Journal of Intelligent Manufacturing manuscript No.
(will be inserted by the editor)

Swarm lexicographic goal programming for fuzzy open shop scheduling

Juan José Palacios · **Inés González-Rodríguez** ·

Camino R. Vela · **Jorge Puente**

Received: date / Accepted: date

Abstract In this work we consider a multiobjective open shop scheduling problem with uncertain processing times and flexible due dates, both modelled using fuzzy sets. We adopt a goal programming model based on lexicographic multiobjective optimisation of both makespan and due-date satisfaction and propose a particle swarm algorithm to solve the resulting problem. We present experimental results which show that this multiobjective approach achieves as good results as single-objective algorithms for the objective with the highest priority, while greatly improving on the second objective.

Keywords open shop scheduling · fuzzy processing times · flexible due dates · particle swarm optimisation · lexicographic goal programming

Juan José Palacios · Camino R. Vela · Jorge Puente

Department of Computing. University of Oviedo, Spain. Campus de Viesques, 33204 Gijón.

E-mail: {palaciosjuan,crvela,puente}@uniovi.es

Inés González-Rodríguez

Department of Mathematics, Statistics and Computing. University of Cantabria, Spain.

Los Castros s/n, 39005 Santander. E-mail: ines.gonzalez@unican.es

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 (see (Pinedo, 2008)). For a number of machines $m \geq 3$ this problem is NP-complete; in consequence, it is usually tackled via metaheuristics techniques. For instance, for makespan minimisation, Guéret and Prins (1998) describe two heuristic methods to obtain a list of operation priorities later used in a list-scheduling algorithm; Liaw (1999) proposes a tabu search algorithm; Blum (2005) hybridises ant colony optimisation with beam search and Sha and Cheng-Yu (2008) propose a solution based on particle swarm optimisation. To minimise total tardiness, Naderi et al (2011) propose two metaheuristics based on genetic algorithms and variable neighbourhood search and for multiobjective open shop we find an ant colony algorithm combined with simulated annealing in (Panahi et al, 2008) and particle swarm optimisation in (Sha et al, 2010).

Traditionally, scheduling has been treated as a deterministic problem that assumes 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 (Herroelen and Leus, 2005). 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 and as an interesting tool for improving solution robustness and stability (Guiffrida and Nagi, 1998; Dubois et al, 2003) and (Petrovic et al, 2008).

Far from being trivial, extending heuristic strategies to uncertain settings usually requires a significant reformulation of both the problem and solving methods. This is patent in the available literature on job shop problems with uncertain processing times and/or flexible constraints. For instance, Dubois et al (1995) extend a constrained-based approach, Fortemps (1997) uses simulated annealing and Sakawa and Kubota (2000) propose a genetic algorithm in what can be seen as pioneering works in the application of metaheuristic strategies, followed by many authors, e.g. (González Rodríguez et al, 2008),(Puente et al, 2010),(Niu et al, 2008) or (Zheng et al, 2011). However, while there are many contributions to solve fuzzy job shop problems, the literature on fuzzy open shop is still scarce. Indeed, the open shop with uncertainty constitutes a relatively new and complex research line. Among the few existing proposals, in (Alcaide et al, 2006) a heuristic approach is proposed to minimise the expected makespan for an open shop problem with stochastic processing times and random breakdowns; González-Rodríguez et al (2010) minimise the expected makespan of an open shop with fuzzy durations using a genetic algorithm hybridised with local search, while Palacios et al (2011) use a particle swarm optimisation algorithm for the same problem. Finally, a possibilistic mixed-integer linear programming method is proposed in (Noori-Darvish et al, 2012) for an OSP with setup times, fuzzy processing times and fuzzy due dates to minimise total weighted tardiness and total weighted completion times.

Another issue that must be taken into account to reduce the gap between academic and real-world problems is the fact that many real-life applications require taking into account several conflicting points of view corresponding to multiple objectives. This is one of the reasons why the applications of multiobjective decision making techniques in engineering have grown in the recent decades (Pasandideh et al, 2013). Although Pareto optimality is undoubtedly the most extended approach to multicriteria optimisation, as Ehrgott (2005) puts it, “it is not the end of the story”, with other approaches to multiobjective optimisation in the

literature (Ehrgott and Gandibleux, 2000). Among these techniques, lexicographic and goal programming methods are some of the most popular ones (Farahani et al, 2010). The philosophy behind goal programming (Romero, 2001) can be traced back to the theories of rational decision developed in the 1950s, especially the concept of satisficing solutions: in a complex environment, the decision maker's aim may be to reach a certain satisfactory level for every relevant objective, rather than optimising its value. Also, lexicographic problems arise naturally when conflicting objectives exist in a decision problem but for reasons outside the control of the decision maker the objectives have to be considered in hierarchical manner. Recent examples of real-world problems where these techniques are applied can be found, for instance, in (Ehrgott, 2005), (Diaz-Balteiro and Romero, 2008), (Puente et al, 2013), (Coshall and Charlesworth, 2011), and (Liberatore et al, 2013). Additionally, there exist interesting relationships between lexicographic and Pareto-optimal solutions. Indeed, "lexicographic minimisation is well-suited to seek a compromise between conflicting interests, as well as reconciling this requirement with the crucial notion of Pareto-optimality" (Bouveret and Lemaître, 2009).

To our knowledge, a lexicographical goal programming approach to solve multiobjective instances of fuzzy open shop has never been taken in the still scarce literature on this problem. This paper attempts to contribute to filling this gap. To this end, in the sequel we propose a multiobjective particle swarm optimisation (MOPSO) algorithm to solve instances of open shop where uncertain processing times are modelled with triangular fuzzy numbers and flexible due dates are modelled with fuzzy sets. In Section 2 we provide some background on fuzzy sets, which will be used in Section 3 to formulate the Fuzzy Open Shop Problem (FOSP). We adopt a lexicographic goal programming approach to define an objective function which combines minimisation of the expected fuzzy makespan and maximisation of overall due-date satisfaction. The resulting problem is solved by means of a

particle swarm optimization method searching in the space of possibly active schedules, as proposed in Section 4. Section 5 reports results from the experimental study which illustrate the potential of the proposed method. Finally, in Section 6 we summarise the main conclusions and propose some ideas for future work.

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 be able to estimate, for instance, an interval for the possible processing time or its most typical value. In 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.

2.1 Fuzzy interval arithmetic to model processing times

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 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:

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

Triangular fuzzy numbers and more generally fuzzy intervals have been extensively studied in the literature (cf. (Dubois and Prade, 1986)). 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* and its *support* 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*. However, computing the resulting equation is in general cumbersome, if not intractable. It can be somewhat simplified for two fuzzy numbers M and N, so the α-cuts M_α and N_α are closed bounded intervals of the form $[\underline{m}_\alpha, \overline{m}_\alpha]$ and $[\underline{n}_\alpha, \overline{n}_\alpha]$, 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 $f(M,N)$:

$$f(M,N) = \cup_{\alpha \in (0,1]} [f(\underline{m}_\alpha, \underline{n}_\alpha), f(\overline{m}_\alpha, \overline{n}_\alpha)] \tag{2}$$

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

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

Unfortunately, for the maximum of TFNs there is no such simplified expression. Being an isotonic function, we can use equation (2) above, but in general this still requires an infinite number of computations, since 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,

1997) 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 (Niu et al, 2008) and (Chen and Chang, 2001). Therefore, for any two TFNs M and N, their maximum will be approximated as follows:

$$\max(M,N) \sim M \sqcup N = (\max(m^1,n^1), \max(m^2,n^2), \max(m^3,n^3)). \tag{4}$$

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

For a fuzzy number N, its membership function μ_N can be interpreted as a possibility distribution on the real numbers. This allows to define the *expected value* of a fuzzy number (Liu and Liu, 2002), given for a TFN A by

$$E[A] = \frac{1}{4}(a^1 + 2a^2 + a^3). \tag{5}$$

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 (Dubois et al, 2003). It induces a total ordering $\leq_E$ in the set of fuzzy intervals (Fortemps, 1997), where for any two fuzzy intervals M,N $M \leq_E N$ if and only if $E[M] \leq E[N]$.

2.2 Modelling flexible due dates

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:

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

This expresses a flexible threshold "less than", representing the satisfaction level $sat(t) = \mu_D(t)$ for the ending date t of the job (Dubois et al, 2003). 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* (Sakawa and Kubota, 2000; Celano et al, 2003):

$$AI(C,D) = \frac{area(D \cap C)}{area(C)} \tag{7}$$

where $area(D \cap C)$ and $area(C)$ denote the areas under the membership functions of $(D \cap C)$ and C respectively. The intuition behind this definition is to measure the degree to which C is contained in D (the degree of subsethood).

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

3.1 Fuzzy schedules from crisp task orderings

A schedule s for an open shop problem of size $n \times m$ (n jobs and m machines) may be determined by a decision variable $\mathbf{z} = (z_1,\ldots,z_{nm})$ 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 belonging to 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 $\mathbf{p}$ of size $n \times m$. For a given task processing order $\mathbf{z}$ and a task o_{ij}, its starting time $S_{ij}(\mathbf{z},\mathbf{p})$ is the maximum (eq. 4) between the completion times of the task preceding o_{ij} in its job, let it be denoted o_{ik}, and the task preceding o_{ij} in its machine, let it be denoted o_{lj}:

$$S_{ij}(\mathbf{z},\mathbf{p}) = C_{ik}(\mathbf{z},\mathbf{p}) \sqcup C_{lj}(\mathbf{z},\mathbf{p}) \tag{8}$$

where $C_{ik}(\mathbf{z},\mathbf{p})$ or $C_{lj}(\mathbf{z},\mathbf{p})$ are taken to be zero if o_{ij} is the first task to be processed either in its job or its machine. Then, its completion time $C_{ij}(\mathbf{z},\mathbf{p})$ is obtained by adding its duration p_{ij} to $S_{ij}(\mathbf{z},\mathbf{p})$:

$$C_{ij}(\mathbf{z},\mathbf{p}) = S_{ij}(\mathbf{z},\mathbf{p}) + p_{ij} \tag{9}$$

The completion time of a job J_i will then be the maximum completion time of all its tasks, that is, $C_i(\mathbf{z},\mathbf{p}) = \sqcup_{1\leq j\leq m}\{C_{ij}(\mathbf{z},\mathbf{p})\}$.

For this schedule, the *fuzzy makespan* $C_{max}(\mathbf{z},\mathbf{p})$ is defined as the maximum of job completion times:

$$C_{max}(\mathbf{z},\mathbf{p}) = \sqcup_{1\leq i\leq n}\left(C_i(\mathbf{z},\mathbf{p})\right) \tag{10}$$

Notice that when uncertain durations are given as fuzzy intervals the schedule s will be fuzzy in the sense that the starting and completion times of all tasks as well as the makespan are fuzzy intervals. These may be interpreted as possibility distributions on the values that each time may take. Fuzzy intervals are thus used to represent our incomplete knowledge of problem parameters related to durations and, in consequence, our incomplete knowledge of starting and completion times for all tasks. However, the task processing order represented by $\mathbf{z}$ that determines such schedule is crisp: there is no uncertainty regarding the order in which tasks are to be processed.

Given a fuzzy schedule, it is necessary to give a precise definition of what "optimal makespan" means, 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 of minimising the makespan translates, in practice, into minimising its expected value $E[C_{max}]$ (eq. 5).

While also being fuzzy sets, due dates d_i for jobs J_i, $i = 1,\ldots,n$, do not model uncertainty. Instead, they model flexible constraints, introducing grades in the traditionally

Boolean notion of feasibility (cf. (Dubois, 2011) and the references therein for the semantics of fuzzy sets and their role in decision making). In this setting, the *agreement index*, $AI(C_i(\mathbf{z},\mathbf{p}),d_i)$ (eq. 7), denoted $AI_i(\mathbf{z},\mathbf{p})$ for short, measures to what degree the flexible due date d_i is satisfied by the fuzzy time $C_i(\mathbf{z},\mathbf{p})$. The degree of overall due-date satisfaction for schedule s may then be obtained by aggregating the satisfaction degrees $AI_i(\mathbf{z},\mathbf{p})$, $i = 1,\ldots,n$. In particular, we shall consider two aggregation functions, the minimum and the average, previously used in the literature concerning shop scheduling with soft constraints, for instance, in (Sakawa and Kubota, 2000; González Rodríguez et al, 2008; Lei, 2008). The minimum is inspired by the seminal paper on fuzzy decision making (Bellman and Zadeh, 1970), while the average provides an alternative for which the compensation property holds. Hence, the degree $AI_{ag}(\mathbf{z},\mathbf{p})$ to which a schedule s determined by an ordering $\mathbf{z}$ satisfies due dates will be determined by one of the two following formula:

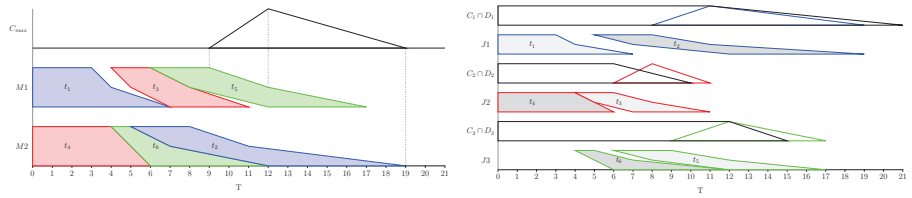
$$AI_{av}(\mathbf{z},\mathbf{p}) = \frac{1}{n}\sum_{i=1}^{n} AI_i(\mathbf{z},\mathbf{p}), \tag{11}$$

$$AI_{min}(\mathbf{z},\mathbf{p}) = \min_{i=1,\ldots,n} AI_i(\mathbf{z},\mathbf{p}) \tag{12}$$

Clearly both $AI_{av}(\mathbf{z},\mathbf{p})$ and $AI_{min}(\mathbf{z},\mathbf{p})$ should be maximised. Notice however that they model different requirements and encourage different behaviours. In the cases when there is no possible confusion regarding the order $\mathbf{z}$ or the processing times $\mathbf{p}$, we may simplify the notation and write AI_{ag} or C_{max}.

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

$$\mathbf{p} = \begin{pmatrix} (3,4,7) & (3,4,7) \\ (2,3,4) & (4,5,6) \\ (3,4,5) & (1,2,6) \end{pmatrix} \mathbf{d} = \begin{pmatrix} (11,21) \\ (6,10) \\ (12,15) \end{pmatrix}$$



(a) Machine oriented

(b) Job oriented

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

Here $p_{21} = (2,3,4)$ is the processing time of task o_{21}, the task of job J_2 to be processed in machine M_1 and $d_2 = (6,10)$ is the flexible due date for job J_2. Figures 1 (a) and (b) show the Gantt charts (both machine and job oriented) adapted to TFNs of the schedule given by the decision variable **z**=(1, 4, 6, 3, 5, 2). They represent the partial schedules on each machine and each job obtained from this 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}, which are respectively 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 $C_{21} = S_{21} + p_{21} = (4,5,7) + (2,3,4) = (6,8,11)$. Also, it is easy to see that $C_{max} = (9,12,19)$ (see Figure 1 (a)), so $E[C_{max}] = 13$. Regarding due dates, in Figure 1 (b) we can see that the completion time of job J_1 always satisfies its due date, so $AI_1 = 1$, whereas for job J_2 $area(C_2) = 5/2$ and $area(d_2 \cap C_2) = 4/3$, so $AI_2 = 0.53$, and analogously $AI_3 = 0.75$. Hence, the aggregated degrees of due date satisfaction will be $AI_{min} = 0.53$ and $AI_{av} = 0.76$.

3.2 Multiobjective model

For the fuzzy open shop problem we are interested both in maximising the aggregated due-date satisfaction AI_{ag} and minimising the expected makespan $E[C_{max}]$. A well-established

approach dealing with multiple and possibly conflicting objectives is lexicographic goal programming (Ehrgott, 2005; Tamiz et al, 1998), assuming that the decision makers establish a priority structure as well as target levels for the different objectives.

Before we formulate the resulting problem, notice that $AI_{ag}(\mathbf{z},\mathbf{p}) \in [0,1]$ for both aggregation operators. Hence, maximising $AI_{ag}(\mathbf{z},\mathbf{p})$ is equivalent to minimising $1-AI_{ag}(\mathbf{z},\mathbf{p})$, which could be interpreted as the degree to which due dates are violated. In consequence, we can restate the objective of our problem as minimising both $E[C_{max}(\mathbf{z},\mathbf{p})]$ and $1-AI_{ag}(\mathbf{z},\mathbf{p})$.

Let C_{max} and $1-AI_{ag}$ be ordered according to their priority, and let f_1 denote the objective with highest priority and f_2 denote the secondary objective. Also, let us assume that the decision makers establish target values $b_1, b_2 \geq 0$ for f_1 and f_2. Clearly, these values should not be exceeded, which translates into the following goal constraints:

$$f_i(\mathbf{z},\mathbf{p}) + \Delta_i^- - \Delta_i^+ = b_i, \quad i = 1,2 \tag{13}$$

where $\Delta_1^+, \Delta_2^+ \geq 0$, the positive deviations from the targets, should be minimised. This results in the following *lexicographic goal programming model* for the fuzzy open shop problem (FOSP):

$$\begin{cases} \text{lexmin} & (\Delta_1^+, \Delta_2^+) \\ \text{subject to:} & \\ & f_i(\mathbf{z},\mathbf{p}) + \Delta_i^- - \Delta_i^+ = b_i, \quad i = 1,2, \\ & b_i \geq 0, \quad i = 1,2, \\ & \Delta_i^-, \Delta_i^+ \geq 0, \\ & 1 \leq z_l \leq nm, \quad l = 1,\ldots,nm, \\ & z_l \neq z_k, \quad k \neq l \\ & z_l \in \mathbb{Z}^+, \quad l = 1,\ldots,nm, \end{cases} \tag{14}$$

where lexmin denotes lexicographically minimising the objective vector (Δ_1^+, Δ_2^+).

The resulting problem can be denoted $O|fuzz\ p_i, fuzz\ d_i|LexGP(E[C_{max}], 1 - AI_{av})$ according to the three-field notation from (Graham et al, 1979), extended to multicriteria scheduling in the spirit of (T'kindt and Billaut, 2006).

4 Particle swarm optimization for the FOSP

Particle swarm optimisation (PSO) is a population-based stochastic method inspired by bird flocking or fish schooling, first proposed in (Kennedy and Eberhart, 1995) which has been successfully applied to solve complex combinatorial optimization problems; recent examples of this success can be found in (Belmecheri et al, 2013), (Jia and Seo, 2013), and (Kim and Son, 2012). In particular, it has been applied to scheduling problems, among others, in (Tassopoulos and Beligiannis, 2012), (Vijay chakaravarthy et al, 2013), and (Marinakis and Marinaki, 2013) as well as the already mentioned references devoted to the open shop problem (Sha and Cheng-Yu, 2008; Sha et al, 2010).

In PSO, each position in a multidimensional 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 are represented by multidimensional particles flying through the problem space, changing their position and velocity by following the current optimum particles *pbest* and *gbest*. A generic PSO algorithm is given 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.

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**

Update velocity $\mathbf{v}^k$;

Update position $\mathbf{x}^k$;

Evaluate particle k;

Update *pbest* and *gbest* values;

return The schedule from the best particle evaluated so far;

Algorithm 1: A generic PSO algorithm

In the following, we present a multiobjective PSO algorithm for the FOSP with lexicographic goal programming defined in the previous section. A preliminary version of this algorithm was presented in (Palacios et al, 2011) to minimise the expected makespan of fuzzy open shop.

4.1 Position Representation and Evaluation

For each particle k in the swarm, its position $\mathbf{x}^k$ is represented with a priority-based representation. Thus, the decision variable $\mathbf{z}^k$ is encoded as a *priority array* $\mathbf{x}^k = (x_l^k)_{l=1\ldots nm}$ 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.

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

$$\mathbf{z} = (4,1,5,2,3,6) \qquad \mathbf{pos^z} = (2,4,5,1,3,6)$$

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

$$\mathbf{x} = (2.3,3.7,5.4,0.8,2.8,5.9)$$

Conversely, from every particle position $\mathbf{x}$ we can obtain a position array $\mathbf{pos^x}$ (and the corresponding decision variable) where pos^x_i is the position of x_i if the elements of $\mathbf{x}$ were reordered in non-decreasing order.

A particle may be decoded in several ways. For deterministic job shop and, by extension, for open shop scheduling, it is common to use the G&T algorithm (Giffler and Thompson, 1960), 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 (Gonçalves et al, 2005) 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 parametrised 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

Input A FOSP instance and a particle position $\mathbf{x}^k$

Output A schedule for the input instance considering the priorities given by $\mathbf{x}^k$

$\Omega \leftarrow \{1,\ldots,nm\}$;

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 S_{l^*}}

$\Omega \leftarrow \Omega - \{l^*\}$;

return The schedule s given by $\{S_l : l \in \{1,\ldots,nm\}\}$

Algorithm 2: The $pFG\&T$

$\delta = 0$ the search is constrained 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 (Sha and Cheng-Yu, 2008) to the deterministic OSP, under the name "parameterized active schedule generation algorithm". Algorithm 2, denoted $pFG\&T$, is an extension of parametrised G&T to the case of fuzzy processing times proposed in (Palacios et al, 2011). Throughout the algorithm, Ω denotes the set of tasks that have not been scheduled yet, $\mathbf{x}^k$ denotes the priority array and S_l and C_l denote the starting and completion time of task o_{ij} such that $l = (i-1)m + j$. 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*.

4.2 Particle movement

4.2.1 Velocity update

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 greater 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. Velocity and particle updating is controlled by the inertia weight w according to Algorithm 3. In the updating process of each particle k and dimension d an element of randomness is introduced, making it dependent on $pbest_d^k$ with probability p_1 and on $gbest_d$ with probability p_2, where $p_1, p_2 \in [0,1]$ are constants such that $p_1 + p_2 \leq 1$.

4.2.2 Mutation

When adapting PSO to discrete optimisation, there is a risk of getting stuck in local minima when velocity is limited to absolute values (Hu et al, 2003). In order to introduce diversity, after a particle k moves to a new position, we randomly choose a dimension d 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 $x_d^k \geq (nm/2)$), x_d^k will take a random value in $[0,n]$ and $v_d^k = -1$.

4.2.3 Diversification strategy

In the case that all particles had the same *pbest* solution, they could be trapped into local optima. To prevent such situation, a diversification strategy is proposed in (Sha and Cheng-Yu, 2008) in order to keep the different *pbest* solutions. According to this strategy, the *pbest* solution of each particle is not the best solution found by the particle itself, but one of the

Input A particle position $\mathbf{x}^k$ and velocity $\mathbf{v}^k$, best particle and swarm positions $pbest^k$ and $gbest$, inertia w and updating probabilities p_1, p_2

Output The updated particle position $\mathbf{x}^k$ and velocity $\mathbf{v}^k$

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 updated particle position $\mathbf{x}^k$ and velocity $\mathbf{v}^k$;

Algorithm 3: Particle movement

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 has

a different makespan from all *pbest* solutions, then the worst *pbest* solution is replaced by the new one; else, the set of N *pbest* solutions remains unchanged.

5 Experimental Results

For the experimental study, we use the fuzzy open shop instances proposed in (González-Rodríguez et al, 2010). These were obtained by fuzzyfying the well-known benchmark from (Brucker et al, 1997), consisting of 6 families, denoted *J3, J4,..., J8*, of sizes 3×3 to 8×8, with 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 and compare our results on expected makespan to those of the memetic algorithm (MA) proposed in (González-Rodríguez et al, 2010), which combines a genetic algorithm with a local search schema. According to the results reported in (González-Rodríguez et al, 2010), this MA outperforms the genetic algorithm alone when run under equivalent running conditions; additionally, on crisp instances of OSP it improves two GAs from (Liaw, 2000) and (Prins, 2000) and is competitive with two PSO algorithms from (Sha and Cheng-Yu, 2008), one of them hybridised with beam search.

For each original deterministic problem instance there are 10 fuzzy versions, generated by transforming the original crisp processing times into symmetric TFNs such that their modal value corresponds to the original duration. To add a due date d_i for each job J_i we follow Andresen et al (2008): first, we define a generic due date $d_i = TF \times \sum_{j=1}^{m} p_{ij}^2$, where TF is a tightness factor; then, we use two different tightness factors to have the earliest and latest due dates: d_i^1, with $TF = 1.10$, and d_i^2, with $TF = 1.15$.

Given the method for generating due dates, in *per0* instances, where all jobs have the same workload (and consequently the same due date), the makespan and due date satisfaction are strongly correlated objectives, making these instances unsuitable for our multiobjective study. Therefore, the experimental analysis will be conducted on the instances *per10* and *per20* of size 7×7 and 8×8, making it a total of 120 instances, these being the hardest ones to solve when both objectives are considered.

For each problem instance, we have run the PSO algorithm using different objectives: we have considered the three single-objective functions $E[C_{max}], AI_{av}$ and AI_{min} and the four multiobjective functions that result from combining the two choices of aggregation function for due date satisfaction ($AI_{ag} = AI_{min}$ or $AI_{ag} = AI_{av}$) and the two possible priority structures for objectives ($f_1 = C_{max}, f_2 = AI_{ag}$ or $f_1 = AI_{ag}, f_2 = C_{max}$).

For the multiobjective cases, it is necessary that the target values for both objectives be fixed. As already mentioned, in practice these target values should be given by the DM based on his/her expertise in the problem. Unfortunately, such expert knowledge is not available for the set of synthetic instances used herein. Instead, we emulate the DM and try to gain insight into the problem instances with some preliminary runs of the PSO using $E[C_{max}], AI_{av}$ and AI_{min} as single objectives, using the parameter values proposed in (Sha and Cheng-Yu, 2008). Then, we set b_1 (resp. b_2 for $1 - AI_{ag}$) equal to the worst value of $E[C_{max}]$ ($1 - AI_{ag}$) across 30 runs of the PSO.

5.1 Parameter setting

To ensure that the algorithm yields reliable solutions within a reasonable amount of time, the Taguchi method is used for parameter tuning. Table 1 shows the parameters of our algorithm together with the four possible values (factor levels in the Taguchi terminology) considered

for each of them. A caveat in changing the swarm size N is its considerable effect on the algorithm's runtime if a constant number of iterations is considered. Now, it is common in literature to measure the computational effort of a metaheuristic in terms of the number of objective-function evaluations, which is independent of the computer system. This suggests adjusting the number of iterations in such a way that the PSO evaluates roughly the same number of particles for all possible swarm sizes: for $N = 60, 80, 100$ and 120, the number of iterations $Niter$ is set respectively to 3000, 2250, 1800 and 1500. As for the second parameter, the inertia weight w, it should be linearly decreasing from a starting value, thus stimulating the exploration of the PSO. We consider two possible starting values, 0.9 and 0.7, and two possible slopes, $0.6/Niter$ and $0.2/Niter$, which should allow to analyse the behaviour of the PSO with either more exploration or more exploitation in the last iterations. In consequence, w will be linearly decreasing in four possible intervals, as shown in Table 1. Regarding the guiding probabilities, p_1 and p_2, since their sum must be less or equal to 1, we consider them as a single factor: given the values 0.7, 0.5, 0.3 and 0.1, p_1 and p_2 simultaneously traverse these values in increasing and decreasing order respectively, that is, first $p_1 = 0.7$ and $p_2 = 0.1$, then $p_1 = 0.5$ and $p_2 = 0.3$ and so forth. Thus, we always ensure that the constraint $p_1 + p_2 \leq 1$ holds, while covering a varied sample of values for both probabilities. Finally, for the delay parameter we consider the two extremes values, $\delta = 0$ — which in the deterministic case restricts the search to the space of *non-delay* schedules— and $\delta = 1$, together with two intermediate values $\delta = 0.25$ and $\delta = 0.75$.

With a total of four parameters and four factor levels each, the orthogonal array L'_{16} is pertinent for the Taguchi analysis. For every combination of parameter values given by the orthogonal array we run the PSO with the four multiobjective functions: $L(C_{max}, AI_{av})$, $L(C_{max}, AI_{min})$, $L(AI_{av}, C_{max})$, and $L(AI_{min}, C_{max})$ on a fuzzy instance of each 8×8 problem.

Table 1: Parameter settings

Parameters	Factor level			
	1	2	3	4
Swarm Size (N)	60	80	100	120
Inertia Weight (w) linearly decreasing [from,to]	$[0.9,0.3]$	$[0.7,0.1]$	$[0.9,0.7]$	$[0.7,0.5]$
Guiding Probabilities ($gp = (p_1, p_2)$)	$(0.7,0.1)$	$(0.5,0.3)$	$(0.3,0.5)$	$(0.1,0.7)$
Delay Parameter (δ)	0	0.25	0.75	1

To measure the quality of each configuration we need a value that can consistently combine such heterogeneous values as C_{max}, AI_{av} and AI_{min} while taking into account the lexicographical goal programming nature of the model. First, we consider the distance of each value to its corresponding target, averaged across ten runs of the algorithm and normalised so as to unify scales (notice that such distance is taken to be zero if the target is reached). Let d_1 and d_2 denote, respectively, the normalised distance values for the primary and secondary objective. These values will allow us to characterise the algorithm's performance for the Taguchi analysis as follows: if the first target is reached, i.e. $d_1 = 0$, then the performance is given by d_2 (the distance to the second objective); in the worse case that the primary objective does not reach its target ($d_1 > 0$), then the performance is given by $1 + d_1$. Since $0 \leq d_2 \leq 1$, this guarantees that the algorithm is always considered to perform worse when the target for the primary objective is not reached, as well as discriminating among solutions taking into account how far they are from reaching each target. We have opted for using this performance measure directly, instead of the classical signal-to-noise ratio, in the line of the use of the Taguchi method in (Jia and Seo, 2013) and (Wang et al, 2013) for scheduling problems.

Table 2 shows, for every combination of factor levels in the orthogonal array, the average performance value for each of the four multiobjective functions considered. It is based on these values that we can compute the response value of each parameter and analyse their significance rank. As a summary, Figure 2 depicts the response values of each parameter for each of the four objective functions, illustrating the effect of the parameter levels on the algorithm's performance. Clearly, the most significant parameter for all objective functions is δ, with a difference between the highest and lowest level over 1.25 of a maximum possible

Table 2: Orthogonal tabulation and average performance values

Exp.	Parameter levels				Average performance			
	N	w	gp	δ	$L(C_{max}, AI_{min})$	$L(AI_{min}, C_{max})$	$L(C_{max}, AI_{av})$	$L(AI_{av}, C_{max})$
1	1	1	1	1	0.919	0.727	0.524	0.727
2	1	2	2	2	0.051	0.043	0.208	0.178
3	1	3	3	3	1.266	1.344	1.302	1.409
4	1	4	4	4	1.701	1.759	1.661	1.871
5	2	1	2	3	1.084	1.104	1.100	1.015
6	2	2	1	4	1.282	1.423	1.361	1.546
7	2	3	4	1	0.970	0.792	1.035	0.764
8	2	4	3	2	0.324	0.258	0.556	0.186
9	3	1	3	4	1.650	1.693	1.702	1.776
10	3	2	4	3	1.222	1.217	1.193	1.294
11	3	3	1	2	0.293	0.226	0.175	0.004
12	3	4	2	1	1.023	0.775	0.881	0.740
13	4	1	4	2	0.822	0.429	0.441	0.197
14	4	2	3	1	1.055	0.802	1.063	0.953
15	4	3	2	4	2.000	2.000	2.000	1.983
16	4	4	1	3	0.912	0.930	0.920	0.605

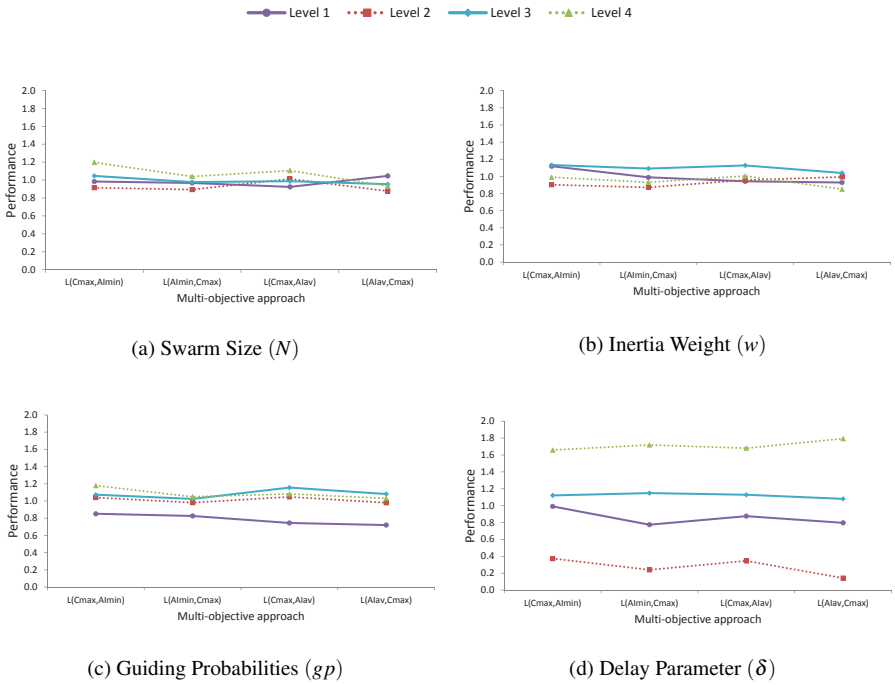
difference of 2.00 (see Figure 2(d)). The second most significant parameter is the pair of guiding probabilities (Figure 2(c)), although their effect is significantly smaller. Finally, the smallest effect on the performance for all functions is obtained with the swarm size and the inertia weight (see Figure 2(a),(b))). Additionally, for the two most significant parameters it can be clearly seen that the best level remains the same for all four objective functions. This is not the case for swarm size and inertia weight, where the best levels differ for $L(C_{max}, AI_{av})$ and $L(AI_{av}, C_{max})$; however, the difference is relatively small, 0.079 for swarm size and 0.142 for inertia weight. In consequence, we will take the factor level that performs best for all but one objective functions, this being a good value in all cases.

As a result of this analysis, the parameter setting in what follows will be $\delta = 0.25$, $gp = (p_1, p_2) = (0.7, 0.1)$, w linearly decreasing from 0.7 to 0.1, and swarm size $N = 80$ for all objective functions.

5.2 Highest priority for makespan minimisation

Let us first consider the case where C_{max} is the objective with highest priority and let $L(C_{max}, AI_{av})$ and $L(C_{max}, AI_{min})$ denote the resulting multiobjective functions for $AI_{ag} = AI_{av}$ and $AI_{ag} = AI_{min}$ respectively.

In order to illustrate the algorithm's convergence, we first focus on a single problem instance. Figure 3 shows the convergence pattern of $L(C_{max}, AI_{min})$ for a fuzzy instance generated from *J8-per20-1*, one of the largest and hardest instances. We can see how the algorithm shows a proper convergence: initially the algorithm minimises the expected makespan $E[C_{max}]$ while the behaviour of AI_{min} is erratic. However, once the algorithm has reached the expected makespan target (around the 250th iteration), it starts maximising AI_{min} instead. We can also observe the evolution of the AI_{av} value and its correlated behaviour w.r.t. AI_{min}.



(a) Swarm Size (N)

(b) Inertia Weight (w)

(c) Guiding Probabilities (gp)

(d) Delay Parameter (δ)

Fig. 2: Average performance of the four multiobjecive-PSO for each parameter level.

Analogous convergence curves show that the number of iterations can be reduced for 7×7 to 2100 iterations.

Tables 3 and 4 contain a summary of the results obtained when makespan minimisation has the highest priority. For each objective function used by the PSO they report the values of $E[C_{max}]$, AI_{av} and AI_{min} in the solution, averaged across the 30 executions of the PSO and the 10 fuzzy instances generated from the same original problem, together with the standard deviations. The average values are shown in bold when they reach the target for the corresponding objective.

A first look at Tables 3 and 4 confirms the strong correlation between the values of AI_{min} and A_{av}, both measuring the overall due-date satisfaction. In most cases, the single-objective version using any of these aggregated values reaches the target value established

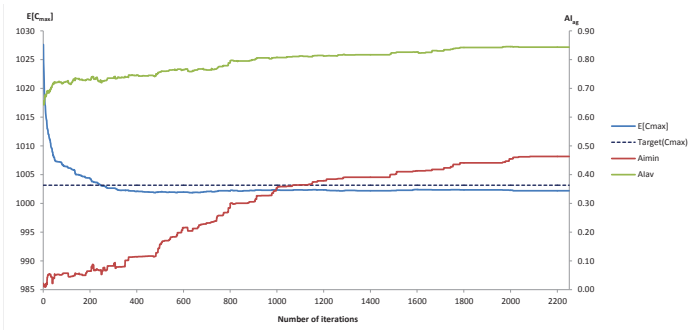


Fig. 3: Evolution of $E[C_{max}]$ and $E[AI_{min}]$ on the $L(C_{max}, AI_{min})$ version for the *J8-per20-1* instance

for the other aggregated measure. That is, when any one of these aggregated measures is optimised, the alternative one is also optimised.

Let us now compare results obtained by the proposed multiobjective approach using $L(C_{max}, AI_{min})$ and $L(C_{max}, AI_{av})$ with the results obtained when optimising a single criterion. For the objective with the highest priority—minimisation of expected makespan—we see that both multiobjective approaches behave similarly to the single-objective function. In particular, they always reach the expected makespan target. Additionally, the multiobjective approach obtains a clear improvement in due-date satisfaction. Indeed, for all instances, AI_{min} values obtained with $L(C_{max}, AI_{min})$ are in average 159% better than those obtained using $E[C_{max}]$ as single-objective function. There are however remarkable differences in the improvement rate depending on the instance type. For example, due-date satisfaction improves only 8% for *J7-per10* instances and 16% for *J8-per10*, but this improvement scales up to 164% and 450% in *per20* instances of sizes 7×7 and 8×8 respectively. This variability is due to the fact that, as mentioned above, the dependency between $E[C_{max}]$ and AI_{min} is greater for *per10* instances, given the way in which the original benchmark was created. In consequence, for *per10* problems, when the makespan is optimised, due-date satisfaction is

also being optimised to a certain extent; however this is not always the case for an arbitrary open shop problem.

Regarding AI_{av} values, they improve 7.2% when using $L(C_{max}, AI_{av})$. Again there is a remarkable variability in the improvement depending on the family of problems: 2.1% for *per10* instances and 12.2% for *per20* instances. It is also tempting to conclude that the gain obtained with $L(C_{max}, AI_{min})$ is much higher than that obtained with $L(C_{max}, AI_{av})$. However, this is only a scale effect. If instead of considering absolute gains, we measure the reduction of the gap between the AI_{min} and the AI_{av} values and their corresponding targets, the multiobjective approach $L(C_{max}, AI_{min})$ is in average over 36% better than $E[C_{max}]$ and $L(C_{max}, AI_{av})$ is also over 36% better than $E[C_{max}]$ w.r.t. the corresponding secondary target. In any case, it is worth noticing that for *per10* instances $L(C_{max}, AI_{min})$ performs better than $L(C_{max}, AI_{av})$ whereas for *per*20 instances the best performance corresponds to $L(C_{max}, AI_{av})$ in terms of gap reduction w.r.t. its secondary target. A possible explanation is that AI_{min} is a more demanding aggregation operator. If it is relatively "easy" to satisfy the due dates for all jobs (at least to a certain extent), then $0 < AI_{min} \leq AI_{av}$ and AI_{min} will probably provide a better guide for maximising due date satisfaction. However, as long as it is likely that one of the due dates is not satisfied at all in schedules with good makespan values (as is the case for *per20* problems), then $AI_{min} = 0$ with high probability, thus providing a poor guide for the optimisation process.

Finally, the correlation between both aggregation operators is further confirmed if we look at the behaviour of AI_{av} in case of $L(C_{max}, AI_{min})$ and AI_{min} in case of $L(C_{max}, AI_{av})$: both multiobjective approaches significantly reduce the alternative due-date objective, with a gap-improvement of approximately 26% in both cases.

Let us now compare the multiobjective PSO using $L(C_{max}, AI_{av})$ or $L(C_{max}, AI_{min})$ with the single-objective memetic algorithm (MA) from (González Rodríguez et al, 2010) in

terms of expected makespan minimisation. Table 5 contains the expected makespan results for each method—MA optimising only $E[C_{max}]$, PSO with $L(C_{max},AI_{av})$ and PSO with $L(C_{max},AI_{min})$— with values averaged across the 10 instances of each size and 30 runs of the algorithm. Clearly, the PSO with both multiobjective functions $L(C_{max},AI_{av})$ and $L(C_{max},AI_{min})$ compares favourably with the single-objective MA in terms of makespan values. Indeed, the multiobjective PSO reduces $E[C_{max}]$ values about 2.25% (slightly over 3% for *per10* instances and slightly below 1.5% for *per20* instances), with no significant differences between different problem sizes or different aggregated measures for due-date satisfaction. This reduction may not seem very important in absolute values. However, on a closer look we can see that the MA never reaches the expected makespan target value, whereas the multiobjective PSO reaches this target in all instances. We can conclude that our multiobjective PSO outperforms the previous single-objective algorithm when it comes to optimising the objective with the highest priority (makespan), while also optimising the secondary objective.

5.3 Highest priority for due-date satisfaction

We now consider the alternative priority structure where due-date satisfaction becomes the primary objective; let $L(AI_{ag},C_{max})$ denote the resulting lexicographic goal programming multiobjective function. If we now compare each $L(AI_{ag},C_{max})$ with the corresponding aggregated due-date satisfaction value AI_{ag} (AI_{min} or AI_{av}), the results are analogous to the case where makespan was the first objective. In all instances $L(AI_{ag},C_{max})$ reaches the corresponding target for due-date satisfaction value whereas the gap between the expected makespan and its target value is reduced 36% in average when $AI_{ag} = AI_{min}$ and 40% in the case that $AI_{ag} = AI_{av}$. Figure 4 shows the $E[C_{max}]$ values (averaged across the 10 fuzzy

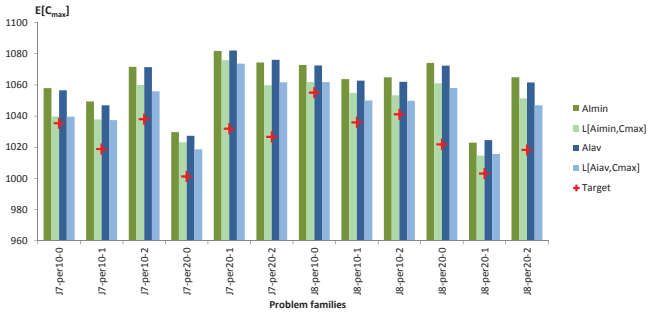


Fig. 4: Average $E[C_{max}]$ values obtained with AI_{min}, AI_{av} and the corresponding multiobjective $L(AI_{min}, C_{max})$ and $L(AI_{av}, C_{max})$.

instances of every original problem and the 30 executions of the PSO algorithm) obtained with AI_{min}, AI_{av} and the corresponding multiobjective functions on each family of problems. It also depicts the $E[C_{max}]$ target values for each family. We can clearly appreciate how the expected makespan behaves better in the multiobjective approach. We can also observe that AI_{av} used as single objective function obtains in general slightly better $E[C_{max}]$ values than the alternative AI_{min}. Also, its multiobjective counterpart $L(AI_{av}, C_{max})$ performs slightly better (in terms of makespan minimisation) than $L(AI_{min}, C_{max})$. The explanation, again, lies in the fact that AI_{min} is a more pessimistic aggregator of individual job due-date satisfaction. The figure also illustrates that, as above, the solutions are in general closer to the target values for *per10* instances than for *per20* ones.

6 Conclusions and future work

We have proposed a multiobjective approach for solving the open shop scheduling problem with uncertain durations and flexible due dates modelled using fuzzy sets. We have adopted a lexicographic goal programming framework to deal with the multiple objectives of minimis-

ing the project's makespan and maximising due-date satisfaction. The resulting problem has been solved by adapting a particle swarm optimisation algorithm to the hierarchical multiobjective framework. The experimental results, on fuzzy instances of well-known benchmark problems, illustrate the potential of our proposal. In general, the multiobjective approaches perform as well as their single-objective counterparts when it comes to optimising the objective with the highest priority, reaching the target levels in all cases. Additionally, the multiobjective approaches greatly improve on the secondary objective. Also, the multiobjective PSO algorithm compares favourably to a memetic algorithm from the literature in terms of makespan minimisation, when this is the objective with the highest priority.

In the future, we would like to contemplate an alternative approach to multiobjective optimisation, appropriate for the case when no priority structure among multiple objectives can or needs to be established. We would like to explore the known relationships between lexicographic and Pareto optimality, as well as extending the PSO algorithm to directly work with sets of non-dominated solutions. We would also like to adapt the PSO algorithm to other scheduling problems with uncertainty, such as job shop or resource-constrained project scheduling.

Acknowledgements We would like to thank the anonymous referees for their insightful and constructive comments. This research has been supported by the Spanish Government under research grants FEDER TIN2010-20976-C02-02 and MTM2010-16051 and by the Principality of Asturias (Spain) under grant Severo Ochoa BP13106.

References

Alcaide D, Rodriguez-Gonzalez A, Sicilia J (2006) A heuristic approach to minimize expected makespan in open shops subject to stochastic processing times and failures. Inter-

national Journal of Flexible Manufacturing Systems 17:201–226

Andresen M, Bräsel H, Mörig M, Tusch J, Werner F, Willenius P (2008) Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop. Mathematical and Computer Modelling 48:1279–1293

Bellman RE, Zadeh LA (1970) Decision-making in a fuzzy environment. Management Science 17(4):141–164

Belmecheri F, Prins C, Yalaoui F, L A (2013) Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. Journal of Intelligent Manufacturing 24(4):775–789

Blum C (2005) Beam-ACO—hybridizing ant colony optimization with beam search: an application to open shop scheduling. Computers & Operations Research 32(6):1565–1591

Bouveret S, Lemaître M (2009) Computing leximin-optimal solutions in constraint networks. Artificial Intelligence 173:343–364

Brucker P, Hunrink J, Jurisch B, Wöstmann B (1997) A branch & bound algorithm for the open-shop problem. Discrete Applied Mathematics 76:43–59

Celano G, Costa A, Fichera S (2003) An evolutionary algorithm for pure fuzzy flowshop scheduling problems. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 11:655–669

Chen SM, Chang TH (2001) Finding multiple possible critical paths using fuzzy PERT. IEEE Transactions on Systems, Man, and Cybernetics–Part B: 31(6):930–937

Coshall JT, Charlesworth R (2011) A management orientated approach to combination forecasting of tourism demand. Tourism Management 32:759–769

Diaz-Balteiro L, Romero C (2008) Making forestry decisions with multiple criteria: A review and an assessment. Forest Ecology and Management 255:3222–3241

Dubois D (2011) The role of fuzzy sets in decision sciences: Old techniques and new directions. Fuzzy Sets and Systems 184:3–28

Dubois D, Prade H (1986) Possibility Theory: An Approach to Computerized Processing of Uncertainty. Plenum Press, New York (USA)

Dubois D, Fargier H, Prade H (1995) Fuzzy constraints in job-shop scheduling. Journal of Intelligent Manufacturing 6:215–234

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

Ehrgott M (2005) Multicriteria Optimization, 2nd edn. Springer

Ehrgott M, Gandibleux X (2000) A survey and annotated bibliography of multiobjective combinatorial optimization. OR Spektrum 22:425–460

Farahani RZ, SteadieSeifi M, Asgari N (2010) Multiple criteria facility location problems: A survey. Applied Mathematical Modelling 34:1689–1709

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

Giffler B, Thompson GL (1960) Algorithms for solving production scheduling problems. Operations Research 8:487–503

Gonçalves J, Mendes J, de M RM (2005) A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research 167:77–95

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

González-Rodríguez I, Palacios JJ, Vela CR, Puente J (2010) Heuristic local search for fuzzy open shop scheduling. In: Proceedings IEEE International Conference on Fuzzy Systems,

FUZZ-IEEE2010, IEEE, pp 1858–1865

González Rodríguez I, Vela CR, Puente J (2010) A genetic solution based on lexicographical goal programming for a multiobjective job shop with uncertainty. Journal of Intelligent Manufacturing 21:65–73

Graham R, Lawler E, Lenstra J, Rinnooy Kan A (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 4:287–326

Guéret C, Prins C (1998) Classical and new heuristics for the open-shop problem: A computational evaluation. European Journal of Operational Research 107:306–314

Guiffrida AL, Nagi R (1998) Fuzzy set theory applications in production management research: a literature survey. Journal of Intelligent Manufacturing 9:39–56

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

Hu X, Eberhart RC, Shi Y (2003) Swarm intelligence for permutation optimization: a case study of n-queens problem. In: Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, IEEE, pp 243–246

Jia Q, Seo Y (2013) An improved particle swarm optimization for the resource-constrained project scheduling problem. International Journal of Advanced Manufacturing Technology In press

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: IEEE International Conference on Neural Networks, IEEE Press, New Jersey, pp 1942–1948

Kim BI, Son SJ (2012) A probability matrix based particle swarm optimization for the capacitated vehicle routing problem. Journal of Intelligent Manufacturing 23:1119–1126

Lei D (2008) Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems. International Journal of Advanced Manufacturing Technol-

ogy 37:157–165

Liaw CF (1999) A tabu search algorithm for the open shop scheduling problem. Computers and Operations Research 26:109–126

Liaw CF (2000) A hybrid genetic algorithm for the open shop scheduling problem. European Journal of Operational Research 124:28–42

Liberatore F, Ortuño MT, Tirado G, Vitoriano B, Scaparra MP (2013) A hierarchical compromise model for the joint optimization of recovery operations and distribution of emergency goods in humanitarian logistics. Computers & Operations Research In press

Liu B, Liu YK (2002) Expected value of fuzzy variable and fuzzy expected value models. IEEE Transactions on Fuzzy Systems 10:445–450

Marinakis Y, Marinaki M (2013) Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem. Soft Computing 17(7): 1159–1173

Naderi B, Fatemi Ghomi SMT, Aminnayeri M, Zandieh M (2011) A study on open shop scheduling to minimise total tardiness. International Journal of Production Research 49(15):4657–4678

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

Noori-Darvish S, Mahdavi I, Mahdavi-Amiri N (2012) 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

Palacios JJ, González-Rodríguez I, Vela CR, Puente J (2011) Particle swarm optimisation for open shop problems with fuzzy durations. In: Proceedings of IWINAC 2011, Part I, Springer, Lecture Notes in Computer Science, vol 6686, pp 362–371

Panahi H, Rabbani M, Tavakkoli-Moghaddam R (2008) Solving an open shop scheduling problem by a novel hybrid multi-objective ant colony optimization. In: Eighth International Conference on Hybrid Intelligent Systems, IEEE, pp 320–325

Pasandideh SHR, Niaki STA, V H (2013) A multi-objective facility location model with batch arrivals: two parameter-tuned meta-heuristic algorithms. Journal of Intelligent Manufacturing 24(2):331–348

Petrovic S, Fayad S, Petrovic D (2008) Sensitivity analysis of a fuzzy multiobjective scheduling problem. International Journal of Production Research 46(12):3327–3344

Pinedo ML (2008) Scheduling. Theory, Algorithms, and Systems., 3rd edn. Springer

Prins C (2000) Competitive genetic algorithms for the open-shop scheduling problem. Mathematical Methods of Operations Research 52:389–411

Puente J, Vela CR, González-Rodríguez I (2010) Fast local search for fuzzy job shop scheduling. In: Proceedings of ECAI 2010, IOS Press, pp 739–744

Puente J, Vela CR, González-Rodríguez I, Rodríguez LJ, Palacios JJ (2013) GRASPing examination board assignments for university-entrance exams. In: IEA-AIE 2013, Proceedings of, Springer, Lecture Notes in Computer Science, vol 7906, p 171–180

Romero C (2001) Extended lexicographic goal programming: a unifying approach. Omega 29:63–71

Sakawa M, Kubota R (2000) 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

Sha D, Lin HH, Hsu C (2010) A modified particle swarm optimization for multi-objective open shop scheduling. In: Proceeding of the International MultiConference of Engineers and Comuter Scientists, vol 3

Sha DY, Cheng-Yu H (2008) A new particle swarm optimization for the open shop scheduling problem. Computers & Operations Research 35:3243–3261

Tamiz M, Jones D, Romero C (1998) Goal programming for decision making: An overview of the current state-of-the-art. European Journal of Operations Research 111:569–581

Tassopoulos IX, Beligiannis GN (2012) Using particle swarm optimization to solve effectively the school timetabling problem. Soft Computing 16:1229–1252

T'kindt V, Billaut JC (2006) Multicriteria Scheduling. Theory, Models and Algorithms, 2nd edn. Springer

Vijay chakaravarthy G, Marimuthu S, Naveen Sait A (2013) Performance evaluation of proposed differential evolution and particle swarm optimization algorithms for scheduling m-machine flow shops with lot streaming. Journal of Intelligent Manufacturing 24:175–191

Wang L, Zhou G, Xu Y, Min L (2013) A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem. International Journal of Production Research pp 1–16

Zheng Y, Li Y, Lei D (2011) Swarm-based neighbourhood search for fuzzy job shop scheduling. International Journal of Innovative Computing and Applications 3(3):144–151

Table 3: Comparison of results for $E[C_{max}]$ highest priority on instances of size 7×7

	Objective	$E[C_{max}]$			AI_{min}			AI_{av}		
		Targ.	Avg	S.Dev	Target	Avg	S.Dev	Target	Avg	S.Dev
J7-per10-0	$L[C_{max}, AI_{min}]$	1035	**1032**	1.655	0.9569	0.9155	0.0992	0.9923	0.9818	0.0225
	$L[C_{max}, AI_{av}]$	1035	**1032**	1.878	0.9569	0.9114	0.0925	0.9923	0.9815	0.0205
	$E[C_{max}]$	–	**1030**	1.946	–	0.8182	0.1157	–	0.9669	0.0226
	AI_{min}	–	1058	9.363	–	**0.9873**	0.0108	–	**0.9968**	0.0035
	AI_{av}	–	1057	10.473	–	**0.9841**	0.0162	–	**0.9971**	0.0029
J7-per10-1	$L[C_{max}, AI_{min}]$	1019	**1017**	1.478	0.9303	0.7562	0.0321	0.9857	0.9352	0.0117
	$L[C_{max}, AI_{av}]$	1019	**1017**	1.664	0.9303	0.7529	0.0340	0.9857	0.9351	0.0137
	$E[C_{max}]$	–	**1017**	1.157	–	0.7502	0.0253	–	0.9346	0.0102
	AI_{min}	–	1049	8.212	–	**0.9723**	0.0212	–	**0.9916**	0.0083
	AI_{av}	–	1047	9.655	–	**0.9653**	0.0298	–	**0.9934**	0.0049
J7-per10-2	$L[C_{max}, AI_{min}]$	1038	**1033**	3.410	0.9358	0.7685	0.1091	0.9817	0.9252	0.0301
	$L[C_{max}, AI_{av}]$	1038	**1034**	2.935	0.9358	0.7495	0.1322	0.9817	0.9347	0.0275
	$E[C_{max}]$	–	**1031**	2.915	–	0.6873	0.1511	–	0.9091	0.0353
	AI_{min}	–	1072	13.352	–	**0.9713**	0.0183	–	**0.9898**	0.0073
	AI_{av}	–	1071	14.196	–	**0.9670**	0.0257	–	**0.9938**	0.0045
J7-per20-0	$L[C_{max}, AI_{min}]$	1001	**1001**	0.294	0.8278	0.3915	0.1116	0.9459	0.7322	0.0683
	$L[C_{max}, AI_{av}]$	1001	**1001**	0.343	0.8278	0.3140	0.1501	0.9459	0.7838	0.0433
	$E[C_{max}]$	–	**1001**	0.145	–	0.1412	0.0888	–	0.6426	0.0695
	AI_{min}	–	1030	7.829	–	**0.8700**	0.0167	–	0.9419	0.0170
	AI_{av}	–	1027	7.934	–	**0.8367**	0.0412	–	**0.9635**	0.0086
J7-per20-1	$L[C_{max}, AI_{min}]$	1032	**1031**	1.561	0.8337	0.3143	0.1404	0.9531	0.7730	0.0560
	$L[C_{max}, AI_{av}]$	1032	**1031**	1.603	0.8337	0.2204	0.1767	0.9531	0.7960	0.0468
	$E[C_{max}]$	–	**1028**	2.329	–	0.0884	0.1144	–	0.7312	0.0435
	AI_{min}	–	1082	9.013	–	**0.8781**	0.0210	–	**0.9550**	0.0147
	AI_{av}	–	1082	10.664	–	**0.8534**	0.0400	–	**0.9698**	0.0069
J7-per20-2	$L[C_{max}, AI_{min}]$	1027	**1024**	2.246	0.8658	0.4303	0.2055	0.9617	0.8225	0.0562
	$L[C_{max}, AI_{av}]$	1027	**1024**	2.974	0.8658	0.3934	0.2045	0.9617	0.8333	0.0517
	$E[C_{max}]$	–	**1021**	2.742	–	0.2688	0.2400	–	0.7972	0.0606
	AI_{min}	–	1074	15.165	–	**0.9158**	0.0211	–	**0.9665**	0.0121
	AI_{av}	–	1076	13.906	–	**0.8979**	0.0412	–	**0.9771**	0.0082

Table 4: Comparison of results for $E[C_{max}]$ highest priority on instances of size 8×8

	Objective	$E[C_{max}]$			AI_{min}			AI_{av}		
		Targ.	Avg	S.Dev	Target	Avg	S.Dev	Target	Avg	S.Dev
J8-per10-0	$L[C_{max}, AI_{min}]$	1055	**1052**	2.587	0.9026	0.8292	0.0889	0.9756	0.9515	0.0261
	$L[C_{max}, AI_{av}]$	1055	**1052**	2.676	0.9026	0.8017	0.0999	0.9756	0.9545	0.0225
	$E[C_{max}]$	–	**1050**	3.105	–	0.7504	0.1156	–	0.9399	0.0271
	AI_{min}	–	1073	10.333	–	**0.9598**	0.0273	–	**0.9874**	0.0100
	AI_{av}	–	1072	9.972	–	**0.9453**	0.0384	–	**0.9898**	0.0063
J8-per10-1	$L[C_{max}, AI_{min}]$	1036	**1032**	3.391	0.8653	0.7242	0.0987	0.9664	0.9062	0.0360
	$L[C_{max}, AI_{av}]$	1036	**1033**	2.904	0.8653	0.6721	0.1333	0.9664	0.9141	0.0355
	$E[C_{max}]$	–	**1030**	3.744	–	0.6087	0.1316	–	0.8858	0.0375
	AI_{min}	–	1064	11.386	–	**0.9386**	0.0342	–	**0.9801**	0.0151
	AI_{av}	–	1063	12.490	–	**0.9342**	0.0485	–	**0.9881**	0.0088
J8-per10-2	$L[C_{max}, AI_{min}]$	1041	**1036**	5.139	0.8656	0.7958	0.1082	0.9658	0.9357	0.0364
	$L[C_{max}, AI_{av}]$	1041	**1037**	4.521	0.8656	0.7533	0.1275	0.9658	0.9436	0.0330
	$E[C_{max}]$	–	**1033**	5.225	–	0.6735	0.1283	–	0.9108	0.0371
	AI_{min}	–	1065	13.246	–	**0.9330**	0.0339	–	**0.9778**	0.0149
	AI_{av}	–	1062	13.724	–	**0.9279**	0.0472	–	**0.9862**	0.0088
J8-per20-0	$L[C_{max}, AI_{min}]$	1022	**1020**	2.339	0.8644	0.1645	0.1259	0.9668	0.7168	0.0712
	$L[C_{max}, AI_{av}]$	1022	**1020**	2.001	0.8644	0.0771	0.1324	0.9668	0.7728	0.0550
	$E[C_{max}]$	–	**1015**	2.255	–	0.0219	0.0515	–	0.6685	0.0604
	AI_{min}	–	1074	11.741	–	**0.9394**	0.0322	–	**0.9814**	0.0123
	AI_{av}	–	1072	11.650	–	**0.9250**	0.0481	–	**0.9870**	0.0083
J8-per20-1	$L[C_{max}, AI_{min}]$	1003	**1002**	0.800	0.7574	0.2573	0.1703	0.9225	0.7709	0.0716
	$L[C_{max}, AI_{av}]$	1003	**1002**	0.793	0.7574	0.1541	0.1846	0.9225	0.7914	0.0630
	$E[C_{max}]$	–	**1001**	0.862	–	0.0411	0.0790	–	0.6946	0.0589
	AI_{min}	–	1023	9.613	–	**0.8358**	0.0417	–	**0.9288**	0.0255
	AI_{av}	–	1025	11.223	–	**0.7799**	0.0920	–	**0.9513**	0.0147
J8-per20-2	$L[C_{max}, AI_{min}]$	1018	**1017**	2.292	0.8246	0.3463	0.2001	0.9502	0.7961	0.0700
	$L[C_{max}, AI_{av}]$	1018	**1017**	2.191	0.8246	0.2714	0.2040	0.9502	0.8239	0.0567
	$E[C_{max}]$	–	**1014**	2.538	–	0.1269	0.1498	–	0.7593	0.0561
	AI_{min}	–	1065	15.408	–	**0.9026**	0.0347	–	**0.9628**	0.0185
	AI_{av}	–	1062	15.974	–	**0.8879**	0.0611	–	**0.9758**	0.0127

Table 5: Comparison between PSO and MA in terms of $E[C_{max}]$

Problem	Target	MA	PSO	
		$E[C_{max}]$	$L(C_{max}, AI_{min})$	$L(C_{max}, AI_{av})$
J7-per10-0	1035	1066	1032	1032
J7-per10-1	1019	1052	1017	1017
J7-per10-2	1038	1067	1033	1034
J7-per20-0	1001	1004	1001	1001
J7-per20-1	1032	1044	1031	1031
J7-per20-2	1027	1042	1024	1024
J8-per10-0	1055	1083	1052	1052
J8-per10-1	1036	1066	1032	1033
J8-per10-2	1041	1071	1036	1037
J8-per20-0	1022	1037	1020	1020
J8-per20-1	1003	1014	1002	1002
J8-per20-2	1018	1035	1017	1017