Int J Adv Manuf Technol (2014) 75:883–895
DOI 10.1007/s00170-014-6181-0

ORIGINAL ARTICLE

Particle swarm optimization-based algorithm for fuzzy parallel machine scheduling

J. Behnamian

Received: 19 February 2014 / Accepted: 18 July 2014 / Published online: 8 August 2014
© Springer-Verlag London 2014

Abstract This research extends the hybrid particle swarm optimization-based metaheuristic to solve the fuzzy parallel machine scheduling problems with bell-shaped fuzzy processing times. In this paper, we propose a discrete particle swarm optimization (DPSO) which comprises two components: a particle swarm optimization and genetic algorithm. In this paper, fuzzy arithmetic on bell-shaped fuzzy numbers is used to determine the completion time of jobs. We also use a defuzzification function to rank the fuzzy numbers. Under this ranking concept among fuzzy numbers, we plan to minimize the fuzzy makespan. An extensive numerical study on large-scale scheduling problems up to 100 jobs is conducted to assess the performance of the DPSO algorithm. The results show the proposed algorithm in comparison with lower bound to be very efficient for different structure instances.

Keywords Fuzzy parallel machine scheduling · Bell-shaped fuzzy number · Discrete particle swarm optimization · Genetic algorithm · Lower bound

1 Introduction

Scheduling problems are the allocation of machines to perform a set of jobs in a period of time. In general, scheduling requires both sequencing and machine allocation decisions. When there is only one machine, the allocation of that machine is completely determined by sequencing decisions. As a consequence, in the single-machine model, no distinction exists between sequencing and machine allocation. To appreciate that distinction, we must examine models with more than one machine. In parallel systems, jobs consist of one operation, as in the single-machine model. A parallel machine scheduling is one of the most important scheduling problems. As in the basic model, this problem consists of a stage which has several machines operating in parallel for processing. Other characteristics which are considered in this paper are as follows:

1. There are n independent jobs that are available at time 0 and are always processed without error.
2. Processing time of jobs is bell-shaped fuzzy number.
3. A machine can process only one job at a time.
4. A job can be processed by any of the machines and will be processed by a single-machine in each stage.
5. The number of jobs and machines is fixed.
6. To schedule without preemption, job processing cannot be interrupted and jobs have no associated priority values.
7. Processors are available with no breakdowns.

Since the first scheduling paper was undertaken in the mid-1950s, many variants of the basic scheduling problem have been formulated by differentiating between machine environments, constraints, and objective functions. Until the late 1990s, the majority of these papers assumed that the task-related times are crisp values. However, this assumption is deemed as insufficient for real industrial settings. The use of fuzzy set theory as a methodology for modeling and analyzing decision systems is of particular interest to researchers in scheduling due to fuzzy set theory's ability to quantitatively and qualitatively model problems which involve vagueness and imprecision. Fuzzy scheduling is a very important research topic not only because of the fuzzy nature of most real-world problems, but also because there are still many open questions in this area. The earliest paper on fuzzy scheduling appeared in 1979 [1]. Dumitru and Luban [2] investigated the application of fuzzy mathematical programming

J. Behnamian (✉)
Department of Industrial Engineering, Faculty of Engineering, Bu-Ali Sina University, Hamedan, Iran
e-mail: behnamian@basu.ac.ir

model on the problem of the production scheduling. Especially from the beginning of the 1990s fuzzy logic applications on the scheduling problems have increased.

Particle swarm optimization algorithm mimics the behavior of flying birds and their means of information exchange to solve optimization problems. It has been introduced as an optimization technique in real number spaces, but many optimization problems are set in a discrete space. Typical examples include problems that require ordering, such as scheduling. Particle swarm optimization (PSO) has been successfully applied to a variety of problems such as continuous and combinatorial optimization problems, artificial neural network training, and multi-objective optimization problems [3]. There are several difficulties in applying PSO for discrete optimization problems because Kennedy and Eberhart [4] originally developed it for continuous problems. Another shortcoming of PSO is its premature convergence. To overcome these two shortcomings, in this paper, we introduce discrete particle swarm optimization (DPSO) as a powerful generalization of particle swarm optimization (PSO). DPSO is inspired from a society whose members behave anarchically to improve their situations.

Since scheduling problems usually are NP-hard, the best exact algorithms proposed are efficient to solve small instances. To obtain an optimal solution for this type of complex problem in reasonable computational time, using traditional approaches is extremely difficult. Our goal in this paper is to combine a particle swarm optimization and genetic algorithm to solve parallel machine with fuzzy bell-shaped fuzzy processing time.

The paper has the following structure. Section 2 gives a literature review about PSO applications and fuzzy scheduling. Section 3 described the bell-shaped fuzzy numbers and their arithmetic. The characteristics of the proposed discrete hybrid metaheuristic are described in Section 4. Section 5 presents the computational results and finally, Section 6 is devoted to conclusions and future works.

2 Literature review

2.1 Applications of PSO

Kennedy and Eberhart [5] proposed the first discrete PSO algorithm which is characterized by a binary solution representation and a stochastic velocity model. Several binary solution representation models are proposed and tested in Mohan and Al-kazemi [6]. Another approach for tackling discrete optimization problems by PSO also has been proposed by Laskari et al. [7], which is based on the truncation of the real values to their nearest integer. A few relevant papers for discrete PSO are as follows: *n*-queens problem [8], polygonal approximation of digital curves [9], no-wait flow shop, vehicle routing problem [10], resource constrained project scheduling, job shop [11], transmission network expansion planning [12], data clustering [13], estimation of distribution for polygonal approximation problems [14], and traveling salesman problem [15]. Some recent applications of PSO to the optimization problems are listed in Table 1.

Table 1 Applying PSO-based algorithm to optimization problem

Authors	Algorithm descriptions	Application
Zhang et al. [16]	Adaptive PSO	Reservoir operation optimization
Shakibian and Charkari [17]	n-Cluster vector evaluated PSO	Distributed regression in WSNs
Wang et al. [18]	Scale-free Gaussian-dynamic PSO	Real power loss minimization
Zhang et al. [19]	Population statistics for PSO	Resampling methods in noisy optimization
Das et al. [20]	Neural network trained by PSO	Non-linear channel equalization
Zhang et al. [21]	Extended bare-bones PSO	Reliability redundancy allocation
Wang and Yeh [22]	Modified PSO	Aggregate production planning
Xue et al. [23]	PSO with novel initialisation and updating mechanisms	Feature selection in classification
Sadeghi et al. [24]	Improved PSO	Hybrid vendor-managed inventory and transportation
Liu et al. [25]	PSO-based simultaneous learning	Clustering and classification
Du et al. [26]	PSO	Roundness error evaluation
Elsayed et al. [27]	Self-adaptive mix of particle swarm methodologies	Constrained optimization
Wang et al. [28]	Hybrid PSO	Power plant fault diagnosis
Bagheri et al. [29]	Quantum-behaved PSO	Financial forecasting
Koulinas et al. [30]	PSO-based hyper-heuristic algorithm	Resource constrained project scheduling

PSO has also been applied to scheduling problem. Some of them are listed in Table 2.

2.2 Fuzzy parallel machine scheduling

A limited amount of the literature has been devoted to fuzzy parallel machine scheduling problems. Li et al. [31] presented a fuzzy optimization methodology for scheduling single operation parts on identical machines with possible breakdowns. A fuzzy optimization formulation is first developed. The Lagrangian relaxation technique is used to decompose the problem into part-level sub-problems and a fuzzy membership sub-problem with the authors. Hong et al. [32] demonstrated how discrete fuzzy concepts can easily be used in the longest processing time (LPT) algorithm for managing uncertain scheduling. They extended application to the continuous fuzzy domain and assumed that membership functions of the scheduled tasks are triangular. At the end, they proposed three heuristic fuzzy LPT scheduling methods to yield scheduling results with triangular final completion time membership functions. Peng and Song [33] proposed a fuzzy programming expected value model for fuzzy parallel machine scheduling problem. In that work, a fuzzy simulation-based genetic algorithm is designed to solve the proposed model. Peng and Liu [34] developed a methodology for modeling parallel machine scheduling problems with fuzzy processing times. Three types of fuzzy scheduling models are presented. In that research, a hybrid intelligent algorithm is also designed for minimizing the fuzzy maximum tardiness, the fuzzy maximum completion time, and the fuzzy maximum idleness. Anglani et al. [35] proposed an approximated model for solving the scheduling problem of parallel machines with sequence-dependent set-up costs. In this paper, a fuzzy mathematical programming model is formulated by taking into account the uncertainty in

Table 2 Applying PSO-based algorithm to scheduling problem

Authors	Algorithm descriptions	Shop environment
Tavakkoli-Moghaddam et al. [49]	Hybrid multi-objective Pareto archive PSO	Bi-objective job shop scheduling
Sha and Lin [50]	Multi-objective PSO	Job shop scheduling
Lei [51]	Pareto archive particle swarm optimization	Multi-objective job shop scheduling
Zhang et al. [52]	Hybrid PSO with tabu search	Multi-objective flexible job shop scheduling
Lin et al. [53]	Hybrid PSO with simulated annealing technique (SA) and multi-type individual enhancement scheme	Job shop scheduling
Liu et al. [54]	Hybrid PSO with SA and multi-neighborhood guided by an adaptive meta-Lamarckian learning strategy	Permutation flow shop scheduling with limited buffers
Liou and Liu [55]	PSO with novel encoding scheme	Two-machine group scheduling
Tang et al. [56]	Hybrid PSO/GA	Job shop scheduling
Tavakkoli-Moghaddam et al. [57]	Hybrid PSO with genetic operators as variable neighborhood search (VNS)	Job shop scheduling with ready times
Tu et al. [58]	PSO with improved strategy and topology	Job shop scheduling
Tavakkoli-Moghaddam et al. [59]	Pareto archive PSO with genetic operators and VNS	Job shop scheduling with sequence-dependent set-up times
AitZai et al. [60]	Discrete PSO	No-wait job shop scheduling
Zhang and Zuo [61]	Standard PSO in a hybrid cloud	Deadline constrained task scheduling
Dousthaghi et al. [62]	Hybrid PSO	Economic lot and delivery scheduling in a flexible job shop with unrelated parallel machines
Niu et al. [63]	PSO with clonal selection algorithm	Parallel machine scheduling
Sha and Lin [64]	Multi-objective PSO	Multi-objective flowshop scheduling
Chakaravarthy et al. [65]	PSO	*m*-machine flow shops with lot streaming
Jamili et al. [66]	Hybrid PSO with SA	Periodic job shop scheduling
Shiau and Huang [67]	Hybrid two-phase encoding PSO	Proportionate flexible flow shop scheduling
Marinakis and Marinaki [68]	PSO with expanding neighborhood topology	Permutation flowshop scheduling
Akhshabi et al. [69]	PSO based on memetic algorithm (MA)	No-wait flow shop
Wang and Tang [70]	PSO with a stochastic iterated local search	Permutation flowshop
Damodaran et al. [71]	PSO	Batch processing machines in a permutation flowshop

processing times to provide the optimal solution as a trade-off between total set-up cost and robustness in demand satisfaction.

Petrovic and Duenas [36] presented a new fuzzy logic-based decision support system for parallel machine scheduling/rescheduling in the presence of uncertain disruptions. They are modeled and combined using standard fuzzy sets and level 2 fuzzy sets, respectively. Two sets of Sugeno-type rules are proposed to support rescheduling decision making. One set of the fuzzy rules determines when to reschedule, while the other one determines which rescheduling method to use. Franke et al. [37] presented a methodology based on a rule system for automatically generating online scheduling strategies for a complex objective defined by a machine provider. To do this, the authors assumed that the parallel jobs are independent and there are multiple identical machines at the stage. This rule system classifies all possible scheduling states and assigns a corresponding scheduling strategy. Each state is described by several parameters. Gharehgozli et al. [38] presented a new mixed-integer goal programming (MIGP) model for a parallel machine scheduling problem with sequence-dependent set-up times and release dates. Two objectives are considered in the model to minimize the total weighted flow time and the total weighted tardiness simultaneously under the hypothesis of fuzzy processing time's knowledge and two fuzzy objectives as the MIGP model. In addition, a heuristic for solving the above fuzzy model is presented. Muralidhar and Alwarsamy [39] used fuzzy logic and simulated annealing techniques to select the best optimal schedule which minimizes the makespan, total tardiness, and total earliness in parallel machines. In that paper, fuzzy logic is used for obtaining approximate solutions to combinatorial optimization problems and simulated annealing is used as a stochastic approach which strives to overcome local optimality by accepting bad solutions with definite probability.

Chyu and Chang [40] presented two simulated annealing and a greedy randomized adaptive search procedure to solve unrelated parallel machine scheduling problems. To minimize two fuzzy optimization objectives, i.e., makespan and average tardiness, some schemes such as (1) matching-based decoding; (2) acceptance rule based on Pareto dominance, objective fitness, or Pareto reference point distance; and (3) random or fixed weighted direction search, are incorporated into their algorithm. A genetic algorithm was proposed by Balin [41] to minimize the makespan in parallel machine scheduling problems with fuzzy processing times in a simulation model. In this study, the obtained results of genetic algorithm (GA) are compared with the longest processing time rule. In another work, Balin [42] focused on non-identical parallel machine scheduling problem with fuzzy processing times and proposed a genetic algorithm that embedded in a simulation model to makespan. To minimize the makespan, Alcan and Başlıgil [43] proposed a GA for large-scale non-identical parallel machine scheduling problem as a common problem in the industry. They used triangular fuzzy processing times in order to adapt the GA to the problem. Torabi et al. [44] proposed a multi-objective model for a fuzzy unrelated parallel machine scheduling problem considering fuzzy processing times and due dates, ready times, sequence, and machine-dependent set-up times, and secondary resource constraints for jobs. For this problem, they proposed a multi-objective particle swarm optimization (MOPSO) algorithm that minimized the total weighted flow time, total weighted tardiness, and total machine load variation.

According to Bojadziev and Bojadziev [45], the bell-shaped fuzzy numbers depicts the fuzzy normal distribution. Furthermore, the most real-world problems are usually needed to use the well-known normal or Gauss distribution. For these reasons and since in the literature of fuzzy scheduling, the most researchers considered other types of fuzzy numbers, in this study, we used bell-shaped fuzzy processing time. The main contributions of this paper can be concluded as follows:

(1) We pursue a new research by considering the features of parallel machine scheduling with bell-shaped processing time.
(2) We also propose the new formulas for the elementary operations between *bell-shaped fuzzy numbers*.
(3) For scheduling problems with large size, which is the reality in practice, we suggest a novel PSO-GA hybrid algorithm and meanwhile pay special attention to discretize the proposed algorithm.

To the best of our knowledge, our research is the first attempt to include these features simultaneously in a fuzzy scheduling in parallel machine environment.

3 Bell-shaped fuzzy numbers

The bell function depends on three parameters a, b, and c as given by

$$f(x;a,b,c)=\frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \tag{1}$$

where the parameter b is usually positive. The parameter c locates the center of the curve. Figure 1 shows a typical form of bell-shaped fuzzy number, where for a fuzzy set whose universe of discourse is X.

- *Support of a fuzzy set* is all the elements in X that have nonzero membership values.

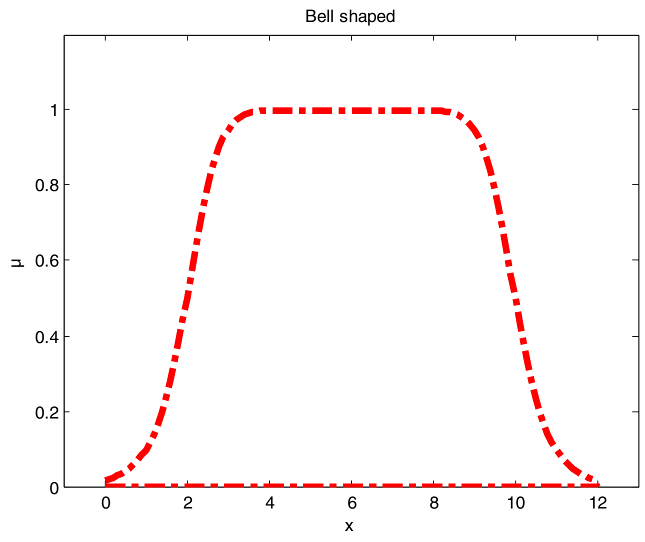


Fig. 1 Bell-shaped fuzzy number with support={0:12}, height=1, and core x=6

- *Height of a fuzzy set* is the largest membership value of a fuzzy set.
- *Core of a fuzzy set* is defined for four different situations. If the membership function of a fuzzy set reaches its maximum at only one element of the universe of discourse, the element is called the center of the fuzzy set. If the membership function of a fuzzy set achieves its maximum at more than one element of the universe of discourse and all these elements are bounded, the middle point of the element is the center. If the membership function of a fuzzy set attains its maximum at more than one element of the universe of discourse and not all of the elements are bounded, the largest element is the center if it is bounded; otherwise, the smallest element is the center. For practical applications, bell-shaped or symmetric-triangular fuzzy numbers are the most popular representations whenever input or output is a fuzzy quantity or linguistically represented.

3.1 Fuzzy arithmetic on bell-shape numbers

In the following, we proposed the following formulas for the elementary operations between *bell-shaped fuzzy numbers*. Given that two fuzzy numbers n_1 and n_2, represented as three field fuzzy numbers of the form

$$n_1 = (a_1, b_1, c_1)$$
$$n_2 = (a_2, b_2, c_2).$$

The sum of the two bell-shaped numbers can be written as follows.

$$\begin{aligned} n_1 + n_2 &= (a_1, b_1, c_1) + (a_2, b_2, c_2) \\ &= (a_1 + a_2, b_1 + b_2, c_1 + c_2). \end{aligned} \tag{2}$$

If we make use of the opposite -n of the bell-shaped fuzzy number n, which can be defined as

$$-n_2 = (-a_2,\ b_2,\ c_2).$$

Now, we can deduce the following formula from sum equation for the subtraction:

$$\begin{aligned} n_1 - n_2 &= (a_1,\ b_1,\ c_1) + (-a_2,\ b_2,\ c_2) \\ &= (a_1 - a_2,\ b_1 + b_2,\ c_1 + c_2). \end{aligned} \tag{3}$$

Note that in the case of subtraction of three field fuzzy numbers, the fuzzy numbers n_1 and n_2 have to be opposite of three field types to guarantee the closure of the operation.

Based on extension principle, the multiplication of numbers n_1 and n_2 can be formulated as the following.

$$\begin{aligned} n_1 . n_2 &= (a_1, b_1, c_1).(a_2, b_2, c_2) \\ &= (a_1 a_2,\ a_1 b_2 + a_2 b_1,\ a_1 c_2 + a_2 c_1). \end{aligned} \tag{4}$$

In the multiplication of numbers n_1 and n_2, if we make use of the *1/n* as

$$1/n_2 = (1/a_2,\ 1/c_2,\ 1/b_2).$$

We can deduce the following formula from the multiplication equation for the division.

$$\begin{aligned} n_1 / n_2 &= (a_1,\ b_1,\ c_1)/(a_2,\ b_2,\ c_2) \\ &= \left(a_1/a_2, (a_1 c_2 + a_2 b_1)/a_2^2, (a_1 b_2 + a_2 c_1)/a_2^2\right). \end{aligned} \tag{5}$$

To generalize the equations for sum operator, we consider the sums of the first two numbers as a new number N_1, then the third number adds to this number,

$$\begin{aligned} N_1 &= n_1 + n_2 = (a_1, b_1, c_1) + (a_2, b_2, c_2) \\ &= (a_1 + a_2, b_1 + b_2, c_1 + c_2), \\ N_2 &= n_1 + n_2 + n_3 = N_1 + n_3 \\ &= (a_1 + a_2, b_1 + b_2, c_1 + c_2) + (a_3, b_3, c_3) \\ &= \left((a_1 + a_2) + a_3, (b_1 + b_2) + b_3, \left(c_1 + c_2\right) + c_3\right), \\ N_3 &= n_1 + n_2 + n_3 + n_4 = N_2 + n_4 \end{aligned} \tag{6}$$

and so on.

In the same way, for three number multiplications, we can write:

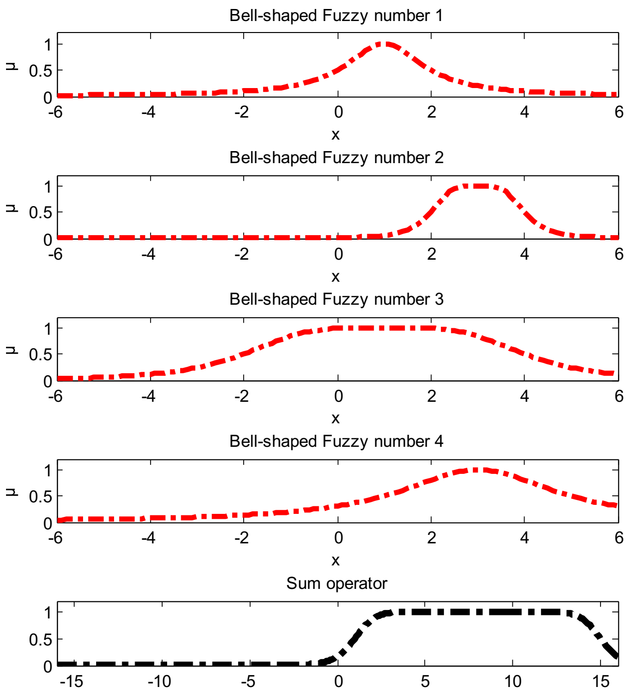
$$\begin{aligned} N_1 &= n_1 . n_2 = (a_1, b_1, c_1).(a_2, b_2, c_2) = (a_1 a_2,\ a_1 b_2 + a_2 b_1,\ a_1 c_2 + a_2 c_1), \\ N_2 &= n_1 . n_2 . n_3 = N_1 + n_3 = (a_1 a_2,\ a_1 b_2 + a_2 b_1,\ a_1 c_2 + a_2 c_1).(a_3, b_3, c_3) \\ N_3 &= n_1 . n_2 . n_3 . n_4 = N_3 . n_4 \end{aligned} \tag{7}$$

For all cases and for n number, the generalized results can be done as calculated above. In the following, the fuzzy arithmetic on bell-shaped fuzzy number is shown.

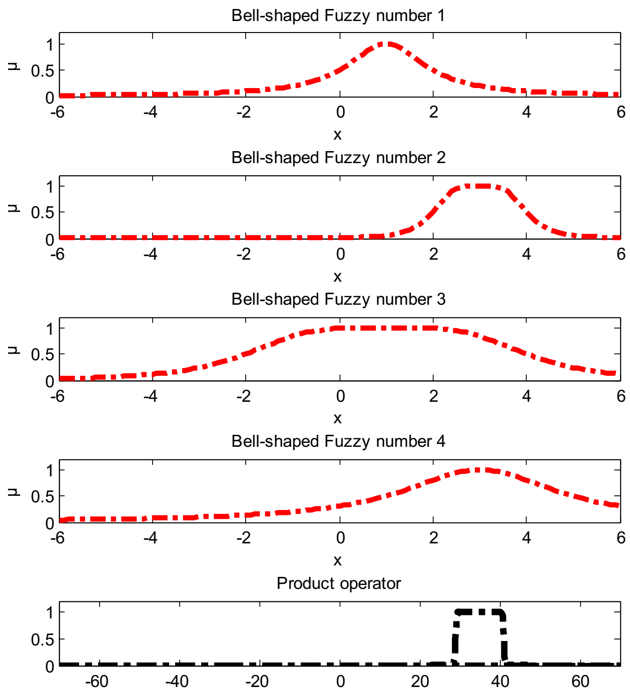
As it can be seen in Fig. 2, there are four fuzzy numbers (1, 2, 3, and 4 in red lines) represented in a, b, c, and d. The last fuzzy number is defined as the sum, multiplication, subtraction and division of these fuzzy numbers.

3.2 Fuzzy ranking function

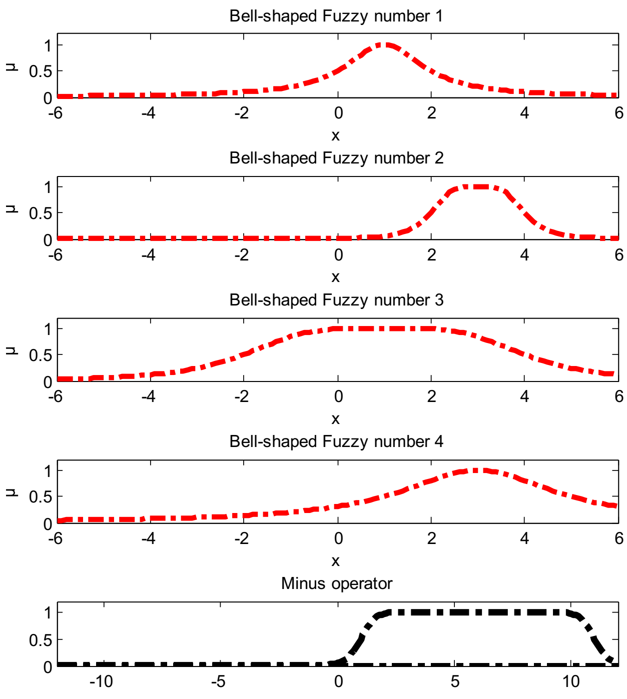
The best sequence is the job sequence with minimum makespan. In our problem, the obtained makespan is still a



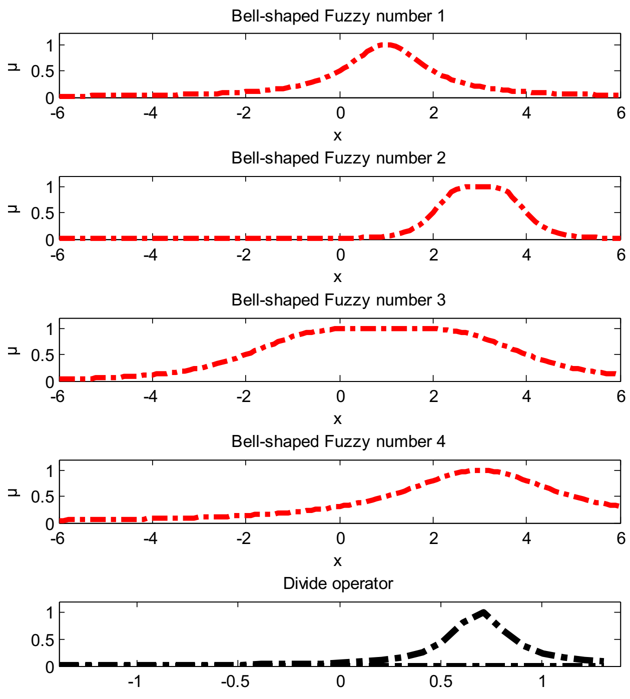
(a) Sum of the bell-shaped fuzzy number



(b) Product of the bell-shaped fuzzy number



(c) Minus of the bell-shaped fuzzy number



(d) Divide of the bell-shaped fuzzy number

Fig. 2 Fuzzy arithmetic on bell-shape fuzzy numbers

fuzzy number. Ranking is an important issue in fuzzy theory because solving a decision problem with fuzzy quantities usually requires choosing among different fuzzy numbers. To solve the problem, we have to compare the quality of feasible schedules. This emerges the need to identify an appropriate ranking method. Although many methods have been suggested in the literature, it is hard to find a method which can provide satisfactory results in most cases. In this study, we use the center of area defuzzification function to rank the fuzzy numbers.

4 The discrete hybrid metaheuristic

Recall that our problem is NP-hard, i.e., in general, it is impossible to find an optimal solution without the use of an enumerative algorithm which has an exponential time and so the use of metaheuristic can be appropriate for solving it in a large-size in a reasonable time. In this respect, the hybridizing of several metaheuristics can improve the efficiency of the searching algorithm [46]. In this paper, we improved the particle swarm optimization algorithm through hybridizing it with genetic algorithm.

An important issue is how each particle can generate a movement policy based on another particle. In continuous problems, there are several ideas in this matter as we have seen in PSO, but in discrete problems, it needs more attention and innovation. As an inspiring example, when each solution of a discrete problem is coded as a chromosome, we can use a crossover and mutation operators as movement policy. This hybridizing allows the possible integration of several basic features of a reference set of metaheuristics and aims at analyzing the effectiveness of the resultant customizable algorithm for a difficult problem and a test problem. Let us now discuss which aspects have been borrowed from PSO and GA.

4.1 Solution representation

The proposed representation to solve the scheduling problem is based on coding all n jobs as genes in a 1-by-($n+m-1$) string. In this type of representation, ($m-1$) genes consisting of "$*$"s are used to differentiate from one machine to the other one.

An example for representation is shown in Fig. 3. In this example, there are 10 jobs and the number of identical parallel machines is 3. As shown in Fig. 1, jobs 1, 4, and 8 with order 4→8→1 are assigned to machine 1; jobs 2 and 3 are assigned

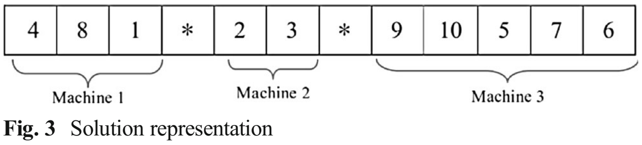


Fig. 3 Solution representation

to machine 2. Jobs 9, 10, 5, 7, and 6 are assigned to the last machine.

It is important to note that the sequence of jobs is represented by the numbers of genes from the left side to the right side.

4.2 Particle swarm optimization

According to what scientists have found, in order to search for food, each member in a flock of birds determines its velocity based on their personal experience as well as information gained through interaction with other members of the flock. This idea was the main principle for PSO. Each bird, called particle, flies through the solution space of the optimization problem searching for the optimum solution and thus its position represents a potential solution for the problem. In particle swarm terminology, the available solutions in each iteration are called the "swarm" which is equivalent to the "population" in genetic algorithms. In this metaheuristic, each particle tries to modify its position using the following terms:

1. *Inertia term*: this term forces the particle to move in the following own way using old velocity.
2. *Cognitive term*: this term forces the particle to go back to the previous best position
3. *Social term*: this term forces the particle to move to the best position of other neighborhoods.

4.2.1 Mathematical concepts of PSO

Suppose a D-dimensional searching space and a swarm of N particles searching for the globally optimal solution within the searching space. Three D-dimensional vectors are assigned to each particle, that is position, denoted by $X_i=(x_{i1}, x_{i2}, \ldots, x_{id})$, velocity, denoted by $V_i=(v_{i1}, v_{i2} \ldots, v_{id})$ and best personal position, denoted by $P_i=(p_{i1}, p_{i2}, \ldots, p_{id})$.

In a continuous searching space, each dimension of the position vector corresponds to the value of a decision variable for the problem. In other words, the position of each particle is a potential solution for the problem at hand and the fitness of this particle (solution) can be calculated by putting these values into a predetermined objective function. When the fitness is more desirable in terms of the objective function, the particle's position is better. Velocity vector represents the distance a particle will traverse in each dimension in each iteration of the algorithm. Best personal position vector is the best visited position of a particle. The particles also need to be aware of the best global position visited by the whole swarm which is denoted by $P_g=(p_{g1}, p_{g2,} \ldots, p_{gd})$. The new velocity of each particle is calculated as follows:

$$v_{\mathrm{id}}(k+1) = \omega\, v_{\mathrm{id}}(k) + \lambda_1 r_1 [p_{\mathrm{id}}(k) - x_{\mathrm{id}}] + \lambda_2 r_2 \left[p_{\mathrm{gd}}(k) - x_{\mathrm{id}}(k)\right] \quad (8)$$

where λ_1 and λ_2 are constants called acceleration coefficients, ω is called the inertia factor, and r_1 and r_2 are two independent random numbers uniformly distributed in the range [0, 1].

Thus, the position of each particle is updated in each generation according to the Eq. (9).

$$x_{\mathrm{id}}(k+1) = x_{\mathrm{id}}(k) + v_{\mathrm{id}}(k+1) \quad (9)$$

In the standard PSO, Eq. (8) is used to calculate the new velocity according to its previous velocity and to the distance of its current position from both its own best historical position and its neighbors' best position. Generally, the value of each component in v_i can be clamped to the range $[-v_{\max}, v_{\max}]$ to control the excessive roaming of particles outside the search space. Then, the particle flies towards a new position according to Eq. (9). This process is repeated until a user-defined stopping criterion is reached. The basic PSO structure is illustrated by Algorithm 1.

Algorithm 1: Basic PSO structure

Step 1: Initialize a population of particles with random positions and velocities, where each particle contains d variables.
Step 2: Evaluate the objective values of all particles. Let p_{best} of each particle and its objective value be equal to its current position and objective value, and let g_{best} and its objective value be equal to the position and objective value of the best initial particle.
Step 3: Update the velocity and position of each particle according to Eqs. (8) and (9).
Step 4: Evaluate the objective values of all particles.
Step 5: For each particle, compare its current objective value with the objective value of its p_{best}. If the current value is better, then update p_{best} and its objective value with the current position and objective value.
Step 6: Determine the best particle of the current swarm with the best objective value. If the objective value is better than the objective value of g_{best}, update g_{best} and its objective value with the position and objective value of the current best particle.
Step 7: If a stopping criterion is met, output g_{best} and its objective value; otherwise, go to Step 3.

4.3 The proposed Discrete PSO

The most important issue in applying PSO to schedule problems is to find a suitable method between the job sequence and the position of particles in PSO. In this study, for discretization, the PSO algorithm, genetic operators are used.

Genetic algorithm is classified as a stochastic optimization algorithm. During the last decades, GA has become one of the most well-known metaheuristics and is widely used in many combinatorial optimization problems, including machine scheduling [47]. Genetic operators such as crossover and mutation are used to generate new individuals and to keep away from local optimum or premature convergence. So, in this paper, we will introduce a discrete PSO algorithm which uses from the crossover and mutation operators to update the particle. The operators performed for updating particle positions are explained in the following.

4.3.1 Swarm initialization

Random initial solutions for constructive metaheuristics are a common and popular procedure. Randomness characteristic of this technique causes varying results in different runs. To generate random solutions for the initial population, the procedure is as follows:

Generate $(m-1)$ asterisks "*" and randomly assign to blocks of the chromosome in which none of them are assigned to the blocks. Then, assign the numbers from 1 to n to the rest of the unfilled blocks. The size of the population depends on the solution space.

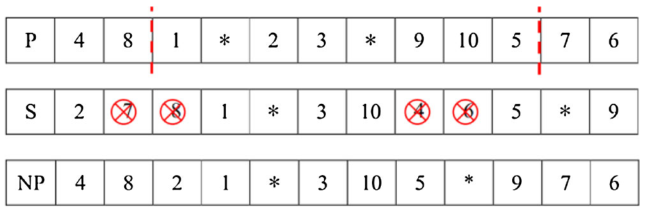
4.3.2 Crossover operator

We use two-point order crossover. As seen in Fig. 4, in this crossover, two crossover points are randomly chosen to divide in each particle into three parts. All blocks (4, 8, 7, and 6) outside the crossover points are inherited from a chosen particle (P) to the new position (NP) with no changes. The other blocks (2, 1, 3, 10, 5, and 9) are sorted in the same order as in the other p_{best} or g_{best} (S_{p} or S_{s}).

By applying this operator, not only assignment can be changed, but also sequencing can be changed in the new position of the particle.

4.3.3 Mutation operators

We use insertion mutation because good results were obtained by this combination in literature. This mutation is often referred to as a shift. As seen in Fig. 5, in this mutation a randomly chosen block (* marked by yellow block) is inserted



P	4	8	1	*	2	3	*	9	10	5	7	6
S	2	$\cancel{7}$	$\cancel{8}$	1	*	3	10	$\cancel{4}$	$\cancel{6}$	5	*	9
NP	4	8	2	1	*	3	10	5	*	9	7	6

Fig. 4 Two-point crossover operator

4	8	1	*	2	3	*	9	10	5	7	6

4	8	1	5	*	2	3	*	9	10	7	6

Fig. 5 Insertion mutation

into a randomly chosen position (the second locus marked by blue block).

4.3.4 Constraint handling

Due to mechanism of generating neighborhood, e.g., mutation and crossover, it is not guaranteed that the individual obtained from the metaheuristic operators will be feasible over the problem constraints. In our proposed algorithm, if between the asterisks there is no unfilled block, it means that there is no assigned job to the corresponding machine. For example, consider the following solution.

In this solution, we have 10 jobs with 3 machines in parallel. As shown in Fig. 6, jobs 4, 8, 1, and 5 are assigned to machine 1 and jobs 3, 2, 9, 10, 7, and 6 are assigned to the last machine.

4.3.5 A hybrid algorithm

The basic proposed hybrid algorithm structure is designed as shown in Algorithm 2.

Algorithm 2: DPSO structure

Step 1: Randomly initialize a population with 20 particles
Step 2: Evaluate the objective values of all particles for determining p_{best} and g_{best}
Step 3: Update the velocity of each particle according to following procedure.

- Through the mutation process, we make a slight change in the sequence, i.e., generating a new but similar sequence as a solution based on inertia (first part of Eq. 8).
- Use crossover for generating the solution based on p_{best} and based on g_{best} (second and third parts of Eq. 8) for each particle.

Step 4: Update the position of each particle: After the calculating three components of velocity (solution based on inertia (S_{i}), solution based on p_{best} (S_{p}), solution based on social (S_{s})), elitism among $\{S_{\text{i}}, S_{\text{p}}, S_{\text{s}}\}$ is used to update the new population and enforce the search ability.

4	8	1	5	*	*	3	2	9	10	7	6

Fig. 6 Constraint handling

Step 5: Evaluate each particle by its corresponding objective value
Step 6: Determine the best particle of the current swarm with the best objective value. If the objective value is better than the objective value of g_{best}, update g_{best} and its objective value with the position and objective value of the current best particle.
Step 7: If a stopping criterion is met, output g_{best} and its objective value; otherwise, go to Step 3.

5 Computational results

In order to evaluate the effectiveness of the algorithm proposed in the previous section, we compared our proposed algorithm with lower bound and GA proposed by Balin [41]. In this paper, we have run the DPSO algorithm ten times for small, medium, and large test problems.

The metaheuristic algorithms were implemented in MATLAB 7 and run on an Intel Pentium IV dual core 2.00 GHz PC at 1,022 MB RAM under a Microsoft Windows Vista environment.

5.1 Data generation and settings

To analyze the algorithm developed in this problem, several classes of instances are defined. In each class, there is a change in one of the inputs. Following Kurz and Askin [48], the problem data can be characterized by three factors, and each of these factors can have at least two levels. These levels are shown in Table 3.

For the test on our algorithm against algorithm presented previously, 108 instances were randomly generated, i.e., six problems for each of the 18 combinations of three levels of the number of jobs, number of machines, and two levels of core of processing times. In general, all combinations of these levels are tested. In each class, there is a change in one of the inputs. However, some further restrictions are introduced. The largest number of machines on the stage must be two times less than the number of jobs. For example, the instance with 10 machines and 6 jobs will be replaced with the combination of 4 machines with 6 jobs.

Table 3 Factor levels

Factor	Levels		
Number of jobs	6	30	100
Number of machines	2	5	10
Core of processing times	(10, 20)	(50, 70)	

Table 4 Computational results

Instance (job × machine)	PSO-based algorithm			Genetic algorithm		
	Wor.	Opt.	Avg.	Wor.	Opt.	Avg.
6×2	0.02719	0.00328	0.01620	0.044440	0.005328	0.021792
6×3	0.02925	0.01360	0.02391	0.063220	0.017689	0.032767
6×4	0.24733	0.20195	0.23822	0.405967	0.313987	0.395950
6 Jobs	**0.10126**	**0.07294**	**0.09277**	**0.171209**	**0.112335**	**0.150170**
30×2	0.00585	0.00033	0.00170	0.161240	0.111381	0.122035
30×5	0.13570	0.02396	0.06843	0.211227	0.030052	0.147894
30×10	0.28154	0.09853	0.19187	0.353045	0.146480	0.267466
30 Jobs	**0.14103**	**0.04094**	**0.08733**	**0.241837**	**0.095971**	**0.179132**
100×2	0.00220	0.00013	0.00067	0.003615	0.000219	0.000915
100×5	0.16011	0.02421	0.07958	0.222896	0.031258	0.120167
100×10	0.42327	0.16960	0.29424	0.669971	0.223065	0.429662
100 Jobs	**0.19520**	**0.06465**	**0.12483**	**0.298828**	**0.084847**	**0.183581**
Average	**0.14583**	**0.05951**	**0.10164**	**0.237291**	**0.097717**	**0.170961**

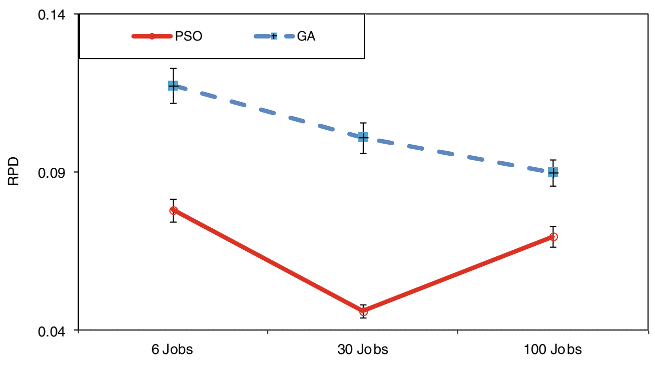
Data in bold are average of the corresponding above column

5.2 Stopping rule

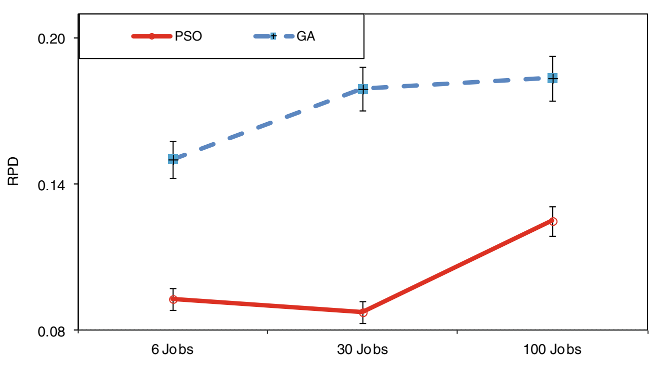
The stopping criterion is set to a number of iterations to 50 repetitions.

5.3 Evaluation Metric

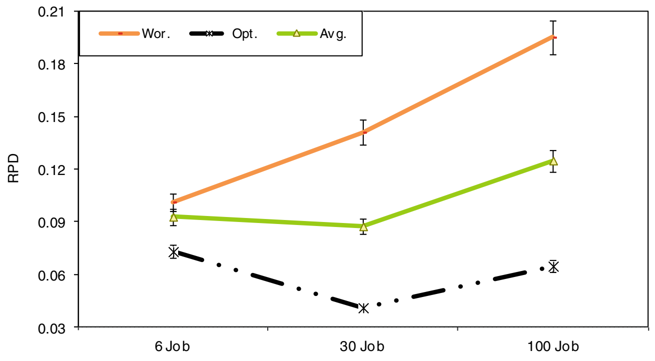
Another important decision is how to evaluate the quality of the solutions. After computation of $C_{\max}$ of our algorithm for



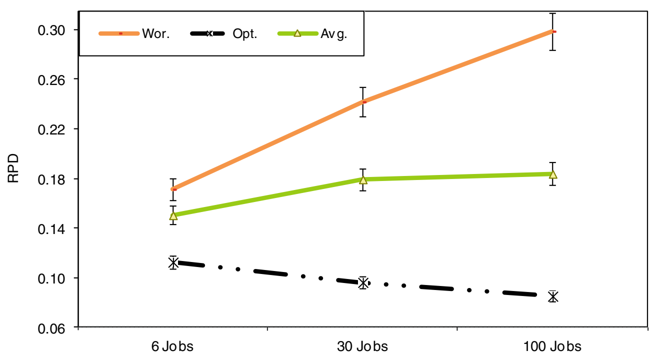
(a) Best case comparison



(b) Average case comparison

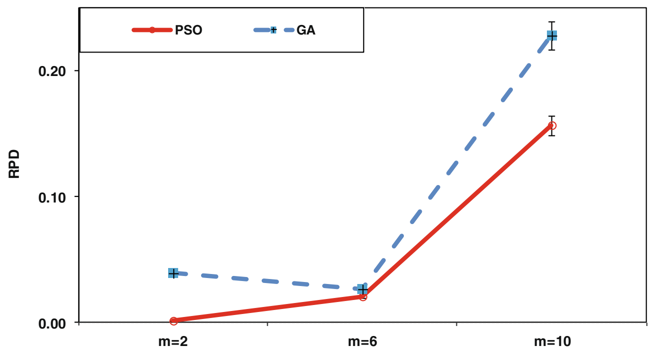


(c) DPSO analysis

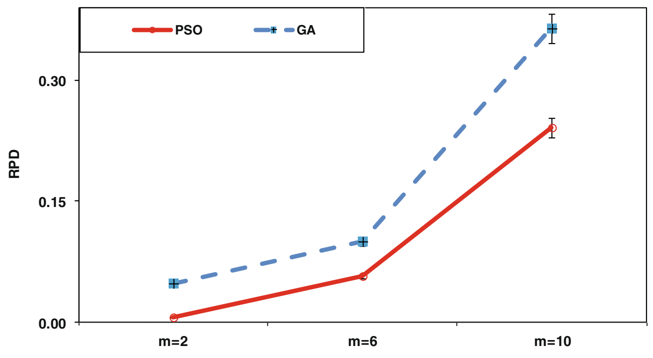


(d) GA analysis

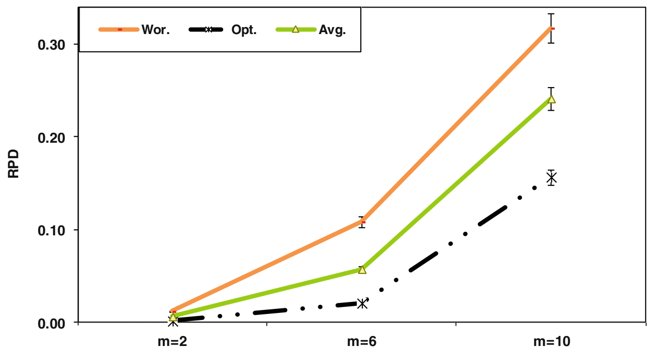
Fig. 7 Plot of *RPD* for the interaction between the type of algorithm and the number of jobs



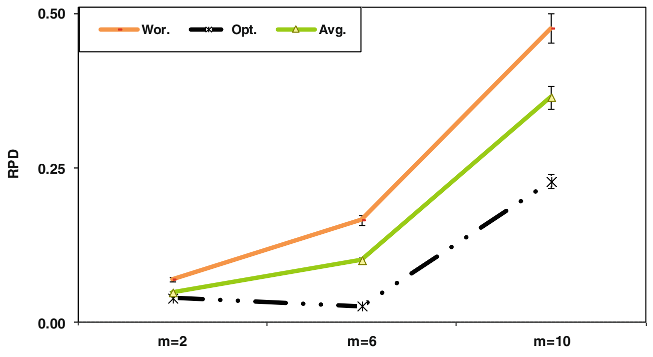
(a) Best case comparison



(b) Average case comparison



(c) DPSOanalysis



(d) GA analysis

Fig. 8 Plot of *RPD* for the interaction between the type of algorithm and magnitude of machines

an instance, the lower bound (LB_{cmax}) is calculated. Relative percentage deviation (RPD) is calculated as follows:

$$\mathrm{RPD} = \frac{\mathrm{Alg}_{\mathrm{DPSO}} - \mathrm{LB}}{\mathrm{LB}} \tag{10}$$

Where LB_{cmax} can be calculated as follows.

$$LB_{\mathrm{cmax}} = \frac{\text{Sum of the processing time of job}}{\text{Number of machines in parallel}} \tag{11}$$

5.4 Numerical results

In order to evaluate the efficacy and performance of the algorithms, the following instances are used. Each instance is solved using 10 different seeds and the average, best, and worst solutions are considered. Table 4 represents the comparison results.

5.5 Analysis of controlled factors

In order to see the effect of control factor in our proposed algorithm, a two-way ANOVA is applied.

5.5.1 Analyzing factor n (number of jobs)

Figure 7 shows the least significant difference (LSD) intervals and means plot at the 95 % confidence level for the interaction between the factors of type of solution (worse case (Wor.), best solution (Opt.) and average (Avg.)) and number of jobs.

As we can see, in general, with increasing the size of problems the performance of the algorithm is worsening.

5.5.2 Analyzing factor m (number of machines)

Another LSD test is applied to see the effect of magnitude of machines on the quality of the algorithm. The interaction plot is shown in Fig. 8.

In this figure, we can see that the performance of the algorithm is declined when the number of machines on stage is increased.

6 Conclusions and future works

This paper presented a new particle swarm optimization (PSO) which discretize with hybridizing it with genetic algorithm (GA) operators to fuzzy parallel machine scheduling problem. The hybrid metaheuristic which combines PSO and GA has the ability to balance exploration and exploitation and the local improvement ability. In this paper, imprecise processing times are modeled as bell-shaped fuzzy numbers, which results in a makespan that is a bell-shaped fuzzy number. Our experimental results indicated that the proposed DPSO generally has good performance. This work facilitates not only the fuzzy machine scheduling problem, but also can be used as a tool to extend several other discrete optimization problems with the fuzzy context. Since most practical problems involve multiple objectives, another worthy direction of research is to develop our proposed hybrid algorithm for multi-objective scheduling.

References

1. Prade H (1979) Using fuzzy set theory in a scheduling problem: a case study. Fuzzy Sets Syst 2:153–165
2. Dumitru V, Luban F (1982) Membership functions, some mathematical programming models and production scheduling. Fuzzy Sets Syst 8:19–33
3. García-Villoria RP (2009) Introducing dynamic diversity into a discrete particle swarm optimization. Comput Oper Res 36(3):951–966
4. Kennedy J, Eberhart R (1995) Particle swarm optimization. Proceeding of the 1995 I.E. international conference on neural network, Perth, Australia, pp. 1942–1948
5. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In Proceedings of the IEEE 1997 International Conference on Systems, Man and Cybernetics, pp. 4104–4109
6. Mohan CK, Al-kazemi B (2001) Discrete particle swarm optimization. Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis, IN: Purdue School of Engineering and Technology, IUPUI. pp. 22–29
7. Laskari E.C., Parsopoulos, K.E., Vrahatis, M.N. (2002) Particle swarm optimization for integer programming. In: Proceedings of the IEEE 2002 Congress on Evolutionary Computation, Honolulu (HI), pp. 1582–1587
8. Hu et al (2003) Swarm intelligence for permutation optimization: a case study of N-queens problem. Proceedings of the IEEE Swarm Intelligence Symposium, 243–246
9. Yin P-Y (2004) A discrete particle swarm algorithm for optimal polygonal approximation of digital curves. J Vis Commun Image Represent 15(2):241–260
10. Chen et al (2006) Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. J Zhejiang Univ Sci A 7:607–614
11. Sha DY, Hs C-Y (2006) A hybrid particle swarm optimization for job shop scheduling problem. Comput Ind Eng 51(4):791–808
12. Xiong Y, Cheng H-Z, Yan J-y, Zhang L (2007) New discrete method for particle swarm optimization and its application in transmission network expansion planning. Electr Power Syst Res 77(3–4):227–233
13. Karthi R, Arumugam S, Ramesh Kumar K (2009) Discrete particle swarm optimization algorithm for data clustering nature inspired cooperative strategies for optimization, NICSO 2008, Volume 236
14. Wang X, Tang L (2009) A tabu search heuristic for the hybrid flowshop scheduling with finite intermediate buffers. Comput Oper Res 36(3):907–918
15. Król D, Drożdżowski M (2010) Use of MaSE methodology and swarm-based metaheuristics to solve the traveling salesman problem, J Intell Fuzzy Syst 21, in press
16. Zhang Z, Jiang S, Zhang Y, Geng S, Wang H, Sang G (2014) An adaptive particle swarm optimization algorithm for reservoir operation optimization. Appl Soft Comput 18:167–177
17. Shakibian H, Charkari NM (2014) n-cluster vector evaluated particle swarm optimization for distributed regression in WSNs. J Netw Comput Appl 42:80–91
18. Wang C, Liu Y, Zhao Y, Chen Y (2014) A hybrid topology scale-free Gaussian-dynamic particle swarm optimization algorithm applied to real power loss minimization. Eng Appl Artif Intell 32:63–75
19. Rada-Vilela J, Johnston M, Zhang M (2014) Population statistics for particle swarm optimization: Resampling methods in noisy optimization problems, Swarm and Evolutionary Computation, In Press
20. Das G, Kumar Pattnaik P, Kumari Padhy S (2014) Artificial neural network trained by particle swarm optimization for non-linear channel equalization. Expert Syst Appl 41(7):3491–3496
21. Zhang E, Wu Y, Chen Q (2014) A practical approach for solving multi-objective reliability redundancy allocation problems using extended bare-bones particle swarm optimization. Reliab Eng Syst Saf 127:65–76
22. Wang S-C, Yeh M-F (2014) A modified particle swarm optimization for aggregate production planning. Expert Syst Appl 41(6):3069–3077
23. Xue B, Zhang M, Browne WN (2014) Particle swarm optimisation for feature selection in classification: novel initialisation and updating mechanisms. Appl Soft Comput 18:261–276
24. Sadeghi J, Sadeghi S, Taghi Akhavan Niaki S (2014) Optimizing a hybrid vendor-managed inventory and transportation problem with fuzzy demand: an improved particle swarm optimization algorithm. Inf Sci 272:126–144
25. Liu R, Chen Y, Jiao L, Li Y (2014) A particle swarm optimization based simultaneous learning framework for clustering and classification. Pattern Recogn 47(6):2143–2152
26. Du C-L, Luo C-X, Han Z-T, Zhu Y-S (2014) Applying particle swarm optimization algorithm to roundness error evaluation based on minimum zone circle. Measurement 52:12–21
27. Elsayed SM, Sarker RA, Mezura-Montes E (2014) Self-adaptive mix of particle swarm methodologies for constrained optimization, Information Sciences, In Press
28. Wang X, Ma L, Wang T (2014) An optimized nearest prototype classifier for power plant fault diagnosis using hybrid particle swarm optimization algorithm. Int J Electr Power Energy Syst 58:257–265
29. Bagheri A, Mohammadi Peyhani H, Akbari M (2014) Financial forecasting using ANFIS networks with quantum-behaved particle swarm optimization, expert systems with applications, In Press
30. Koulinas G, Kotsikas L, Anagnostopoulos K (2014) A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem, Information Sciences, In Press
31. Li Y, Luh PB, Guan X (1994) Fuzzy Optimization-based scheduling of identical machines with possible breakdown, robotics and automation, Proceedings, IEEE International Conference, San Diego, CA, May 1994, vol.4, 3447–3452
32. Hong TP, Yu KM, Huang CM (1998) LPT scheduling on fuzzy tasks with Triangular membership function, Second International Conference on Knowledge-Based Intelligent Elecwonic Systems, April 1998, Adelaide, Australia. Editors, L.C. Jain and R.K. Jab
33. Peng J, Song K (2001) Expected value goal programming models for fuzzy scheduling problem. Proceedings of the Tenth IEEE International Conference on Fuzzy Systems, December, 2001. pp. 292–295, Melbourne, Australia
34. Peng J, Liu B (2004) Parallel machine scheduling models with fuzzy processing times. Inf Sci 166(1–4):49–66

35. Anglani A, Grieco A, Guerriero E, Musmanno R (2005) Robust scheduling of parallel machines with sequence-dependent set-up costs. Eur J Oper Res 161:704–720
36. Petrovic D, Duenas A (2006) A fuzzy logic based production scheduling/rescheduling in the presence of uncertain disruptions. Fuzzy Sets Syst 157:2273–2285
37. Franke C, Hoffmann F, Lepping J, Schwiegelshohn U (2008) Development of scheduling strategies with genetic fuzzy systems. Appl Soft Comput 8:706–721
38. Gharehgozli AH, Tavakkoli-Moghaddam R, Zaerpour N (2009) A fuzzy-mixed-integer goal programming model for a parallel-machine scheduling problem with sequence-dependent setup times and release dates. Robot Comput Integr Manuf 25:853–859
39. Muralidhar A, Alwarsamy T (2009) Multi-objective optimization of parallel machine scheduling using fuzzy logic and simulated annealing. Int J Appl Eng Res 4:11
40. Chyu C-C, Chang W-S (2011) Optimizing fuzzy makespan and tardiness for unrelated parallel machine scheduling with archived metaheuristics. Int J Adv Manuf Technol 57(5–8):763–776
41. Balin S (2011) Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation. Inf Sci 181(17):3551–3569
42. Balin S (2012) Non-identical parallel machine scheduling with fuzzy processing times using genetic algorithm and simulation. Int J Adv Manuf Technol 61(9–12):1115–1127
43. Alcan P, Başlıgil H (2012) A genetic algorithm application using fuzzy processing times in non-identical parallel machine scheduling problem. Adv Eng Softw 45(1):272–280
44. Torabi SA, Sahebjamnia N, Mansouri SA, Aramon Bajestani M (2013) A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem. Appl Soft Comput 13(12):4750–4762
45. Bojadziev G, Bojadziev M (1996) Fuzzy sets, fuzzy logic, applications. World Scientific Pub Co Inc, Hackensack
46. Yuan Q, Qian F, Du W (2010) A hybrid genetic algorithm with the Baldwin effect. Inf Sci 180(5):640–652
47. Low C, Yuling Y (2009) Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separated. Robot Comput Integr Manuf 2(25):314–322
48. Kurz ME, Askin RG (2003) Comparing scheduling rules for flexible flow lines. Int J Prod Econ 85:371–388
49. Tavakkoli-Moghaddam R, Azarkish M, Sadeghnejad-Barkousaraie A (2011) A new hybrid multi-objective Pareto archive PSO algorithm for a bi-objective job shop scheduling problem. Expert Syst Appl 38(9):10812–10821
50. Sha DY, Lin H-H (2010) A multi-objective PSO for job-shop scheduling problems. Expert Syst Appl 37(2):1065–1070
51. Lei D (2008) A Pareto archive particle swarm optimization for multi-objective job shop scheduling. Comput Ind Eng 54(4):960–971
52. Zhang G, Shao X, Li P, Gao L (2009) An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. Comput Ind Eng 56(4):1309–1318
53. Lin T-L et al (2010) An efficient job-shop scheduling algorithm based on particle swarm optimization. Expert Syst Appl 37(3):2629–2636
54. Liu B, Wang L, Jin Y-H (2008) An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. Comput Oper Res 35(9):2791–2806
55. Liou C-D, Liu C-H (2010) A novel encoding scheme of PSO for two-machine group scheduling. Adv Swarm Intell Lect Notes ComputSci 6145:128–134
56. Tang J, Zhang G, Lin B, Zhang B (2010) A hybrid PSO/GA algorithm for job shop scheduling problem. Adv Swarm Intell Lect Notes Comput Sci 6145:566–573
57. Tavakkoli-Moghaddam R, Azarkish M, Sadeghnejad A (2010) A new hybrid multi-objective Pareto archive PSO algorithm for a classic job shop scheduling problem with ready times. Adv Intell Comput Theor Appl Commun Comput Inf Sci 93:61–68
58. Tu K, Hao Z, Chen M (2006) PSO with improved strategy and topology for job shop scheduling. Adv Nat Comput Lect Notes Comput Sci 4222:146–155
59. Tavakkoli-Moghaddam R, Azarkish M, Sadeghnejad-Barkousaraie A (2011) Solving a multi-objective job shop scheduling problem with sequence-dependent setup times by a Pareto archive PSO combined with genetic operators and VNS. Int J Adv Manuf Technol 53(5–8):733–750
60. AitZai A, Benmedjdoub B, Boudhar M (2014) Branch-and-bound and PSO algorithms for no-wait job shop scheduling, Journal of Intelligent Manufacturing, in press
61. Zhang G, Zuo X (2013) Deadline constrained task scheduling based on standard-PSO in a hybrid cloud. Adv Swarm Intell Lect Notes Comput Sci 7928:200–209
62. Dousthaghi S, Tavakkoli-Moghaddam R, Makui A (2013) Solving the economic lot and delivery scheduling problem in a flexible job shop with unrelated parallel machines and a shelf life by a proposed hybrid PSO. Int J Adv Manuf Technol 68(5–8):1401–1416
63. Niu Q, Zhou T, Wang L (2010) A hybrid particle swarm optimization for parallel machine total tardiness scheduling. Int J Adv Manuf Technol 49(5–8):723–739
64. Sha DY, Lin HH (2009) A particle swarm optimization for multi-objective flowshop scheduling. Int J Adv Manuf Technol 45(7–8): 749–758
65. Chakaravarthy GV, Marimuthu S, Naveen Sait A (2013) Performance evaluation of proposed Differential Evolution and Particle Swarm Optimization algorithms for scheduling m-machine flow shops with lot streaming. J Intell Manuf 24(1):175–191
66. Jamili A, Shafia MA, Tavakkoli-Moghaddam R (2011) A hybrid algorithm based on particle swarm optimization and simulated annealing for a periodic job shop scheduling problem. Int J Adv Manuf Technol 54(1–4):309–322
67. Shiau D-F, Huang Y-M (2012) A hybrid two-phase encoding particle swarm optimization for total weighted completion time minimization in proportionate flexible flow shop scheduling. Int J Adv Manuf Technol 58(1–4):339–357
68. Marinakis Y, Marinaki M (2013) Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem. Soft Comput 17(7):1159–1173
69. Akhshabi M, Tavakkoli-Moghaddam R, Rahnamay-Roodposhti F (2014) A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time. Int J Adv Manuf Technol 70(5–8):1181–1188
70. Wang X, Tang L (2010) An improved particle swarm optimization for permutation flowshop scheduling problem with total flowtime criterion. Adv Swarm Intell Lect Notes Comput Sci 6145:144–151
71. Damodaran P, Gangadhara Rao A, Mestry S (2013) Particle swarm optimization for scheduling batch processing machines in a permutation flowshop. Int J Adv Manuf Technol 64(5–8):989–1000