Authors' version of the paper published at http://ieeexplore.ieee.org/abstract/document/4481218/
DOI: 10.1109/TSMCA.2008.918603
Reference as:
Gonzalez-Rodriguez, I., Puente, J., Vela, C. R., & Varela, R. (2008).
Semantics of schedules for the fuzzy job-shop problem. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 38(3), 655-666.

Semantics of Schedules for the Fuzzy Job Shop Problem

Inés González-Rodríguez, Jorge Puente, Camino R. Vela, and Ramiro Varela

***Abstract*— In the sequel we consider the *fuzzy job shop problem*, a variation of the job shop problem where duration of tasks may be uncertain and where due-date constraints are allowed to be flexible. Uncertain durations are modelled using triangular fuzzy numbers and due-date constraints are fuzzy sets with decreasing membership functions expressing a flexible threshold "less than". Also, the objective function is built using fuzzy decision-making theory. We propose the use of a genetic algorithm to find solutions to this problem. Our aim is to provide a semantics for this type of problems and use this semantics in a methodology to analyse, evaluate and therefore compare solutions. Finally, we present the results obtained using the genetic algorithm and evaluate them using the proposed methodology.**

***Index Terms*— Fuzzy systems, Scheduling, Genetic algorithms**

I. Introduction

In the last decades, scheduling problems have been subject to intensive research due to their multiple applications in different areas of industry, finance and science [2]. In particular, considerable effort has been made to develop heuristic strategies, for instance genetic algorithms, local search, etc, that are used as solving methods for highly complex scheduling problems such as shop problems [3].

Genetic Algorithms, or GAs, constitute one of the paradigms of general purpose search meta-heuristics. Since their appearance towards the end of the 1960s [4], they have demonstrated to be a powerful and flexible tool to confront certain problems in industry that had proved difficult if not impossible to solve with the classical methods available at the time. In particular, due to their ability to cope with huge search spaces involved in optimizing schedules [5], GAs have been successfully and widely used to solve scheduling problems [6],[7].

In most cases, scheduling has been treated as a deterministic problem that assumes precise knowledge of all data involved, such as durations, due dates, etc. Unfortunately, this is not a realistic approach in some applications. Indeed, modelling real-world problems usually involves processing uncertainty and flexibility, either due to a lack of knowledge relating to concepts, to inherent vagueness in concepts themselves or to a relaxation in certain constraints. Many papers have been published recently that use evolutionary strategies to solve flexible and dynamic scheduling problems. Research on evolutionary optimisation in presence of uncertainty is reviewed in [8] in a unified framework: uncertainty appearing in evolutionary computation is classified in four classes and several references are included that justify that evolutionary strategies "perform better than several local search methods on noisy environments". A recent paper [9] considers the dynamic job shop problem, a variant of JSSP where new orders may arrive dynamically over the time and have to be integrated into the schedule. The authors propose and compare two solving methods: a randomized priority rule-based approach and an evolutionary algorithm. In [10], a GA with dominant genes is proposed to deal with distributed scheduling problems, specially in flexible manufacturing systems environments; compared with other classical heuristic approximations to scheduling problems, the GA obtains satisfactory results. In [11] and [12], other GAs have been proposed to solve the flexible job shop scheduling, which allows for an operation to be processed by any machine from a given set of resources. The proposed GAs need to solve two subproblems: the first one is to assign operations to a subset of machines and the second one is to determine the processing order of jobs in each machine. In [11] the objective is to find a schedule with minimal makespan and four different codifications of the chromosomes are analysed. In [12], the objective is to both minimise the makespan and the idle machine cost, and the objective function is defined as a linear combination of both values. Two GAs are described, one for each subproblem and the presented experimental results are obtained on benchmark problems, obtained after modifying well-known job shop problems, which had been proposed in previous works.

A possible approach to modelling uncertainty and flexibility is to use fuzzy sets. This has resulted in a particular branch of scheduling, known as fuzzy scheduling [13] [14]. Here we find a great variety of approaches, connected with the three semantics of fuzzy sets. They range from representing incomplete or vague states of information to using fuzzy priority rules with linguistic qualifiers or preference modelling. It is also possible to find models combining more than one of these approaches.

There are many examples in the literature where fuzzy numbers are used to represent uncertain processing times and heuristic strategies are considered. A job shop problem is introduced in [15], where durations are modelled using 6-point fuzzy numbers, there are no due dates, and the single objective of minimising makespan is achieved based on fuzzy num-

This work was supported by MCYT-FEDER Grant TIC2003-04153. A preliminary version was presented at IWINAC2005 Conference [1].

Inés González Rodríguez is with the Department of Mathematics, Statistics and Computing, University of Cantabria, Los Castros s/n, Santander, 39005, Spain (phone: +34 942201532; email: ines.gonzalez@unican.es).

Jorge Puente, Camino R. Vela and Ramiro Varela are with the A.I. Centre and Department of Computer Science, University of Oviedo, Campus de Viesques s/n, Gijón, 33271, Spain (phone: +34 985182479, +34 985182134, +34 985182508; email: {puente,crvela,ramiro}@uniovi.es).

ber comparison using simulated annealing. In [16], uncertain durations are modelled using both triangular fuzzy numbers or λ-sets; the authors minimise the makespan by solving a crisp job shop problem that results from defuzzifying the durations. Job shop problems with triangular fuzzy numbers as processing times are also considered in [17], [18] and [19]. All papers propose the use of GAs to confront this problem but with different objectives. In [17], there is a single objective of satisfying flexible due-date constraints whilst both [18] and [19] use multiobjective GAs. In the first case, they try to maximise due-date satisfaction and minimise the makespan. In the second case, the authors concentrate on satisfying due dates and try to maximise the satisfaction degree of the average tardiness and minimise the number of tardy jobs. A simpler problem, the fuzzy flow shop scheduling problem, has been considered in various papers [14], [20],[21], with different definitions of the objective function and different methods used to solve it. In [21], the two-machine fuzzy flow shop problem is solved using Johnson's algorithm (that provides the optimal solution for a non-fuzzy problem) combined with a ranking method for triangular fuzzy numbers. In [20] different objective functions are compared to solve generic fuzzy flow shop problems with an evolutionary algorithm. In [14], two GAs are described to solve a fuzzy flow shop problem and a sequencing job problem. Finally, in [22] the authors study how the use of triangular fuzzy numbers to model uncertain durations in activity networks affects critical paths.

In the following, we shall define a job shop problem where task durations are uncertain and where due-date constraints may be flexible. A genetic algorithm will be described to solve the fuzzy job shop problem. We shall then propose a semantics for this type of problem that will provide a methodology to evaluate and, hence, compare different solutions to the same fuzzy job shop. Finally, we will analyse the solutions obtained by the GA on problems from the literature using the proposed methodology.

II. Description of the Problem

A. The Job Shop Scheduling Problem

The classical *job shop scheduling problem*, also denoted *JSSP*, consists in scheduling a set of jobs $\{J_1, \ldots, J_n\}$ on a set of physical resources or machines $\{M_1, \ldots, M_m\}$, subject to a set of constraints. There are *precedence constraints*, so each job J_i, $i = 1, \ldots, n$, consists of m tasks $\{\theta_{i1}, \ldots, \theta_{im}\}$ to be sequentially scheduled. Also, there are *capacity constraints*, whereby each task θ_{ij} requires the uninterrupted and exclusive use of one of the machines for its whole processing time. Hence, the execution of two tasks requiring the same machine cannot overlap in time. In addition, we may consider *due-date constraints*, where each job has a maximum completion time and all its tasks must be scheduled to finish before this time.

The goal of this problem is twofold: we need to find a *feasible* schedule, so that all constraints hold and then we want this schedule to be *optimal*, in the sense that its *makespan* (i.e., the time it takes to finish all jobs) is minimal.

B. Uncertain Processing Times

It is common in the literature to use fuzzy numbers to represent uncertain processing times. Fuzzy sets proved an alternative to probability distributions, which require a deeper knowledge of the problem and usually yield a complex calculus. 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 a fuzzy number. The simplest model of fuzzy interval is a *triangular fuzzy number* or *TFN*, using only an interval $[a^1, a^3]$ of possible values and a single plausible value a^2 in it. That is, for a TFN A, denoted $A = (a^1, a^2, a^3)$, the membership function takes a triangular shape completely determined by the three real numbers, $a^1 \leq a^2 \leq a^3$ as follows:

$$\mu_A(x) = \begin{cases} 0, & \text{for } x < a^1 \\ \frac{x-a^1}{a^2-a^1}, & \text{for } a^1 \leq x \leq a^2 \\ \frac{x-a^3}{a^2-a^3}, & \text{for } a^2 < x \leq a^3 \\ 0, & \text{for } a^3 < x. \end{cases} \quad (1)$$

Any real number $r \in \mathbb{R}$ can be seen as a special case of TFN; this allows us to deal with problems where tasks have both uncertain and precise processing times, which is usually the case in real-life applications.

In this framework, the *completion time* of a task is found by adding the task's duration to its starting time. Given two TFNs $A = (a^1, a^2, a^3)$ and $B = (b^1, b^2, b^3)$, *fuzzy number addition* is defined as

$$A + B = (a^1 + b^1, a^2 + b^2, a^3 + b^3) \quad (2)$$

so completion times are TFNs as well [23]. Secondly, the *starting time* for a given task θ is given by the maximum between two TFNs: the completion times of the tasks preceding θ in its job and in its resource. The *maximum* $A \vee B$ of two TFNs is obtained by extending the lattice operation max on real numbers using the Extension Principle. However, computing the membership function is not trivial and the result is not guaranteed to be a TFN. For these reasons, we approximate $A \vee B$ by the following TFN

$$A \sqcup B = (a^1 \vee b^1, a^2 \vee b^2, a^3 \vee b^3). \quad (3)$$

This approximation was first proposed in [15] for 6-point fuzzy numbers, a particular case of which are TFNs. Both sets $A \vee B$ and $A \sqcup B$ have identical support and the modal point in $A \sqcup B$ also has full membership in $A \vee B$.

Finally, given the completion time for each job, the *makespan* C_{max} corresponds to the greatest of these TFNs. Unfortunately, neither the maximum $\vee$ nor its approximation $\sqcup$ define a total ordering in the set of TFNs. Hence, we use a method for *fuzzy number ranking* [24] that establishes a total ordering based on using the following three real numbers for each TFN A:

$$Cr_1(A) = \frac{a^1 + 2a^2 + a^3}{4}, Cr_2(A) = a^2, Cr_3(A) = a^3 - a^1. \quad (4)$$

Given these real numbers, it is possible to establish a total ordering in a set of TFNs using lexicographical ordering, as shown in Algorithm 1. For instance, let us consider four TFNs A_1, A_2, A_3, A_4, where $A_1 = (2, 5, 8), A_2 = (3, 4, 9), A_3 = (3, 5, 7)$, and $A_4 = (4, 5, 8)$. Using the first criterion $Cr_1, Cr_1(A_4) = 5.5$ and $Cr_1(A_1) = Cr_1(A_2) = Cr_1(A_3) = 5.0$. Hence, the maximal element is $A_{max} = A_4$. By the second criterion $Cr_2, Cr_2(A_1) = Cr_2(A_3) = 5$ and $Cr_2(A_2) = 4$. Thus, the minimal element $A_{min} = A_2$. The third criterion Cr_3 shows $Cr_3(A_1) = 6$ and $Cr_3(A_3) = 4$. In this way, the increasing ordering of the four TFNs is $A_2 \ll A_3 \ll A_1 \ll A_4$.

1: order the *TFNs* according to the value of Cr_1
2: **if** there are TFNs with identical value of Cr_1 **then**
3: order these TFNs using the real value Cr_2
4: **if** there are TFNs with identical value of Cr_1 and Cr_2 **then**
5: rank them using Cr_3

Alg. 1: Ranking Method for TFNs.

In practice, if due-date constraints exist, they are often flexible: we are completely satisfied if the job finishes before a delivery date d^1 and after this time our satisfaction decreases until a later date d^2, after which we are clearly dissatisfied. A common approach to modelling such satisfaction degrees is to use a fuzzy set D with linear decreasing membership function as follows:

$$\mu_D(x) = \begin{cases} 1, & x < d^1 \\ \frac{x-d^2}{d^1-d^2}, & d^1 < x \leq d^2 \\ 0, & d^2 < x. \end{cases} \quad (5)$$

According to Dubois *et al* [13], the above membership function expresses a flexible threshold "less than" and expresses the satisfaction degree $sat(t) = \mu_D(t)$ for the ending date t of the job.

When the job's completion time is a TFN C, the degree to which C satisfies the due-date constraint D is measured by the *agreement index* [25],[18],[20]

$$ai(C, D) = \frac{area(D \cap C)}{area(C)} \quad (6)$$

where, $area(D \cap C)$ and $area(C)$ denote the areas under the membership functions of $D \cap C$ and C respectively (see Figure 1). It could be the case that $D \cap C$ is not a triangle; for the sake of simplicity, we shall then approximate $area(D \cap C)$ by the area of the greatest triangle contained in $D \cap C$. This provides a lower bound for $area(D \cap C)$ and, in consequence, for the agreement index $ai(C, D)$.

C. Objective of FJSSP

Having established how to model uncertain duration times and flexible due-dates, it is possible to find a schedule for a given problem and, for every job J_i, $i = 1, \ldots, n$, we may decide to what degree its completion time C_i satisfies the flexible due-date D_i, as given by the agreement index $AI_i = ai(C_i, D_i)$. We may also obtain a fuzzy makespan, C_{max}, a TFN indicating the time when all jobs are finished. The goal of a classical job shop problem is to find a feasible

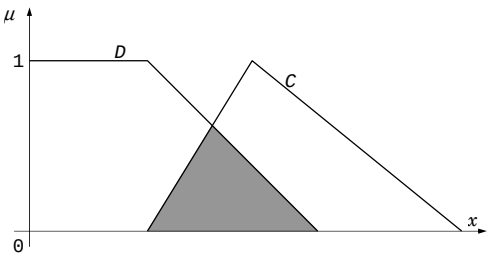

Fig. 1. Agreement Index $ai(C, D)$

schedule with a minimum makespan. Let us see how this translates into the fuzzy scheduling framework.

The agreement index AI_i measures the degree to which the due-date constraint for job J_i is satisfied for $i = 1, \ldots, n$. It seems then natural to decide on the feasibility of the given schedule by somehow combining these satisfaction degrees. We use two different aggregation operators, the average and the minimum aggregation operators, and define the following [18],[20]:

$$z_1 = AI_{av} = \frac{1}{n} \sum_{i=1}^{n} AI_i \quad (7)$$

$$z_2 = AI_{min} = \min_{i=1..n} AI_i. \quad (8)$$

The expression for z_1 can be interpreted as the probability $\Pr(F)$ of the fuzzy event F "*the schedule is feasible*" over the finite domain of jobs $D = \{J_1, \ldots, J_n\}$, provided that the membership of job J_i in F is $\mu_F(J_i) = AI_i$, $i = 1, \ldots, n$. Similarly, z_2 corresponds to asking that all due dates be satisfied and can be interpreted as the necessity measure $N(F)$ of the fuzzy event F over the finite domain D [26] [27]. For both approaches, $z_1 \in [0, 1]$ and $z_2 \in [0, 1]$ measure the degree to which due-date constraints are satisfied and our aim should be to maximise both.

Another goal is to minimise the makespan of the proposed schedule, $C_{max} = \max_{i=1,\ldots,n} C_i$, where max denotes the maximum obtained using the ranking method. If we consider the total ordering defined by the proposed ranking method, this would mean to minimise the defuzzified makespan $Cr_1(C_{max})$:

$$z_3 = Cr_1(C_{max}). \quad (9)$$

Overall, we have three different goals: maximise z_1 and z_2 (maximise feasibility) and minimise z_3 (minimise makespan). Hence, given a set of possible schedules S, we need to find a schedule $s \in S$ satisfying the following goals:

$$G_1 : \text{maximise } z_1 = \frac{1}{n} \sum_{i=1}^{n} AI_i \quad (10)$$

$$G_2 : \text{maximise } z_2 = AI_{min} = \min_{i=1,\ldots,n} (AI_i) \quad (11)$$

$$G_3 : \text{minimise } z_3 = Cr_1(C_{max}). \quad (12)$$

If we solve this problem in the framework of fuzzy decision making [28], the degree to which a given schedule $s \in S$ satisfies the three goals G_1, G_2, G_3 would be given by:

$$\mu_D(s) = \min\left(\mu_{G_1}(s), \mu_{G_2}(s), \mu_{G_3}(s)\right) \quad (13)$$

where μ_{G_i} represents the degree to which s satisfies goal G_i, $i = 1, 2, 3$. Our aim is to find a schedule $s \in S$ maximising this satisfaction degree $\mu_D(s)$.

For the problem to be well-posed, the satisfaction degrees μ_{G_i} must be defined. For the first two goals G_1 and G_2, if $z_i = 0, i = 1, 2$, our satisfaction degree should be null, and if $z_i = 1, i = 1, 2$ we should be totally satisfied. Moreover, as z_i increases from 0 to 1, satisfaction should also increase. Hence, the satisfaction degrees for the first two goals are given by $\mu_{G_i}(s) = \mu_i(z_i), i = 1, 2$, where $\mu_i : [0, 1] \to [0, 1]$ is an increasing function such that $\mu_i(0) = 0, \mu_i(1) = 1$, $i = 1, 2$. Its exact definition should be dependent on the nature of the scheduling problem. Ideally, μ_i should be elicited by an expert. In practice, such an expert might not be available; in this case, we take a standard approach in the literature [14],[18] and use a simple linear function of the form:

$$\text{for } i = 1, 2 \; \mu_i(z_i) = \begin{cases} 0, & z_i \leq z_i^0, \\ \frac{z_i - z_i^0}{z_i^1 - z_i^0}, & z_i^0 < z_i < z_i^1, \\ 1, & z_i \geq z_i^1 \end{cases} \tag{14}$$

where $z_i^0 < z_i^1$ represent the values providing minimum and maximum satisfaction respectively (see Figure 2). The adequate values for these two parameters need then to be obtained in the experimentation process. Notice that, in the absence of an expert, this is, in itself, a hard problem which, to our knowledge, has not been considered in the literature.

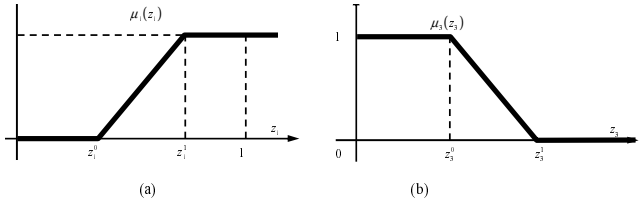

Fig. 2. Satisfaction degrees for the objectives: (a) $\mu_{G_i}(s) = \mu_i(z_i)$ for $i = 1, 2$ and (b) $\mu_{G_3}(s) = \mu_3(z_3)$.

Finally, the satisfaction degree for the third goal G_3 is given by $\mu_{G_3}(s) = \mu_3(z_3)$ where $\mu_3 : [0, \infty] \to [0, 1]$ is some decreasing function. Here, the exact definition of μ_3 is even more dependent on the nature of the problem and ideally should also be elicited by some expert. In practice, if no expert is available, we take a standard approach as above and μ_3 is defined as a lineal function of the form:

$$\mu_3(z_3) = \begin{cases} 1, & z_3 \leq z_3^0, \\ \frac{z_3 - z_3^1}{z_3^0 - z_3^1}, & z_3^0 < z_3 < z_3^1, \\ 0, & z_3 \geq z_3^1 \end{cases} \tag{15}$$

where $z_3^0 < z_3^1$ represent the values providing maximum and minimum satisfaction respectively (see Figure 2). Again, the adequate values for these two parameters need to be obtained in the experimentation process.

The above definition of the satisfaction degrees μ_{G_i}, $i = 1, 2, 3$ may be incorporated to the expression in (13), so the *objective function* of the FJSSP is:

$$f(s) = \min\{\mu_1(z_1), \mu_2(z_2), \mu_3(z_3)\} \tag{16}$$

where s is a possible schedule. The solution to the FJSSP will be a schedule maximising the value of such objective function.

III. Solving FJSSP using Genetic Algorithms

Essentially, GAs perform an stochastic search in the space of possible solutions to a given problem. A GA starts with an initial population, where individuals represent potential solutions to the problem. This population undergoes an evolution process, where new (and hopefully better) populations are iteratively generated. At each stage of the process, evolution is modelled via selection, crossover and mutation of individuals. To perform selection, the quality of each individual, i.e. the degree to which the individual has adapted itself to the environment, is evaluated according to a fitness function. If the algorithm converges adequately, the average quality of individuals will increase with the successive generations. In order to characterise a GA, we need to define its key aspects, namely, the codification of individuals as chromosomes, the generation of the initial population, the evaluation of individuals (fitness function), and the operations of selection, crossover and mutation.

Let us describe a possible GA for the FJSSP, inspired by that proposed in [18] and based in finding active schedules as described in the following.

A. Generation of Active Schedules

In classical JSSP, the search for an optimal schedule is usually limited to the space of active schedules. A schedule is *active* if one task must be delayed for any other one to start earlier. One of the best-known algorithms to find active schedules is the *G&T Algorithm* [29], which allows to use complementary techniques to reduce the search space [7]. Also, it can be used as a basis for efficient GAs successful in solving job shop problems. In Algorithm 2 we propose a possible extension of G&T. It should be noted nonetheless that, due to the uncertainty in task durations, we cannot guarantee that the produced schedule will indeed be active when it is actually performed (and tasks have exact durations). We may only say that the obtained fuzzy schedule is possibly active. Throughout the algorithm, given any task θ, its starting time will be denoted by $ST(\theta)$ and its duration will be denoted by $du(\theta)$. Also, remember that its completion time is simply a TFN given by:

$$C(\theta) = (C(\theta)^1, C(\theta)^2, C(\theta)^3) = ST(\theta) + du(\theta). \tag{17}$$

Notice that at certain stages of this fuzzy G&T, some heuristic decisions have been taken and other alternatives might as well be implemented. In general, we have opted for the alternatives with low computational cost. The reason is the intensive use that the GA makes of G&T algorithm. The criterion applied to select a task from the conflict set depends on the use of the algorithm. When it is used to generate the initial population, the task is chosen at random and, when it is used in the crossover operator, the task is chosen according to the information given by the parents, as we shall see.

1: $A = \{\theta_{i1},\ i = 1, \ldots, n\}$; /*first task of each job*/
2: **while** $A \neq \emptyset$ **do**
3: Find the task $\theta' \in A$ with minimum earliest completion time /*$C(\theta')^1$*/;
4: Let M' be the machine required by θ' and B the subset of tasks in A requiring machine M';
5: Remove from B any task θ that cannot overlap with θ'; /*$ST(\theta)^1 > C(\theta')^3$*/
6: Select $\theta^\star \in B$ according to some criterion (e.g. randomly) to be scheduled;
7: Remove $\theta^\star$ from A and insert in A the task following $\theta^\star$ in the job if $\theta^\star$ is not the last task of its job;

Alg. 2: Fuzzy G&T

B. A Genetic Algorithm for FJSSP

The GA used in this paper to solve the FJSSP was first proposed in [18]. Chromosomes are a direct codification of schedules and the *fitness function* for a chromosome is given by the expression for the objective function in (16), assuming that satisfaction degrees μ_i, $i = 1, 2, 3$ are known. Crossover and mutation (with probability pc and pm respectively) are based on the extended G&T. Let us suppose that two individuals have been selected to cross and a new individual is being generated using fuzzy G&T. Each time the conflict set B has more than one element (step 6 of Algorithm 2), one of the parents I is selected at random and for each task $\theta_i \in B$, $i = 1, \ldots, |B|$, we consider its completion time in that parent, $C(\theta_i|I)$. In total, there are $|B|$ completion times, TFNs that can be ranked according to Algorithm 1. The task with the minimum completion time is the one selected from the conflict set. The mutation operator is embedded in the crossover operator, so with a given probability, the task from the conflict set B is selected at random.

The extended G&T is also used to generate initial individuals. These are incorporated to the initial population only if similarity to other chromosomes is less than a given threshold σ, thus ensuring diversity. The *similarity between two individuals* I_1 and I_2 is defined as:

$$Sim(I_1, I_2) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(|Pr_{I_1\cap I_2}(\theta_{ij})| + |Su_{I_1\cap I_2}(\theta_{ij})|\right)}{n \cdot m \cdot (m-1)} \quad (18)$$

where $|Pr_{I_1\cap I_2}(\theta_{ij})|$ denotes the cardinal of $Pr_{I_1}(\theta_{ij}) \cap Pr_{I_2}(\theta_{ij})$ and $|Su_{I_1\cap I_2}(\theta_{ij})|$ denotes de cardinal of $Su_{I_1}(\theta_{ij}) \cap Su_{I_2}(\theta_{ij})$, being $Pred(\theta|I)$ the set of tasks preceding θ in its machine given the ordering induced by individual I and $Su(\theta|I)$ the set of tasks following θ in its machine given that ordering. Using this similarity measure, Sakawa and Kubota [18] recommend to use a similarity threshold of 0.8 when generating the initial population.

Finally, the GA uses a niche-based system to avoid premature convergence: the population is initially divided in sub-populations, that evolve separately for I_{min} generations (termination condition T_1 in Algorithm 3) and are later merged into a single population of N individuals, to further evolve for a total of I_{max} generations (termination condition T_2 in Algorithm 3).

A pseudo-code description of the resulting GA can be seen in Algorithm 3.

1: Generate initial population divided in K groups $P_1, \ldots, P_K$ containing k individuals each;
2: **while** terminating condition T_1 is not satisfied **do**
3: **for** $i = 1; i \leq K; i{+}{+}$ **do**
4: **repeat**
5: select 2 parents from P_i at random;
6: obtain 3 children by crossover and mutation;
7: select the best of 3 children and the best of remaining children and parents for the new population NP_i;
8: **until** a new population NP_i is complete
9: Replace the worst individual in NP_i with the best of P_i.
10: Join $P_1, \ldots, P_K$ into a single population P;
11: **while** Terminating condition T_2 is not satisfied **do**
12: Obtain a new population from P following the scheme above;

Alg. 3: Genetic Algorithm for FJSSP

IV. Semantics for a Solution to the FJSSP

A. Motivation

In the traditional deterministic framework, a solution to the JSSP provides an ordering according to which tasks are to be performed and consequently a time-schedule for these tasks. This provides an exact makespan for the whole project, as well as crisp completion times for each job. It is possible to compare two solutions in terms of makespan and check whether due-date constraints are satisfied, i.e., whether a given solution is indeed a feasible one.

As in the deterministic case, a solution to the FJSSP also provides an ordering according to which tasks are to be performed. However, it cannot provide a precise time-schedule for the tasks. Instead, it gives a fuzzy time-schedule, a possibility distribution on the values of each time-related parameter. It becomes more difficult to evaluate the quality of a solution and, hence, to compare different solutions to the same problem in order to select the best one.

The value of the objective function is the degree to which a solution satisfies the three objectives. However, this value is strongly dependent on the definition of the satisfaction degrees, so it is only informative as long as we know and agree with such definitions. Even in this case, it is possible to obtain two different solutions to the FJSSP with the same value of the objective function.

A solution to the FJSSP also provides a completion time for each job in the form of a TFN. This is a possibility distribution on the values that the completion time might take. To obtain an estimation for the makespan, a ranking method must be used. Unfortunately, different ranking methods yield different estimations for the makespan. Moreover, to compare the makespan of different solutions, a ranking method must be used again and depending on the chosen method, a solution might be considered to be better or worse than other. In consequence, our estimations of makespan and our choice of a solution over other are strongly dependent on the ranking method used (see [30] for a study of the influence of ranking methods). To illustrate this dependency, let us recall the example from SectionII-B. Three criteria were applied in ascending order,

Cr_1, Cr_2, Cr_3, and the obtained total ordering for the TFNs was $A_2 \ll A_3 \ll A_1 \ll A_4$. However, if the three criteria are applied in a different order Cr_3, Cr_2, Cr_1, the new ordering of TFNs becomes $A_3 \ll A_4 \ll A_2 \ll A_1$.

In summary, for a given solution we may evaluate our satisfaction degree (knowing the definition of μ_i, $i = 1, 2, 3$) and we may obtain an estimation of the completion time for each job. These certainly help us to interpret the solution. However, this information is not enough to fully evaluate a single solution or to discriminate between different ones. The need arises of further studying a solution to obtain a thorough understanding of it.

B. Interpretation as A-priori Solutions

Given the above, our aim is to provide some semantics for the FJSSP to analyse its solutions. The underlying concept is that solutions to the FJSSP are *a-priori solutions*, found when the duration of tasks is not exactly known. 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. Each schedule provides an ordering of tasks and, it is not until tasks are executed according to this ordering that we know their real duration and, hence, obtain a real schedule, the *a-posteriori solution* with crisp job completion times and makespan. Similar ideas although in a somewhat different framework can be found in [31].

According to this interpretation, the main interest of a solution to the FJSSP lies in the ordering of tasks that it provides a priori, when information about the problem is incomplete. Ideally, this ordering should yield good schedules in the moment of its practical use, when tasks do have real durations. For this reason, we propose to evaluate the behaviour of a solution to the FJSSP when faced with a family of N crisp job shop problems, generated from the fuzzy problem so that they can be interpreted as its realisations.

A means of simulating possible realisations of the FJSSP is to generate durations for each task at random, according to a probability distribution which is coherent with the fuzzy durations (namely, the TFNs membership functions normalized so that the additivity axiom holds). We thus obtain a job shop problem with crisp durations (the real durations of tasks). Fuzzy sets modelling due-date constraints do not represent partial ignorance. Instead, they represent preference related to finishing times. It is not until the latest possible date d^2 that a schedule becomes unfeasible (however little satisfactory it might be). For this reason, when transforming due dates into a crisp number, we choose d^2.

Having thus generated a family of N crisp versions of the FJSSP, we use them to evaluate the fuzzy solution. More precisely, a solution to the FJSSP provides an ordering of tasks ord. Given this ordering and a crisp version of the FJSSP (a possible *a posteriori* realisation), an algorithm of direct ordering (semiactive schedules building) can be used to obtain a crisp time-schedule. For such crisp schedule, we define the following quality measures:

- the *Feasibility Error*, F, the proportion of due-date constraints that do not hold;
- the *Relative Makespan Error*, E, the relative difference in time units between the obtained crisp makespan $C_{max} = \max_{i=1,\ldots,n} C_i$ and a given lower bound for the makespan LB, that is:
$$E = \frac{C_{max} - LB}{LB} \tag{19}$$
where this lower bound may be obtained with some of the existing methods from the literature, as explained in Section V.

The above provide a means of evaluating the solution to the FJSSP on a single possible realisation, a crisp JSSP. If we consider the whole family of N crisp problems, for each of them we may obtain the values of F and E, denoted by F_l and E_l, $l = 1, \ldots, N$. The overall performance of the fuzzy solution across the family of N crisp problems can be measured by the average value of F and E as follows:

$$F = \frac{\sum_{l=1}^{N} F_l}{N} \quad E = \frac{\sum_{l=1}^{N} E_l}{N} \tag{20}$$

We may now compare different solutions to the FJSSP based on feasibility (using F), based on makespan (using E) or even based on the overall achievement of both objectives (using some combination of F and E). In any case, we should bear in mind that the quality of a given ordering ord is measured on a family of problems which may be quite diverse. In fact, the greater the uncertainty in the FJSSP, the greater the variety of possible crisp realisations and hence, the diversity within the family of associated crisp JSSPs.

Let us illustrate this semantics with a toy example of size 3×3. The fuzzy durations are given by:

$$T = \begin{pmatrix} (9,13,17) & (5,8,11) & (9,11,15) \\ (5,8,9) & (3,4,5) & (4,7,10) \\ (3,5,6) & (3,4,5) & (1,3,4) \end{pmatrix}$$

where $T_{i,j}$ is the processing time of task j of job i, $\theta_{i,j}$, $i, j = 1, 2, 3$. Resource constraints are specified by the following:

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

where $M_{i,j}$ indicates the machine where task $\theta_{i,j}$ must be processed, $i, j = 1, 2, 3$. Finally, due dates are given by:

$$D_1 = (39, 48), D_2 = (34, 38), D_3 = (19, 23).$$

A solution to this problem can be found with the above GA, with fitness calculated according to (16), where the satisfaction degrees are defined by $z_1^0 = 0.6, z_1^1 = 1$, $z_2^0 = 0.0, z_2^1 = 1$, $z_3^0 = 39, z_3^1 = 54$. The solution provides the following task processing order:

$$\theta_{3,1}\theta_{2,1}\theta_{2,2}\theta_{3,2}\theta_{3,3}\theta_{2,3}\theta_{3,1}\theta_{3,2}\theta_{3,3}$$

For this solution, the objective values are $z_1 = 0.835081$, $z_2 = 0.505245$ and $z_3 = 44$, which yield a fitness value of 0.505245 according to the given satisfaction degrees.

Three possible crisp realisations of task durations may be given by matrices T_1, T_2 and T_3 as follows:

$$\begin{pmatrix} 10 & 9 & 12 \\ 8 & 4 & 9 \\ 3 & 4 & 4 \end{pmatrix} \begin{pmatrix} 11 & 7 & 14 \\ 8 & 5 & 10 \\ 4 & 4 & 3 \end{pmatrix} \begin{pmatrix} 16 & 10 & 15 \\ 9 & 4 & 9 \\ 5 & 4 & 2 \end{pmatrix}.$$

Lower bounds for the makespan of the resulting crisp job shop problems would be 39, 40 and 50 respectively. The following table shows, for each of these crisp problems, the results obtained when tasks are executed according to the above ordering: makespan C_{max}, relative makespan error E, number of respected due dates N_{dd} and feasibility error F.

Realisation	C_{max}	E	N_{dd}	F
$T1$	43	0.102	3	0
$T2$	44	0.1	3	0
$T3$	54	0.074	2	0.333
Average		0.092		0.111

V. Experimental Results

The results shown hereafter correspond to the benchmark problems from [18], three problems of size 6×6 and three problems of size 10×10. For each problem, we are given the duration of each task as a TFN, $du = (t^1, t^2, t^3)$, and the flexible due-date for each job, $D = (d^1, d^2)$. However, no definition of the satisfaction degrees used in the objective function are provided, so the problem definitions are not complete.

We shall now see how the semantics of schedules of FJSSP proposed in Section IV can be used for two different purposes. First, we shall propose a methodology to fully determine the objective function in the absence of an expert. Secondly, we will evaluate the quality of solutions obtained by the GA presented in Section III.

A. Heuristic Definition of Objective Function

Before doing any experimental analysis of the problem, it is necessary to completely define the objective function, i.e. define functions $\mu_i, i = 1, 2, 3$. In the absence of an expert who elicits them, it is necessary to somehow find the minimum and maximum satisfaction degrees z_i^0, z_i^1, $i = 1, 2, 3$, to complete the problem's statement.

According to the semantics proposed in Section IV, any solution to the FJSSP is an a-priori solution providing an ordering for the execution of tasks when durations are ill-known. This is then evaluated on a family of N crisp realisations of the FJSSP, using the overall feasibility error F and the overall makespan error E. To compute the latter, it is necessary to obtain a lower bound for the makespan. Here, we consider a relaxed crisp problem without due-date constraints and use the Branch & Bound Method [2] to find an optimal solution. This may not satisfy the due dates but provides a lower bound for the makespan of feasible solutions.

The methodology to automatically determine the satisfaction degrees μ_i, $i = 1, 2, 3$, consists in running the GA from Section III with different values of the parameters $z_i^0, z_i^1, i = 1, 2, 3$ and evaluate the fuzzy solutions, in order to select the parameter values that minimise the overall errors. For this experiments, the GA parameters are those proposed in [18] (see Table I). Due to the stochastic nature of GAs, for every fixed set of values for the parameters $z_i^0, z_i^1, i = 1, 2, 3$ we run the GA M times. Each of the M executions yields an overall feasibility error F_k and an overall makespan error E_k. The average performance of the GA is then evaluated using the following average quality measures:

$$\overline{F} = \frac{\sum_{k=1}^{M} F_k}{M}, \quad \overline{E} = \frac{\sum_{k=1}^{M} E_k}{M} \tag{21}$$

The optimal configuration would be that which simultaneously minimises the average relative makespan error $\overline{E}$ and the average feasibility error $\overline{F}$. However, it is very rare that a configuration exists optimising both criteria simultaneously. For most problems, there is a tradeoff between feasibility and makespan. For this reason, we choose to compromise between both error measures, so the optimal configuration will be that which minimises $\max(\overline{E}, \overline{F})$.

TABLE I
Parameters for the GA proposed in [18]

Problem	K	k	I_{min}	I_{max}	P_c	P_m
6×6	10	10	50	100	0.9	0.03
10×10	20	10	100	200	0.9	0.03

Figure 3 shows the average values across 20 executions of the GA of $\overline{E}$, $\overline{F}$ and f for approximately 200 different definitions of satisfaction degrees $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3)$, in problem 10×10-3. The different definitions of satisfaction degrees are obtained for varying values $z_i^0, z_i^1 \in [0, 1]$, $i = 1, 2$, and $z_3^0, z_3^1 \in [\min_{i=1}^{n}(d_i^1), \max_{i=1}^{n}(d^2)]$, where $D_i = (d_i^1, d_i^2)$ is the due date of job J_i, $i = 1, \ldots, n$. The resulting definitions $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3)$ are ordered in the coordinate axis according to the fitness value they yield. The plotted values of $\overline{E}$ and $\overline{F}$ illustrate the difficulty of simultaneously optimising both errors. It also shows that there exist different definitions of (μ_1, μ_2, μ_3) that yield maximum fitness value f but perform differently on the family of crisp problems. Here, $\overline{E}$ and $\overline{F}$ may help to select the "best" solution.

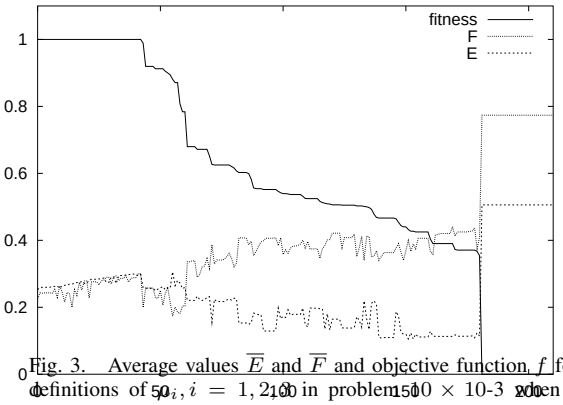

Fig. 3. Average values $\overline{E}$ and $\overline{F}$ and objective function, f, for 211 different definitions of $\mu_i, i = 1, 2, 3$ in problem 10×10-3 when $M = 20$ and $N = 50$.

Not surprisingly, there exists a strong correlation between $\overline{E}$ and goal G_3 and $\overline{F}$ and goals G_1, G_2. In all six problems, for those configurations of z_i^0, z_i^1 $i = 1, 2, 3$ with the best values for $\overline{E}$, the satisfaction function μ_3 does have influence in the fitness value, whilst in those configurations with the best

values for $\overline{F}$, it is mainly function μ_2 the one that determines the fitness value.

For our experimental results, we explored approximately 200 configurations of values for $z_i^0, z_i^1, i = 1, 2, 3$ as explained above. According to the proposed methodology, from these configurations we chose that with the minimum value of $\max(\overline{E}, \overline{F})$ as the optimal one. Interestingly, for all problems of the same size we also found two configurations, denoted $z_6^\star$ and $z_{10}^\star$, which performed very well despite not being the optimal ones. This specific configuration corresponds to that obtaining the best average performance across all problems of the same size. For $z_6^\star$ we have $z_1^0 = 0.5, z_1^1 = 1$; $z_2^0 = 0.1, z_2^1 = 0.4$; $z_3^0 = LB(t^3)$, the lower bound for the makespan of the crisp JSSP where all tasks have their maximum durations, and $z_3^1 = \max_{i=1,\ldots,n} d_i^1$, the maximum of the due-date upper bounds. For $z_{10}^\star$ we have $z_1^0 = 0.3, z_1^1 = 0.8$; $z_2^0 = 0, z_2^1 = 0.3$; $z_3^0 = LB(t^3)$ and $z_3^1 = \max_{i=1,\ldots,n} d_i^1$. Notice that, with these values, μ_2 is less strict than μ_1, which is coherent with the property $z_2 \leq z_1$. Table II contains both the optimal configuration and $z^\star$ for all problems of size 10×10.

TABLE II

OPTIMAL CONFIGURATION FOR $z_i^0, z_i^1, i = 1, 2, 3$ COMPARED WITH $z^\star$ ON PROBLEMS OF SIZE 10×10

Problem	Conf.	z_1^0	z_1^1	z_2^0	z_2^1	z_3^0	z_3^1
10×10-1	optimal	0.5	1	0	0.3	151	169
	$z^\star$	0.3	0.8	0	0.3	158	184
10×10-2	optimal	0.5	1	0	0.5	89	158
	$z^\star$	0.3	0.8	0	0.3	154	158
10×10-3	optimal	0	1	0	0.3	143	170
	$z^\star$	0.3	0.8	0	0.3	143	178

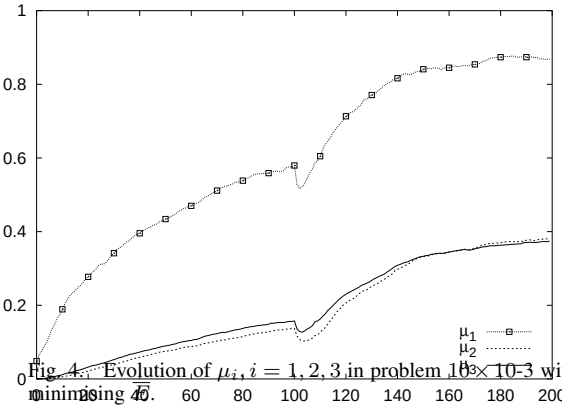

Fig. 4. Evolution of $\mu_i, i = 1, 2, 3$ in problem 10×10-3 with configuration minimising $\overline{E}$.

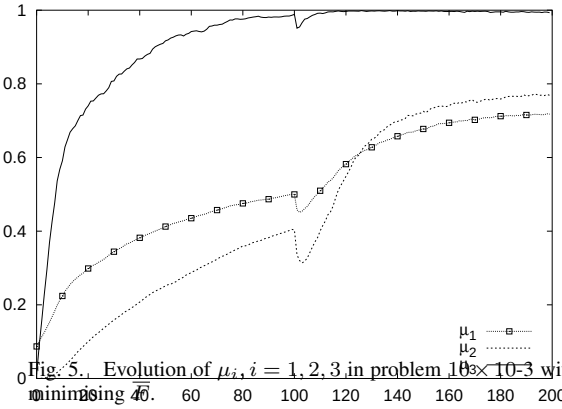

Fig. 5. Evolution of $\mu_i, i = 1, 2, 3$ in problem 10×10-3 with configuration minimising $\overline{F}$.

TABLE III

RESULTS OF THE OPTIMAL CONFIGURATION COMPARED WITH $z^\star$ ON PROBLEMS OF SIZE 10×10 (IN BOLD, THE VALUE OF $\% \max(\overline{E}, \overline{F})$.)

Problem	Conf.	$\%\overline{F}$	$\%\overline{E}$
10×10-1	optimal	17.25	**19.69**
	$z^\star$	**23.73**	18.13
10×10-2	optimal	**6.48**	6.34
	$z^\star$	10.15	**18.53**
10×10-3	optimal	25.01	**25.15**
	$z^\star$	**27.44**	26.75

Table III shows the results on problems of size 10×10 with both the optimal configuration and $z^\star$. Figures 4–7 show the average value of $\mu_i, i = 1, 2, 3$, for problem 10×10-3 on a second family of N crisp realisations (that used in Section V-C) and illustrates the evolution of these satisfaction degrees along 200 generations for four different configurations of $z_i^0, z_i^1, i = 1, 2, 3$: a) that minimising $\overline{E}$, b) that maximising $\overline{F}$, c) that minimising $\max(\overline{E}, \overline{F})$ and d) the heuristic configuration $z^\star$.

In summary, for each problem, we may obtain an optimal configuration for $\mu_i, i = 1, 2, 3$, through intensive experimentation using the methodology proposed above. Alternatively, we may use configuration $z^\star$. Despite not being the optimal one, it performs very well and avoids any further experimentation. Therefore, in the absence of an expert, we propose to use $z^\star$ to provide a heuristic definition of the satisfaction degrees for the objective function.

B. Parameter Analysis for the GA

Once the problems are completely defined, it is interesting to see if the GA parameters from Table I are the optimal ones. We have done some preliminary testing, similar to that in [32]. We performed $3 \times 3 \times 2$ factorial design of experiments involving three parameters: we considered three different values for the crossover and mutation probabilities, $p_c = 0.7, 0.9, 1$ and $p_m = 0.03, 0.05, 0.07$ and two values for the number of generations for which the population evolves in different niches, $I_{min} = 50, 100$. The remaining parameters are more dependent on the problem size, so we decided to fix them as in [18]. The GA was executed 50 times with the same seed for the random generator, in order to avoid any bias in the initial population. To determine the best combination of parameter values, we analysed the maximum of the makespan and feasibility errors obtained on a posterior evaluation on 53 crisp problems. The results showed that for a small value of p_c, the error was never too big nor were the best results obtained.

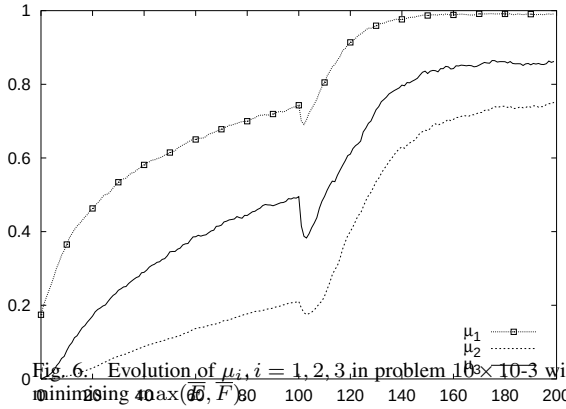

Fig. 6. Evolution of $\mu_i, i = 1, 2, 3$ in problem 10 × 10-3 with configuration minimising $\max(\overline{E}, \overline{F})$

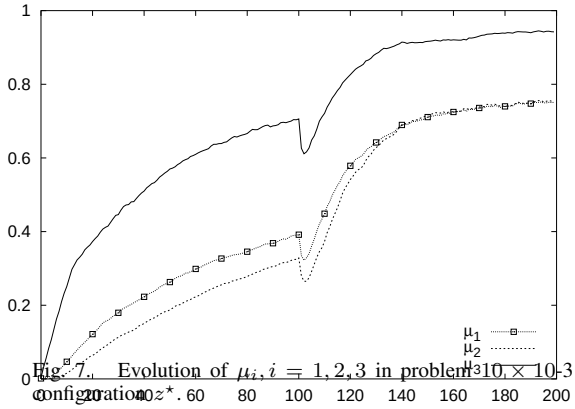

Fig. 7. Evolution of $\mu_i, i = 1, 2, 3$ in problem 10 × 10-3 with heuristic configuration $z^\star$.

It was also the case that the best value for the number of generations prior to niche fusion is different for each value of p_c. Regarding p_m, the best results are always for the largest value of this parameter ($p_m = 0.07$). Given this, we performed new experiments with p_m taking values larger or equal to 0.1, but the error would increase as the value of p_m increased. We concluded that changes in the parameters original values do not produce significant improvements in the GA solutions. Therefore, we decided to keep the parameter values proposed in [18].

C. Analysis of GA Solutions

Once the objective function has been completely defined and the GA parameters have been fixed, it is possible to obtain solutions to the FJSSP. To analyse the GA's performance, we first consider the fitness value, which is meaningful because it is accompanied by the definition of the satisfaction degrees $\mu_i, i = 1, 2, 3$. Secondly, the proposed semantics suggest evaluating the fuzzy solution in terms of its meaning as an a-priori solution, using $\overline{F}$ and $\overline{E}$.

1) Performance: Following these ideas, Tables IV and V present the results obtained with the proposed GA (labelled as FGA). They correspond to 20 executions of the GA and, for each execution, to the individual with best fitness value. Table IV includes the number of GA executions where the best fitness value is obtained *NB*, the best fitness value *Best*, the average best fitness value across the 20 executions *Av*, the worst out of the 20 best fitness values *Worst* and the variance in the best fitness value *Var*. The table also presents the fitness values obtained in [18] (denoted by *S&K*). These values are included for the sake of completeness, but it should be noticed that it is not known to which definition of the objective function they correspond (the satisfaction degrees μ_i are not known). Table V shows the values of $\overline{E}$ and $\overline{F}$ on a new family of $N = 50$ crisp problems, in order to further illustrate the quality of the obtained solutions, as well as the average due-date satisfaction, given by:

$$\overline{S} = \frac{1}{NMn} \sum_{k=1}^{N} \sum_{j=1}^{M} \sum_{i=1}^{n} \mu_{D_i}(C_i) \tag{22}$$

where C_i denotes the crisp completion time for job J_i and D_i its flexible due date, $i = 1, \ldots, n$.

TABLE IV
RESULTS OBTAINED BY THE GA (I).

Problem		NB	Best	Av.	Worst	Var.
6x6-1	S&K	18	0.775	0.761	0.628	0.002
	FGA	3	1.00	0.819	0.595	1.34E-02
6x6-2	S&K	19	0.792	0.779	0.542	0.003
	FGA	4	0.798	0.780	0.683	1.00E-03
6x6-3	S&K	20	0.580	0.580	0.580	0.00
	FGA	20	1.000	1.000	1.000	0
10x10-1	S&K	1	0.714	0.574	0.439	0.010
	FGA	14	1.000	0.977	0.831	1.90E-03
10x10-2	S&K	8	0.818	0.722	0.545	0.008
	FGA	20	1.000	1.000	1.000	0
10x10-3	S&K	1	0.560	0.525	0.475	0.003
	FGA	3	1.000	0.848	0.723	6.79E-03

Average execution time on a Pentium IV at 3Ghz:
size 6 × 6: 9 seconds (0.88 to generate the initial population)
size 10 × 10: 97 seconds (27 to generate the initial population)

TABLE V
RESULTS OBTAINED BY THE GA (II).

Problem	$\%\overline{F}$	$\%\overline{E}$	$\%\overline{S}$
6x6-1	8.95	9.02	80.12
6x6-2	0.93	6.23	95.09
6x6-3	5.22	5.26	75.94
10x10-1	23.73	18.25	62.92
10x10-2	10.15	18.68	78.18
10x10-3	27.44	26.62	55.86

2) Need of the A-Posteriori Semantics: We have argued that different solutions with equal fitness value may not be equally good a posteriori, when tasks have exact durations. This point is illustrated by variance analysis on the results of the GA. Once normality is checked in all sets of crisp problems and taking into account that these sets have been generated independently, we run an ANOVA test with factor 95% for each of the six problems. The obtained p-value

(0.00 in all 6 cases) concludes that differences in terms of relative makespan error $\overline{E}$ and degree of feasibility $\overline{F}$ among different solutions are important. The values of these quality measures differs greatly within different executions. This is not surprising, since we have already seen that low fitness values imply high errors, but high fitness values may correspond to different solutions with different errors $\overline{E}$ and $\overline{F}$. This simply confirms that two different solutions to the FJSSP may yield identical satisfaction degrees; it is then necessary to use $\overline{E}$ and $\overline{F}$ to further discriminate between them.

3) Comparison with Random Orderings: The proposed semantics assumes that the solution to the FJSSP is used to provide an ordering for the execution of tasks. Someone sceptical might argue that it would be better just to obtain a task ordering with a simpler method and avoid the cost of finding a solution to the FJSSP. However, we would expect the solution to the FJSSP to produce a "better" ordering, given that it is using all the available information. An obvious experiment is to compare the ordering provided by the solution to the FJSSP with a random ordering of tasks (the simplest ordering we can think of). Results in Table VI correspond to the average feasibility and makespan errors obtained with 100 random permutations of tasks and with $M = 20$ executions of the proposed GA. They confirm that the solution to the FJSSP obtained with the GA is indeed making use of the available information and provides a better ordering of tasks than a simple random ordering.

TABLE VI
GA VS. RANDOM ORDERINGS 10×10.

Problem	Ordering	$\%\overline{F}$	$\%\overline{E}$
10x10-1	GA	23.73	18.25
	random	81.79	47.80
10x10-2	GA	10.15	18.68
	random	85.76	40.91
10x10-3	GA	27.44	26.62
	random	79.23	46.55

4) Convergence: The average fitness value and its variance across several executions of the GA in Table IV suggest that this algorithm converges adequately with an average fitness value in all executions close to 1. To further confirm this fact, a t-test at 95% was performed for problems 10×10-1 and 10×10-3, after checking the normality hypotheses. These problems were chosen for being the only ones where the optimal solution is not always obtained. The null hypothesis H0 was "mean=0.97" for problem 10×10-1 and "mean=0.81" for problem 10×10-3. The alternative hypothesis Ha was "mean < 0.97" and "mean< 0.81" respectively. The obtained p-value (0.183 and 0.462 respectively) is too big to reject the null hypotheses. Hence, nothing seems to indicate that the fitness value converges to a value lower than 0.97 and 0.81 respectively. All data related to this statistical analysis for problem 10×10-3 can be seen in Table VII. Figure 7 further illustrates the convergence of the GA for problem 10×10-3, showing the evolution of the values of μ_i, $i = 1, 2, 3$, along the 200 generations. We observe a decrease in the satisfaction degrees when niches are merged into a single population, but only few generations are needed to recover previous satisfaction values and the slope of the evolution curves is then higher.

TABLE VII
t-TEST ANALYSIS FOR FITNESS VALUE f IN PROBLEM 10×10-3.

Parameters						
Analysis		1 Sample t	H0: Mean = 0.81			0.81
Input Variable(s)		f	Ha: Mean < 0.81			-1
			Confidence			95
t-Test Analysis						
	N	Mean	Std. Dev.	Std. Err.	t	df	p-value
f	100	0.81	0.083	0.008	-0.095	99.0	0.462

5) Comparative analysis: We have proposed a methodology to deal with uncertainty in the solutions to FJSSP obtained by the GA. Let us compare this with classical methodologies used for evolutionary algorithms in uncertain environments, as described in [8]. Although the problems considered in our work do not fall clearly in any of the described categories, according to the a-posteriori interpretation, we might consider that durations are somehow subject to changes once the optimal solution has been found. Our algorithm tries to take these changes into account in the optimisation process. In that sense, we are looking for robust solutions and, in consequence, they are not optimal solutions. This is also the case in [9], where the authors argue that "there is usually a tradeoff between the quality and robustness".

There are many possible kinds of robustness [8]. The robustness we are looking for in our solutions refers only to the optimal solution being insensitive to small variations in the task durations. It would not be natural that the solution be insensitive to variations in environmental parameters, specially for z_i^0, z_i^1, because these parameters determine the definition of the objective function. To measure the robustness of our solutions, we have studied the variation of the error for a processing order obtained by the GA on a set of 50 crisp problems. After all, these problems are a sample of possible variations in the variable values generated according to their probability distributions. More precisely, we have looked at the standard deviation for these errors, and we have obtained deviations between 2% and 4%, depending on the problem and processing order used.

In this case, fitness evaluation is similar to that defined in the presence of noise [8]. In fact, there are certain similarities between the "Explicit Averaging" methodology proposed in this paper and the one that we propose herein to evaluate solutions. In [8], a sample is generated to estimate the fitness value. Here, we generate a sample to select a task-processing order from the solutions obtained by the fuzzy optimisation algorithm.

VI. CONCLUSIONS AND FUTURE WORK

We have considered the fuzzy job shop problem or FJSSP, a job shop problem where durations of tasks are uncertain, due-dates are allowed to be flexible and where the objective function is obtained in the framework of fuzzy decision

making. We have also described a genetic algorithm from the literature that finds solutions to this problem. We have proposed a new semantics for this type of problem and its solutions, which are seen as a-priori schedules, obtained before durations of tasks can be precisely known. This semantics suggests an evaluation methodology for solutions, based on simulating possible realisations of task durations and using the obtained family of crisp problems to provide two new measures for the quality of a solution. Finally, we have shown some results on problems from the literature. These problems are not complete, in the sense that the objective function is not completely defined. For this reason, we have first used the proposed quality measures to obtain a heuristic definition of satisfaction degrees used in the objective function. Having done this, the potential of the proposed approach has been illustrated with results obtained with the genetic algorithm, analysed based on both the objective function values and the proposed evaluation methodology.

In the future, it would be interesting to obtain more benchmark problems, based on those available for the classical JSSP. These problems could be used to further test the existing methods and new ones. Moreover, the proposed semantics could be used both to design these new methods and compare the different solutions obtained. It is also possible to extend some of the proposed ideas to other scheduling problems, such as flow shop. Finally, it is our view that future work on critical path analysis and local search and heuristic methods would help to shed more light on the FJSSP, as well as other scheduling problems.

ACKNOWLEDGMENTS

We would like to thank our colleague María Sierra for her enthusiastic help. We are also grateful to the anonymous referees for their insightful comments.

REFERENCES

[1] I. González Rodríguez, C. R. Vela, and J. Puente, "An evolutionary approach to designing and solving fuzzy job-shop problems," *Proc. of IWINAC2005, Lecture Notes in Computer Science*, vol. 3562, pp. 74–83, 2005.

[2] P. Brucker, *Scheduling Algorithms*, 4th ed. Springer, 2004.

[3] P. Brucker and S. Knust, *Complex Scheduling*. Springer, 2006.

[4] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. The University of Michigan Press, 1975.

[5] D. C. Mattfeld, *Evolutionary Search and the Job Shop Investigations on Genetic Algorithms for Production Scheduling*. Springer-Verlag, 1995.

[6] C. Bierwirth and D. C. Mattfeld, "Production scheduling and rescheduling with genetic algoritms," *Evolutionary Computation*, vol. 7, pp. 1–17, 1999.

[7] R. Varela, C. R. Vela, J. Puente, and A. Gómez, "A knowledge-based evolutionary strategy for scheduling problems with bottlenecks," *European Journal of Operational Research*, vol. 145, pp. 57–71, 2003.

[8] Y. Jin and J. Branke, "Evolutionay optimization in uncertain environments–a survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, pp. 303–317, 2005.

[9] J. Branke and D. Mattfeld, "Anticipation and flexibility in dynamic scheduling," *International Journal of Production Research*, vol. 43, pp. 3103–3129, 2005.

[10] F. Chan, S. Chung, and P. Chan, "Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems," *International Journal of Production Research*, vol. 44, pp. 523–543, 2006.

[11] J. Tay and D. Wibowo, "An effective chromosome representation for evolving flexible job shop schedules," *Genetic and Evolutionay Computation Conference (GECCO), Lecture Notes in Computer Science*, vol. 3103, pp. 210–221, 2004.

[12] F. Chan, T. Wong, and P. Chan, "Flexible job-shop scheduling problemunder resource constraints," *International Journal of Production Research*, vol. 44, pp. 2071–2089, 2006.

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

[14] R. Słowiński and M. Hapke, Eds., *Scheduling Under Fuzziness*, ser. Studies in Fuzziness and Soft Computing. Physica-Verlag, 2000, vol. 37.

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

[16] F.-T. Lin, "Fuzzy job-shop scheduling based on ranking level $(\lambda, 1)$ interval-valued fuzzy numbers," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 4, pp. 510–522, 2000.

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

[18] 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*, vol. 120, pp. 393–407, 2000.

[19] 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*, vol. 3533, pp. 524–533, 2005.

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

[21] S. Petrovic and X. Song, "A new approach to two-machine flow shop problem with uncertain processing time," in *Fourth International Symposium on Uncertainty, Modeling and Analysis (ISUMA)*, B. Ayyub and N. Attoh-Okine, Eds. IEEE Computer Society, 2003, pp. 110–115.

[22] S. Chanas, D. Dubois, and P. Zieliński, "On the sure criticality of tasks in activity networks with imprecise durations," *IEEE Transactions on Systems, Man and Cybernetics–Part B: Cybernetics*, vol. 32, pp. 393–407, 2002.

[23] H. T. Nguyen and E. A. Walker, *A First Course in Fuzzy Logic*, 2nd ed. Chapman & Hall, 2000.

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

[25] A. Kaufmann and M. Gupta, *Introduction to Fuzzy Arithmetic*. New York: Van Nostrand Reinhold, 1991.

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

[27] ——, "Weighted minimum and maximum operations in fuzzy set theory," *Information Sciences*, vol. 39, pp. 205–210, 1986.

[28] R. E. Bellman and L. A. Zadeh, "Decision-making in a fuzzy environment," *Management Science*, vol. 17, no. 4, pp. 141–164, 1970.

[29] B. Giffler and G. L. Thomson, "Algorithms for solving production scheduling problems," *Operations Research*, vol. 8, pp. 487–503, 1960.

[30] M. Litoiu and R. Tadei, "Real-time task scheduling with fuzzy deadlines and processing times," *Fuzzy Sets and Systems*, vol. 117, pp. 23–45, 2001.

[31] S. Chanas and A. Kasperski, "Possible and necessary optimality of solutions in the single machine scheduling problem with fuzzy parameters," *Fuzzy Sets and Systems*, vol. 142, pp. 359–371, 2004.

[32] C. Oguz and F. Ercan, "A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks," *Journal of Scheduling*, vol. 8, pp. 321–351, 2005.

Inés González Rodríguez received the MSc degree in Mathematics from the University of Oviedo, Spain, in 1999 and the PhD degree in Artificial Intelligence from the University of Bristol, UK, in 2002. She is currently a Temporary Lecturer in the Department of Mathematics, Statistics and Computing of the University of Cantabria, Spain. Previously, she worked as Teaching Assistant at the University Rey Juan Carlos and the University of Oviedo, Spain. Her research interests include uncertainty theories, soft computing and their application to scheduling and machine learning problems.

Jorge Puente received the MEng and PhD degrees in Computer Science from the University of Oviedo, Spain, in 1995 and 2001 respectively. He is currently Assistant Professor in the Department of Computer Science and member of the Artificial Intelligence Centre of the University of Oviedo. His current research interests include soft computing, evolutionary computation and other meta-heuristics and Internet collaborative applications. He is a member of the Spanish Association for the Artificial Intelligence (AEPIA).

Camino R. Vela received the MSc degree in Mathematics from the University of Zaragoza, Spain, in 1986 and the PhD degree in Mathematics in the University of Oviedo, Spain, in 1990. She is currently Assistant Professor in the Department of Computer Science and member of the Artificial Intelligence Centre of the University of Oviedo. Her research interests include heuristic search, evolutionary computation, soft computing and other meta-heuristics techniques and their application to constraint satisfaction problems such as scheduling She is a member of the Spanish Association for the Artificial Intelligence (AEPIA).

Ramiro Varela received the MSc degree in Physics from the University of Santiago de Compostela, Spain, in 1984 and the PhD degree in Computer Science from the University of Oviedo, Spain, in 1995. He is currently Assistant Professor with the Department of Computer Science and member of the Artificial Intelligence Centre of the University of Oviedo. His research interests include heuristic search, evolutionary computation and other meta-heuristics and the application of these techniques to problem solving. He is member of the Spanish Association for Artificial Intelligence (AEPIA).