Particle Swarm Optimisation for Open Shop Problems with Fuzzy Durations

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

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

Abstract. In this paper we confront a variation of the open shop problem where task durations are allowed to be uncertain and where uncertainty is modelled using fuzzy numbers. We propose a particle swarm optimization (PSO) approach to minimise the expected makespan using priorities to represent particle position, as well as a decoding algorithm to generate schedules in a subset of possibly active ones. Finally, the performance of the PSO is tested on several benchmark problems, modified so as to have fuzzy durations, compared with a memetic algorithm from the literature.

1 Introduction

The open shop scheduling problem, with an increasing presence in the literature, is a problem with 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 [19]. Given its NP-hardness for a number of machines $m \geq 3$ [19], practical approaches to solving it usually involve heuristic strategies: simulated annealing [1], tabu search [16], genetic algorithms with local search [7], ant colony optimization [2], etc.

To enhance the range of applications of scheduling, part of the research is devoted to model the uncertainty and vagueness pervading real-world situations. The approaches are diverse and, among these, fuzzy sets have been used in a wide variety of ways [4]. Incorporating uncertainty usually requires a significant reformulation of the problem and solving methods, in order that the problem can be precisely stated and solved efficiently and effectively. Some heuristic methods have so far been proposed for fuzzy flow and job shop problems, where uncertain durations are modelled via fuzzy intervals, e.g. [11], [15], [24], [21]. However, to the best of our knowledge, the open shop problem has received little attention in the fuzzy framework: in [14] fuzzy sets are used to represent flexible job start and due dates, in [18] a genetic algorithm is proposed to solve the open shop

with fuzzy durations, and in [10] this genetic algorithm is combined with a local search method.

In the following, we consider the fuzzy open shop problem with expected makespan minimisation, denoted $FuzO||E[C_{max}]$ and propose a bio-inspired technique to solve it. The rest of the paper is organized as follows. In Section 2 we formulate the problem and introduce the notation used across the paper. Then, in Section 3, we describe the main components of the PSO algorithm. Section 4 reports results from the experimental study. Finally, in Section 5 we summarise the main conclusions and propose ideas for future work.

2 Open Shop Scheduling with Uncertain Durations

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 consists of m tasks or operations, each requiring the exclusive use of a different machine for its whole processing time without preemption, i.e. all operations must be processed without interruption. In total, there are mn operations, $\{O_k, 1 \leq k \leq mn\}$. A solution to this problem is a *schedule* –an allocation of starting times for all operations– which is *feasible*, in the sense that all constraints hold, and is also optimal according to some criterion. Here, the objective will be minimising the makespan C_{max}, that is, the time lag from the start of the first operation until the end of the last one, a problem often denoted $O||C_{max}$ in the literature.

2.1 Uncertain Durations

In real-life applications, it is often the case that it is not known in advance the exact time it will take to process one operation and only some uncertain knowledge is available, for instance, an interval of possible durations, or a most likely duration with a certain error margin. 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 open shop, we essentially need two operations on processing times (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 is not necessarily so 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 [17], 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 intervals [5], where for any two fuzzy intervals M, N $M \leq_E N$ if and only if $E[M] \leq E[N]$.

2.2 Fuzzy Open Shop Scheduling

If processing times of operations are allowed to be imprecise and such imprecision or uncertainty is modelled using TFNs, the resulting schedule is fuzzy in the sense that starting and completion times for each operation and hence the makespan are TFNs. Each TFN can be seen as a possibility distributions on the values that the time may take. Notice however that there is no uncertainty regarding the task processing ordering given by the schedule.

An important issue with fuzzy times is to decide on the meaning of "optimal makespan". It is not trivial to optimise a fuzzy makespan, since neither the maximum nor its approximation define a total ordering in the set of TFNs. Using ideas similar to stochastic scheduling, we follow the approach taken for the fuzzy job shop in [9] and use the total ordering provided by the expected value and consider that the objective is to minimise the expected makespan $E[C_{max}]$. The resulting problem may be denoted $FuzO||E[C_{max}]$.

3 Particle Swarm Optimization for the FOSP

Given the complexity of the open shop, different metaheuristic techniques have been proposed to solve the general m-machine problem. Among these, [12] describes two heuristic methods to obtain a list of operation priorities later used in a list-scheduling algorithm; [16] proposes a tabu search algorithm; [7] introduces a genetic algorithm hybridised with local search; a genetic algorithm using heuristic seeding is given in [20], and [23] proposes a solution based on particle swarm optimisation. As mentioned in Section 1, among these only a genetic algorithm [18] and its combination with local search [10] have been applied to the case where durations are fuzzy numbers.

PSO is a population-based stochastic optimisation technique inspired by bird flocking or fish schooling [13]. In PSO, each position in the search space corresponds to a solution of the problem and particles in the swarm cooperate to find the best position (hence best solution) in the space. Particle movement is mainly affected by three factors:

- Inertia: Velocity of the particle in the latest iteration.

1. generate and evaluate the initial swarm.
2. compute gbest and pbest for each particle.
while no termination criterion is satisfied **do**
  for each particle k **do**
    for each dimension d **do**
      3. update velocity v_d^k.
      4. update position x_d^k.
    5. evaluate particle k.
    6. update pbest and gbest values.
return the schedule from the best particle evaluated so far;

Alg. 1: A generic PSO algorithm

– *pbest*: The best position found by the particle.
– *gbest*: The best position found by the swarm so far (the best *pbest*).

The potential solutions or particles fly through the problem space changing their position and velocity by following the current optimum particles *pbest* and *gbest*.

Algorithm 1 describes the structure of a generic PSO algorithm. First, the initial swarm is generated and evaluated. Then the swarm evolves until a termination criterion is satisfied and in each iteration, a new swarm is built from the previous one by changing the position and velocity of each particle towards its *pbest* and *gbest* locations.

In [23], a PSO algorithm was proposed to solve the deterministic OSP which improved the best published results. Here we shall extend this algorithm to the fuzzy framework and test it compared to the state-of-the-art methods.

3.1 Position Representation and Evaluation

Following [23], we use a priority-based representation for particle positions. Thus a schedule is encoded as a priority matrix $X^k = (x_{ij}^k)_{i=1...m,j=1...n}$, where x_{ij}^k denotes the priority of operation o_{ij}, the task of job j processed on machine i. An operation with smaller x_{ij}^k has a higher priority to be scheduled.

If we represent an OSP solution as a task processing order S, which is a permutation of tasks, we can transfer this permutation to a priority matrix and viceversa. For instance, given the following solution:

$$S = \left(o_{11}\ o_{13}\ o_{23}\ o_{12}\ o_{31}\ o_{33}\ o_{21}\ o_{23}\ o_{22} \right)$$

a particle in the space can be obtained by randomly setting x_{ij} in the interval $(p - 0.5, p + 0.5)$ where p is the location of o_{ij} in S. Therefore, the operation with smaller x_{ij} has higher priority to be scheduled. The above permutation list can be transferred to:

$$X^k = \begin{pmatrix} 1.2 & 4.0 & 1.7 \\ 6.6 & 9.4 & 2.7 \\ 5.3 & 7.9 & 6.4 \end{pmatrix}$$

while $\Omega \neq \emptyset$ **do**
  $f^* \leftarrow \min_{o_{ij} \in \Omega}\{E[f_{ij}]\}$.
  $s^* \leftarrow \min_{o_{ij} \in \Omega}\{E[s_{ij}]\}$.
  Identify the conflict set $O \leftarrow \{o_{ij} : E[s_{ij}] < s^* + \delta \times (f^* - s^*), o_{ij} \in \Omega\}$.
  Choose the operation o^*_{ij} from O with smallest x^k_{ij}.
  Schedule the operation o^*.
  $\Omega \leftarrow \Omega - \{o^*\}$.

Alg. 2: The $pFG\&T$ algorithm

Decodification of a particle may be done in different ways. For the crisp job shop and by extension for the open shop, it is common to use the G&T algorithm [6], 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 [7] a narrowing mechanism was incorporated to the G&T algorithm in order to limit machine idle times by means of a delay parameter $\delta \in [0, 1]$, thus searching over the space of so-called parameterised active schedules. In the deterministic case, for $\delta < 1$ the search space is reduced so it may no longer contain optimal schedules and, at the extreme $\delta = 0$ the search is constrained to non-delay schedules, where a resource is never idle if a requiring operation is available. This variant of G&T has been applied in [23] to the deterministic OSP, under the name "parameterized active schedule generation algorithm".

In Algorithm 2 we propose an extension of parameterised G&T to the case of fuzzy processing times, denoted $pFG\&T$. It should be noted that, due to the uncertainty in task durations, even for $\delta = 1$, we cannot guarantee that the produced schedule will indeed be active when it is actually performed (and tasks have exact durations). We may only say that the obtained fuzzy schedule is *possibly active*. Throughout the algorithm, Ω detones the set of the operations that have not been scheduled, X^k the priority matrix, s_{ij} the starting time of the operation o_{ij} and f_{ij} the completion time of the operation o_{ij}.

Notice that the $pFG\&T$ algorithm may change the task processing order given by the particle position. Therefore the PSO does not record in *gbest* and *pbest* the best positions found so far, but rather the best operation sequences of the schedules generated by the decoding operator.

3.2 Particle movement and velocity

Particle velocity is traditionally updated depending on the distance to *gbest* and *pbest*. Instead, this PSO only considers whether the position value x^k_{ij} is larger or smaller than $pbest^k_{ij}$ ($gbest_{ij}$). 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 $\{-1, 0, 1\}$. Updating is controlled by the inertia weight w at the beginning of

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 $rand \leq C_1$ **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 $C_1 < rand \leq C_1 + C_2$ **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
    $x_d^k \leftarrow x_d^k + v_d^k$.

Alg. 3: Particle movement

each iteration as follows. For each particle k and operation o_{ij} (in the following, to simplify, dimension d), if $v_d^k \neq 0$, v_d^k will be set to 0 with probability $1 - w$, meaning that if x_d^k was either increasing or decreasing, x_d^k stops at this iteration with probability $1 - w$. Otherwise, if $v_d^k = 0$, with probability C_1, v_d^k and x_d^k will be updated depending on $pbest_d^k$ and with probability C_2 they will be updated depending on $gbest_d$, always introducing an element of randomness and where C_1 and C_2 are constants between 0 and 1 such that $C_1 + C_2 \leq 1$. The details on how the updating of particle k takes place are given in Algorithm 3.

Position mutation. After a particle moves to a new position, we randomly choose an operation and then mutate its priority value x_d^k disregarding v_d^k. As in [23], for a problem of size $n \times m$, if $x_d^k < (nm/2)$, x_d^k will take a random value in $[mn - n, mn]$, and $v_d^k = 1$. Otherwise, if $x_d^k > (nm/2)$, x_d^k will take a random value in $[0, n]$ and $v_d^k = -1$.

Diversification strategy. If all particles have the same *pbest* solutions, they will be trapped into local optima. To prevent such situation, a diversification strategy is proposed in [23] that keeps the *pbest* solutions different. In this strategy, the *pbest* solution of each particle is not the best solution found by the particle itself, but one of the best N solutions found by the swarm so far, where N is the size of the swarm. Once any particle generates a new solution, the *pbest* solutions will be updated in these situations:

– if the makespan of the particle solution is equal to any *pbest* solution, replace that *pbest* solution with the new particle solution;
– if the makespan of the particle solution is less than the worst *pbest* solution and different from all *pbest* solutions, set the worst *pbest* solution equal to the particle solution.

4 Experimental Results

For the experimental study we use the test bed given in [10], where the authors follow [5] and generate a set of fuzzy problem instances from well-known benchmark problems from [3]. Given a crisp problem instance, each crisp processing time t is transformed into a symmetric fuzzy processing time $p(t)$ such that its modal value is $p^2 = t$ and p^1, p^3 are random values, symmetric w.r.t. p^2 and generated so the TFN's maximum range of fuzziness is 30% of p^2. By doing this, the optimal solution to the crisp problem provides a lower bound for the expected makespan of the fuzzified version [5]. The original problem instances consist of 6 families, denoted J3, J4,. . . , J8, of sizes 3×3, 4×4,. . . ,8×8, containing 8 or 9 instances each. From each crisp problem instance 10 fuzzy versions were generated, so in total there are 520 problem instances. The obtained benchmarks for the fuzzy open shop are available at `http://www.di.uniovi.es/tc`.

In [10], the authors propose a neighbourhood structure which combined with the GA from [18] provides a MA that not only obtains better solutions but is also more "reliable" in the sense that there is less variability in quality solution across different executions. In this experimental study we compare our PSO with this MA.

For the PSO we take the best values for each parameter obtained after a parameter analysis in [23]: swarm size $N = 60$, $C_1 = 0.7$, $C_2 = 0.1$, and inertia weight w linearly decreasing from 0.9 to 0.3. The number of iterations has been adapted for each problem size so as to obtain similar running times to the MA. Regarding the filtering mechanism of the search space given in the schedule generator, the efficiency of this reduction depends of the problem size [8], in fact our experimentation suggest taking $\delta = 1$ (no reduction) in small instances (families J3 and J4) and $\delta = 0.25$ in larger ones.

To evaluate its performance, we run the proposed PSO 30 times for each problem instance, recording the best, average expected makespan values across these 30 runs. Table 1 shows a summary of the results, with average values across 30 executions on each the 80–90 instances of the same size (detailed results for each problem would require 520 rows). It contains three columns for each method, PSO and MA, showing the Average of the Best values (AoB), the Average of the Average values (AoA), and the Average CPU Times in seconds ($Time$). We can see that both the PSO and the MA perform equally well on the small (3×3, 4×4) problems (the AoA value of PSO is slightly worse in the J4 family). As the problem size increases, also does increase the difference in solution quality between the PSO and the MA, with the former obtaining better results.

Table 1. Comparison between PSO and MA

Problem Family	PSO			MA		
	$E[C_{max}]$		$Time$	$E[C_{max}]$		$Time$
	AoB	AoA		AoB	AoA	
J3	1063.1	1063.1	0.06	1063.1	1063.1	0.13
J4	1048.8	1057.2	0.11	1048.8	1050.4	0.23
J5	1028.7	1029.6	0.70	1030.8	1044.9	1.29
J6	1033.3	1036.0	4.86	1033.9	1052.1	7.57
J7	1036.0	1041.6	14.84	1044.1	1067.5	14.46
J8	1031.6	1039.0	30.69	1045.5	1068.2	26.15

Table 2. Average relative makespan error (in %) for sets of problems of size 8×8.

Problem	PSO		MA	
	B	A	B	A
`j8-per0-1`	6.415	7.586	9.395	12.406
`j8-per0-2`	7.292	7.858	8.806	11.646
`j8-per10-0`	4.467	5.189	5.793	8.308
`j8-per10-1`	2.405	3.293	4.03	6.586
`j8-per10-2`	2.380	3.602	4.603	7.077
`j8-per20-0`	1.255	1.844	1.870	3.672
`j8-per20-1`	0.038	0.204	0.185	1.379
`j8-per20-2`	1.062	1.621	1.733	3.466

More detailed results are presented in Table 2, where each row corresponds to one set of ten fuzzy versions of one of the crisp instances of size 8×8. It shows relative makespan errors w.r.t. a lower bound for the expected makespan, which is 1000 for all problem instances. As expected, the PSO compares favourably with MA in all instances. Notice as well that the relative errors for the best (B) and average (A) solution do not differ greatly, suggesting that the PSO is quite stable. Figure 1 illustrates the reduction of the makespan error in average for each set of fuzzy problems; we can observe this is higher for mean values, which are more significant in stochastic algorithms.

5 Conclusions and Future Work

We have considered an open shop problem with uncertain durations modelled as triangular fuzzy numbers, $FuzO||E[C_{max}]$, and have proposed a particle swarm optimization technique to solve this problem. The PSO has obtained good results both in terms of relative makespan error and also in comparison to a memetic algorithm from the literature. These promising results suggest directions for future

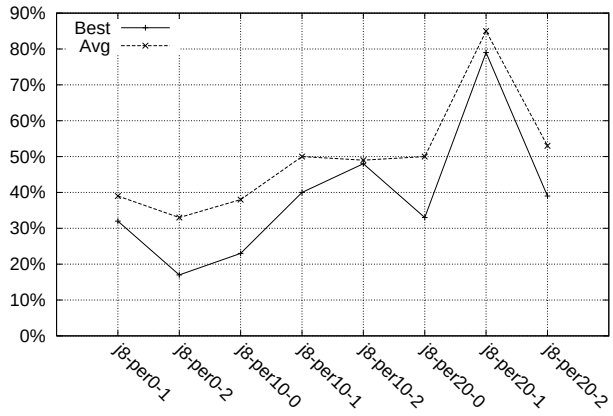

Fig. 1. Percentage of reduction in average relative error of PSO w.r.t. MA.

work. First, the PSO should be tested on more difficult problems, fuzzy versions of other benchmark problems from the literature. Also, the PSO provides a solid basis for the development of more powerful hybrid methods, in combination with local search techniques, an already successful approach in fuzzy job shop problems [22].

Acknowledgments

This work is supported by the Spanish Ministry of Science and Education under research grant MEC-FEDER TIN2010-20976-C02-02.

References

1. Andresen, M., Bräsel, H., Mörig, M., Tusch, J., Werner, F., Willenius, P.: Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop. Mathematical and Computer Modelling 48, 1279–1293 (2008)
2. Blum, C.: Beam-ACO—hybridizing ant colony optimization with beam search: an application to open shop scheduling. Computers & Operations Research 32(6), 1565–1591 (2005)
3. Brucker, P., Hunrink, J., Jurisch, B., Wöstmann, B.: A branch & bound algorithm for the open-shop problem. Discrete Applied Mathematics 76, 43–59 (1997)
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. Giffler, B., Thompson, G.L.: Algorithms for solving production scheduling problems. Operations Research 8, 487–503 (1960)

7. Gonçalves, J., Mendes, J., de M, R.M.: A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research 167, 77–95 (2005)
8. González, M.A., Sierra, M., Vela, C.R., Varela, R.: Genetic algorithms hybridized with greedy algorithms and local search over the spaces of active and semi-active schedules. Current Topics in Artificial Intelligence. 11th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2005. Revised Selected Papers. LNCS 4177 pp. 231–240 (2006)
9. González Rodríguez, I., Vela, C.R., Puente, J.: Sensitivity analysis for the job shop problem with uncertain durations and flexible due dates. IWINAC 2007, Lecture Notes in Computer Science 4527, 538–547 (2007)
10. González-Rodríguez, I., Palacios, J.J., Vela, C.R., Puente, J.: Heuristic local search for fuzzy open shop scheduling. In: Proceedings IEEE International Conference on Fuzzy Systems, FUZZ-IEEE2010. pp. 1858–1865. IEEE (2010)
11. 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)
12. Guéret, C., Prins, C.: Classical and new heuristics for the open-shop problem: A computational evaluation. European Journal of Operational Research 107, 306–314 (1998)
13. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks. pp. 1942–1948. IEEE Press, New Jersey (1995)
14. Konno, T., Ishii, H.: An open shop scheduling problem with fuzzy allowable time and fuzzy resource constraint. Fuzzy Sets and Systems 109, 141–147 (2000)
15. Lei, D.: Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems. International Journal of Advanced Manufacturing Technology 37, 157–165 (2008)
16. Liaw, C.F.: A tabu search algorithm for the open shop scheduling problem. Computers and Operations Research 26, 109–126 (1999)
17. Liu, B., Liu, Y.K.: Expected value of fuzzy variable and fuzzy expected value models. IEEE Transactions on Fuzzy Systems 10, 445–450 (2002)
18. Palacios, J.J., Puente, J., Vela, C.R., González-Rodríguez, I.: A genetic algorithm for the open shop problem with uncertain durations. In: Proceedings of IWINAC 2009, Part I. Lecture Notes in Computer Science, vol. 5601, pp. 255–264. Springer (2009)
19. Pinedo, M.L.: Scheduling. Theory, Algorithms, and Systems. Springer, third edn. (2008)
20. Puente, J., Diez, H., Varela, R., Vela, C., Hidalgo, L.: Current Topics in Artificial Intelligence (Revised Selected Papers CAEPIA 2003), vol. LNAI 3040, chap. Heuristic Rules and Genetic Algorithms for Open Shop Scheduling Problem, pp. 394–403. Springer (2003)
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. Sha, D.Y., Cheng-Yu, H.: A modified parameterized active schedule generation algorithm for the job shop scheduling problem. In: Proceedings of the 36th International Conference on Computers and Industrial Engineering, ICCIE2006. pp. 702–712 (2006)
23. Sha, D.Y., Cheng-Yu, H.: A new particle swarm optimization for the open shop scheduling problem. Computers & Operations Research 35, 3243–3261 (2008)
24. 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)