assess the benefit in terms of robustness of using fuzzy sets. Indeed, although modelling uncertainty seems to be a natural approach to exploit all the available information, it may happen that from a practical point of view this makes no great difference with respect to solving the deterministic problem obtained by taking only the most likely duration of each task, that is the modal values. If this were the case, then it would not be worth in practice to increase the complexity of the problem by considering fuzzy numbers. To check if this is the case, we have run some additional experiments in the same line as we did above, now comparing the predictive fuzzy schedule obtained with $E_{\frac{1}{2}}$ and the predictive deterministic schedule obtained solving the defuzzified problem, again with 30 runs of CELS. The predictive schedules obtained from the defuzzified problem have a slightly better makespan value, compared to the expected makespan of the fuzzy schedules. However, the deterministic solutions are much worse than the the fuzzy ones in terms of robustness, as illustrated in Figure 3. Indeed, the $\overline{\epsilon}$ values obtained after solving the deterministic instances are in average 73.62% worse than the $\overline{\epsilon}_{1/2}$ values obtained with the fuzzy schedules as shown above. We conclude that, even though considering the defuzzified problem in principle seems to yield to better solutions, in fact these solutions are not robust enough and the outcome of a real execution is much more unpredictable than when fuzzy information about durations is taken into account.

7. Conclusions

We have tackled the flexible job shop scheduling problem with fuzzy durations and have proposed a new cooperative co-evolutionary algorithm hybridised with local search, named CELS, to solve it. The experimental results have assessed the quality of the initial heuristic seeding and the synergy between coevolution and local search. They have also shown that CELS outperforms

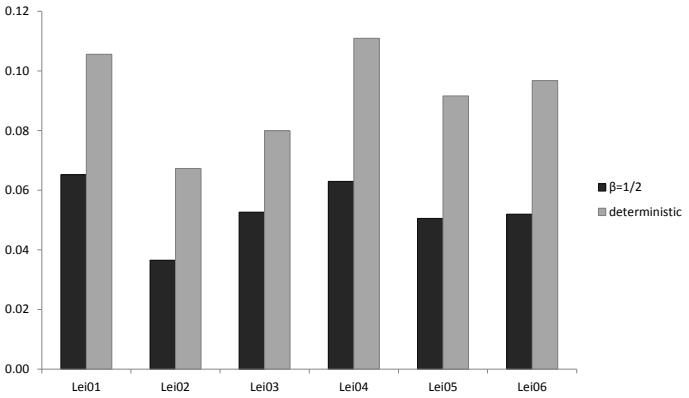

Figure 3: $\bar{\epsilon}$ values for deterministic and fuzzy predictive schedules

the state-of-the-art methods, establishing new best known solutions and, in two cases, even finding the optimal solution. In addition, we have assesed the behaviour of several ranking methods for TFNs in these problems by means of a robustness measure. This measure accounts for the average behaviour of predictive schedules in real situations. We have seen that the use of different ranking methods actually affects the robustness of the algorithm's outcome. Due to the nature of scheduling problems, this outcome is more robust when ranking is based on the expected value of the TFNs, modelling a decision maker who compromises between pessimism and optimism, followed by the approach modelling a pessimistic decision maker, with the optimistic method being definitely worse. These results strongly support the use of the expected value as a ranking method for fuzzy scheduling problems.

Acknowledgements

This research has been supported by the Spanish and Asturias Governments under grants FEDER TIN2010-20976-C02-02, MTM2010-16051 and FICYT grants FC-13-COF13-035 and BP13106.

References

[1] M. L. Pinedo, Scheduling. Theory, Algorithms, and Systems., 3rd Edition, Springer, 2008.

[2] F. Herrera, J. L. Verdegay, Fuzzy sets and operations research: Perspectives, Fuzzy Sets and Systems 90 (1997) 207–218.

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

[4] P. Fortemps, Editorial. fuzzy sets in scheduling and planning, European Journal of Operational Research 147 (2003) 229–230.

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

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

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

[8] J. J. Palacios, I. González-Rodríguez, C. R. Vela, J. Puente, Robust swarm optimisation for fuzzy open shop scheduling, Natural Computingdoi:10.1007/s11047-014-9413-1.

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

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

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

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

[13] D. Lei, X. Guo, Swarm-based neighbourhood search algorithm for fuzzy flexible job shop scheduling, International Journal of Production Research 50 (6) (2012) 1639–1649. doi:10.1080/00207543.2011.575412.

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

[15] M. A. Potter, K. A. De Jong, Cooperative coevolution: An architecture for evolving coadapted subcomponents, Evolutionary Computation 8 (1) (2000) 1–29.

[16] E. Popovici, A. Bucci, R. P. Wiegand, E. D. de Jong, Coevolutionary principles, in: Handbook of Natural Computing, Springer, 2012, pp. 987–1033. doi:10.1007/978-3-540-92910-9-31.

[17] L. Antonio, C. Coello Coello, Use of cooperative coevolution for solving large scale multiobjective optimization problems, in: 2013 IEEE Congress on Evolutionary Computation (CEC), 2013, pp. 2758–2765. doi:10.1109/CEC.2013.6557903.

[18] J. Derrac, S. García, F. Herrera, IFS-CoCo: Instance and feature selection based on cooperative coevolution with nearest neighbor rule, Pattern Recognition 43 (2010) 2082–2015. doi:0.1016/j.patcog.2009.12.012.

[19] C.-K. Goh, K. C. Tan, A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization, IEEE Transactions on Evolutionary Computation 13 (1) (2009) 103–127.

[20] Y. K. Kim, K. Park, J. Ko, A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling, Computers & Operations Research 30 (8) (2003) 1151–1171.

[21] J. Gu, M. Gu, C. Cao, X. Gu, A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem, Computers & Operations Research 37 (5) (2010) 927–937.

[22] M. Kuroda, Z. Wang, Fuzzy job shop scheduling, International Journal of Production Economics 44 (1996) 45–51.

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

[24] M. Sakawa, 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 (2000) 393–407.

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

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

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

[28] W. Van Leekwijck, E. E. Kerre, Defuzzification: criteria and classification, Fuzzy Sets and Systems 108 (1999) 159–178.

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

[30] W. Wang, Z. Wang, Total orderings defined on the set of all fuzzy numbers, Fuzzy Sets and Systems 243 (2014) 131–141.

[31] L. M. Campos Ibañez, A. González Muñoz, A subjective approach for ranking fuzzy numbers, Fuzzy Sets and Systems 29 (1989) 145–153.

[32] T. L. Liou, M. J. Wang, Ranking fuzzy numbers with integral value, Fuzzy Sets and Systems 50 (1992) 247–255.

[33] R. R. Yager, A procedure for ordering fuzzy subsets of the unit interval., Information Sciences 24 (1981) 143–161.

[34] S. Heilpern, The expected value of a fuzzy number, Fuzzy Sets and Systems 47 (1992) 81–86.

[35] P. Fortemps, M. Roubens, Ranking and defuzzification methods based on area compensation, Fuzzy Sets and Systems 82 (1996) 319–330.

[36] S. Chanas, M. Nowakowski, Single value simulation of fuzzy variable, Fuzzy Sets and Systems 25 (1988) 43–57.

[37] B. Liu, Y. K. Liu, Expected value of fuzzy variable and fuzzy expected value models, IEEE Transactions on Fuzzy Systems 10 (2002) 445–450.

[38] B. Asady, A. Zendehman, Ranking fuzzy numbers by distance minimization, Applied Mathematical Modelling 31 (2007) 2589–2598.

[39] D. Dubois, H. Prade, The mean value of a fuzzy number, Fuzzy Sets and Systems 24 (1987) 279–300.

[40] D. Dubois, Possibility theory an statistical reasoning, Computational Statistics & Data Analysis 51 (2006) 47–69.

[41] J. M. Adamo, Fuzzy decision trees, Fuzzy Sets and Systems 4 (3) (1980) 207–219.

[42] X. Wang, E. E. Kerre, Reasonable properties for the ordering of fuzzy quantitites (I), Fuzzy Sets and Systems 118 (2001) 375–385.

[43] X. Wang, E. E. Kerre, Reasonable properties for the ordering of fuzzy quantitites (II), Fuzzy Sets and Systems 118 (2001) 387–405.

[44] I. Couso, S. Destercke, Ranking of fuzzy numbers seen through the imprecise probabilistic lense, in: B. De Baets, J. Fodor, S. Montes (Eds.), Proceedings of EUROFUSE 2013, University of Oviedo, 2013, pp. 73–82.

[45] D. Dubois, The role of fuzzy sets in decision sciences: Old techniques and new directions, Fuzzy Sets and Systems 184 (2011) 3–28. doi:10.1016/j.fss.2011.06.003.

[46] K. Nakamura, Preference relation on a set of fuzzy utilities as a basis for decision making, Fuzzy Sets and Systems 20 (2) (1986) 147–162.

[47] C. Carlsson, R. Fullér, On possibilistic mean value and variance of fuzzy variables, Fuzzy Sets and Systems 122 (2) (2001) 315–326.

[48] S.-H. Chen, Ranking fuzzy numbers with maximizing set and minimizing set, Fuzzy Sets and Systems 17 (2) (1985) 113–129.

[49] S. Bodjanova, Median value and median interval of a fuzzy number, Information Sciences 172 (1) (2005) 73–89.

[50] D. Dubois, H. Prade, Unfair coins and necessity measures: Towards a possibilistic interpretations of histograms, Fuzzy Sets and Systems 10 (1983) 15–20.

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

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

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

[54] D. Dubois, H. Prade, S. Sandri, Fuzzy Logic, Vol. 12 of Theory and Decision Library, Kluwer Academic, 1993, Ch. On possibility/probability transformations, pp. 103–112.

[55] H. Aissi, C. Bazgan, D. Vanderpooten, Min-max and min-max regret versions of combinatorial optimization problems: A survey, European Journal of Operational Research 197 (2009) 427–438. doi:10.1016/j.ejor.2008.09.012.

[56] C. Bierwirth, A generalized permutation approach to jobshop scheduling with genetic algorithms, OR Spectrum 17 (1995) 87–92.

[57] I. Ono, M. Yamamura, 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, IEEE, 1996, pp. 547–552.

[58] Z. Hong, W. Jian, A cooperative coevolutionary algorithm with application to job shop scheduling problem, in: Service Operations and Logistics, and Informatics 2006, SOLI'06, IEEE International Conference on, IEEE, 2006, pp. 746–751.

[59] I. González Rodríguez, C. R. Vela, J. Puente, R. Varela, A new local search for the job shop problem with uncertain durations, in: Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS-2008), AAAI Press, Sidney, 2008, pp. 124–131.

[60] M. Mastrolilli, L. Gambardella, Effective neighborhood functions for the flexible job shop problem, Journal of Scheduling 3 (1) (2000) 3–20.

[61] M. González, C. R. Vela, R. Varela, An efficient memetic algorithm for the flexible job shop with setup times, in: Proceedings of the 23th International Conference on Automated Planning and Scheduling (ICAPS-2013), 2013, pp. 91–99.

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