A Multiobjective Approach to Fuzzy Job Shop Problem Using Genetic Algorithms

Inés González-Rodríguez[1], Jorge Puente[2], and Camino R. Vela[2]

[1] Department of Mathematics, Statistics and Computing,
University of Cantabria, (Spain)
ines.gonzalez@unican.es
[2] A.I. Centre and Department of Computer Science,
University of Oviedo, (Spain)
{puente,crvela}@uniovi.es
http://www.aic.uniovi.es/Tc

Abstract. We consider a job shop problem with uncertain durations and flexible due dates and introduce a multiobjective model based on lexicographical minimisation. To solve the resulting problem, a genetic algorithm and a decoding algorithm to generate possibly active schedules are considered. The multiobjective approach is tested on several problem instances, illustrating the potential of the proposed method.

1 Introduction

In the last decades, scheduling problems have been subject to intensive research due to their multiple applications in areas of industry, finance and science [1]. To enhance the scope of applications, fuzzy scheduling has tried to model the uncertainty and vagueness pervading real-life situations, with a great variety of approaches, from representing incomplete or vague states of information to using fuzzy priority rules with linguistic qualifiers or preference modelling [2],[3].

The complexity of problems such as shop problems means that practical approaches to solving them usually involve heuristic strategies, such as genetic algorithms, local search, etc [1]. It is not trivial to extend these strategies to fuzzy scheduling. Indeed, incorporating uncertainty to scheduling usually requires a significant reformulation of the problem and solving methods. In the literature, we find some attempts to extend heuristic methods for job shop solving to the fuzzy case. For instance, 6-point fuzzy numbers and simulated annealing are used for single objective problem in [4], while triangular fuzzy numbers and genetic algorithms are considered for multiobjective problems in [5], [6] and [7]. The latter also proposes a semantics for solutions to job shop with uncertainty.

In the sequel, we describe a fuzzy job shop problem with uncertain durations and flexible due dates. A leximin approach is taken to define an objective function that combines minimisation of the expected fuzzy makespan and maximisation of due-date satisfaction. The resulting problem is solved by means of a genetic algorithm (GA) based on permutations with repetitions that searches in the space of possibly active schedules. We analyse the performance of the resulting multiobjective GA on a set of problem instances.

D. Borrajo, L. Castillo, and J.M. Corchado (Eds.): CAEPIA 2007, LNAI 4788, pp. 80–89, 2007.

2 Uncertain Processing Times and Flexible Constraints

In real-life applications, it is often the case that the exact duration of a task is not known in advance. However, based on previous experience, an expert may have some knowledge about the duration, thus being able to estimate, for instance, an interval for the possible processing time or its most typical value. In the literature, it is common to use fuzzy numbers to represent such processing times, as 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 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. For a TFN A, denoted $A = (a^1, a^2, a^3)$, the membership function takes the following triangular shape:

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

Two arithmetic operations on TFNs are of interest herein. The first one is *fuzzy number addition*, which in the case of TFNs $A = (a^1, a^2, a^3)$ and $B = (b^1, b^2, b^3)$ is reduced to adding three pairs of real numbers so $A + B = (a^1 + b^1, a^2 + b^2, a^3 + b^3)$. The second one is the *maximum* $A \vee B$, obtained by extending the lattice operation max on real numbers using the Extension Principle. Computing the membership function is not trivial and the result is not guaranteed to be a TFN, so in practice we approximate $A \vee B$ by a TFN, $A \sqcup B = (a^1 \vee b^1, a^2 \vee b^2, a^3 \vee b^3)$. This approximation was first proposed in [4] for 6-point fuzzy numbers, a particular case of which are TFNs. The approximated maximum can be trivially extended to the case of $n > 2$ TFNs.

When a TFN models an uncertain duration, its membership function may be interpreted as a possibility distribution on the values that the duration may take. Given this interpretation and based on credibility theory, the *expected value* [8] of a TFN A is given by $E[A] = \frac{1}{4}(a^1 + 2a^2 + a^3)$.

In practice, if due-date constraints exist, they are often flexible. For instance, customers may have a preferred delivery date d^1, but some delay will be allowed until a later date d^2, after which the order will be cancelled. The satisfaction of a due-date constraint becomes a matter of degree, our degree of satisfaction that a job is finished on a certain date. A common approach to modelling such satisfaction levels is to use a fuzzy set D with linear decreasing membership function:

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

Such membership function expresses a flexible threshold "less than", representing the satisfaction level $sat(t) = \mu_D(t)$ for the ending date t of the job [2]. When the job's completion time is no longer a real number t but a TFN C, the degree to which C satisfies the due-date constraint D may be measured using the following *agreement index* [9],[5]:

$$AI(C, D) = \frac{area(D \cap C)}{area(C)} \tag{3}$$

3 The Job Shop Scheduling Problem

3.1 Description of the Problem

The *job shop scheduling problem*, also denoted *JSP*, 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. 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. A solution to this problem is a schedule (a starting time for all tasks) which, besides being *feasible*, in the sense that due precedence and capacity constraints hold, is optimal according to some criteria, for instance, that due-date satisfaction is maximal or makespan is minimal.

A schedule s for a job shop problem of size $n \times m$ (n jobs and m machines) is fully determined by a decision variable representing a task processing order $\boldsymbol{x} = (x_1, \ldots, x_{nm})$, where $1 \leq x_l \leq n$ for $l = 1, \ldots, nm$ and $|\{x_l : x_l = i\}| = m$ for $i = 1, \ldots, n$. This is a permutation with repetition as proposed by Bierwirth [10]; a permutation of the set of tasks, where each task is represented by the number of its job. Thus a job number appears in such decision variable as many times as different tasks it has. The order of precedence among tasks requiring the same machine is given by the order in which they appear in the decision variable $\boldsymbol{x}$. Hence, the decision variable represents a task processing order that uniquely determines a feasible schedule. This permutation should be understood as expressing partial schedules for every set of operations requiring the same machine.

Let us assume that the processing time p_{ij} of each task θ_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, m$ is a fuzzy variable (a particular case of which are TFNs). Let $\boldsymbol{\xi}$ be the matrix of fuzzy processing times such that $\xi_{ij} = p_{ij}$, let $\boldsymbol{\nu}$ be a machine matrix such that ν_{ij} is the machine required by task θ_{ij}, let $C_i(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})$ denote the completion time of job J_i and let $C_{ij}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})$ denote the completion time of task θ_{ij}, $i = 1, \ldots, n$ $j = 1, \ldots, m$. Clearly, the completion time of a job is the completion time of its last task, that is: $C_i(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) = C_{im}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}), \quad i = 1, \ldots, n$. The starting time for task θ_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, m$ will be the maximum

between the completion times of the tasks preceding θ_{ij} in its job and its machine. Hence, the completion time of task θ_{ij} is given by the following:

$$C_{ij}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) = \left(C_{i(j-1)}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) \sqcup C_{rs}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})\right) + p_{ij}$$

where θ_{rs} is the task preceding θ_{ij} in the machine according to the processing order given by $\boldsymbol{x}$. $C_{i0}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})$ is assumed to be zero and, analogously, $C_{rs}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})$ is taken to be zero if θ_{ij} is the first task to be processed in the corresponding machine. Finally, the *fuzzy makespan* $C_{max}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})$ is the maximum completion time of jobs $J_1, \ldots, J_n$ as follows:

$$C_{max}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) = \sqcup_{1 \leq i \leq n} \left(C_i(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})\right)$$

3.2 A Multiobjective Model

It is not trivial to optimise a schedule in terms of fuzzy makespan, since neither the maximum $\vee$ nor its approximation $\sqcup$ define a total ordering in the set of TFNs. In the literature, this problem is tackled using some ranking method for fuzzy numbers, lexicographical orderings, comparisons based on λ-cuts or defuzzification methods. Here the modelling philosophy is similar to that of stochastic scheduling, which optimises some expected objective functions subject to some expected constraints. For this purpose, we use the concept of expected value for a fuzzy variable, so the objective is to minimise the expected makespan $E[C_{max}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})]$, a crisp value. In the absence of due-date constraints, this provides an *expected makespan model* for fuzzy job shop scheduling problems [11].

If flexible due dates D_i exist for jobs J_i, $i = 1, \ldots, n$, the agreement index $AI(C_i(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}), D_i)$, denoted $AI_i(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})$ for short, is a crisp value measuring to what degree the due date is satisfied. The degree of overall due-date satisfaction for schedule s may be obtained by combining the satisfaction degrees $AI_i(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})$, $i = 1, \ldots, n$. We may expect due dates to be satisfied in average or, being more restrictive, expect that all due dates be satisfied. The degree to which schedule s, determined by an ordering $\boldsymbol{x}$, satisfies due dates is then given, respectively, by the following:

$$AI_{av}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) = \frac{1}{n} \sum_{i=1}^{n} AI_i(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}), \quad AI_{min}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) = \min_{i=1,\ldots,n} AI_i(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) \quad (4)$$

Clearly, both $AI_{av}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})$ and $AI_{min}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})$ should be maximised. Notice however that they model different requirements and encourage different behaviours.

In order to maximise both measures of due-date satisfaction and minimise the expected makespan, we may formulate a multiobjective problem as a fuzzy goal programming model according to a priority structure and target levels established by the decision makers as follows:

Priority 1. $f_1(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) = E[C_{max}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})]$ should be minimised and should not exceed a given target value b_1, i.e. we have the following goal constraint:

$$f_1(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) + d_1^- - d_1^+ = b_1 \quad (5)$$

where d_1^+, the positive deviation from the target, should be minimised.

Priority 2. $f_2(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) = AI_{av}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})$ should be maximised and should not be less than a given target value b_2, i.e. we have the following goal constraint:

$$f_2(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) + d_2^- - d_2^+ = b_2 \tag{6}$$

where d_2^-, the negative deviation from the target, should be minimised.

Priority 3. $f_3(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) = AI_{min}(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})$ should be maximised and should not be less than a given target value b_3, i.e. we have the following goal constraint:

$$f_3(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu}) + d_3^- - d_3^+ = b_3 \tag{7}$$

where d_3^-, the negative deviation from the target, should be minimised.

Thus, we have the following *lexmin scheduling model* for the fuzzy job shop problem (FJSP):

$$\begin{cases} \text{lexmin} & (d_1^+, d_2^-, d_3^-) \\ \text{subject to:} & \\ & f_i(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\nu})] + d_i^- - d_i^+ = b_i, \quad i = 1, 2, 3, \\ & b_i \geq 0, \quad i = 1, 2, 3, \\ & d_i^-, d_i^+ \geq 0, \\ & 1 \leq x_l \leq n, \quad l = 1, \ldots, nm, \\ & |\{x_l : x_l = i\}| = m, \quad i = 1, \ldots, n, \\ & x_l \in \mathbb{Z}^+, \quad l = 1, \ldots, nm. \end{cases} \tag{8}$$

where lexmin denotes lexicographically minimising the objective vector.

4 Using Genetic Algorithms to Solve FJSP

The crisp job shop problem is a paradigm of constraint satisfaction problem and has been approached using many heuristic techniques. In particular, genetic algorithms have proved to be a promising solving method [10],[12],[13]. The structure of a conventional genetic algorithm for the FJSP is described in Algorithm 1. First, the initial population is generated and evaluated. Then the genetic algorithm iterates for a number of steps or generations. In each iteration, a new population is built from the previous one by applying the genetic operators of selection, recombination and acceptation.

To codify chromosomes we use the decision variable $\boldsymbol{x}$, a permutation with repetition, which presents a number of interesting characteristics [14]. The quality of a chromosome is evaluated by the fitness function, which is taken to be the objective function of the leximin problem lexmin(d_1^+, d_2^-, d_3^-) as defined above.

In the selection phase, chromosomes are grouped into pairs using tournament. Each of these pairs is mated to obtain two offsprings and acceptance consists in selecting the best individuals from the set formed by the pair of parents and their offsprings. For chromosome mating we consider the *Job Order Crossover*

Require: an instance of fuzzy JSP, P
Ensure: a schedule H for P
1. Generate the initial population;
2. Evaluate the population;
while No termination criterion is satisfied **do**
3. Select chromosomes from the current population;
4. Apply the recombination operator to the chromosomes selected at step 3. to generate new ones;
5. Evaluate the chromosomes generated at step 4;
6. Apply the acceptance criterion to the set of chromosomes selected at step 3. together with the chromosomes generated at step 4.;
return the schedule from the best chromosome evaluated so far;

Algorithm 1. Conventional Genetic Algorithm

(JOX) [10]. Given two parents, JOX selects a random subset of jobs, copies their genes to the offspring in the same positions as they appear in the first parent, and the remaining genes are taken from the second parent so as to maintain their relative ordering. This operator has an implicit mutation effect. Therefore, no explicit mutation operator is actually necessary and parameter setting is simplified, as crossover probability is 1 and mutation probability need not be specified.

From a given decision variable $\boldsymbol{x}$ we may obtain a *semi-active* schedule as explained in Section 3, meaning that for any operation to start earlier, the relative ordering of at least two tasks must be swapped. However, other possibilities may be considered. For the crisp job shop, it is common to use the G&T algorithm [15], which is an active schedule builder. A schedule is *active* if one task must be delayed for any other one to start earlier. Active schedules are good in average and, most importantly, the space of active schedules contains at least an optimal one, that is, the set of active schedules is *dominant*. For these reasons it is worth to restrict the search to this space. Moreover, the G&T algorithm is complete for the job shop problem.

In Algorithm 2 we propose an extension of G&T to the case of fuzzy processing times. It should be noted nonetheless that, due to the uncertain 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 and completion times will be denoted by S_θ and C_θ respectively.

Recall that operator JOX tries to maintain for each machine a subsequence of tasks in the order as they appear in parent 1 and the remaining tasks in the same order as they are in parent 2. It often happens that these two subsequences are not compatible with each other in order to obtain an active schedule, so the decoding algorithm given in Algorithm 2 has to exchange the order of some operations. This new order is translated to the chromosome, for it to be passed

Require: a chromosome $\boldsymbol{x}$ and a fuzzy JSP P
Ensure: the schedule s given by chromosome $\boldsymbol{x}$ for problem P
1. $A = \{\theta_{i1},\ i = 1, \ldots, n\}$; /*set of first tasks of all jobs*/
2. **while** $A \neq \emptyset$ **do**
3. Determine the task $\theta' \in A$ with minimum earliest completion time $C^1_{\theta'}$ if scheduled in the current state;
4. Let M' be the machine required by θ' and $B \subseteq A$ the subset of tasks requiring machine M';
5. Remove from B any task θ that starts later than $C_{\theta'}$: $C^i_{\theta'} \leq S^i_\theta$, $i = 1, 2, 3$;
6. Select $\theta^\star \in B$ such that it is the leftmost operation in the sequence $\boldsymbol{x}$;
7. Schedule θ^* as early as possible to build a partial schedule;
8. 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;
9. **return** the built schedule;

Algorithm 2. Extended G&T for triangular fuzzy times

onto subsequent offsprings. In this way, the GA exploits the so called lamarckian evolution. As mentioned above, an implicit mutation effect is obtained.

The GA described above has been successfully used to minimise the expected makespan using semi-active schedules, comparing favourably to a simulated annealing algorithm from the literature [4]. Also, the GA combined with the extended G&T improves the expected makespan results obtained by a niche-based GA where chromosomes are matrices of completion times and recombination operators are based on fuzzy G&T [11].

5 Experimental Results

For the experimental results, we follow [4] and generate a set of fuzzy problem instances from well-known benchmark problems: FT06 of size 6×6 and LA11, LA12, LA13 and LA14 of size 20×5. For a given crisp processing time x, a symmetric fuzzy processing time $p(x)$ is generated such that its centre value, p^2, is equal to x. The value of p^1 is selected at random so that the TFN's maximum range of fuzziness is 30% of p^2, taking into account that p^3 is the symmetric point to p^1 with respect to p^2, $p^3 = 2p^2 - p^1$. In [4], only uncertain durations are considered. To generate a flexible due date for a given job J_i, let $\iota_i = \sum_{j=1}^{m} p^2_{i,j}$ be the sum of most typical durations across all its tasks. Also, for a given task $\theta_{i,j}$ let $\rho_{i,j}$ be the sum of most typical durations of all other tasks requiring the same machine as $\theta_{i,j}$, $\rho_{i,j} = \sum_{r \neq i, s \neq j : \nu_{rs} = \nu_{ij}} p^2_{r,s}$, where $p^2_{r,s}$ denotes the most typical duration of task $\theta_{r,s}$. Finally, let $\rho_i = \max_{j=1,\ldots,m} \rho_{i,j}$ be the maximum of such values across all tasks in job J_i. The earlier due date d^1 is a random value from $[d_m, d_M]$, where $d_m = \iota_i + 0.5\rho_i$ and $d_M = \iota_i + \rho_i$, and the later due date d^2 is a random value from $[d^1, int(1.3d^1)]$, where $int(x)$ denotes the smallest integer greater than or equal to x. [16].

For each problem instance, we have run the GA 30 times, using the three single-objective functions f_1, f_2 and f_3 and the multi-objective function

proposed in this work lexmin(d_1^+, d_2^-, d_3^-). The configuration parameters of the GA are population size 100 and number of generations 200. To fix the target value for the expected makespan b_1, we use our experience obtained with previous experimentation using f_1 as single objective and set b_1 equal to the makespan's average value across 30 runs of the single-objective GA (see Table 1). The target values for due date satisfaction are in all cases $b_2 = b_3 = 1$. Finally, we also include results obtained with a different multi-objective function based on fuzzy decision making, proposed in [7] and denoted f hereafter. In this approach, the decision maker must define gradual satisfaction degrees for each objective. The results shown herein are obtained with maximum satisfaction for each objective equal to the above target values; the minimum satisfaction degrees for makespan are $int(1.1b_1)$ and, in all cases, the minimum satisfaction for f_2 and f_3 is achieved at 0. For each fitness function we measure $E[C_{max}]$, AI_{av} and AI_{min} of the obtained schedule and compute the best, average and worst of these values across the 30 executions of the GA. The results are shown in Table 1; under each problem name and between brackets we include optimal value of the makespan for the original crisp problem, which provides a lower bound for the expected makespan of the fuzzified version [4].

Let us first analyse the results obtained by the proposed multiobjective approach, compared to the results obtained when optimising a single criterion. For the most prioritary objective, minimisation of makespan, we see that the multiobjective approach obtains exactly the same expected makespan values than the single-objective function. These expected values also coincide with the optimal value for the crisp problem in all cases except LA12. For this problem, the fuzzy makespan for the 30 runs of the GA is $C_{max} = (972, 1039, 1110)$, so the most typical value coincides with the optimal value of the crisp problem, but $E[C_{max}] = 1040$. Besides, there is a clear improvement in due date satisfaction.

Regarding the second objective, AI_{av}, the results obtained by lexmin compared to $1 - f_2$ yield a relative error lower than 1%, except for FT06, where the relative error is close to 6%. Notice however that, for this problem, the single objective $1 - f_2$ has a relative error w.r.t. the best expected makespan values up to 36%. The use of multiobjective optimisation sets this error to 0, at the expense of reasonably worse results for the second objective. In the remaining problems, the benefits of multiobjective optimisation are even clearer: the makespan errors (w.r.t. the best values) go from 2.26%-28.12% when using $1 - f_2$ to zero using the multiobjective approach and, at the same time, the multiobjective approach has notably higher values of AI_{av} than when only makespan (f_1) is optimised, at the same computational cost.

The behaviour for the third objective, AI_{min} is similar. There is a slight worsening in the value of AI_{min} when lexmin is used instead of $1 - f_3$ (with the exception of FT06, where worsening is higher), but this is largely compensated by the improvement in makespan. Notice as well that the errors obtained in AI_{min} when only makespan minimisation is considered may be up to 100% and they are reduced by lexmin between 60% and 100%, with an average reduction of 77.72%, again with the exception of FT06.

Table 1. Results obtained by the GA

Problem	Fitness	f_1			f_2			f_3		
		Best	Avg	Worst	Best	Avg	Worst	Best	Avg	Worst
FT06 (55)	f_1	55	55	55	0.94	0.94	0.94	0.64	0.64	0.64
	$1-f_2$	59	70.47	75	1	1	1	1	1	1
	$1-f_3$	61	72.28	75	1	1	1	1	1	1
	lexmin	55	55	55	0.94	0.94	0.94	0.64	0.64	0.64
	f	59	69.07	87	0.98	0.89	0.68	0.89	0.51	0
LA11 (1222)	f_1	1222	1222	1222	0.93	0.83	0.74	0.37	0.07	0
	$1-f_2$	1238	1326.91	1371	1	1	1	1	1	1
	$1-f_3$	1241	1327.88	1372.50	1	1	1	1	1	1
	lexmin	1222	1222	1222	1	1	0.98	1	0.99	0.80
	f	1279.75	1339.22	1488.75	0.86	0.75	0.61	0.32	0.03	0
LA12 (1039)	f_1	1040	1040	1040	0.95	0.88	0.82	0.64	0.15	0
	$1-f_2$	1084	1112.53	1212.75	1	1	1	1	1	1
	$1-f_3$	1084	1102.16	1192.25	1	1	1	1	1	1
	lexmin	1040	1040	1040	1	0.99	0.99	0.93	0.92	0.90
	f	1064.50	1145.97	1293	0.91	0.84	0.77	0.47	0.08	0
LA13 (1150)	f_1	1150	1150	1150	0.96	0.92	0.88	0.80	0.54	0.34
	$1-f_2$	1198	1291.62	1359.25	1	1	1	1	1	1
	$1-f_3$	1151	1272.12	1363	1	1	1	1	1	1
	lexmin	1150	1150	1150	1	1	1	1	1	1
	f	1185.75	1289.16	1378.25	0.96	0.87	0.80	0.75	0.24	0
LA14 (1292)	f_1	1292	1292	1292	0.95	0.86	0.81	0.60	0.04	0
	$1-f_2$	1292	1321.21	1432.75	1	1	1	1	1	1
	$1-f_3$	1292	1315.63	1445	1	1	1	1	1	1
	lexmin	1292	1292	1292	1	1	1	1	1	1
	f	1292	1337.54	1427.50	0.94	0.86	0.79	0.47	0.05	0

Finally, if we compare the two multiobjective approaches, lexmin is clearly better than f. The latter, based on fuzzy decision making, uses the minimum to aggregate the objectives' satisfaction degrees and this operator might be too coarse in some cases, for instance, when one objective is more difficult to achieve than the others or when objectives are partially incompatible.

6 Conclusions and Future Work

We have considered a job shop problem with uncertain durations, modelled using TFNs, and flexible due dates, also modelled with fuzzy sets. The goal is to find an ordering of tasks that yields a feasible schedule with minimal makespan and maximum due-date satisfaction. We propose to formulate the multiobjective problem as a fuzzy goal programming model according to a priority structure and target levels established by the decision maker, using the expected value of the makespan and lexicographical minimisation. The resulting problem is solved using a GA with codification based on permutations with repetitions. Experimental results on fuzzy versions of well-known problem instances illustrate the

potential of both the proposed multiobjective formulation and the GA. Indeed, in most cases the expected makespan values coincide with optimal values for the original problems and due-date satisfaction is maximal.

In the future, the multiobjective approach will be further analysed on varied set of problem instances, incorporating the semantics proposed in [7]. Also, the GA may be hybridised with other heuristic techniques such as local search, which implies further studying task criticality when durations are fuzzy.

References

1. Brucker, P., Knust, S.: Complex Scheduling. Springer, Heidelberg (2006)
2. Dubois, D., Fargier, H., Fortemps, P.: Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. European Journal of Operational Research 147, 231–252 (2003)
3. Słowiński, R., Hapke, M. (eds.): Scheduling Under Fuzziness. Studies in Fuzziness and Soft Computing, vol. 37. Physica-Verlag, Heidelberg (2000)
4. Fortemps, P.: Jobshop scheduling with imprecise durations: a fuzzy approach. IEEE Transactions of Fuzzy Systems 7, 557–569 (1997)
5. Sakawa, M., Kubota, R.: Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms. European Journal of Operational Research 120, 393–407 (2000)
6. Fayad, C., Petrovic, S.: A fuzzy genetic algorithm for real-world job-shop scheduling. In: Ali, M., Esposito, F. (eds.) IEA/AIE 2005. LNCS (LNAI), vol. 3533, pp. 524–533. Springer, Heidelberg (2005)
7. González Rodríguez, I., Puente, J., Vela, C.R., Varela, R.: Semantics of schedules for the fuzzy job shop problem. IEEE Transactions on Systems, Man and Cybernetics, Part A, Accepted for publication (2007)
8. Liu, B., Liu, Y.K.: Expected value of fuzzy variable and fuzzy expected value models. IEEE Transactions on Fuzzy Systems 10, 445–450 (2002)
9. Celano, G., Costa, A., Fichera, S.: An evolutionary algorithm for pure fuzzy flowshop scheduling problems. Fuzziness and Knowledge-Based Systems 11, 655–669 (2003)
10. Bierwirth, C.: A generalized permutation approach to jobshop scheduling with genetic algorithms. OR Spectrum 17, 87–92 (1995)
11. González Rodríguez, I., Vela, C.R., Puente, J.: A memetic approach to fuzzy job shop based on expectation model. In: Proceedings of FUZZ-IEEE 2007 (2007)
12. Mattfeld, D.C.: Evolutionary Search and the Job Shop Investigations on Genetic Algorithms for Production Scheduling. Springer, Heidelberg (1995)
13. Varela, R., Vela, C.R., Puente, J., Gómez, A.: A knowledge-based evolutionary strategy for scheduling problems with bottlenecks. European Journal of Operational Research 145, 57–71 (2003)
14. Varela, R., Serrano, D., Sierra, M.: New codification schemas for scheduling with genetic algorithms. In: Mira, J.M., Álvarez, J.R. (eds.) IWINAC 2005. LNCS, vol. 3562, pp. 11–20. Springer, Heidelberg (2005)
15. Giffler, B., Thomson, G.L.: Algorithms for solving production scheduling problems. Operations Research 8, 487–503 (1960)
16. González Rodríguez, I., Vela, C.R., Puente, J.: Study of objective functions in fuzzy job-shop problem. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 360–369. Springer, Heidelberg (2006)