Surrogate-Assisted Multiobjective Evolutionary Algorithm for Fuzzy Job Shop Problems

Juan José Palacios, Jorge Puente, Camino R. Vela, Inés González-Rodríguez, and El-Ghazali Talbi

Abstract We consider a job shop scheduling problem with uncertain processing times modelled as triangular fuzzy numbers and propose a multiobjective surrogate-assisted evolutionary algorithm to optimise not only the schedule's fuzzy makespan but also the robustness of schedules with respect to different perturbations in the durations. The surrogate model is defined to avoid evaluating the robustness measure for some individuals and estimate it instead based on the robustness values of neighbouring individuals, where neighbour proximity is evaluated based on the similarity of fuzzy makespan values. The experimental results show that by using fitness estimation, it is possible to reach good fitness levels much faster than if all individuals are evaluated.

1 Introduction

Scheduling problems form an important body of research since the late fifties with multiple applications in industry, finances, welfare, etc. Traditionally, scheduling has been treated as a deterministic problem that assumes precise knowledge of all

Juan José Palacios
European Center for Soft-Computing, Mieres, (Spain), e-mail: juanjose.palonso@gmail.com

Jorge Puente, Camino R. Vela
Dept. of Computing, University of Oviedo, (Spain), e-mail: \{puente,crvela\}@uniovi.es

Inés González-Rodríguez
Dept. of Mathematics, Statistics and Computing, University of Cantabria, (Spain), e-mail: gonzalezri@unican.es

El-Ghazali Talbi
DOLPHIN Team, Inria Lille-Nord Europe and LIFL, Université Lille 1, (France), e-mail: el-ghazali.talbi@univ-lille1.fr

data. However, modelling real-world problems usually involves processing uncertainty and flexibility. In the literature we find different proposals for dealing with uncertainty in scheduling [14], either finding solutions which adapt dynamically to changes or incorporating available knowledge about possible changes to the solution. In particular, fuzzy sets have contributed to bridge the gap between classical techniques and real-world user needs, serving both for handling flexible constraints and uncertain data [26]. They are also emerging as an interesting tool for improving solution robustness, a much-desired property in real-life applications [18].

When the improvement in robustness must not be obtained at the cost of loosing performance quality in the solutions, we face a bi-objective scheduling problem. In general there is a growing interest in multiobjective optimisation for scheduling and, given its complexity, in the use of metaheuristic techniques to solve these problems, as shown in [7] among others. Specifically, the multiobjective fuzzy job shop problem is receiving an increasing attention, mostly to optimise objective functions related to makespan and due-date satisfaction. Existing proposals include genetic algorithms [12], differential evolution algorithms [15], or hybrid strategies like the genetic simulated-annealing algorithm from [25]. Interestingly, the latter contemplates finding both robust and satisfactory schedules, although the robustness optimisation criterion is based on the worst-case approach which can be too conservative in cases where the worst case is not that critical and instead an overall acceptable performance might be more adequate.

Roughly speaking, a schedule is said to be *robust* if it minimises the effect of executional uncertainties on its primary performance measure [1], the makespan in our case, so the robustness of one schedule can be only measured after executing it in a real environment. In absence of real execution, we can use Monte-Carlo simulations to approximate the robustness value of every schedule, but even with this approximation the number of simulations required to compute this value translate into an excessive computational cost for a fitness function in a evolutionary algorithm. This suggests resorting to surrogate-assisted evolutionary computation, which was mainly motivated to reduce computational time in problems where complex simulations are involved [16].

The idea of approximating fitness values of some individuals based on information generated during the run, i.e. based on fitness values of individuals generated previously, has lately gained increasing attention [16]. In the simplest case, the fitness of a new individual is derived from its parents' fitnesses. Other approaches attempt to construct a more global model of the fitness landscape based on previous evaluations. Several such approaches can be found in the literature, mainly differing in the model that is used to approximate the landscape and the selection of data points used to construct the model [5]. The idea is to keep previous evaluations in a history and select the closest neighbours to build a specific estimation model.

In the following, we will consider the bi-objective fuzzy job shop problem with the goal of optimising both makespan and robustness. We will propose to solve it with a multiobjective evolutionary algorithm (MOEA) where the fitness related to robustness is found via surrogates, considering closest neighbours in terms of

approximation of fuzzy values, using the degree of similarity between fuzzy sets to have a proximity measure.

The rest of the paper is organised as follows. Section 2 introduces the fuzzy job shop scheduling problem and a related robustness measure. Section 3 describes the multiobjective evolutionary algorithm proposed to solve the problem, including the surrogate model for evaluating the robustness fitness function. We then present and analyse some experimental results in 4 and finish with some conclusions and future work in Section 5.

2 Job Shop Scheduling with Uncertain Durations

The *job shop scheduling problem*, also denoted *JSP*, consists in scheduling a set of jobs $\{J_1,\ldots,J_n\}$ on a set of physical resources or machines $\{M_1,\ldots,M_m\}$, subject to a set of constraints. There are *precedence constraints*, so each job J_i, $i=1,\ldots,n$, consists of m tasks $\{\theta_{i1},\ldots,\theta_{im}\}$ to be sequentially scheduled. Also, there are *capacity constraints*, whereby each task θ_{ij} requires the uninterrupted and exclusive use of one of the machines for its whole processing time. A solution to this problem is a schedule (an allocation of starting times 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 the makespan is minimal or its robustness is maximal.

2.1 Uncertain Durations

In real-life applications, it is often the case that the exact duration of a task, i.e. the time it takes to be processed, is not known in advance, and only some uncertain knowledge is available. Such knowledge can be modelled using a *triangular fuzzy number* or TFN, given by an interval $[a^1,a^3]$ of possible values and a modal 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:

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

In the job shop, we essentially need two operations on fuzzy numbers, the sum and the maximum. These are obtained by extending the corresponding operations on real numbers using the *Extension Principle*. However, computing the resulting expression is cumbersome, if not intractable. For the sake of simplicity and tractability of numerical calculations, we follow [11] and approximate the results of these operations, evaluating the operation only on the three defining points of each TFN.

It turns out that for any pair of TFNs A and B, the approximated sum $A+B \approx (a^1+b^1, a^2+b^2, a^3+b^3)$ coincides with the actual sum of TFNs; this may not be the case for the maximum $\max(A,B) \approx (\max(a^1,b^1), \max(a^2,b^2), \max(a^3,b^3))$, although they have identical support and modal value.

The membership function of a fuzzy number can be interpreted as a possibility distribution on the real numbers. This allows to define its expected value, given for a TFN A by $E[A] = \frac{1}{4}(a^1 + 2a^2 + a^3)$. It coincides, among others, with the neutral scalar substitute of a fuzzy interval and the centre of gravity of its mean value [9]. It induces a total ordering $\leq_E$ in the set of fuzzy numbers, where for any two fuzzy numbers A, B $A \leq_E B$ if and only if $E[A] \leq E[B]$.

2.2 Robust scheduling

A fuzzy schedule does not provide exact starting times for each task. Instead, it gives a fuzzy interval of possible values for each starting time, provided that tasks are executed in the order determined by the schedule. In fact, it is impossible to predict what the exact time-schedule will be, because it depends on the realisation of the task's durations, which is not known yet. This idea is the basis for a semantics for fuzzy schedules from [13] by which solutions to the fuzzy job shop should be understood as a-priori solutions, also called baseline or predictive schedules in the literature [14]. When tasks are executed according to the ordering provided by the fuzzy schedule we shall know their real duration and, hence, obtain a real (executed) schedule, the a-posteriori solution with deterministic times. Clearly, it is desirable that a fuzzy solution yields reasonably good executed schedules at the moment of its practical use, in clear relation with the concept of schedule robustness.

As already mentioned, we consider that a schedule is *robust* if it minimises the effect on the makespan of executional uncertainties. This straightforward definition may, however, be subject to many different interpretations when it comes to specifying robustness measures [22]. In this work, we will consider only uncertainties in task processing times and we shall adopt the concept of ε-robustness proposed in [3] for stochastic scheduling, already adapted to the fuzzy flexible job shop in [19]. This definition states that a predictive schedule is considered to be robust if the quality of the eventually executed schedule is close to the quality of the predictive schedule. In particular, for the fuzzy job shop, a predictive schedule with makespan value $C_{max,pred}$ (a TFN) is ε-*robust* for a given ε if the objective value $C_{max,ex}$ of the eventually executed schedule (a real value) is such that:

$$(1-\varepsilon) \leq \frac{C_{max,ex}}{E[C_{max,pred}]} \leq (1+\varepsilon) \tag{2}$$

or, equivalently,

$$\frac{|C_{max,ex} - E[C_{max,pred}]|}{E[C_{max,pred}]} \leq \varepsilon. \tag{3}$$

That is, the relative error of the estimation made by the predictive schedule (i.e. its expected makespan) is bounded by ε. Obviously, the smaller ε is, the better.

Notice however that this definition requires a real execution of the problem which may not always be available. Consider for instance the synthetic problems commonly used in the literature as benchmarks. In this case, following [19], we provide an approximation of the ε-robustness measure by means of a Monte-Carlo simulation. Given a fuzzy instance, we generate a sample of K possible realisations of that instance by assigning an exact duration to each task, that is K deterministic instances on which we can evaluate the ε-robustness of the solution. Now for each realisation $k = 1, \ldots, K$, let $C_{max,k}$ denote the exact makespan obtained by executing tasks according to the ordering provided by a predictive schedule. Then, the *average ε-robustness* of the predictive schedule, denoted $\overline{\varepsilon}$, is calculated as:

$$\overline{\varepsilon} = \frac{1}{K} \sum_{k=1}^{K} \frac{|C_{max,k} - E[C_{max}]|}{E[C_{max}]}, \tag{4}$$

where $E[C_{max}]$ is the expected makespan estimated by the predictive schedule.

A crucial factor in this method is the way in which we sample deterministic durations for the tasks based on their fuzzy values. This is done by simulating exact durations for tasks following a probability distribution that is consistent with the possibility distribution μ_A defined by each fuzzy duration A. A simple approach consists in considering the uniform probability distribution that is bounded by the support of the TFN. This possibility-probability transformation is motivated by several results from the literature (see [10, 2]) that justify the use of TFNs as fuzzy counterparts to uniform probability distributions and model-free approximations of probability distributions with bounded support.

2.3 The multiobjective approach

In scheduling, when there is uncertainty in some of the input data, solution robustness becomes an important factor to be taken into account. Indeed, an optimal solution found for an ideal deterministic scenario (for instance, assuming that all durations take their modal value) may be of little or no use when it is executed if changes in the input data affect its real performance. Therefore, our aim in this work is to optimise both a performance or quality function, the expected makespan $E[C_{max}]$, as well as the robustness of the solution with respect to that function, the approximate measure $\overline{\varepsilon}$.

To optimise these two objective functions, we shall take a dominance-based approach. In general, for a minimisation problem with f_i, $i = 1, \ldots, n$ objective functions, a solution s is said to be *dominated* by a solution s', denoted $s' \succ s$ iff for each objective function f_i, $f_i(s') \leq f_i(s)$ and there exists at least one objective function such that $f_i(s) < f_i(s)$. Our goal will then be to find non-dominated solutions to the

FJSP with respect to $E[C_{max}]$ and $\overline{\varepsilon}$. To achieve this, we propose a dominance-based multiobjective evolutionary algorithm (MOEA) [23].

3 Multiobjective evolutionary algorithm

We propose a MOEA based on the well-known NSGA-II template [8]. An initial population is randomly created and evaluated and then the algorithm iterates over a number of generations, keeping a set of non-dominated solutions. At each iteration i, a new population $\mathrm{Off}(P_i)$ is built from the current one P_i by applying the genetic operators of selection and recombination and then a replacement strategy is applied to obtain the next generation P_{i+1}. Finally, the stopping criterion can be at least one of the following: stop when no solution belonging to the set of non-dominated solutions is removed from this set after ni_{iter} iterations or after a fixed number of iterations or after a given running time.

Solutions are encoded into chromosomes using permutations with repeated elements, which are permutations of the set of tasks, each being represented by its job number [4]. A given chromosome is decoded into the associated schedule, using an insertion strategy in the schedule-generating scheme [20]. This immediately gives the makespan expected value. The other fitness function is evaluated based on the degree of similarity between the fuzzy makespan of the current solution and a set of solutions, named *cache* hereafter, for which the $\overline{\varepsilon}$ value has been previously calculated. We shall refer to this algorithm as sMOEA.

3.1 The surrogate model

Evolutionary algorithms usually need a large number of fitness evaluations before obtaining a satisfying result. Either when an explicit fitness function does not exist or when the evaluation of the fitness is computationally very expensive, it becomes necessary to estimate the fitness value by an approximate model. This is indeed the case of one of our fitness functions, related to the objective of robustness.

Fitness approximation has been addressed from different areas, as can be seen for instance in [5, 16]. Techniques to manage surrogates for fitness evaluation include evaluating the fitness function only in some of the generations or in some individuals within a generation, having a pre-selection of offspring before evaluation, evaluating only those individuals that potentially have a good fitness value or choosing for re-evaluation representative individuals by clustering the population, to mention but a few.

Here we propose a new double approximation by using data sampling techniques. First, we run a Monte-Carlo simulation to provide a surrogate of the ε-robustness measure for the individuals in the cache, as explained in Section 2.2. Then we approximate the ε-robustness of a solution with that of the most similar solution in

the set for which $\overline{\varepsilon}$ values have been previously computed, provided that this similarity exceeds a given threshold. The set of pre-evaluated solutions is named cache inspired in the cache memory of the computers. For this second part of the surrogate model there are several design decision that have to be made, namely, the similarity measure and the update strategy of the cache list. The similarity threshold and the size of the cache list are to be experimentally determined.

3.1.1 The similarity measure

Every decoded solution has a fuzzy makespan value, so we propose to use a similarity measure for TFNs to compare a given solution with every solution in the cache list based on fuzzy makespan values. The rationale behind this choice is that, on one hand, $\overline{\varepsilon}$ depends on the expected value of the fuzzy makespan and, on the other hand, the makespan of every deterministic realisation used to compute $\overline{\varepsilon}$ lies in the support of the fuzzy makespan.

In the literature we can find numerous proposals to quantify the degree of similarity between two fuzzy numbers using different descriptive parameters, such as the geometric distance, the perimeter, the area or the distance between the centres of gravity, etc [6, 24]. However, most similarity functions are not adequate for our framework. In particular we need normalised similarity values in order to establish threshold values which are independent of the instance and this normalisation must be invariant to translations if they are to correspond to similarities in $\overline{\varepsilon}$ values. Additionally, similarity degrees must be easy to compute. That is, we look for simple but still representative measures.

In this paper, we consider a measure based on the so-called shared area between the fuzzy numbers. The shared area between fuzzy numbers with respect to the total area of these fuzzy numbers has been incorporated as a component of the measure of similarity of generalised fuzzy trapezoidal numbers in [24]. Here we are using TFNs, less complex than generalised fuzzy numbers, so we need only consider this value. The degree of similarity $S_{A,B}$ of A and B is then defined as

$$S_{A,B} = \frac{Area(A \cap B)}{Area(A \cup B)} \tag{5}$$

When $A \cap B$ is not a triangle, we approximate the area by the maximum triangle inscribed in this plane area. We will say that A and B are *approximately equal* given a small nonnegative number δ iff $S_{A,B} \leq \delta$.

3.1.2 The update strategy

The update strategy for the cache is motivated by the fact that, as the algorithm converges, the chance that a new solution lies in areas of the search space with bad solutions becomes smaller. A new solution is thus added to the cache only if it is

not similar enough to any of the elements already in the list, in the sense that they are not approximately equal as defined above. When this is the case, the solution is fully evaluated to obtain its $\overline{\varepsilon}$ value and added to the list. If this is full, the new solution replaces the one in the list that has not been used for longer to estimate the value of the robustness for another individual.

The cache is initially empty. Then, every individual in the initial population is processed to be inserted in the list following the replacement strategy above. Notice that by doing this the cache list, which has a prefixed size, may not be full once the whole initial population is analysed.

3.2 Genetic operators

In the selection phase all chromosomes are randomly grouped into pairs, and then each of these pairs is mated to obtain two offspring. For the mating we have implemented one of the most extended crossover operators for the *JSP*, the Generalized Order Crossover (GOX). In order to preserve the diversity of individuals inside the population and prevent the algorithm from getting stuck in local optima, the insertion mutation strategy is also introduced.

The replacement strategy is a key factor in MOEA algorithms. Here we adopt a strategy based on the non-dominated sorting approach with diversity preservation from [8], that is, solutions belonging to a lower (better) non-domination rank are preferred and, between two solutions in the same non-dominance level, we prefer the solution located in the less crowded region. To introduce greater diversity in the algorithm, we remove from the pool of individuals those which are repeated, in the sense that there exists in the pool at least another individual having identical values for all objective functions. In the case that such elimination causes the pool to contain less individuals than the population size, all the non-repeated individuals pass onto the next generation, which is later completed with the best repeated individuals according to their rank level and crowding distance.

4 Experimental study

For the experimental study, we consider one hard and well-known instance obtained by fuzzifying task durations of a crisp JSP classical benchmark, La29, as proposed in [21].

We start by trying to gain some insight into the effects of considering different similarity thresholds and cache sizes, respectively denoted δ and λ hereafter. We have generated a set of 1000 random solutions and, for different similarity thresholds δ ranging from 0.10 to 0.95, we have recorded the percentage of times (called hit rate) that a solution is approximately equal to at least one solution in a set of size λ, with λ ranging from 10 to 200. We have seen that, for similarity thresholds

under 0.80, the cache almost always contains an approximately equal solution for most of the λ values. Moreover, even when $\delta = 0.80$ and λ values over 100 are considered, we have hit rates near 100%. Translated into our surrogate algorithm this could mean that actual $\overline{\varepsilon}$ values will most likely be computed only for those solutions inserted in the cache in the first iterations. However, a different behaviour is observed for the same cache sizes ($\lambda \geq 100$) when $\delta = 0.90$ and $\delta = 0.95$ are considered. On the other hand, when these stricter similarity thresholds are considered with small λ values, there is little chance of finding an approximately equal solution in the cache, so most of the times a new solution will need to be completely evaluated, neutralising the potential effect of the surrogate approach in running times. That said, $\delta = 0.8$ offers more reasonable hit rates for small λ values, which support considering these smaller cache sizes in the experimental study.

We now proceed to consider different values for δ and λ and analyse their effects in running times and solution quality. For the sake of clarity, we will refer to the multiobjective algorithm using the surrogate model as sMOEA, while MOEA will refer to the algorithm that avoids surrogates and evaluates every new solution. As a result of a preliminary parametric analysis, the parameter setup is as follows: population size 100, crossover and mutation probabilities 1.0 and 0.1 respectively. Given the stochastic nature of the algorithm, it is run 10 times, so 10 different sets of non-dominated solutions are stored in order to obtain representative data.

In the literature we find many proposals to compare multiobjective algorithms. In this work we consider two metrics: the hypervolume, which is a quality indicator that combines both convergence and diversity measures and the unary additive $\varepsilon - Indicator$, which is a distance-based indicator [23]. This last indicator is calculated with respect to a reference set RF, which ideally should be the optimal Pareto front PO^*. However, since the benchmark instance used here has not been solved yet, this Pareto front is unknown. In consequence, we approximate it by the set of non-dominated elements of the union of all sets of solutions obtained so far in this experimentation [23]. Additionally, to avoid problems derived from the different scales of the objective functions, we normalise their values. More precisely, let $f_i^-(S) = \min\{f_i(s) : s \in S\}$ be a lower bound of the objective function f_i in the set S, and

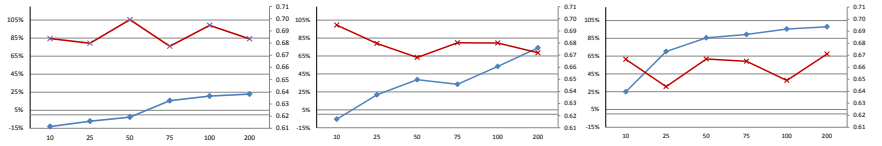
$$f_i^+(S) = \max\{f_i(s) : s \in S\} + 0.05 * (\max\{f_i(s) : s \in S\} - \min\{f_i(s) : s \in S\})$$

an upper bound thereof, then the objective value $f_i(s)$ of each solution $s \in S$ is normalised as follows:

$$\forall i \, f_i(s) = \frac{f_i(s) - f_i^-(S)}{f_i^+(S) - f_i^-(S)}. \tag{6}$$

By taking this upper bound, we prevent the solutions from having a value equal to 1, which can be troublesome when computing the hypervolume. Indeed, solutions with objective values equal to 1 define a rectangle with null area, making them unsuitable for fair comparisons.

In our experimental study, we first run the MOEA that evaluates $\overline{\varepsilon}$ for every solution, using $ni_{iter} = 25$ non-improving consecutive iterations as stopping criterion.



(a) $\delta = 0.95$ (b) $\delta = 0.90$ (c) $\delta = 0.80$

Fig. 1: Improvement and hypervolume values for different sizes of cache. Red lines represent hypervolumes and blue lines time reductions

Besides the sets of non-dominated solutions, we register the number of generations needed for MOEA to converge (3950). Then, using this number of generations as stopping criterion, sMOEA is executed considering three δ values: 0.80, 0.90 and 0.95 and six λ values: 10, 25, 50, 75, 100 and 200. The sets of non-dominated solutions obtained after every run are then fully evaluated in order to obtain their actual $\overline{\varepsilon}$ values that allow for fairer comparisons, so solutions that are dominated after the full evaluation are removed from the sets.

Figure 1 shows the improvement in running time of sMOEA w.r.t. MOEA and the hypervolume values for sMOEA given different combinations of λ and δ: Figure 1(a) for $\delta = 0.95$, Figure 1(b) for $\delta = 0.90$ and Figure 1(c) for $\delta = 0.80$. All figures show a primary Y-axis for time improvement in percentage (from -15% to 105%) and a secondary Y-axis for hypervolume values (from 0.61 to 0.71), being 0.6939 the hypervolume value reached by MOEA.

We can observe that, in general, increasing the similarity threshold improves the hypervolume and reduces the time improvement. This behavior is quite natural, as having a stricter threshold δ causes the algorithm to fully evaluate more solutions, thus having more accurate information, at the cost of having a longer runtime.

Notice as well that, even though time reductions are comparable, this is not the case for hypervolume (HV) values. When $\delta = 0.80$, the HV values obtained are in general much worse than those obtained with $\delta \geq 0.90$ (only with a cache size of 200 do they begin to be competitive). This leads us to reject 0.80 as an appropriate threshold for the similarity of solutions.

Focusing on Figures 1(a) and 1(b) we can observe that, when the similarity threshold is high and the cache size is small, the processing time not only does not decrease, it even increases. Specifically, with $\delta = 0.95$ and $\lambda \leq 50$, and with $\delta = 0.90$ and $\lambda = 10$, CPU times of sMOEA are worse than those of MOEA. This is explained by the fact that it is unlikely to find an approximately equal individual in a small-sized cache, so most of individuals of the sMOEA are fully evaluated (as is the case with MOEA) and, additionally, sMOEA has to compare solutions and update the cache frequently, incurring in higher computational cost. For values such as $\delta = 0.95$ and $\lambda = 75$ or $\delta = 0.90$ and $\lambda = 50$, for which the cache is actually used by sMOEA, we can observe time improvements at some cost in HV values. More interestingly, when the cache size increases to 100, time reduction does not imply a loss in quality. We believe this is because the most similar solution in the cache is

more likely to be really close to the current solution, having the effect of more reliability in the surrogates, leading to better HV values. This effect is nonetheless lost for $\lambda = 200$, probably because an excessively large cache causes the robustness fitness value to be estimated too often. Finally, we look at the ε-indicator for those sets of experiments where sMOEA does not consume more time resources than MOEA. We see that, only for $\lambda = 100$ and $\delta = 0.90$ or $\delta = 0.95$, the ε-indicator values obtained for sMOEA are similar to those of the MOEA (0.0712, 0.0791, and 0.0731 respectively).

Once a first filter has been applied to the different options for δ and λ values, we shall compare the two best variants of sMOEA (with $\lambda = 100$ and $\delta = 0.95$ or 0.90) and MOEA. Regarding runtime, sMOEA(0.90,100) obtains a greater reduction in CPU times (53% versus 21% of sMOEA(0.95,100)). In terms of quality, a first impression can be obtained from the box-plots in Figure 2, which correspond to the HV values obtained using the three configurations. With the exception of two outsider values, HV values corresponding to the configuration sMOEA(0.95,100) appear to be slightly better than the rest. Having said this, differences in the box-plot are not big enough for a solid conclusion, making further comparisons necessary.

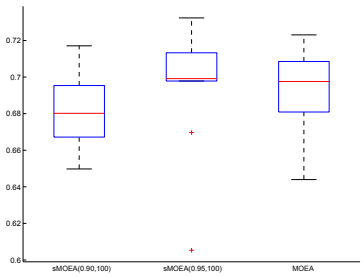


Fig. 2: *HV* for all runs of sMOEA(0.90,100), sMOEA(0.95,100) and MOEA (with no cache)

A more detailed comparison between the three algorithms, taking into account their multiobjective nature, can be done by means of the empirical attainment functions (EAFs), which characterise the output of a stochastic multiobjective optimisation algorithm [17].Figure 3 graphically plots the difference (using a gray scale) between the EAFs for algorithms MOEA (with no cache), sMOEA(0.95,100) and sMOEA(0.90,100), which allows us to identify the regions where one algorithm performs better than another. We can see that the solutions obtained with MOEA dominate those obtained by sMOEA(0.90,100) in almost every region with a low probability, slightly higher in the middle of the front, whereas solutions from sMOEA(0.90,100) almost never dominate those of MOEA with not null probability. The comparison between MOEA and sMOEA(0.95,100) reveals more equilibrium: the solutions from MOEA dominate with lower probability those of the surrogate version in some regions of the space, but they are dominated in other regions with a probability higher than 0. Finally, between both variants of the surrogate algorithm,

sMOEA(0.95,100) is clearly better than sMOEA(0.90,100), since the difference between their EAFs is positive in almost every region and it is null when we make the opposite comparison.

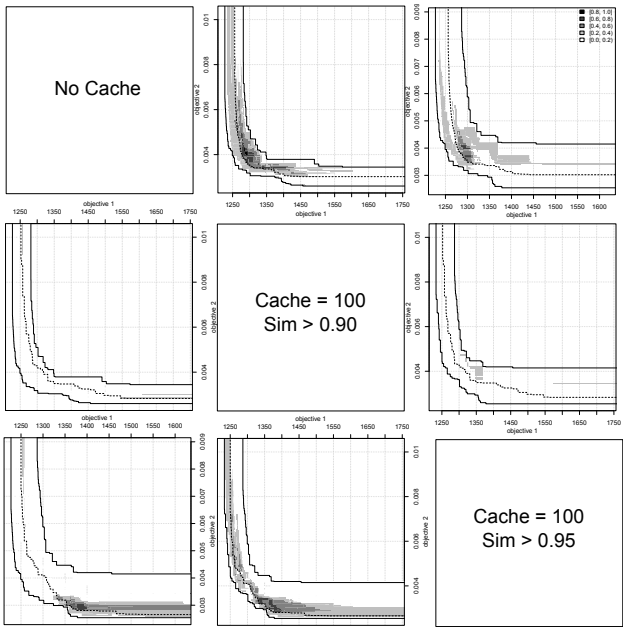


Fig. 3: Comparison between MOEA (no cache) and sMOEA(0.90/0.95,100) based on EAFs.

Finally, a statistical analysis between HVs of the sets of solutions obtained with sMOEA(0.95,100) and MOEA is carried out, with a level of significance of 0.05 in every test considered. Once the normality of the samples and the homocedasticity are verified, a t-test is done with the result that there are no significant differences in terms of solution quality (p-value 0.918709). Notice that the lack of statistically significant differences, far from being a bad result, supports the interest of our proposal. Indeed, it indicates that the HV values obtained with the proposed sMOEA and MOEA are similar and therefore means that the use of surrogates does not have a bad influence in solution quality (in terms of hypervolume values), while it provides a reduction of over 20% in running time.

5 Conclusions

We have considered the job shop problem with uncertainty in the task durations modeled as fuzzy numbers. We have proposed to simultaneously optimise the

makespan and the robustness of the solutions, understood as an overall acceptable performance under variations in the input data.

We have seen that a multiobjective evolutionary algorithm (MOEA) may produce good results for the FJSP, but at the same it is too time consuming. To overcome this drawback we have proposed a new surrogate model to spend less time in robustness evaluation. The resulting algorithm, termed sMOEA, has shown to be quite sensitive to the values of the parameters δ and λ. In particular, the combination of small δ values with large λ values gives rise to a fast method, at the cost of losing quality on solutions, while with large δ and small λ values the opposite happens. From a thorough experimental study, we have found a reasonable tradeoff between these parameters that allows sMOEA to reduce running times in more than 20% w.r.t. the original MOEA, without significant loss of solution quality in terms of hypervolumes.

Acknowledgements

This research has been supported by the Spanish Government under Grant FEDER TIN2013-46511-C2-2-P.

References

1. Aytung, H., Lawley, M.A., McKay, K., Shantha, M., Uzsoy, R.: Executing production schedules in the face of uncertainties: A review and some future directions. European Journal of Operational Research **161**, 86–110 (2005)
2. Baudrit, C., Dubois, D.: Practical representations of incomplete probabilistic knowledge. Computational Statistics & Data Analysis **51**, 86–108 (2006)
3. Bidot, J., Vidal, T., Laboire, P.: A theoretic and practical framework for scheduling in stochastic environment. Journal of Scheduling **12**, 315–344 (2009)
4. Bierwirth, C.: A generalized permutation approach to jobshop scheduling with genetic algorithms. OR Spectrum **17**, 87–92 (1995)
5. Branke, J., Schmidt, C.: Faster convergence by means of fitness estimation. Soft Computing **9**, 13–20 (2005)
6. Chen, C.T.: Extensions of the topsis for group decision-making under fuzzy environment. Fuzzy Sets and Systems **114**, 1–9 (2000)
7. Dabia, S., Talbi, E.G., van Woensel, T., De Kok, T.: Approximating multi-objective scheduling problems. Computers & Operations Research **40**, 1165–1175 (2013)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
9. Dubois, D., Fargier, H., Fortemps, P.: Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. European Journal of Operational Research **147**, 231–252 (2003)
10. Dubois, D., Foulloy, L., Mauris, G., Prade, H.: Probability-possibility transformations, triangular fuzzy sets and probabilistic inequalities. Reliable Computing **10**, 273–297 (2004)
11. Fortemps, P.: Jobshop scheduling with imprecise durations: a fuzzy approach. IEEE Transactions of Fuzzy Systems **7**, 557–569 (1997)

12. Ghrayeb, O.A.: A bi-criteria optimization: minimizing the integral value and spread of the fuzzy makespan of job shop scheduling problems. Applied Soft Computing **2**(3), 197–210 (2003)
13. González Rodríguez, I., Puente, J., Vela, C.R., Varela, R.: Semantics of schedules for the fuzzy job shop problem. IEEE Transactions on Systems, Man and Cybernetics, Part A **38**(3), 655–666 (2008)
14. Herroelen, W., Leus, R.: Project scheduling under uncertainty: Survey and research potentials. European Journal of Operational Research **165**, 289–306 (2005)
15. Hu, Y., Yin, M., Li, X.: A novel objective function for job-shop scheduling problem with fuzzy processing time and fuzzy due date using differential evolution algorithm. International Journal of Advanced Manufactoring Technology **56**, 1125–1138 (2011)
16. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation **1(2)**, 61–70 (2011)
17. Lopez-Ibañez, M., Paquete, L., Stützle, T.: Exploratory analysis of stochastic local search algorithms in biobjective optimization. In: Experimental Methods for the Analysis of Optimization Algorithms, chap. 9, pp. 209–222. Springer (2010)
18. Palacios, J.J., González-Rodríguez, I., Vela, C.R., Puente, J.: Robust swarm optimisation for fuzzy open shop scheduling. Natural Computing **13**(2), 145–156 (2014)
19. Palacios, J.J., González-Rodríguez, I., Vela, C.R., Puente, J.: Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop. Fuzzy Sets and Systems **278**, 81–97 (2015)
20. Palacios, J.J., Vela, C.R., González-Rodríguez, I., Puente, J.: Schedule generation schemes for job shop problems with fuzziness. In: T. Schaub, G. Friedrich, B. O'Sullivan (eds.) Proceedings of ECAI 2014, *Frontiers in Artificial Intelligence and Applications*, vol. 263, pp. 687–692. IOS Press (2014). DOI 10.3233/978-1-61499-419-0-687
21. Puente, J., Vela, C.R., González-Rodríguez, I.: Fast local search for fuzzy job shop scheduling. In: Proceedings of ECAI 2010, pp. 739–744. IOS Press (2010)
22. Roy, B.: Robustness in operational research and decision aiding: A multi-faceted issue. European Journal of Operational Research **200**, 629–638 (2010)
23. Talbi, E.G.: Metaheuristics. From Design to Implementation. Wiley (2009)
24. Vicente, E., Mateos, A., Jiménez, A.: A new similarity function for generalized trapezoidal fuzzy numbers. In: Artificial Intelligence and Soft Computing, pp. 400–411. Springer (2013)
25. Wang, B., Li, Q., Yang, X., Wang, X.: Robust and satisfactory job shop scheduling under fuzzy processing times and flexible due dates. In: Proc. of the 2010 IEEE International Conference on Automation and Logistics, pp. 575–580 (2010)
26. Wong, B.K., Lai, V.S.: A survey of the application of fuzzy set theory in production and operations management: 1998–2009. International Journal of Production Economics **129**, 157–168 (2011)