Hybrid Tabu Search for Fuzzy Job Shop

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

[1] A.I. Centre and Department of Computer Science,
University of Oviedo, (Spain) {palaciosjuan,puente,crvela}@uniovi.es,
http://di002.edv.uniovi.es/iscop
[2] Department of Mathematics, Statistics and Computing,
University of Cantabria, (Spain) ines.gonzalez@unican.es

Abstract. We consider the fuzzy job shop scheduling problem, which is a variant of the well-known job shop problem, with uncertainty in task durations that we model using fuzzy numbers. We propose a tabu search algorithm for minimising the expected makespan based on reversing arcs within critical blocks. We test the algorithm and then combine it with a genetic algorithm from the literature so we can observe the synergy effect, obtaining better results with the hybrid algorithm than with its components by separate. Finally we compare our hybrid algorithm with a memetic algorithm from the literature and show that even in similar times, our method is better in terms of expected makespan.

1 Introduction

Scheduling problems have formed an important body of research during the last decades as they are present in multiple applications in industry, finance and science [15]. Part of that research is focused on dealing with the uncertainty and vagueness pervading real-world situations [9]. Among the different ways of representing the uncertainty, fuzzy sets have emerged as a very interesting tool and have been extensively used in different manners, ranging from representing incomplete or vague states of information to using fuzzy priority rules with linguistic qualifiers or preference modelling [4],[18].

In deterministic scheduling the complexity of problems such as shop problems means that practical approaches to solving them usually involve heuristic strategies: simulated annealing, genetic algorithms, local search, etc. [2]. Some attempts have been made to extend these heuristic methods to the case where uncertain durations are modelled via fuzzy intervals, among others: a genetic algorithm is hybridised with a local search procedure in [10] for the flow shop problem, and in [13] a particle swarm is proposed to solve the open shop problem. For the job shop with different optimisation criteria, we find a neural approach [19], genetic algorithms [17],[14], simulated annealing [5], genetic algorithms hybridised with local search [6] or particle swarm optimisation [12].

In this paper, we intend to advance in the study of local search methods to solve the fuzzy job shop problem with expected makespan minimisation, denoted

$J|fuzz\, p_i|E[C_{max}]$ according to the three field notation. We shall propose a new local search algorithm and see how it can be combined with a genetic algorithm to improve the quality of the best solutions found so far.

2 The Fuzzy Job Shop Scheduling Problem

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

2.1 Uncertain Durations

In real-life applications, it is often the case that the exact time it takes to process a task 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 $[n^1, n^3]$ of possible values and a modal value n^2 in it. For a TFN N, denoted $N = (n^1, n^2, n^3)$, the membership function takes the following triangular shape:

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

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 [5] 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 M and N, the approximated sum $M + N \approx (m^1 + n^1, m^2 + n^2, m^3 + n^3)$ coincides with the actual sum of TFNs; this may not be the case for the maximum $\max(M, N) \approx (\max(m^1, n^1), \max(m^2, n^2), \max(m^3, n^3))$, 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 [11], given for a TFN N by $E[N] = \frac{1}{4}(n^1 + 2n^2 + n^3)$. It coincides with the neutral scalar substitute of a fuzzy interval and the centre of gravity of its mean value [4]. It induces a total ordering $\leq_E$ in the set of fuzzy numbers [5], where for any two fuzzy numbers M, N $M \leq_E N$ if and only if $E[M] \leq E[N]$.

2.2 Fuzzy Job Shop Scheduling

A job shop problem instance may be represented by a directed graph $G = (V, A \cup D)$. V contains one node $x = m(i-1)+j$ per task θ_{ij}, $1 \leq i \leq n$, $1 \leq j \leq m$, plus two additional nodes 0 (or *start*) and $nm+1$ (or *end*), representing dummy tasks with null processing times. Arcs in A, called *conjunctive arcs*, represent precedence constraints (including arcs from node *start* to the first task of each job and arcs form the last task of each job to node *end*). Arcs in D, called *disjunctive arcs*, represent capacity constraints; $D = \cup_{j=1}^{m} D_j$, where D_j corresponds to machine M_j and includes two arcs (x, y) and (y, x) for each pair x, y of tasks requiring that machine. Each arc (x, y) is weighted with the processing time p_x of the task at the source node (a TFN in our case). A feasible task processing order σ is represented by a *solution graph*, an acyclic subgraph of G, $G(\sigma) = (V, A \cup R(\sigma))$, where $R(\sigma) = \cup_{j=1}^{m} R_j(\sigma)$, $R_j(\sigma)$ being a hamiltonian selection of D_j. Using forward propagation in $G(\sigma)$, it is possible to obtain the starting and completion times for all tasks and, therefore, the schedule and the makespan $C_{max}(\sigma)$.

The schedule will be fuzzy in the sense that the starting and completion times of all tasks and the makespan are TFNs, interpreted as possibility distributions on the values that the times may take. However, the task processing ordering σ that determines the schedule is crisp; there is no uncertainty regarding the order in which tasks are to be processed.

Given that the makespan is a TFN and neither the maximum nor its approximation define a total ordering in the set of TFNs, it is necessary to reformulate what is understood by "minimising the makespan". In a similar approach to stochastic scheduling, it is possible to use the concept of expected value for a fuzzy quantity and the total ordering it provides, so the *objective* is to minimise the expected makespan $E[C_{max}(\sigma)]$, a crisp objective function.

Another concept that needs some reformulation in the fuzzy case is that of criticality, an issue far from being trivial. In [5], an arc (x, y) in the solution graph is taken to be critical if and only if the completion time of x and the starting time of y coincide in any of their components. In [8], it is argued that this definition yields some counterintuitive examples and a more restrictive notion is proposed. From the solution graph $G(\sigma)$, three *parallel solution graphs* $G^i(\sigma)$, $i = 1, 2, 3$, are derived with identical structure to $G(\sigma)$, but where the cost of arc $(x, y) \in A \cup R(\sigma)$ in $G^i(\sigma)$ is p_x^i, the i-th component of p_x. Each parallel solution graph $G^i(\sigma)$ is a disjunctive graph with crisp arc weights, so in each of them a critical path is the longest path from node *start* to node *end*. For the fuzzy solution graph $G(\sigma)$, a path will be considered to be *critical* if and only if it is critical in some $G^i(\sigma)$. Nodes and arcs in a critical path are termed critical and a critical path is naturally decomposed into critical blocks, these being maximal subsequences of tasks requiring the same machine.

In order to simplify expressions, we define the following notation for a feasible schedule. For a solution graph $G(\sigma)$ and a task x, let $P\nu_x$ and $S\nu_x$ denote the predecessor and successor nodes of x on the machine sequence (in $R(\sigma)$) and let PJ_x and SJ_x denote the predecessor and successor nodes of x on the job

Generate an initial solution S
$S^* \leftarrow S$
$tabuList \leftarrow \emptyset$
while $\neg StoppingCriterion$ **do**
  $\Omega \leftarrow Neighbourhood(S)$
  $\Omega \leftarrow \Omega - \{\omega_i \in \Omega \mid \omega_i \in tabuList \wedge \neg aspiration(\omega_i)\}$
  $S = ChooseNeighbour(\Omega)$
  if $Cmax(S) <_E Cmax(S^*)$ **then**
    $S^* \leftarrow S$
  Update $tabuList$
return S^*;

Alg. 1: General schema for tabu search

sequence (in A). The *head* of task x is the starting time of x, a TFN given by $r_x = \max\{r_{PJ_x} + p_{PJ_x}, r_{P\nu_x} + p_{P\nu_x}\}$, and the *tail* of task x is the time lag between the moment when x is finished until the completion time of all tasks, a TFN given by $q_x = \max\{q_{SJ_x} + p_{SJ_x}, q_{S\nu_x} + p_{S\nu_x}\}$.

3 Tabu Search for FJSP

Roughly speaking, a typical local search schema starts from a given solution, calculates its neighbourhood and then chooses a promising neighbour, which is usually the neighbour with the best value for the objective function. The chosen neighbour replaces the current solution and the process repeats until a stopping criterion is met. The algorithm finally returns the best solution found so far which in our case means that one with the smallest $E[C_{max}]$. In case we have two solutions A and B with the same $E[C_{max}]$ we use a ranking from [1] which chooses the solution with the minor modal value and, if the tie persists, then chooses the solution with the minimum support width.

The simplest local search schema is Hill Climbing, which moves from a solution to a neighbour only if the latter provides an improvement. This approach is fast but it gets easily stuck in local optima. To prevent this, tabu search allows a solution S to move to a non-improving neighbour. This usually makes the algorithm find better solutions at the cost of more evaluations, and in consequence, longer runtime. In our tabu search, starting from an initial solution S, the algorithm moves towards the neighbour with the best $E[C_{max}]$ value. However, in the case that all neighbours of S are worse than it, we may choose a neighbour S' and find at the next step that the best neighbour of S' is S again so we get trapped in a loop. To avoid this, tabu search uses a *tabu list* that stores forbidden solutions or forbidden movements, which are called *tabu*. In addition to a tabu status, a so-called *aspiration criterion* is associated with each move, so if a movement satisfies the associated aspiration criterion, it is considered an admissible move even if it is in the tabu list. Algorithm 1 shows the general schema for the tabu search.

3.1 Neighbourhood

During the last years, many neighbourhoods have been used for solving the job shop problem with deterministic task durations. In particular, in [20], a neighbourhood structure is introduced based on reversing all the critical arcs in the disjunctive graph $G(\sigma)$. This is extended to the fuzzy framework in [5], where an arc (x, y) is taken to be critical in $G(\sigma)$ if exists $i = 1, 2, 3$ such that $r_x^i + p_x^i = q_y^i$. A second extension to the fuzzy case was proposed in [8], using the definition of criticality based on parallel solution graphs instead. As a consequence of the criticality definitions, the new neighbourhood is a proper subset of the previous one while still containing all the improving solutions. Additionally, all neighbours in this structure are feasible and the connectivity property holds: starting from any solution, it is possible to reach a given global optimum in a finite number of steps using this structure. In [6] a new neighbourhood is proposed, based on reversing only those critical arcs at the extreme of critical blocks of a single path. Although the connectivity property does not hold any more, it contains only feasible solutions and it proves to be a very efficient structure.

More recently, in [16] the authors propose a new neighbourhood for the fuzzy job shop inspired in the work from [3] for the deterministic problem based on also reversing adjacent arcs to (x, y) (machine predecessor $P\nu_x$ and successor $S\nu_y$) as explained in Definition 1.

Definition 1. *Let σ be a task processing order and let $v = (x, y)$ be an arc at the extreme of a critical block in the associated graph $G(\sigma)$. Then, the* neighbourhood structure $\mathcal{N}_3^R(\sigma)$ *is obtained as follows: if (x, y) is the only arc in the critical block, then (x, y) is reversed; if $P\nu_x$ is also critical (and $S\nu_y$ is not), then we consider all possible permutations of $(P\nu_x, x, y)$ where (x, y) is reversed; else, if $S\nu_y$ is critical, then we consider all possible permutations of $(x, y, S\nu_y)$ where (x, y) is reversed.*

In the proposed tabu search, only the best neighbour is of interest. Thus, a makespan lower bound may help find the best neighbour in less time discarding those which are far from being the best of the neighbourhood. Therefore we use the method proposed in [16] which calculates a lower bound for the fuzzy makespan of a $\mathcal{N}_3^R$ neighbour as the longest path that would pass throw the affected nodes if we performed the move. This method also allows to easily discard moves that yield to non-feasible solutions (notice that the neighbourhood itself could generate non-feasible solutions). Details on how neighbours are chosen using lower bounds are given in Algorithm 2.

3.2 The Tabu List

In its conception, a tabu list is a set of forbidden solutions so the local search does not explore them any more. However, storing complete solutions and testing if a neighbour belongs to the list is too inefficient in terms of computational time. Usually, a tabu list stores the opposite of any move applied during the search to transform a solution into a new one, e.g. the case that we reverse the arc (x, y),

Method $ChooseNeighbour(\Omega)$
$\omega^* \leftarrow emptySolution$
Compute the lower bound $estim(\omega_i)$ for each neighbour $\omega_i \in \Omega$
while $\Omega \neq \emptyset$ **do**
  $\omega_c \leftarrow \arg\min_{\omega_i \in \Omega}\{estim(\omega_i)\}$
  $\Omega \leftarrow \Omega - \{\omega_c\}$
  Evaluate ω_c updating heads and tails [6]
  if ω^* is empty or $C_{max}(\omega_c) <_E C_{max}(\omega^*)$ **then**
    $\omega^* \leftarrow \omega_c$
    $\Omega \leftarrow \Omega - \{\omega_i \mid estim(\omega_i) > E[C_{max}(\omega^*)]\}$
return ω^*;

Alg. 2: Neighbour choice

we forbid the reversal of arc (y, x) by including it in the tabu list. This simple case cannot be trivially extended to the neighbourhood used herein, where a critical arc (x, y) can generate up to 5 different neighbours.

If we move to a neighbour generated from the critical arc (x, y), we propose to store the relative order before the change of the involved tasks and forbid any movement that yields to a solution with that relative order. For example, if we are in the state $(P\nu_x, x, y)$ and decide to move to $(y, P\nu_x, x)$, we store the first tuple and forbid any movement that generates any solution in which these 3 tasks are sorted as $(P\nu_x, x, y)$. We also introduce a parameter *tabuSize* that determines the maximum size of the list. The behaviour of the list is FIFO, that is, if we need to introduce a new forbidden order in the list and the *tabuSize* has been reached, we remove the oldest one.

A tabu move is allowed provided that it fulfills an aspiration criterion, in our case, that its expected makespan improves the best solution found so far, provided that there is no non-tabu neighbour with same or better quality.

3.3 Stopping Criterion

Unlike other methods like hill climbing, tabu search algorithms do not have a well-established stopping criterion. In the algorithm we propose, the tabu search runs for a number of iterations and finishes if the most recent improvement of the best solution has not occurred in the last *maxIter* iterations.

4 Experimental Results

The purpose of this section is to provide an experimental evaluation of the proposed algorithm in combination with a genetic algorithm. To this end, as in [6] we shall use a set of instances generated by fuzzifying 12 benchmark problems for job shop: the well-known FT10 (size 10×10) and FT20 (20×5), and La21, La24, La25 (15×10), La27, La29 (20×10), La38, La40 (15×15), and ABZ7, ABZ8, ABZ9 (20×15), a set of 10 problems considered to be hard to solve for classical

job shop. We use a fuzzy instance of each benchmark, generated following [5], so task durations become symmetric TFNs where the modal value is the original duration, thus ensuring that the optimal solution to the crisp problem provides a lower bound (LB) for the fuzzified version. This allows to measure the quality of solutions as a relative error (RE) with respect to that lower bound.

$$RE = \frac{\mid E[C_{max}] - LB \mid}{LB} \tag{2}$$

All the experiments reported in this section, correspond to a C++ implementation running on a PC with Xeon processor at 2,2Ghz and 24 Gb RAM running Linux (SL 6.0.1).

Local search algorithms are very sensitive to the starting solution. It is a common practice to run the algorithm several times using different starting solutions (multi-start LS) that could be generated randomly or using a heuristic to obtain good starting points. Here, the multi-start feature is achieved by combining the tabu search (TS) with a genetic algorithm (GA); the resulting method is denoted *HTS*. This kind of hybridisation generally improves the quality of those methods run independently as the GA keeps the quality of the solutions and the diversity, while the TS provides a deeper exploitation of the solutions. We apply the TS to every individual in the population right after its evaluation, resulting in a so-called *memetic algorithm*. In the GA, individuals are permutations with repetitions which are evaluated using a fuzzyfied G&T algorithm. To obtain the best possible performance, a parametric analysis (not reported here due to lack of space) was conducted. The resulting parameter values were: Population=100, JOX crossover with probability 0.9, selection with random pairs (all the individuals are paired), 4:2 tournament between parents and their offspring for the replacement, tabu list size = 8 and $maxIter = 10$ as stopping criterion for the search.

To estimate the number of generations needed by HTS to converge we run it 10 times on the fuzzy benchmark and record average objective values for the best individual in the population in each run and the average population quality. We conclude the algorithm needs 100 generations to converge. The parameter $maxIter$ for the TS has a special relevance for the algorithm behaviour as it defines the length of the TS and as a consequence the level of exploration around the initial solution. Figure 1 illustrates the algorithm's behaviour with varying values of $maxIter$. We can appreciate that the algorithm is scalable as it can reduce the relative error when given more time. Of course the longer the runtime is, the lower the error reduction is. However, choosing a value for the parameter is not easy, since it depends strongly on the available runtime. Being aware of the sensitivity of this parameter, for this work we set it to 10 so we can compare HTS with other another method from the literature [16] in equal runtime conditions.

We run HTS on the benchmark 30 times and store the best and average solution values as well as average runtime across the 30 runs. In order to asses whether the combination of the TS with a GA adds any value to the TS alone, we have implemented a GRASP algorithm which generates random solutions using the same evaluation function as the GA and then applies the TS to them.

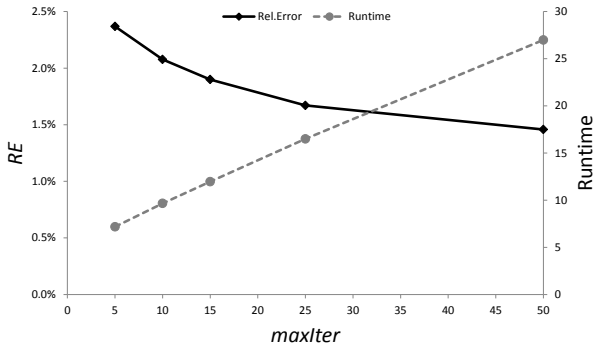

Fig. 1. Average RE depending on parameter $maxIter$

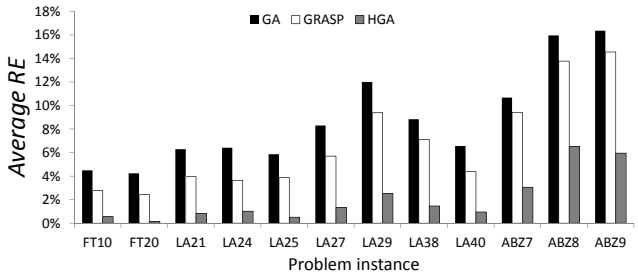

Fig. 2. Average RE obtained by GA, GRASP and HTS on every benchmark instances.

For the sake of comparison, it generates 10.000 different starting solutions (HTS evaluates 100 populations of size 100). In addition, to assess the synergy effect we also run the GA with no local search, letting it evolve with the same population size (100) for the same time HTS takes to finish. Figure 2 shows there is a clear synergy effect; in terms of RE, the improvement of HTS with respect to the GA is 81,0% in average, being the minimum improvement 59,1% for ABZ8 and the maximum 96,3% for FT20. Despite the GRASP being better in RE than the GA, HTS outperforms it in 74,8%, with the improvement ranging from 52,5% (ABZ8) to 93,5% (FT20). In addition, it is remarkable that the runtime for the GRASP is more than 3 times (370%) longer than the runtime for the GA or HTS. This demonstrates the great relevance of the starting solutions for the TS.

Finally, we compare our algorithm with MA from the literature [16] denoted $MA-3$ therein, which combines a GA analogous to the one used here with hill-climbing. Table 1 shows the best and average RE values obtained with both algorithms. We see that HTS is better than MA-3 in average for all the instances, having an average improvement of 33,9%. The highest improvement is in problem FT20 (81,2%) and the smallest is in problem LA24 (15,5%). If we perform an analysis per family, the best results are obtained on the FT problems, and the lesser improvement is for ABZ problems. We can also appreciate that the new

Table 1. Comparison with an MA from the literature

Method	Problem	*RE*		Time	Problem	*RE*		Time
		Best	Avg	(sec.)		Best	Avg	(sec.)
MA-3	FT10	0.30	1.06	2.8	LA29	1.45	3.53	8.4
HTS	(930)	0.30	**0.56**	4.2	(1130)	1.61	**2.54**	9.5
MA-3	FT20	0.04	0.84	3.8	LA38	0.61	2.53	9.0
HTS	(1165)	0.04	**0.16**	4.6	(1196)	0.61	**1.47**	10.1
MA-3	LA21	0.50	1.17	5.3	LA40	0.63	1.47	9.5
HTS	(1046)	0.33	**0.82**	6.0	(1222)	0.43	**0.96**	9.9
MA-3	LA24	0.61	1.21	5.0	ABZ7	2.21	4.00	15.9
HTS	(935)	0.56	**1.02**	5.9	(656)	2.06	**3.06**	16.7
MA-3	LA25	0.15	0.80	5.0	ABZ8	5.85	7.73	16.9
HTS	(977)	0.15	**0.51**	6.0	(645)	5.27	**6.53**	16.3
MA-3	LA27	0.30	2.01	9.0	ABZ9	4.88	7.36	17.2
HTS	(1236)	0.36	**1.35**	9.4	(661)	4.69	**5.95**	17.2

algorithm takes less than an additional second in average in comparison with MA-3 in terms of computational time.

5 Conclusions

We have tackled a variant of the job shop scheduling problem where uncertainty in durations is modelled using triangular fuzzy numbers and where the objective is to minimise the expected makespan. We have proposed a TS algorithm and combined it with a GA in order to have better initial solutions and keep diversity. The experimental results have shown that this combination outperforms both the behaviour of the TS and the GA when they are run independently. Finally we have compared the new memetic algorithm HTS with a MA from the literature and obtained better results for all tested instances using similar computational times. Based in the promising results, in the future we intend to improve on the TS, starting by adapting the most competitive search algorithms for the problem with deterministic durations to the problem with uncertainty. We also intend to create additional harder benchmarks for the fuzzy job shop which provide greater room for improvement for future metaheuristics.

Acknowledgements

This research has been supported by the Spanish Government under research grants FEDER TIN2010-20976-C02-02 and MTM2010-16051.

References

1. Bortolan, G., Degani, R.: A review of some methods for ranking fuzzy subsets. In: Dubois, D., Prade, H., Yager, R. (eds.) Readings in Fuzzy Sets for Intelligence Systems, pp. 149–158. Morgan Kaufmann, Amsterdam (NL) (1993)
2. Brucker, P., Knust, S.: Complex Scheduling. Springer (2006)
3. Dell' Amico, M., Trubian, M.: Applying tabu search to the job-shop scheduling problem. Annals of Operational Research 41, 231–252 (1993)
4. 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)
5. Fortemps, P.: Jobshop scheduling with imprecise durations: a fuzzy approach. IEEE Transactions of Fuzzy Systems 7, 557–569 (1997)
6. González Rodríguez, I., Vela, C.R., Hernández-Arauzo, A., Puente, J.: Improved local search for job shop scheduling with uncertain durations. In: Proceedings of ICAPS-2009. pp. 154–161. AAAI Press, Thesaloniki (2009)
7. González Rodríguez, I., Vela, C.R., Puente, J.: A memetic approach to fuzzy job shop based on expectation model. In: Proceedings of IEEE International Conference on Fuzzy Systems, FUZZ-IEEE2007. pp. 692–697. IEEE, London (2007)
8. González Rodríguez, I., Vela, C.R., Puente, J., Varela, R.: A new local search for the job shop problem with uncertain durations. In: Proceedings of ICAPS-2008. pp. 124–131. AAAI Press, Sidney (2008)
9. Herroelen, W., Leus, R.: Project scheduling under uncertainty: Survey and research potentials. European Journal of Operational Research 165, 289–306 (2005)
10. Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews 67(3), 392–403 (1998)
11. Liu, B., Liu, Y.K.: Expected value of fuzzy variable and fuzzy expected value models. IEEE Transactions on Fuzzy Systems 10, 445–450 (2002)
12. Niu, Q., Jiao, B., Gu, X.: Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time. Applied Mathematics and Computation 205, 148–158 (2008)
13. Palacios, J.J., González-Rodríguez, I., Vela, C.R., Puente, J.: Particle swarm optimisation for open shop problems with fuzzy durations. In: Proceedings of IWINAC 2011, Part I. LNCS 6686, pp. 362–371. Springer (2011)
14. Petrovic, S., Fayad, S., Petrovic, D., Burke, E., Kendall, G.: Fuzzy job shop scheduling with lot-sizing. Annals of Operations Research 159, 275–292 (2008)
15. Pinedo, M.L.: Scheduling. Theory, Algorithms, and Systems. Springer, third edn. (2008)
16. 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)
17. Sakawa, M., Kubota, R.: Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms. European Journal of Operational Research 120, 393–407 (2000)
18. Słowiński, R., Hapke, M. (eds.): Scheduling Under Fuzziness, Studies in Fuzziness and Soft Computing, vol. 37. Physica-Verlag (2000)
19. Tavakkoli-Moghaddam, R., Safei, N., Kah, M.: Accessing feasible space in a generalized job shop scheduling problem with the fuzzy processing times: a fuzzy-neural approach. Journal of the Operational Research Society 59, 431–442 (2008)
20. Van Laarhoven, P., Aarts, E., Lenstra, K.: Job shop scheduling by simulated annealing. Operations Research 40, 113–125 (1992)