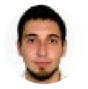
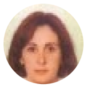
A Particle Swarm Solution based on Lexicographical Goal Programming for a Multiobjective Fuzzy Open Shop Problem

CONFERENCE PAPER · JUNE 2012

4 AUTHORS:


Juan Jose Palacios
University of Oviedo
12 PUBLICATIONS 13 CITATIONS
SEE PROFILE


Camino Rodríguez Vela
University of Oviedo
62 PUBLICATIONS 287 CITATIONS
SEE PROFILE

Ines Gonzalez Rodriguez
Universidad de Cantabria
42 PUBLICATIONS 151 CITATIONS
SEE PROFILE

Jorge Puente Peinador
University of Oviedo
50 PUBLICATIONS 193 CITATIONS
SEE PROFILE

A Particle Swarm Solution based on Lexicographical Goal Programming for a Multiobjective Fuzzy Open Shop Problem

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

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

Abstract. In the sequel, we consider a multiobjective open shop scheduling problem with uncertain durations modelled as fuzzy numbers. Given crisp due dates, the objective is to minimise both the makespan and the maximum tardiness. We formulate the multiobjective problem as a fuzzy goal programming model based on lexicographical minimisation of expected values. The resulting problem is solved using a particle swarm optimization approach searching in the space of possibly active schedules. Experimental results are presented on several problem instances to evaluate the proposed method, illustrating its potential.

1 Introduction

The open shop scheduling problem (OSP) is a problem with an increasing presence in the literature and clear applications in industry—consider for instance testing facilities, where units go through a series of diagnostic tests that need not be performed in a specified order and where different testing equipment is usually required for each test [22]. Traditionally, the open shop as well as other scheduling problems have been treated as deterministic, assuming precise knowledge of all data involved, in contrast with the uncertainty and vagueness pervading real-world problems. To enhance the range of applications of scheduling, an increasing part of the research is devoted to modelling this lack of certainty with great diversity of approaches [14]. In particular, fuzzy sets have been used in different manners, ranging from representing incomplete or vague states of information to using fuzzy priority rules with linguistic qualifiers or preference modelling [5]. They are also emerging as an interesting tool for improving solution robustness, a much-desired property in real-life applications [24],[15].

Proceedings of the 19th RCRA workshop on *Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion* (RCRA 2012).
In conjunction with AI*IA 2012, Rome, Italy, June 14–16, 2012.

The open shop is NP-complete for a number of machines $m \geq 3$, so approaches to solving it usually make use of metaheuristic techniques. In particular, in [1] a heuristic approach is proposed to minimise the expected makespan for an open shop problem with stochastic processing times and random breakdowns; in [21] a genetic algorithm is proposed to minimise the expected makespan of an open shop with fuzzy durations; in [11] this genetic algorithm is combined with a local search method; a particle swarm optimisation algorithm is used to minimise the expected fuzzy makespan in [20] and a possibilistic mixed-integer linear programming method is proposed in [19] for an OSP with setup times, fuzzy processing times and fuzzy due dates to minimize total weighted tardiness and total weighted completion times. Far from being trivial, extending heuristic strategies to uncertain settings usually requires a significant reformulation of both the problem and solving methods.

In the sequel, we describe an open shop problem with fuzzy durations and crisp due dates and where the objective is to minimise both the project's makespan and the maximum tardiness w.r.t. the given job due dates. We adopt a generic multiobjective model so the objective function is defined in order to lexicographically minimise the expected values of several fuzzy goals (here, makespan and tardiness). In addition to the priority structure for the lexicographical minimisation, target levels for each objective are introduced, in order to balance possibly incompatible goals. The resulting problem is solved by means of a multiobjective particle swarm optimisation (MOPSO) algorithm searching in the space of possibly active schedules. We evaluate the performance of the MOPSO algorithm on a set of problem instances based on the expected values of each objective.

2 Uncertain Processing Times as Triangular Fuzzy Numbers

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 be 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 intervals 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.

Fuzzy intervals are a natural extension of human-originated confidence intervals when some values appear to be more plausible than others. The simplest model 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:

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

Triangular fuzzy numbers and more generally fuzzy intervals have been extensively studied in the literature (cf. [6]). A *fuzzy interval* Q is a fuzzy quantity (a fuzzy set on the reals) whose *α-cuts* $Q_\alpha = \{u \in \mathbb{R} : \mu_Q(u) \geq \alpha\}$, $\alpha \in (0, 1]$, are convex, i.e., they are intervals (bounded or not). The *core* of Q consists of those elements with full membership $\mu_Q(u) = 1$, also called *modal values.* The *support* of Q is $Q_0 = \{u \in \mathbb{R} : \mu_Q(u) > 0\}$. A *fuzzy number* is a fuzzy quantity whose α-cuts are closed intervals, with compact (i.e. closed and bounded) support and unique modal value. Thus, real numbers can be seen as a particular case of fuzzy ones.

In order to work with fuzzy numbers, it is necessary to extend the usual arithmetic operations on real numbers. In general, if f is a function $f : \mathbb{R}^2 \to \mathbb{R}$ and Q_1, Q_2 are two fuzzy quantities, the fuzzy quantity $f(Q_1, Q_2)$ is calculated according to the *Extension Principle* as follows:

$$\forall u \in \mathbb{R}, \mu_{f(Q_1,Q_2)}(u) = \sup\{\min(\mu_{Q_1}(w_1), \mu_{Q_2}(w_2)) : f(w_1, w_2) = u\} \tag{2}$$

if $f^{-1}(u) \neq \emptyset$, being equal to 0 otherwise. Computing the above equation is cumbersome, if not intractable. It can be somewhat simplified if M and N are two fuzzy numbers, so the α-cuts M_α and N_α are closed bounded intervals of the form $[\underline{m}_\alpha, \overline{m}_\alpha]$ and $[\underline{n}_\alpha, \overline{n}_\alpha]$, and if f is a continuous isotonic mapping from $\mathbb{R}^2$ into $\mathbb{R}$, that is, if for any $u \geq u'$ and $v \geq v'$ it holds $f(u, v) \geq f(u', v')$. In this case, the First Decomposition Theorem provides us with an alternative formula for $f(M, N)$:

$$f(M, N) = \cup_{\alpha \in (0,1]}[f(\underline{m}_\alpha, \underline{n}_\alpha), f(\overline{m}_\alpha, \overline{n}_\alpha)] \tag{3}$$

In the open shop, we essentially need the following operations on fuzzy durations: addition, substraction and maximum. In the case of TFNs, both the addition and substraction are fairly easy to compute, since they reduce to operating on the three defining points, that is, for any pair of TFNs M and N:

$$M + N = (m^1 + n^1, m^2 + n^2, m^3 + n^3) \tag{4}$$
$$M - N = (m^1 - n^3, m^2 - n^2, m^3 - n^1). \tag{5}$$

Unfortunately, for the maximum of TFNs there is no such simplified expression. Being an isotonic function, we can use equation (3) above to compute the maximum of two fuzzy numbers. However, in general this still requires an infinite number of computations, because we have to evaluate maxima for each value $\alpha \in (0, 1]$. For the sake of simplicity and tractability of numerical calculations, we follow Fortemps [7] and approximate all results of isotonic algebraic operations on TFNs by a TFN. Instead of evaluating the intervals corresponding to all α-cuts, we evaluate only those intervals corresponding to the support and $\alpha = 1$, which is equivalent to working only with the three defining points of each TFN. This is an approach also taken, for instance, in [18] and [4]. Therefore, for any two TFNs M and N, their maximum will be approximated as follows:

$$\max(M, N) \sim M \sqcup N = (\max(m^1, n^1), \max(m^2, n^2), \max(m^3, n^3)). \tag{6}$$

Despite not being equal, for any two TFNs M, N, if $F = \max(N, M)$ denotes their maximum and $G = N \sqcup M$ its approximated value, it holds that $\forall \alpha \in [0,1]$, $\underline{f}_\alpha \leq \underline{g}_\alpha, \overline{f}_\alpha \leq \overline{g}_\alpha$. In particular, F and G have identical support and modal value: $F_0 = G_0$ and $F_1 = G_1$. The approximated maximum can be trivially extended to $n > 2$ TFNs.

The membership function μ_Q of a fuzzy quantity Q can be interpreted as a possibility distribution on the real numbers; this allows to define the *expected value* of a fuzzy quantity [17], given for a TFN A by $E[A] = \frac{1}{4}(a^1 + 2a^2 + a^3)$. The expected value coincides with the *neutral scalar substitute* of a fuzzy interval and can also be obtained as the centre of gravity of its *mean value* or using the *area compensation* method [5]. It induces a total ordering $\leq_E$ in the set of fuzzy intervals [7], where for any two fuzzy intervals M, N $M \leq_E N$ if and only if $E[M] \leq E[N]$.

3 The Fuzzy Open Shop Scheduling Problem

The *open shop scheduling problem*, or *OSP* in short, consists in scheduling a set of n jobs $J_1, \ldots, J_n$ to be processed on a set of m physical resources or machines $M_1, \ldots, M_m$. Each job J_i consists of m tasks or operations o_{ij} ($j = 1, \ldots, m$), where o_{ij} requires the exclusive use of a machine M_j for its whole processing time p_{ij} without preemption, i.e. all tasks must be processed without interruption. In total, there are mn tasks, $\{o_{ij}, 1 \leq i \leq n, 1 \leq j \leq m\}$. Additionally, for each job J_i there may be a due date D_i, $i = 1, \ldots, n$ before which it is desirable that the job be terminated. A solution to this problem is a *schedule*–an allocation of starting times for all tasks– which is *feasible*, in the sense that all constraints hold, and is also optimal according to some criteria. Here, we shall consider the objective of minimising the makespan C_{max}, that is, the time lag from the start of the first task until the end of the last one, as well as minimising the maximum tardiness T_{max}, that is, the maximum delay of any job with respect to its due date. This problem may be denoted $O||multicrit(C_{max}, T_{max})$ using the three-field notation extended to multiple objectives [13].

3.1 Fuzzy Schedules from Crisp Task Orderings

A schedule s for a open shop problem of size $n \times m$ (n jobs and m machines) may be determined by a decision variable $\boldsymbol{z} = (z_1, \ldots, z_{nm})$ representing a task processing order, where $1 \leq z_l \leq nm$ for $l = 1, \ldots, nm$. This is a permutation of the set of tasks where each task o_{ij} is represented by the number $(i-1)m+j$. The task processing order represented by the decision variable uniquely determines a feasible schedule; it should be understood as expressing partial orderings for every set of tasks requiring the same machine and for every set of tasks requiring the same job.

Let us assume that the processing time p_{ij} of each task o_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, m$ is a fuzzy variable (a particular case of which are TFNs), so the problem may be represented by a matrix of fuzzy processing times $\boldsymbol{p}$ of size

$n \times m$. For a given task processing order $\boldsymbol{z}$, let $C_i(\boldsymbol{z}, \boldsymbol{p})$ denote the completion time of job J_i and let $C_{ij}(\boldsymbol{z}, \boldsymbol{p})$ denote the completion time of task o_{ij}. Clearly, the completion time of a job is the maximum completion time of its tasks, that is, $C_i(\boldsymbol{z}, \boldsymbol{p}) = \max_{1 \leq j \leq m}\{C_{ij}(\boldsymbol{z}, \boldsymbol{p})\}$. The completion time for task o_{ij} is obtained by adding its duration p_{ij} to its starting time $S_{ij}(\boldsymbol{z}, \boldsymbol{p})$. The latter will be the maximum between the completion times of the task preceding o_{ij} in its job and its machine, denoted o_{ik} and o_{lj} respectively, according to the processing order given by $\boldsymbol{z}$:

$$S_{ij}(\boldsymbol{z}, \boldsymbol{p}) = C_{ik}(\boldsymbol{z}, \boldsymbol{p}) \sqcup C_{lj}(\boldsymbol{z}, \boldsymbol{p}) \tag{7}$$
$$C_{ij}(\boldsymbol{z}, \boldsymbol{p}) = S_{ij}(\boldsymbol{z}, \boldsymbol{p}) + p_{ij} \tag{8}$$

$C_{ik}(\boldsymbol{z}, \boldsymbol{p})$ and $C_{lj}(\boldsymbol{z}, \boldsymbol{p})$ are taken to be zero if o_{ij} is the first task to be processed in the corresponding job or machine respectively.

For this schedule, the *fuzzy makespan* $C_{max}(\boldsymbol{z}, \boldsymbol{p})$ and the *fuzzy maximum tardiness* (fuzzy tardiness for short) $T_{max}(\boldsymbol{z}, \boldsymbol{p})$ are defined as follows:

$$C_{max}(\boldsymbol{z}, \boldsymbol{p}) = \sqcup_{1 \leq i \leq n} \left(C_i(\boldsymbol{z}, \boldsymbol{p})\right) \tag{9}$$
$$T_{max}(\boldsymbol{z}, \boldsymbol{p}) = \max(\sqcup_{1 \leq i \leq n} \left(C_i(\boldsymbol{z}, \boldsymbol{p}) - D_i\right), 0) \tag{10}$$

Let us illustrate the previous definitions with an example. Consider a problem of 3 jobs and 2 machines with the following matrix for fuzzy processing times:

$$\boldsymbol{p} = \begin{pmatrix} (3,4,7) & (1,2,3) \\ (2,3,4) & (4,5,6) \\ (1,2,4) & (1,2,6) \end{pmatrix}$$

Here $p_{22} = (2,3,4)$ is the processing time of task o_{22}, which is the task of job 2 to be processed in machine 2. Figure 1 shows the Gantt chart (adapted to TFNs) of the schedule given by the decision variable $\boldsymbol{z}$=(1 4 6 3 5 2); it represents the partial schedules on each machine obtained from the decision variable. Tasks must be processed in the following order: $o_{11}, o_{22}, o_{32}, o_{21}, o_{31}, o_{12}$. Given this ordering, the starting time for task o_{21} will be the maximum of the completion times of o_{22} and o_{11}, the preceding tasks in the job and in the machine: $S_{21} = C_{22} \sqcup C_{11} = (4,5,6) \sqcup (3,4,7) = (4,5,7)$. Consequently, its completion time will be $C_{21} = S_{21} + p_{21} = (4,5,7) + (2,3,4) = (6,8,11)$.

Notice that when uncertain durations are given as fuzzy intervals the schedule s will be a fuzzy schedule, in the sense that the starting and completion times of all tasks, the makespan and the tardiness are fuzzy intervals. These fuzzy intervals may be seen as possibility distributions on the values that these times may take. However, the task processing order represented by $\boldsymbol{z}$ that determines such schedule is crisp; there is no uncertainty regarding the order in which tasks are to be processed. In other words, we obtain a fuzzy schedule from a crisp task ordering.

Given a fuzzy schedule, our objective is to *optimise* its makespan and maximum tardiness. However, it is not trivial when dealing with fuzzy values to decide

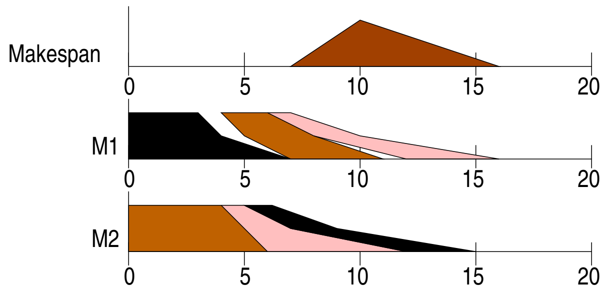


Fig. 1. Gantt chart of the schedule represented by the decision variable (1 4 6 3 5 2)

on the precise meaning of "optimality", since neither the maximum nor its approximation define a total ordering in the set of TFNs. Using ideas similar to stochastic scheduling, we use the total ordering provided by the expected value and consider that the objective is to minimise the expected makespan $E[C_{max}]$ and the expected tardiness $E[T_{max}]$, so the resulting problem may be denoted $FuzO||multicrit(E[C_{max}], E[T_{max}])$.

3.2 Multiobjective Models

With multiple goals it is often the case that some are achievable only at the expense of others. A possible approach to this issue is to establish a hierarchy of importance for these goals so as to satisfy as many as possible in the specified order. In general, for k objectives $f_1, \ldots, f_k$ such priority structure should be established by the decision maker (DM) and may be represented by a one-to-one mapping ρ from $\{f_1, \ldots, f_k\}$ onto $\{1, \ldots, k\}$, such that $\rho(f_i)$ is the priority level of f_i, $i = 1, \ldots, k$, where 1 represents the highest priority. For instance, if $f_1 = C_{max}$ and $f_2 = T_{max}$ and the DM considers that the most prioritary objective is minimising the expected tardiness, then $\rho(f_2) = 1$. Without loss of generality, we can assume that the objective functions f_i $i = 1, \ldots, k$ are ordered according to their priority, that is, $\rho(f_i) = i$. Then, based on similar ideas presented for the job shop in [12] we may formulate the following *expected multiobjective model* for the fuzzy open shop problem (FOSP):

$$\begin{cases} \text{lexmin} & (E[f_1(\boldsymbol{z}, \boldsymbol{p})], \ldots, E[f_k(\boldsymbol{z}, \boldsymbol{p})]) \\ \text{subject to:} & 1 \leq z_l \leq nm, \quad l = 1, \ldots, nm, \\ & z_l \in \mathbb{Z}^+, \quad l = 1, \ldots, nm. \end{cases} \tag{11}$$

where lexmin denotes lexicographically minimising the objective vector.

Pure lexicographical models may get stuck in the first goals and never consider the remaining ones. To balance the multiple conflicting objectives, we may use a goal programming model and consider target levels established by the DM, so $E[f_i(\boldsymbol{z}, \boldsymbol{p})]$ should not exceed a given target value b_i, $i = 1, \ldots, k$. This translates into the following goal constraints:

$$E[f_i(\boldsymbol{z},\boldsymbol{p})] + d_i^- - d_i^+ = b_i, \quad i = 1,\dots,k \tag{12}$$

where d_i^+, the positive deviation from the target, should be minimised. We thus obtain the following *expected fuzzy goal multiobjective model* for the FJSP:

$$\begin{cases} \text{lexmin} & (d_1^+,\dots,d_k^+) \\ \text{subject to:} & E[f_i(\boldsymbol{z},\boldsymbol{p})] + d_i^- - d_i^+ = b_i, \quad i = 1,\dots,k, \\ & b_i, d_i^-, d_i^+ \geq 0, \quad i = 1,\dots,k, \\ & 1 \leq x_l \leq nm, \quad l = 1,\dots,nm, \\ & z_l \in \mathbb{Z}^+, \quad l = 1,\dots,nm. \end{cases} \tag{13}$$

Notice that (11) is a particular case of (13). Indeed, the latter is general enough to comprise all possible fuzzy goals, priority structures and target levels established by the DM.

4 Particle Swarm Optimization for the FOSP

Particle Swarm Optimisation (PSO) is a population-based stochastic optimisation technique inspired by bird flocking or fish schooling [16]. In PSO, each position in the search space corresponds to a solution of the problem and particles in the swarm cooperate to explore the space and find the best position (hence best solution). Particle movement is mainly affected by the three following factors:

- Inertia: Velocity of the particle in the latest iteration.
- *pbest*: The best position found by the particle.
- *gbest*: The best position found by the swarm so far ("the best *pbest*").

Potential solutions or particles fly through the problem space, changing their position and velocity by following the current optimum particles *pbest* and *gbest*. A generic PSO algorithm can be found in Algorithm 1: first, the initial swarm is generated and evaluated and then the swarm evolves until a termination criterion is satisfied. In each iteration, a new swarm is built from the previous one by changing the position and velocity of each particle to move towards its *pbest* and *gbest* locations.

In [20] a PSO algorithm was proposed to minimise the expected makespan of fuzzy open shop. This algorithm was in turn inspired by the method proposed in [23] for the deterministic OSP, which improved the best results published so far. Here we extend this algorithm to the multiobjective setting described in the previous section.

4.1 Position Representation and Evaluation

To represent particle positions, we use a priority-based representation. A decision variable $\boldsymbol{z}$ is encoded as a *priority array* $\boldsymbol{x^k} = (x_l^k)_{l=1\dots nm}$, where x_l^k denotes the priority of task l, so a task with smaller x_l^k has a higher priority to be scheduled.

```
**Input** A FOSP instance
**Output** A schedule for the input instance
  Generate and evaluate the initial swarm;
  Compute *gbest* and *pbest* for each particle;
  **while** no Termination Criterion is satisfied **do**
    **for** each particle $k$ **do**
      **for** each dimension $d$ **do**
        Update velocity $v_d^k$;
        Update position $x_d^k$;
      Evaluate particle $k$;
      Update *pbest* and *gbest* values;
  **return** The schedule from the best particle evaluated so far;
```

Algorithm 1: A generic PSO algorithm

Given an OSP solution represented by a decision variable $\boldsymbol{z}$, which is a permutation of tasks, we can transfer this permutation to a priority array as follows. First, from $\boldsymbol{z}$ we obtain a position array, denoted $pos^{\boldsymbol{z}}$, such that $pos_l^{\boldsymbol{z}}$ is the position of task l in $\boldsymbol{z}$ ($pos_l^{\boldsymbol{z}} = i$ if and only if $z_i = l$). For instance, given the following decision variable for a problem with $n = 2$ jobs and $m = 3$ machines:

$$\boldsymbol{z} = \begin{pmatrix} 4 \; 1 \; 5 \; 2 \; 3 \; 6 \end{pmatrix}$$

the position array for the above decision variable is:

$$pos^{\boldsymbol{z}} = \begin{pmatrix} 2 \; 4 \; 5 \; 1 \; 3 \; 6 \end{pmatrix}.$$

Then, the priority array $\boldsymbol{x}$ is obtained by randomly setting x_l in the interval $(pos_l^{\boldsymbol{z}} - 0.5, pos_l^{\boldsymbol{z}} + 0.5)$, so a task with smaller x_l has higher priority to be scheduled. For the above permutation, a possible particle position would be:

$$\boldsymbol{x} = \begin{pmatrix} 2.3 \; 3.7 \; 5.4 \; 0.8 \; 2.8 \; 5.9 \end{pmatrix}$$

Conversely, from every particle position $\boldsymbol{x}$ we can obtain a position array pos^x where pos_i^x is the position of x_i if the elements of $\boldsymbol{x}$ were reordered in non-decreasing order.

A particle may be decoded in several ways. For the crisp job shop and by extension for the open shop, it is common to use the G&T algorithm [8], 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. In [9] a narrowing mechanism was incorporated to the G&T algorithm in order to limit machine idle times using a delay parameter $\delta \in [0, 1]$, thus searching in the space of so-called parameterised active schedules. In the deterministic case, for $\delta < 1$ the search space is reduced so it may no longer contain optimal schedules and, at the extreme $\delta = 0$ the search is constrained

```
**Input** A FOSP instance and a particle position $\boldsymbol{x^k}$
**Output** A schedule for the input instance considering the priorities given by $\boldsymbol{x^k}$
  $\Omega \leftarrow \{1,\dots,nm\}$;
  **while** $\Omega \neq \emptyset$ **do**
    Compute $\{E[S_l] : l \in \Omega\}$ and $\{E[C_l] : l \in \Omega\}$ considering only tasks previously scheduled;
    $C^* \leftarrow \min_{l\in\Omega}\{E[C_l]\}$;
    $S^* \leftarrow \min_{l\in\Omega}\{E[S_l]\}$;
    Identify the conflict set $O \leftarrow \{l : E[S_l] < S^* + \delta \times (C^* - S^*), l \in \Omega\}$;
    Choose the task $l^*$ from $O$ with smallest $x_l^k$;
    Schedule the operation $l^*$; /*fix the value of $E[S_{l^*}]$*/
    $\Omega \leftarrow \Omega - \{l^*\}$;
  **return** The schedule $s$ given by $\{E[S_l] : l \in \{1,\dots,nm\}\}$
```

Algorithm 2: The *pFG&T*

to non-delay schedules, where a resource is never idle if a requiring operation is available. This variant of G&T has been applied in [23] to the deterministic OSP, under the name "parameterized active schedule generation algorithm".

Algorithm 2, denoted *pFG&T*, is an extension of parameterised G&T to the case of fuzzy processing times proposed in [20]. It should be noted that, due to the uncertainty in task durations, even for $\delta = 1$, 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, Ω denotes the set of tasks that have not been scheduled, x^k the priority array and S_l and C_l the starting and completion time of task o_{ij} such that $l = (i-1)m + j$. Notice that the *pFG&T* algorithm may change the task processing order given by the particle position. Therefore the PSO does not record in *gbest* and *pbest* the best positions found so far, but rather the corresponding priority arrays.

4.2 Particle movement and velocity

Particle velocity is traditionally updated depending on the distance to *gbest* and *pbest*. Instead, this PSO only considers whether the position value x_l^k is larger or smaller than $pbest_l^k$ ($gbest_l$). For any particle, its velocity is represented by an array of the same length as the position array where all the values are in the set $\{-1, 0, 1\}$. The initial values for the velocity array are set randomly. Updating is controlled by the inertia weight w at the beginning of each iteration as follows. For each particle k and dimension d, if $v_d^k \neq 0$, v_d^k will be set to 0 with probability $1 - w$, meaning that if x_d^k was either increasing or decreasing, x_d^k stops at this iteration with probability $1 - w$. Otherwise, if $v_d^k = 0$, with probability p_1, v_d^k and x_d^k will be updated depending on $pbest_d^k$ and with probability p_2 they will be updated depending on $gbest_d$, always introducing an element of randomness and where p_1 and p_2 are constants between 0 and 1 such that $p_1 + p_2 \leq 1$. Further detail on particle updating is given in Algorithm 3.

```
Input  A particle position $\boldsymbol{x^k}$ and velocity $\boldsymbol{v^k}$, best particle and swarm positions $pbest^k$
  and $gbest$, and inertia $w$
Output  The particle position $\boldsymbol{x^k}$ and velocity $\boldsymbol{v^k}$ updated
  for each dimension $d$ do
    generate random value $rand \sim U(0,1)$;
    if $v_d^k \neq 0$ and $rand \geq w$ then
      $v_d^k \leftarrow 0$;
    if $v_d^k = 0$ then
      generate random value $rand \sim U(0,1)$;
      if $rand \leq p_1$ then
        if $pbest_d^k \geq x_d^k$ then $v_d^k \leftarrow 1$;
        else $v_d^k \leftarrow -1$;
        generate random value $rand_2 \sim U(0,1)$;
        $x_d^k \leftarrow pbest_d^k + rand_2 - 0.5$;
      if $p_1 < rand \leq p_1 + p_2$ then
        if $gbest_d \geq x_d^k$ then $v_d^k \leftarrow 1$;
        else $v_d^k \leftarrow -1$;
        generate random value $rand_2 \sim U(0,1)$;
        $x_d^k \leftarrow gbest_d + rand_2 - 0.5$;
    else
      $x_d^k \leftarrow x_d^k + v_d^k$.
  return  The particle position $\boldsymbol{x^k}$ and velocity $\boldsymbol{v^k}$ updated
```

Algorithm 3: Particle movement

Position mutation. After a particle moves to a new position, we randomly choose a task and then mutate its priority value x_d^k independently of v_d^k. For a problem of size $n \times m$, if $x_d^k < (nm/2)$, x_d^k will take a random value in $[mn - n, mn]$, and $v_d^k = 1$; otherwise (if $x_d^k > (nm/2)$), x_d^k will take a random value in $[0, n]$ and $v_d^k = -1$.

Diversification strategy. If all particles have the same *pbest* solutions, they will be trapped into local optima. To prevent such situation, a diversification strategy is proposed in [23] that keeps the *pbest* solutions different. In this strategy, the *pbest* solution of each particle is not the best solution found by the particle itself, but one of the best N solutions found by the swarm so far, where N is the size of the swarm. Once any particle generates a new solution, the *pbest* solutions will be updated as follows: if the new solution equals the makespan of any *pbest* solution, the latter will be replaced with the new solution; else if the new solution has better makespan than the worst *pbest* solution and it is different from all *pbest* solutions, then the worst *pbest* solution is replaced by the new one.

5 Experimental Results

For the experimental results, we use the instances proposed in [11]. These were obtained based on the well-known benchmark from [3], which consists of 6 families, denoted J3, J4,..., J8, of sizes 3×3, 4×4,...,8×8, containing 8 or 9 instances each. Each family is divided into three sets of problems $per0$, $per10$ and $per20$ according to the difference between minimum and maximum workloads of jobs and machines (the number in the name refers to this difference in percentage). We shall only consider the largest instances, pertaining to the blocks of size 7×7 and 8×8, comparing our results to those of the memetic algorithm (MA) proposed in [11]. There are 10 fuzzy versions of each original problem instance, generated by transforming the original crisp processing times into symmetric TFNs such that their modal value corresponds to the original duration. To add a crisp due date D_i for each job J_i we follow [2] and compute $D_i = TF \sum_{j=1}^{m} p_{ij}^2$, where TF is the tightness factor for the due date, in our case, $TF = 1.1$.

Given the method for generating due dates, in $per0$ instances, where all jobs have the same workload (and consequently the same due-date), the difference between tardiness and makespan is only in a constant (the due date value). Both objectives are thus strongly correlated, making these instances unsuitable for our multi-objective study. Hence for the experimental analysis we shall restrict to the instances $per10$ and $per20$ of size 7×7 and 8×8 (in total, 120), these being the hardest problems to solve considering both objectives.

Given the two fuzzy goals $f_1 = C_{max}$ and $f_2 = T_{max}$, we consider four objective functions: two single-objective functions given by the expected values $E[f_1]$ and $E[f_2]$ and two multiobjective functions that result from incorporating two different priority structures in expression (13). The first multiobjective function l_{12} corresponds to the priority structure defined by $\rho(f_i) = i$, that is, the most prioritary goal is the makespan f_1. The second objective function l_{21} corresponds to $\rho(f_1) = 2, \rho(f_2) = 1$, i.e. the most prioritary goal is to minimise tardiness. These hierarchies correspond to probably the most common objectives in the open shop literature, namely minimise makespan or maximise due-date satisfaction.

For the PSO we take the best parameter values obtained in [23] after a parameter analysis: swarm size $N = 60$, $p_1 = 0.7$, $p_2 = 0.1$, and inertia weight w linearly decreasing from 0.9 to 0.3. Regarding the filtering mechanism of the search space given in the schedule generator, the efficiency of this reduction depends of the problem size [10], and our experimentation suggests taking $\delta = 0.25$ for the instances considered herein. The termination criterion is the number of iterations, which depends on the problem size: 2800 for 7×7 instances and 3000 for 8×8 instances. Finally, to fix the target values we emulate the DM and use the experience gained using $E[f_1]$ and $E[f_2]$ as single objectives, setting b_1 (resp. b_2) equal to the worst value of $E[f_1]$ ($E[f_2]$) across 30 runs of the PSO.

Figure 2 shows the average evolution of l_{12} along 3000 iterations for a fuzzy instance generated from $j8 - per20 - 2$. We can see how, initially, the algorithm minimises the expected makespan while the behaviour of the expected tardiness is erratic. However, once the algorithm has reached the expected makespan target (around iteration 600), it starts to mimimise tardiness as well.

Table 1 contains a summary of the results: for each fitness function we measure $E[f_1]$ and $E[f_2]$ in the obtained schedule and compute the average values

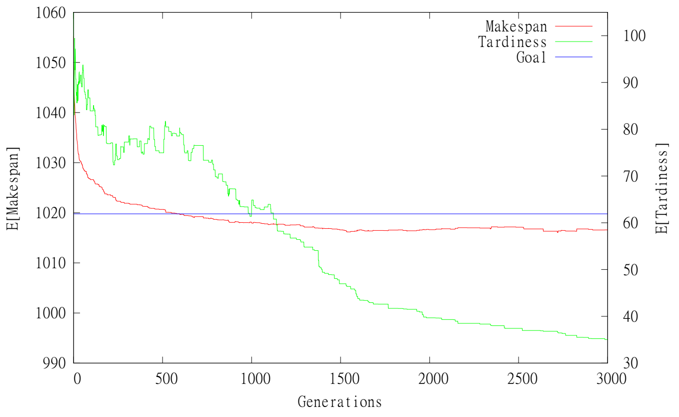


Fig. 2. Evolution of $E[f_1]$ and $E[f_2]$ on the l_{12} version for the *j8-per20-2* instance

and standard deviations across the 30 executions of the PSO and the 10 problem instances generated from the same original problem. According to this table, the multiobjective versions with l_{12} and l_{21} behave similarly to the corresponding single-objective ones, $E[f_1]$ and $E[f_2]$, regarding their most prioritary goal. Besides, they improve considerably on the other goal. Indeed, in all instances, l_{12} reaches the expected makespan target while the expected tardiness values obtained with l_{12} are 27% better in average than the $E[f_1]$ ones, being better in all problem instances. If we measure the reduction of the gap between the expected tardiness and its corresponding target, the multi-objective approach l_{12} is more than 50% better than $E[f_1]$ in average. Clearly, minimising the makespan does not always imply minimising tardiness. If we now compare l_{21} with $E[f_2]$, the results are similar. In all instances l_{21} reaches the expected tardiness target whereas the gap between the expected makespan and its target value is reduced 46% in average when the multi-objective approach is used.

According to Table 1, the improvement in expected tardiness with l_{12} is greater for $per20$ problems than for $per10$ ones. This is not surprising: with higher workload variability, due dates are less uniform in $per20$ instances. Indeed when only expected makespan is optimised, tardiness values are considerably larger for $per20$ than for $per10$ instances, the latter being closer to their target values. In consequence, for $per20$ instances there is greater room for improvement.

Notice that comparisons between different multiobjective functions do not make sense, as they model different priority requirements. In a hierarchical approach such as this, the decision maker is responsible for establishing an adequate hierarchy among goals according to his/her knowledge of the problem. Also, it should be taken into account the relevance of the target values in the model and, in consequence, in a good performance of the algorithm. In the case of a very hard target for the objective with highest priority, the algorithm behaves like the corresponding single-objective one. In experiments not reported here due to lack of space, we have tried different values for tardiness target and for due dates

Table 1. Results obtained by the MOPSO

		Size 7x7				Size 8x8			
Prob.	Fit.	E[Makespan]		E[Tardiness]		E[Makespan]		E[Tardiness]	
		Avg.	Std.Dev	Avg.	Std.Dev	Avg.	Std.Dev	Avg.	Std.Dev
per10-0	$E[f_1]$	1030	1.89	18.9	5.25	1050	2.42	22.4	5.76
	$E[f_2]$	1057	9.50	11.1	1.21	1074	9.37	13.6	1.17
	l_{12}	1031	1.83	14.4	3.07	1052	2.26	18.4	4.10
	l_{21}	1039	5.48	11.8	0.93	1059	7.12	14.6	1.28
per10-1	$E[f_1]$	1017	0.59	20.2	0.91	1030	3.60	29.3	7.51
	$E[f_2]$	1049	8.72	12.1	1.23	1064	11.49	14.2	1.52
	l_{12}	1018	1.22	19.3	1.40	1033	3.32	23.2	5.37
	l_{21}	1035	5.01	13.8	0.72	1050	10.34	15.5	1.30
per10-2	$E[f_1]$	1031	2.68	23.8	6.74	1033	5.50	26.9	7.32
	$E[f_2]$	1067	11.46	11.5	1.25	1063	12.78	14.3	1.45
	l_{12}	1032	3.23	19.9	4.64	1036	5.61	19.6	4.82
	l_{21}	1057	9.35	12.7	1.47	1051	10.17	15.6	1.48
per20-0	$E[f_1]$	1001	0.14	70.7	11.59	1016	2.64	104.2	16.59
	$E[f_2]$	1031	8.09	15.7	0.71	1071	11.60	13.3	1.39
	l_{12}	1001	0.32	46.6	9.33	1019	2.05	73.1	21.27
	l_{21}	1021	7.21	17.1	0.79	1060	11.02	14.6	1.16
per20-1	$E[f_1]$	1028	2.45	91.9	21.32	1001	0.78	93.7	20.11
	$E[f_2]$	1082	7.42	16.4	0.77	1023	10.02	16.9	1.66
	l_{12}	1031	1.40	55.5	17.90	1003	0.88	58.3	20.22
	l_{21}	1072	8.79	17.7	1.20	1013	7.20	18.7	1.64
per20-2	$E[f_1]$	1021	2.70	71.2	26.91	1014	2.58	77.6	21.95
	$E[f_2]$	1072	15.15	14.8	1.13	1062	14.16	15.1	1.54
	l_{12}	1023	2.48	49.7	21.63	1018	2.34	45.0	16.37
	l_{21}	1057	13.75	16.4	1.28	1049	13.13	16.4	1.52

(changing the tightness factor TF), and the results showed that for a difficult target, l_{21} behaves similarly to $E[f_2]$ in terms of makespan and tardiness. On the other hand, when the tardiness target is relaxed, l_{21} reached the target in early iterations of the algorithm and then optimised the makespan, thus behaving like $E[f_1]$.

Finally, we compare the PSO using l_{12} with the single-objective MA algorithm from [12]. Table 2 contains expected makespan results for both methods, with average solutions obtained across the 10 instances of each size by both methods, MA optimising only $E[C_{max}]$ and PSO with l_{12}. In [20] the PSO optimising $E[C_{max}]$ compared favourably with the MA algorithm and we see that this is also the case when using the multiobjective function l_{12}.

6 Conclusions and Future Work

We have considered an open shop problem with uncertain durations modelled using TFNs and where the goal is to find a schedule optimising the fuzzy makespan

Table 2. Comparison of results for $E[C_{max}]$

Problem	Size 7×7		Size 8×8	
	$E[f_1]$ and MA	l_{12} and PSO	$E[f_1]$ and MA	l_{12} and PSO
per10-0	1066.04	1031.34	1083.08	1051.73
per10-1	1052.36	1017.85	1065.86	1032.78
per10-2	1067.27	1032.31	1070.77	1035.61
per20-0	1004.22	1001.03	1036.72	1019.15
per20-1	1043.73	1030.70	1013.79	1002.55
per20-2	1042.05	1022.81	1034.66	1017.63

and fuzzy maximum tardiness. We have proposed to formulate the multiobjective problem as a lexicographical fuzzy goal programming model according to a generic priority structure and target levels established by the decision maker, using the expected value of the fuzzy quantities. As solving method, a PSO with codification based on priority arrays has been described. Experimental results on fuzzy versions of well-known crisp problem instances illustrate the potential of both the proposed multiobjective formulation and the PSO.

In the future, the multiobjective approach will be further analysed on a more varied set of problem instances. This should also enable a thorough parametric analysis of the influence of the target values. Also, we would like to consider the case when the DM cannot establish a priority structure among objectives and there is a need for Pareto-optimal solutions.

Acknowledgements

All authors are supported by Grant MEC-FEDER TIN2010-20976-C02-02.

References

1. D. Alcaide, A. Rodriguez-Gonzalez, and J. Sicilia. A heuristic approach to minimize expected makespan in open shops subject to stochastic processing times and failures. *International Journal of Flexible Manufacturing Systems*, 17:201–226, 2006.
2. M. Andresen, H. Bräsel, M. Mörig, J. Tusch, F. Werner, and P. Willenius. Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop. *Mathematical and Computer Modelling*, 48:1279–1293, 2008.
3. P. Brucker, J. Hunrink, B. Jurisch, and B. Wöstmann. A branch & bound algorithm for the open-shop problem. *Discrete Applied Mathematics*, 76:43–59, 1997.
4. S.-M. Chen and T.-H. Chang. Finding multiple possible critical paths using fuzzy PERT. *IEEE Transactions on Systems, Man, and Cybernetics–Part B:*, 31(6):930–937, 2001.
5. D. Dubois, H. Fargier, and P. Fortemps. Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. *European Journal of Operational Research*, 147:231–252, 2003.

6. D. Dubois and H. Prade. *Possibility Theory: An Approach to Computerized Processing of Uncertainty.* Plenum Press, New York (USA), 1986.
7. P. Fortemps. Jobshop scheduling with imprecise durations: a fuzzy approach. *IEEE Transactions of Fuzzy Systems*, 7:557–569, 1997.
8. B. Giffler and G. L. Thompson. Algorithms for solving production scheduling problems. *Operations Research*, 8:487–503, 1960.
9. J. Gonçalves, J. Mendes, and R. M. de M. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167:77–95, 2005.
10. M. A. González, M. Sierra, C. R. Vela, and R. Varela. Genetic algorithms hybridized with greedy algorithms and local search over the spaces of active and semi-active schedules. *Current Topics in Artificial Intelligence. 11th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2005. Revised Selected Papers. LNCS 4177*, pages 231–240, 2006.
11. I. González-Rodríguez, J. J. Palacios, C. R. Vela, and J. Puente. Heuristic local search for fuzzy open shop scheduling. In *Proceedings IEEE International Conference on Fuzzy Systems, FUZZ-IEEE2010*, pages 1858–1865. IEEE, 2010.
12. I. González-Rodríguez, C. R. Vela, and J. Puente. A genetic solution based on lexicographical goal programming for a multiobjective job shop with uncertainty. *Journal of Intelligent Manufacturing*, 21:65–73, 2010.
13. W. Herroelen, E. Demeulemeester, and De Reyck, Bert. A note on the paper "Resource-constrained project scheduling: Notation, classification, models and methods" by Brucker et al. *European Journal of Operational Research*, 128:679–688, 2001.
14. W. Herroelen and R. Leus. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165:289–306, 2005.
15. A. Kasperski and M. Kule. Choosing robust solutions in discrete optimization problems with fuzzy costs. *Fuzzy Sets and Systems*, 160:667–682, 2009.
16. J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942–1948, New Jersey, 1995. IEEE Press.
17. B. Liu and Y. K. Liu. Expected value of fuzzy variable and fuzzy expected value models. *IEEE Transactions on Fuzzy Systems*, 10:445–450, 2002.
18. Q. Niu, B. Jiao, and X. Gu. Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time. *Applied Mathematics and Computation*, 205:148–158, 2008.
19. S. Noori-Darvish, I. Mahdavi, and N. Mahdavi-Amiri. A bi-objective possibilistic programming model for open shop scheduling problems with sequence-dependent setup times, fuzzy processing times, and fuzzy due-dates. *Applied Soft Computing*, 12:1399–1416, 2012.
20. J. J. Palacios, I. González-Rodríguez, C. R. Vela, and J. Puente. Particle swarm optimisation for open shop problems with fuzzy durations. In *Proceedings of IWINAC 2011, Part I. LNCS 6686*, pages 362–371. Springer, 2011.
21. J. J. Palacios, J. Puente, C. R. Vela, and I. González-Rodríguez. A genetic algorithm for the open shop problem with uncertain durations. In *Proceedings of IWINAC 2009, Part I. LNCS 5601*, pages 255–264. Springer, 2009.
22. M. L. Pinedo. *Scheduling. Theory, Algorithms, and Systems.* Springer, third edition, 2008.
23. D. Y. Sha and H. Cheng-Yu. A new particle swarm optimization for the open shop scheduling problem. *Computers & Operations Research*, 35:3243–3261, 2008.
24. J. Wang. A fuzzy robust scheduling approach for product development projects. *European Journal of Operational Research*, 152:180–194, 2004.