Int J Adv Manuf Technol (2016) 85:2439–2458
DOI 10.1007/s00170-015-8067-1

ORIGINAL ARTICLE

A modified imperialist competitive algorithm for scheduling single batch-processing machine with fuzzy due date

S. Molla-Alizadeh-Zavardehi[1] · R. Tavakkoli-Moghaddam[2] · F. Hosseinzadeh Lotfi[3]

Received: 21 September 2012 / Accepted: 2 November 2015 / Published online: 18 November 2015

Abstract This paper considers a fuzzy single batch-processing machine (SBPM) scheduling problem and aims to make it closer to a real-world application through a fuzzy set theory. For this purpose, jobs' due dates are set to be fuzzy numbers and the membership function of a fuzzy due date assigned to each job represents the degree of satisfaction that a decision maker has with the completion time of that job. The objective is to maximize the total degree of satisfaction or equivalently to minimize the total degree of dissatisfaction over given jobs. Then, the fuzzy mathematical programming model is presented. To solve the model, we propose the imperialist competitive algorithm (ICA) and modify the assimilation policy (i.e., colonies moving) and imperialistic competition in order to overcome its immature convergence and improve its performances. Moreover, due to the significant role of parameters on the quality of random search algorithms, a robust calibration is applied on the parameters using the Taguchi optimization technique. To evaluate the proposed ICA, several random test problems are generated and its performance is compared to the traditional ICA, simulated annealing (SA), particle swarm optimization (PSO), genetic algorithm (GA), ant colony optimization (ACO), and earliest due date (EDD). The obtained computational results demonstrate the superiority and robustness of the modified ICA.

✉ S. Molla-Alizadeh-Zavardehi
saber.alizadeh@gmail.com

[1] Department of Industrial Engineering, Masjed-Soleiman Branch, Islamic Azad University, Masjed-Soleiman, Iran

[2] Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran

[3] Department of Mathematics, Science & Research Branch, Islamic Azad University, Tehran, Iran

Keywords Single batch-processing machine · Fuzzy due date · Imperialist competitive algorithm · Genetic algorithm · Taguchi experimental design

1 Introduction

Batching or grouping of jobs in many manufacturing industries is a common policy. The main advantage is to reduce setups and/or facilitation of material handling. Batching occurs in two different versions: serial batching and parallel batching. On the serial batching machine, the length of a batch equals the sum of the processing times of its jobs. On a parallel batching machine, all the jobs in a batch are processed simultaneously and then released together from the machine, so the length of a batch is the largest processing time of its jobs. In the literature, parallel batching scheduling is known as batch-processing machine (BPM) scheduling.

The BPM scheduling problem is important because the scheduling of batching operations has a significant economic impact. It is mainly motivated by an industrial application, namely, the burn-in operation found in the final testing phase in semiconductor manufacturing [53, 54]. Moreover, BPMs are encountered in various environments such as shoe manufacturing industry, aircraft industry, furniture manufacturing industry, ion plating industry, iron and steel industry, steel casting industry, glass container industry, and the like. A perfect explanation of the basic product flow in BPM and semiconductor manufacturing can be seen in Uzsoy et al. [53] and Knutson et al. [24]. The BPM can process a batch of jobs as long as the sum of all the job sizes in the batch does not violate the capacity of the machine. The processing time of a batch is equal to the longest processing time of all the jobs in that batch. The processing time and the size of each job are known. Once a batch is processed, it cannot be interrupted

and no jobs can be removed from the machine until the process is completed.

The rest of our work is organized as follows. The next section presents literature on the problem under study and a brief review of fuzzy theory for scheduling problems. In Sect. 3, we describe the problem in detail and present the fuzzy mathematical model. Section 4 presents the proposed modified imperialist competitive algorithm (ICA). The numerical examples, the performance analysis using the design of experiments (DOE), and computational results are worked out in Sect. 5. Finally, the conclusions are given.

2 Literature review

In the recent years, batching problems have attracted many researchers both in academia and in industry; however, contrary to classical scheduling problems, little research has been reported on the optimization for these kinds of scheduling problems. In this section, we focus on reviewing the most related studies in their assumptions with ours, especially the papers investigating single batch-processing machine (SBPM), solution algorithms, and the case of fuzzy environment. The literature reviewed shows that the scheduling problems in BPMs are typically solved through a heuristic, dynamic programming algorithm (DPA), or a branch-and-bound procedure (B&B). Since the problem is non-deterministic polynomial-time (NP)-hard, solving and obtaining the optimal solutions for medium- to large-sized problems by use of exact methods require long run time; thus, seeking optimal solutions is often impractical and it is necessary to use efficient metaheuristic methods. Hence, metaheuristic methods are often utilized in the recent related studies in order to solve the problem in a shorter run time.

2.1 Scheduling single batch-processing machine

Ikura and Gimple [20] were the first batch of researchers who studied the BPM problem. They proposed an $O(n^2)$ algorithm with unit job sizes, identical job processing times, dynamic job arrivals, and the objective of minimizing the makespan. Lee et al. [28] first presented a detailed description for burn-in operation and developed DPA for minimizing maximum tardiness, the number of tardy jobs, and maximum lateness, assuming identical job sizes and agreeable release times and due dates.

Uzsoy [50] proved that minimizing the makespan and total completion time is strongly NP-hard. He considered the special case of a SBPM, in which all the jobs have identical processing times, and then, the problem is equivalent to a bin packing problem. Hence, heuristics and B&B methods for bin packing problems are modified to solve SBPM problems. Uzsoy [51] extended his previous research where jobs from different job families cannot be batched together. He presents several heuristics to address these types of problems aimed at minimizing maximum lateness and maximum completion time.

Li and Lee [29] extended the agreeability on release times and due dates to the case for processing times. They proved that the problems are strongly NP-hard and also proposed DPA, which is extended from the algorithm presented in Lee et al. [28]. Uzsoy and Yang [52] developed a B&B method and provided several heuristics to minimize the total weighted completion time. Dupont and Jolai Ghazvini [13] developed heuristics to minimize the makespan. In another study for the same problem, the mean flow time criterion is considered by Hochbaum and Landy (1997) and Jolai Ghazvini and Dupont [23].

Considering dynamic job arrivals, Lee and Uzsoy [27] first presented polynomial and pseudo polynomial time algorithms for several special cases and presented various heuristics to minimize the makespan. Sung and Choung [46] presented some heuristics and a B&B method, with worst-case error bounds derived to minimize makespan with different job release times. Sung et al. [47] presented a DPA for the problem considered by Sung and Choung [46] with one job family. By allowing jobs to be split and processed in different batches, Dupont and Flipo [12] presented some dominance properties and provided a B&B method to minimize the makespan.

Extending the model investigated by Jolai Ghazvini and Dupont [23], Chang and Wang [6] provided a three-phase heuristic method in the presence of dynamic job arrival times, and then, Yao et al. [59] proposed a B&B method on minimizing the total completion time. Jolai Ghazvini [22] showed that the problem of minimizing the number of tardy jobs is NP-hard and presented a polynomial time DPA for the fixed number of job families and batch machine capacity. Perez et al. [39], Mathirajan and Sivakumar [32], Tangudu and Kurz [48], and Kurz and Mason [26] developed several heuristics, some greedy heuristics, a B&B method, and another B&B method, respectively, to minimize the total weighted tardiness under the situation of incompatible job families. Liu et al. [30] considered a case with the assumptions of identical job sizes, agreeable due dates, and processing times and propose a DPA with pseudo polynomial time for minimizing the weighted number of tardy jobs and the total tardiness. He et al. [15] presented a polynomial time algorithm to minimize bi-criteria of the makespan and maximum tardiness simultaneously.

Rafiee Parsa et al. [41] proposed a branch and price algorithm which combines the column generation technique with B&B to minimize the makespan. Bellanger and Oulamara [3] presented some dominance properties and provided a DPA to minimize the total completion time. Liu et al. [31] considered hierarchical bi-criteria scheduling where primary criterion is the makespan. They proved that the problem where the secondary criterion is the weighted number of late jobs is NP-hard and

the problem where the secondary criterion is the total completion time can be solved in polynomial time. Sabouni and Jolai [44] considered the problem of minimizing a linear combination of the objectives of the makespan and the maximum lateness in which jobs are ordered by two customers. Optimal methods are proposed for the case of incompatible customers, and the batch capacity is unbounded, and for the case of compatible customers, the batch capacity is bounded and the processing times are identical for the customer with the maximum lateness objective, respectively.

2.2 Metaheuristic algorithms

For large problem instances, the B&B method and other exact procedures are computationally burdensome. Recently, metaheuristics have been employing to solve the problem, because of their good performance and short run time.

Considering dynamic job arrivals, Wang and Uzsoy [57] extended the case of Li and Lee [29] and combined a DPA with a random key genetic algorithm (GA) to minimize the maximum lateness. Melouk et al. [35] used a simulated annealing (SA) to minimize the makespan and provided a random procedure to produce instances of the problem. Koh et al. [25] proposed some heuristics and a random key representation-based GA for the problems of minimizing the makespan and total weighted completion time with incompatible job families. Sevaux and Peres [45], Husseinzadeh Kashan et al. [18], and Damodaran et al. [11] used a GA and redesigned the coding and decoding methods.

Mönch et al. [37] presented a GA combined with dominance properties to minimize the earliness–tardiness of the jobs under the constraint that the maximum tardiness should be less than or equal to the maximum allowable time value and the situation of unrestrictive late common due date. Chou et al. [10] and Wang et al. [58] presented a hybrid GA and a hybrid forward/backward approach to minimize the makespan. Husseinzadeh Kashan and Karimi [17] developed two versions of an ant colony optimization (ACO) framework, depending on the type of embedded heuristic information under the situation considered in Koh et al. [25].

Chou and Wang [9], Mathirajan et al. [34], and Wang [56] proposed a hybrid GA, SA, and iterated heuristic for the objective of the total weighted tardiness, respectively. Husseinzadeh Kashan et al. [19] considered bi-criteria scheduling with non-identical job sizes. For the simultaneous minimization of the bi-criteria of the makespan and maximum tardiness, they proposed two different multiobjective GAs based on different representation schemes. Chen et al. [7] showed that minimizing the makespan on a single batching machine can be regarded as a special clustering problem and provide a clustering-based algorithm.

2.3 Scheduling problems with fuzzy data

In the classic scheduling problems, it is usually assumed that the aspects of the problem in hand are certain. Most existing models neglect the presence of uncertainty within a scheduling environment. In many real-world scheduling problems, uncertainty and vagueness in due date often do exist that make the models more complex. This uncertainty might come about because of production problems (e.g., defect in raw material, machine malfunctioning) or problems with delivery itself (e.g., transportation delay, traffic jam). However, in many practical situations, due dates may be vague and it is too difficult to decide the due dates accurately. In some cases, due dates are certain intervals and it may be proper to deal with due dates as fuzzy values. To describe the uncertainty in the manufacturing management, fuzzy logic is appropriate [55]. Since Ishii et al. [21] introduced the concept of fuzzy due dates to scheduling problems, fuzzy due date scheduling problems have been investigated by many researchers. Although classic BPM scheduling models are extensively studied in the literature, there are only three studies on fuzzy-based BPM models.

It is generally agreed to accept some deviations from a due date, and therefore, a fuzzy due date can be encountered in this research area. The concept of fuzzy due dates to scheduling problems was introduced by Ishii et al. [21]. Harikrishnan and Ishii [14] studied the serial batching problem with resource-dependent processing time and common setup in the presence of fuzzy due dates. For the fuzzy due dates, a membership function describing non-decreasing satisfaction degree about completion time of each job is defined. They presented a polynomial time for bi-criteria scheduling to minimize the total weighted resource consumption and maximize the minimal satisfaction degree of due dates of jobs.

Yimer and Demirli [60] considered a fuzzy goal programming problem for batch scheduling of jobs on parallel machines in a two-stage flow shop for minimizing the total weighted flow time of all jobs to improve customer responsiveness. They considered the uncertainty for processing and setup times by triangular fuzzy sets. They have also applied a GA for solving large-sized problems. Cheng et al. [8] introduced the fuzzy model of the makespan on an SBPM with non-identical job sizes. The uncertainty of the jobs and the machine in the processing is denoted using triangular fuzzy numbers. The fuzzy model describes the uncertainty in practice including the adjustment times and the processing times of the batches. They also proposed an ACO and employed the metropolis criterion to select the paths of ants. For a further review on BPM scheduling problems, we refer to Potts and Kovalyov [40]. Besides, Mathirajan and Sivakumar [33] did a quite complete survey on scheduling with BPMs.

In the literature, there is no research that considers the objective of minimizing the total weighted fuzzy earliness–tardiness penalties, and a new approach to solving a fuzzy SBPM

(FSBPM) is proposed. A trapezoidal fuzzy number are considered for due dates as an extended triangular fuzzy due date and modeled by fuzzy sets, in which the corresponding membership functions represent satisfaction degree with respect to jobs' completion times. Here, we present a new mathematical programming model for the first time. Since the problem is NP-hard for solving the addressed problem, a new modified imperialist competitive algorithm (MICA) is proposed to obtain good solution results. This algorithm consists of a new assimilation policy (i.e., colonies moving) and imperialistic competition procedure to balance between exploration and exploitation of the original ICA and improve its performances.

3 Fuzzy mathematical model and problem descriptions

3.1 Deterministic model

In this section, the SBPM problem for minimizing total earliness–tardiness penalties is formulated based on the following assumptions:

- No batch can be interrupted or stopped prematurely while it is being processed.
- The buffer between the machines is unlimited.
- Jobs cannot be added to or removed from a batch once processing has been started.
- The processing times of jobs and a size of a job are known as deterministic and non-identical.

The above-mentioned assumptions reflect the actual practice in the manufacturing environment, and also, similar assumptions were considered in most of the existing literature of the studied problem. The objective of this problem is to minimize the total weighted earliness–tardiness penalties. There are n jobs to be processed, and each job $j \in J$ has a processing time p_j and a corresponding size s_j. The total size of all the jobs in a batch does not exceed machine capacity S. The processing time of a batch b is given by the longest job in the batch (i.e., $P^b = max\left\{p_j | j \in batch\ b\right\}$). The formulation is as follows:

Notations:

Sets:

J Jobs, $j \in J$
B Batches, $b \in B$

Parameters:

p_j Processing time of job j
s_j Size of job j
cap Machine capacity
α_j Earliness penalty (/unit/h) of job j
β_j Tardiness penalty (/unit/h) of job j
d_j Due date of job j

Decision variables:

X_{jb} A binary variable indicates the assignment of job j to batch b
p^b Processing time of batch b
c_j Completion time of job j
C^b Completion time of batch b
E_j Earliness of job j
T_j Tardiness of job j

The objective function consists of two sub-functions: earliness and tardiness. At first, the mixed integer linear program (MILP) model proposed by [34] for total weighted tardiness sub-function is presented, and then, a mathematical model of the total weighted earliness–tardiness penalties of jobs is developed. According to the mentioned sets, parameters, and decision variables, the mathematical formulation of the total weighted tardiness penalties of jobs can be written as follows:

$$\text{Min } Z = \sum_{j \in J} \beta_j T_j s_j \tag{1}$$

s.t:

$$\sum_{b \in B} X_j b = 1 \quad \forall\ j \in J \tag{2}$$

$$\sum_{j \in J} s_j X_{jb} \leq cap \quad \forall\ b \in B \tag{3}$$

$$p^b \geq p_j X_{jb} \quad \forall j \in j, \forall\ b \in B \tag{4}$$

$$C^b = \sum_{i=1}^{b} p^i \quad \forall\ b \in B \tag{5}$$

$$c_j \geq C^b - M\left(1 - X_{jb}\right) \quad \forall\ j \in J,\ \forall\ b \in B;\ M \text{ is a very large positive number} \tag{6}$$

$$T_j \geq c_j - d_j \quad \forall\ j \in J \tag{7}$$

$$X_{jb} \in \{0, 1\} \quad \forall\ j \in j,\ \forall\ b \in B \tag{8}$$

Constraint set (2) ensures that each job can be processed in only one batch. Constraint set (3) also ensures that the machine capacity is not exceeded when jobs are assigned to a batch. Constraint set (4) states that the processing time of a batch is the longest processing time among all the jobs in that batch. Completion time of each batch is determined in constraint set (5). Also, constraint set (6) defines the completion time of each job. Constraint set (7) defines the tardiness of a

job as the difference between the due date of a job and its completion time or 0 if it is negative. Constraint set (8) specifies the type of decision variable X_{jb}.

Also, to determine the completion time of each batch, constraint sets (9) and (10) or constraint sets (11) and (12) can be used instead of constraint sets (5).

$$C^1 \geq P^1 \tag{9}$$

$$C^b \geq P^b + C^{b-1} \quad b = 2, \ldots, n \tag{10}$$

$$C^1 = P^1 \tag{11}$$

$$C^b = P^b + C^{b-1} \quad b = 2, \ldots, n \tag{12}$$

Due to minimization of just only tardiness or total weighted tardiness penalties in the objective function, the model chooses the minimum P^b in the constraint sets (4) to reach the longest processing time among all the jobs in that batch. The smaller the completion time of jobs, the more desirable the objective function. Similarly, the model finds the minimum c_j and T_j in the constraint sets (6), (7), (9), and (10).

For the objective function of $\sum_{j \in J} (\beta_j T_j + \alpha_j E_j) s_j$, in addition to constraint set (7), the constraint sets (13) and (14) are needed to calculate the earliness and tardiness of jobs.

$$E_j \geq d_j - c_j \quad \forall\ j \in J \tag{13}$$

$$c_j + E_j - T_j \quad \forall\ j \in J \tag{14}$$

Contrary to tardiness, earliness of jobs decreases when the completion time of jobs increases. So, the above model may not choose the minimum P^b and c_j in the constraint sets (4), (6), (9), and (10) in some situation. For example, consider a simple example of four jobs with machine capacity equal to 2. The other information is given in Table 1. The above model chooses incorrect completion times equal to 3, 3, 5, and 5, respectively, to minimize both earliness and tardiness in the objective function while the correct ones are equal to 2, 2, 4, and 4.

So, to tackle the dilemma, for the objective function with total weighted earliness–tardiness penalties of jobs, the nonlinear constraint sets (15) and (16) should be used instead of linear constraint sets (4) and (6).

$$p_b = max\left(\forall\ j : p_j X_{jb}\right) \quad \forall\ b \in B \tag{15}$$

$$c_j = \sum_{b=1}^{n} X_{jb} C^b \quad \forall\ j \in J \tag{16}$$

These two constraint sets (13) and (14) make the model more complex making it a mixed integer nonlinear program (MINLP). Also, the result of LINGO codes of these two models which is presented in appendices 1 and 2 confirms this dilemma and is presented in Table 2. In order to understand the computational difficulties, the proposed mathematical model is implemented in LINGO solver, to study the model's behavior when the number of jobs increases. For this purpose, required computational time is reported in Table 3. From Table 3, it is observed that the solving time increases very quickly as the problem size (number of jobs) grows. This can also be observed from Fig. 1. This means that the time required to solve even a 14-job problem can easily rise to more than 3 h.

Table 1 Example parameters

Job	1	2	3	4
Processing time	2	2	2	2
Due date	3	3	5	5
Job size	1	1	1	1
α	1	1	1	1
β	2	2	2	2

3.2 Fuzzy model

We briefly introduce some basic concepts and results about fuzzy measure theory initiated by Bellman and Zadeh [4]. Below, we give definitions and notations taken from Bezdek [5].

Definition 3.1 If X is a collection of objects denoted generically by x, then a fuzzy set in X is a set of ordered pairs:

$\tilde{d} = \left\{x, \tilde{d}(x) | x \in X\right\}$, where $\tilde{d}(x)$ is called the membership function that associates with each $x \in X$ a number in $[0, 1]$ indicating to what degree x is a number.

Definition 3.2 $\tilde{d}_j = \left(d_{j,1}, d_{j,2}, d_{j,3}, d_{j,4}\right)$ denotes a trapezoidal fuzzy number (TFN) as shown in Fig. 2.

As mentioned in the literature, the concept of fuzzy due dates has been used in scheduling problems. Here, this concept is being firstly utilized in the BPM scheduling problem. In a fuzzy due date, the membership function assigned to each job represents the customer satisfaction degree for the delivery or

Table 2 Result of LINGO codes of the two MILP and MINLP models

Model	MILP model				MINLP model			
Job	1	2	3	4	1	2	3	4
Batch	1	1	2	2	1	1	2	2
Completion time	3	3	5	5	2	2	4	4
Earliness	0	0	0	0	1	1	1	1
Tardiness	0	0	0	0	0	0	0	0

Table 3 Computational complexity

No. of jobs	4	5	6	7	8	9	10	11	12	13	14
Time (s)	<1	<1	<1	<1	1	1	8	185	1148	1908	>3 h

completion time of that job. The membership function of a trapezoidal fuzzy due date of a job as a generalized triangular fuzzy due date is represented below.

$$\mu_j(C_j) = \begin{cases} 0 & if \quad c_j < d_{j,1} \\ \frac{c_j - d_{j,1}}{d_{j,2} - d_{j,1}} & if \quad d_{j,1} \le c_j \le d_{j,2} \\ 1 & if \quad d_{j,2} \le c_j \le d_{j,3} \\ \frac{d_{j,4} - c_j}{d_{j,4} - d_{j,3}} & if \quad d_{j,3} \le c_j \le d_{j,4} \\ 0 & if \quad c_j > d_{j,4} \end{cases} \tag{17}$$

From Fig. 2, we can see that the full satisfaction (i.e., $\mu_j(C_j) = 1$) is attained if $d_{j,2} \le c_j \le d_{j,3}$, and the satisfaction grade is positive if $d_{j,1} \le c_j \le d_{j,2}$ or $d_{j,3} \le c_j \le d_{j,4}$ in the membership function (1). If $d_{j,2} = d_{j,3}$, the fuzzy trapezoidal due date is called triangular fuzzy due date and can be denoted by triplet $\tilde{d}_j = (d_{j,1}, d_{j,2}, d_{j,3})$. The membership function of a triangular fuzzy number is as follows.

$$\mu_j(C_j) = \begin{cases} 0 & if \quad c_j < d_{j,1} \\ \frac{c_j - d_{j,1}}{d_{j,2} - d_{j,1}} & if \quad d_{j,1} \le c_j \le d_{j,2} \\ \frac{d_{j,3} - c_j}{d_{j,3} - d_{j,2}} & if \quad d_{j,2} \le c_j \le d_{j,3} \\ 0 & if \quad c_j > d_{j,3} \end{cases} \tag{18}$$

In a particular situation (e.g., just-in-time production system), the full satisfaction is not attained if a completion time is too early or tardy. According to the mentioned fuzzy due date, the studied problem can be formulated as a maximization problem of the total degree of satisfaction over given jobs or, equivalently, a minimization problem of

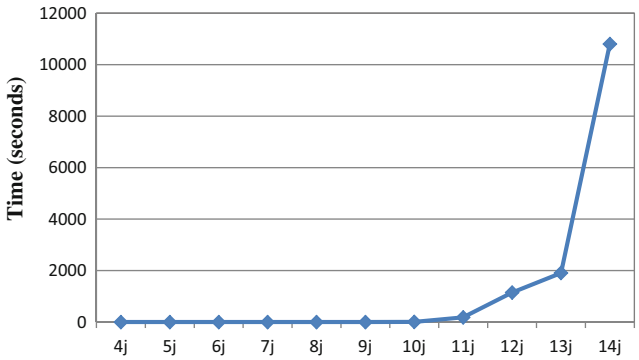

Fig. 1 Computational complexity

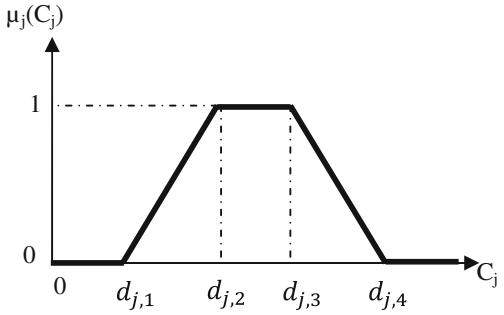

Fig. 2 Trapezoidal membership function

the total degree of dissatisfaction. The fuzzy mathematical formulation of the total degree of satisfaction, considering a trapezoidal fuzzy due date, is as follows.

Model 1. Max Z

$$\begin{aligned} &= \sum_{j \in J} s_j \left(\left(\frac{max\ (0, c_j - d_{j,1})}{c_j - d_{j,1}} \right) \left(\frac{max\ (0, d_{j,2} - c_j)}{d_{j,2} - c_j} \right) \left(\frac{(c_j - d_{j,1})}{d_{j,2} - d_{j,1}} \right) \right. \\ &+ \left(1 - \left(\frac{max\ (0, d_{j,2} - c_j)}{d_{j,2} - c_j} \right) \right) \left(1 - \left(\frac{max\ (0, c_j - d_{j,3})}{c_j - d_{j,3}} \right) \right) \\ &+ \left. \left(\frac{max\ (0, c_j - d_{j,3})}{c_j - d_{j,3}} \right) \left(\left(\frac{max\ (0, d_{j,4} - c_j)}{d_{j,4} - c_j} \right) \left(\frac{(d_{j,4} - c_j)}{d_{j,4} - d_{j,3}} \right) \right) \right) \end{aligned} \tag{19}$$

As mentioned above, equivalent to the model 1, the fuzzy mathematical formulation of the total degree of dissatisfaction is as follows.

Model 2. $Min\ Z = \sum_{j \in J} s_j \left(\frac{max(0, d_{j,1} - c_j)}{d_{j,1} - c_j} \right)$

$$\begin{aligned} &+ \sum_{j \in J} s_j \left(1 - \left(\frac{max(0, c_j - d_{j,1})}{c_j - d_{j,1}} \right) \left(\frac{max(0, d_{j,2} - c_j)}{d_{j,2} - c_j} \right) \left(\frac{(c_j - d_{j,1})}{d_{j,2} - d_{j,1}} \right) \right) \\ &+ \sum_{j \in J} s_j \left(1 - \left(\frac{max(0, c_j - d_{j,3})}{c_j - d_{j,3}} \right) \left(\frac{max(0, d_{j,4} - c_j)}{d_{j,4} - c_j} \right) \left(\frac{(d_{j,4} - c_j)}{d_{j,4} - d_{j,3}} \right) \right) \\ &+ \sum_{j \in J} s_j \left(\frac{max(0, c_j - d_{j,4})}{c_j - d_{j,4}} \right) \\ &= \sum_{j \in J} s_j \left(\frac{max(0, d_{j,1} - c_j)}{d_{j,1} - c_j} \right) \\ &+ \sum_{j \in J} s_j \left(\left(\frac{max(0, c_j - d_{j,1})}{c_j - d_{j,1}} \right) \left(\frac{max(0, d_{j,2} - c_j)}{d_{j,2} - c_j} \right) \left(\frac{(d_{j,2} - c_j)}{d_{j,2} - d_{j,1}} \right) \right) \\ &+ \sum_{j \in J} s_j \left(\left(\frac{max(0, c_j - d_{j,3})}{c_j - d_{j,3}} \right) \left(\frac{max(0, d_{j,4} - c_j)}{d_{j,4} - c_j} \right) \left(\frac{(c_j - d_{j,3})}{d_{j,4} - d_{j,3}} \right) \right) \\ &+ \sum_{j \in J} s_j \left(\frac{max(0, c_j - d_{j,4})}{c_j - d_{j,4}} \right) \end{aligned} \tag{20}$$

Since each customer has a different degree of importance as a decision maker, different earliness tardiness weights or

penalties are considered which is the same as the extreme majority of related papers. Therefore, here, we extend the previous model (i.e., Eq. 11) and reformulate it with earliness tardiness penalties shown below.

$$\begin{aligned}\textbf{Model 3}.\ \text{Min}\ Z = &\sum_{j\in J}\alpha_j s_j\left(\frac{max\ \left(0,d_{j,1}-c_j\right)}{d_{j,1}-c_j}\right)\\ &+\sum_{j\in J}\alpha_j s_j\left(\left(\frac{max\ \left(0,c_j-d_{j,1}\right)}{c_j-d_{j,1}}\right)\left(\frac{max\ \left(0,d_{j,2}-c_j\right)}{d_{j,2}-c_j}\right)\left(\frac{\left(d_{j,2}-c_j\right)}{d_{j,2}-d_{j,1}}\right)\right)\\ &+\sum_{j\in J}\beta_j s_j\left(\left(\frac{max\ \left(0,c_j-d_{j,3}\right)}{c_j-d_{j,1}}\right)\left(\frac{max\ \left(0,d_{j,4}-c_j\right)}{d_{j,4}-c_j}\right)\left(\frac{\left(c_j-d_{j,3}\right)}{d_{j,4}-d_{j,3}}\right)\right)\\ &+\sum_{j\in J}\beta_j s_j\left(\frac{max\ \left(0,c_j-d_{j,4}\right)}{c_j-d_{j,4}}\right)\end{aligned} \tag{21}$$

4 Imperialist competitive algorithm

The ICA is a novel socio-politically algorithm based on imperialistic competition [1]. The ICA has been introduced for dealing with different optimization problems. This evolutionary optimization strategy has shown great performance in both convergence rate and better global optima achievement [1, 2, 42, 43].

The ICA starts with an initial population of solution named country, like other evolutionary algorithms. Some of the best countries in the population are chosen to be the "imperialists," and the rest is the "colonies" of these imperialists. All the colonies of initial population are distributed among the imperialists based on their power. A set of one imperialist and its colonies is called an "empire." Therefore, in an empire, there are one imperialist and several colonies, in which colonies start moving toward their relevant imperialist countries. The total power of an empire depends on both the power of the imperialist country and the power of its colonies. So, the total power of an empire is calculated by adding the percentage of the mean power of colonies to their imperialists.

The imperialistic competition begins among all the empires, and every empire that is not strong enough to compete and cannot increase its power (or at least prevent decreasing it) will be eliminated. The imperialistic competition will lead slightly to an increase in the power of powerful empires and a decrease in the power of weaker ones. Weak empires will lose their power, and finally, they will collapse. The movement of colonies toward their relevant imperialists through the competition among empires and also the collapse mechanism will hopefully cause all the countries to converge to a state in which there is just one empire in the world and all the other countries are colonies of that empire. In this ideal new world, colonies have the same position and power as the imperialist.

4.1 Imperialist competitive algorithm approach to solve the single batch-processing machine (

The ICA starts with an initial population, in which each individual in the population is called country and refers to a candidate solution to a problem. The countries are divided into imperialist states and colonies. Moving colonies toward their relevant imperialist refers the assimilation policy. Imperialistic competition as the main part of the algorithm, accompanied with moving colonies toward their relevant imperialist, causes the colonies to converge to the global minimum of the cost function. The original steps of ICA, firstly proposed by Atashpaz-Gargari and Lucas [1], are as follows.

Begin ICA

1. Initialize the empires.
2. Move the colonies toward their relevant imperialist (i.e., assimilating).
3. If there is a colony in an empire which has lower cost than that of imperialist, exchange the positions of that colony and the imperialist.
4. Compute the total cost of all empires (related to the power of both imperialist and its colonies).
5. Pick the weakest colony from the weakest empire and give it to the empire that has the most likelihood to possess it (i.e., imperialistic competition).
6. Eliminate the empire that has no colonies.
7. If the stopping criteria met, stop the algorithm; if not, go to step 2.

End ICA

In this paper, we utilize the basic idea and steps of the above-mentioned ICA for the problem. Besides, we propose and use some new ideas detailed in the following steps.

4.2 Generating initial empires

Generally, the main purpose of optimization is to find an optimal solution; each solution in this algorithm is shown as a country. A candidate solution is represented as an array. The term country in the ICA stands for a chromosome in the GA, particle in particle swarm optimization (PSO), antibody in artificial immune algorithm (AIA), and the like.

As mentioned earlier in the literature, the random key (RK) method is used for solving BPM scheduling problems. To generate a sequence by this method, random real numbers between zero and one are generated for each job. By ascending sorting of the value corresponding to each job, the job sequence is obtained, and then, the first–first (FF) heuristic is applied to group the jobs into batches.

After having a permutation and forming the batches, we can use it to compute the objective function value of this solution. Each job has a random real number between 0 and 1, and these numbers show the relative order of the jobs. In fact, the problem variables in the algorithms are limited between 0 and 1. For example, consider a problem with ten jobs. The encoding of this sequence through the RKs is shown in Fig. 3. The sequence at position 1 is 2, which means that we schedule job 2 in the beginning position and job 8 at last.

Optimization of an algorithm starts with generating initial population (i.e., countries). Here, let us define N as the population size that is equal to N_{imp} (i.e., number of the most powerful countries selected to be as imperialists) and N_{col} (i.e., the remaining population that will be the colonies belonging to the imperialists). Thus, we have two types of countries (i.e., imperialist and colony) and we can restate that $N=N_{imp}+N_{col}$.

The cost of each country is evaluated by the cost function f and can be shown by C_i as follows:

$$C_i = f(country_i)$$

In this paper, the cost function is to calculate the objective function of the given problem. In other words, the cost of each country as a solution is equal to its objective function. To divide the colonies among imperialists, the normalized cost and the normalized power of each imperialist are shown by NC_i and NP_i and defined below for the nth imperialist, respectively:

$$NC_n = \max\{C_i\} - C_n$$
$$N\,P_n = \frac{NC_n}{\sum_{i=1}^{N_{imp}} NC_i}$$

The more the cost of an imperialist, the lower the normalized cost and its normalized power will be. The initial number of colonies of each imperialist depends on imperialist's normalized power, and it is defined by the following:

$$NCol_n = \text{round}\{NP_n \times N_{col}\}$$

$NCol_n$ is the initial number of colonies of the ith empire. To form each empire, $NCol_n$ of the colonies is randomly selected and given to each imperialist. Therefore, the most powerful empire has the greatest number of colonies. The nth imperialist and its colonies will form the nth empire. Figure 4 shows the initial population of each empire. As depicted, the bigger an empire is, the greater number of colonies it has. In this figure, imperialist 1 has formed the most powerful empire and has the greatest number of colonies.

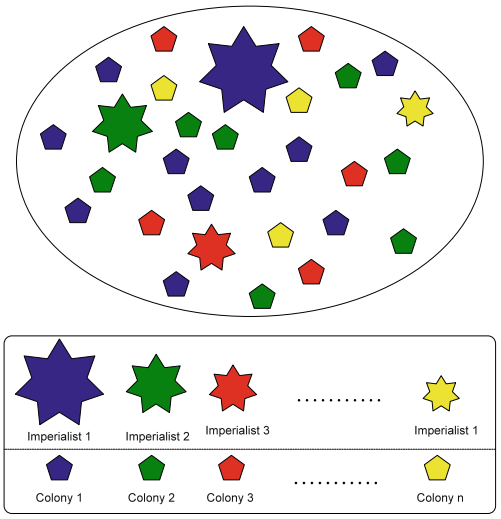

Fig. 4 Generating the initial empires; the more colonies an imperialist possesses, the bigger its relevant mark

4.3 Moving the colonies of an empire toward the imperialist (assimilating)

In the original ICA proposed by Atashpaz-Gargari and Lucas [1], the assimilation policy is happened by moving colonies toward their relevant imperialist. They stated that "imperialist countries improve their countries," and because of this, the assimilation policy will occur. This fact has been modeled by moving all the colonies toward the imperialist. Through this movement, some parts of a colony's structure will be similar to the empire's structure. This similarity is the concept of intensification or local search in metaheuristics and intensifies searching on the imperialist's neighbor. In fact, some regions of the solution space may be unexplored during the search process. Therefore, the assimilation policy has an important role in convergence rate as well as the probability of being trapped in local optimum. The trade-off between these two concepts in the ICA has not been shown and is not clear. In the other words, in long term running of the algorithm, trade-off between diversification and intensification leads to reach the global optimum and the rate of these two concepts should be controllable to understand the behavior of the algorithm.

Fig. 3 Representation encoding for a candidate solution

Before

Jobs	1	2	3	4	5	6	7	8	9	10
RKs	0.23	0.18	0.38	0.87	0.53	0.76	0.46	0.93	0.36	0.84

After

Jobs sequence	2	*1*	*9*	*3*	*7*	*5*	*6*	*10*	*4*	*8*
RKs	0.18	0.23	0.36	0.38	0.46	0.53	0.76	0.84	0.87	0.93

Here, we propose an idea to make sure that the system will not be trapped in local optimum and is controllable by changing or tuning some parameters. We want to use an idea in this step that is most probably taking place in the real world. In the real world, any change or improvement in a colony, not only depends on or toward to the relevant imperialist, but also depends on other imperialists or even other colonies in that empire, may affect the changes on the colony over the time. In other words, the relevant imperialist is not the only country which changes its colony in the name of improvement. Each colony changes over the time, and this change may be the result of having relationship with other colonies in the same empire or having strategies to reach the main characteristics of other empires. In this way, in order to explore more regions, a diversification strategy alongside intensification is applied such that colonies are also allowed to move in other empires and their self-neighbors. Another aspect of this real idea tells us that "distance" is the main factor which affects the changes in a colony. The term distance may stand for the similarity rate of two countries, and often, the lower distance between two countries leads to a greater similarity rate. Simply put, the less the distance of a colony with other colonies in the same empire, or the less distance between a colony with other imperialist countries, the more relationship they have, and it leads to have greater impact.

So, we utilize this real thought in the assimilating step of the ICA. As mentioned earlier, empire i has $NCol_i$ countries. Each country in an empire moves to the relevant empire, the countries in the same empire, and also other imperialist countries, according to the mentioned idea regarding the term distance or similarity. Let us start our explanation with a basic example and then extend it to the given problem.

Assume that we have just two solutions or countries and name them as the colony (i.e., country 1) and its relevant imperialist (i.e., country 2). The colony moves to the relevant imperialist over the time. The more power the imperialist has, the more changes or movement will be occurring from the colony toward the imperialist. We add two more countries as another imperialist (i.e., country 3) and another colony (i.e., country 4) in the same empire of the first colony. Now, the first colony deals with changes or movement that is resulted from the impacts of three countries. We can show these impacts and the movement by the following notation:

$$x_1^{\,*} = \alpha_1 \times x_1 + \alpha_2 \times x_2 + \alpha_3 \times x_3 + \alpha_4 \times x_4$$

where x_i is the ith country in our example, $x_1^{\,*}$ is the country 1 after movement, α_1 is the rate of maintaining the own characteristic of x_1, and $\alpha_i (i=2,3,4)$ stands for the rate of movement into other three countries. In addition, $0 \le \alpha_i \le 1$ and $\sum \alpha_i = 1$ for all of i.

The point is that a colony may be affected from more than three countries in a real world. So, if we consider the real world with more than two imperialists with some colonies in each empire, we can restate the movement by the following expression:

$$\begin{aligned} x_{col}^{\,*} = {} & \alpha_1 \times x_{col} + \alpha_2 \times x_{imp1} + \alpha_3 \\ & \times \left(\left(x_{imp2} + x_{imp3} + \ldots + x_{impN} \right) \Big/ \left(impN - 1 \right) \right) + \alpha_4 \\ & \times \left(\left(x_{col2} + x_{col3} + \ldots + \mathrm{x_{colM}} \right) \Big/ \left(colM - 1 \right) \right) \end{aligned}$$

where x_{col} is the colony number 1 out of the colonies in the empire number 1 out of N empires, which moves toward other country, x_{imp1} is the relevant imperialist of x_{col} in empire 1, $x_{imp2}+x_{imp3}+\ldots+x_{impN}$ are the other imperialists in the world, and $x_{col2}+x_{col3}+\ldots+x_{colM}$ are the other colonies in the empire 1.

As explained earlier, in this world, imperialists and the colonies in the same empire of a colony can affect the colony according to their powers and distances (i.e., similarity rates). Here, we are to employ these terms in the proposed movement step. The power in the minimization problem relates on the reverse objective function (OF) shown by w_i, and the distance of two countries can be calculated with the summation of absolute differences in the relevant characteristics of the two countries depicted by d_i.

In order to explain the term distance, an example is explained here. Suppose that we have two solutions (i.e., countries) and the length of the digits in each solution is equal to 5. Here, we show how the distance is calculated. x_1 moves toward x_2, and then, d_2 is evaluated as follows:

x_1	0.26	0.54	0.61	0.37	0.19
x_2	0.89	0.72	0.09	0.44	0.32

$$\begin{aligned} d_2 &= |0.26-0.89| + |0.54-0.72| + |0.61-0.09| + |0.37-0.44| + |0.19-0.32| \\ &= 0.63 + 0.18 + 0.52 + 0.07 + 0.13 = 1.53 \end{aligned}$$

The impact of an imperialist in attracting a colony can be depicted by γ_i, and the impact of a colony in attracting another colony in the same empire is shown by η_i. They are calculated by the following fraction:

$$\gamma_i = \eta_i = \frac{\dfrac{1}{\dfrac{OF_i}{d_i}}}{\sum_i \dfrac{1}{\dfrac{OF_i}{d_i}}} = \frac{\dfrac{w_i}{d_i}}{\sum_i \dfrac{w_i}{d_i}}$$

By the above notation, we employ the terms power and distance. We can declare the movement function according to the power and distance for a colony by the following expression:

$$x_{col}^{*} = \alpha_1 \times x_{col} + \alpha_2 \times x_{imp1} + \alpha_3 \times \left(\gamma_{imp2} \times x_{imp2} + \gamma_{imp3} \times x_{imp3} + \ldots + \gamma_{impN} \times x_{impN}\right) + \alpha_4 \times \left(\eta_{col2} \times x_{col2} + \eta_{col3} \times x_{col3} + \ldots + \eta_{colM} \times x_{colM}\right)$$

Here, we can explain the controllable diversification and intensification in this algorithm, as mentioned in the first sentences of this step. α_1 shows the rate of heritage in a solution from its own characteristic. The big α_2 will speed up the local search, because the colony moves directly toward its relevant imperialist. α_3 and α_4 let the colony to have chance in moving toward other mentioned countries or solutions. So, we can conclude that to further improve, we use an adaptive controller using four consolable factors that adapts the movement vector and balances both intensification and diversification mechanism. From this adaptive assimilation, we can enhance the ability of local optimum escaping and fast converging to global optimum.

Finally, according to other metaheuristics to give a stochastic characteristic to our proposed algorithm, we use random numbers r_i (i=1,2,3,4) in the interval [0, 1] by multiplying it to α_i as follows:

$$x_{col}^{*} = r_1 \times \alpha_1 \times x_{col} + r_2 \times \alpha_2 \times x_{imp1} + r_3 \times \alpha_3 \times \left(\gamma_{imp2} \times x_{imp2} + \gamma_{imp3} \times x_{imp3} + \ldots + \gamma_{impN} \times x_{impN}\right) + r_4 \times \alpha_4 \times \left(\eta_{col2} \times x_{col2} + \eta_{col3} \times x_{col3} + \ldots + \eta_{colM} \times x_{colM}\right)$$

4.4 Exchanging positions of the imperialist and a colony

Owing to movement toward the other mentioned countries, a colony may reach a position with lower cost than imperialist. In such a condition, the position of imperialist and colony is changed. After that, the algorithm will continue by the imperialist in a new position, and then, colonies start moving toward this position. Figure 5 depicts the position exchange between a colony and the imperialist. In Figs. 5 and 6, the best colony of the empire is shown in a darker color. This colony has a lower cost than that of the imperialist. Figure 6 shows the whole empire after exchanging the position of the imperialist and that colony.

4.5 Total power of an empire

The total power of each empire is the power of imperialist plus percentage of its colonies. It is defined by the total cost. TC_i is the total cost of the ith empire. It defines the total power as follows:

$$TC_n = cost(imperialist_n) + \xi \times \min\{cost(colonies\ \ of\ \ empire_n)\}$$

where TC_n is the total cost of the empire and ξ is a positive number considered to be less than 1. A little value for ξ causes

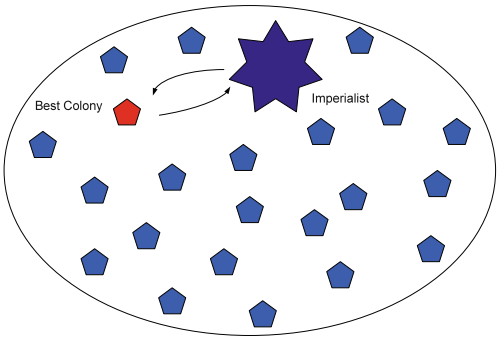

Fig. 5 Position exchange between a colony and the imperialist

the total power of the empire to be determined by just the imperialist, and increasing it will increase the role of the colonies in determining the total power of an empire. The above expression can be restated as follows:

$$TC_n = \lambda \times cost(imperialist_n) + (1-\lambda) \times \min\{cost(colonies\ \ of\ \ empire_n)\}$$

where λ and $1-\lambda$ show the impact of each of the two expressions and they are equal to the following:

$$\lambda = \frac{1}{1+\xi} \quad \text{and} \quad 1-\lambda = \frac{\xi}{1+\xi}$$

By this notation, we can depict the impact of each of the two expressions in determining the total cost of an empire clearly.

4.6 Imperialistic competition

As mentioned earlier, all empires attempt to own the other empires' colonies and manage them. The power of weaker empires will decrease, and as a result, the power of more powerful ones will increase in this imperialistic competition. This competition can be modeled by just picking one of the weakest colonies of the weakest empires and giving them to the empire that has most likelihood to possess them. Figure 7 illustrates the modeled imperialistic competition. Based on their total power, in this competition, each of the empires has a likelihood of taking possession of the mentioned colonies. In other words, these colonies are not possessed by the most powerful empires; however, these

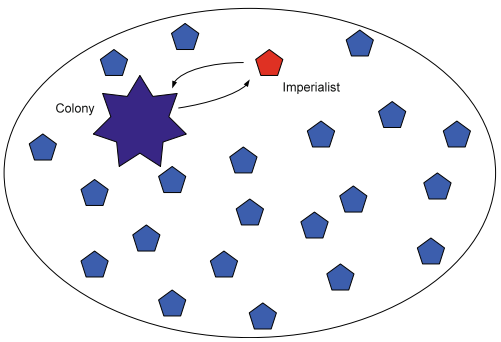

Fig. 6 Whole empire after exchanging the position of the imperialist and of the that colony

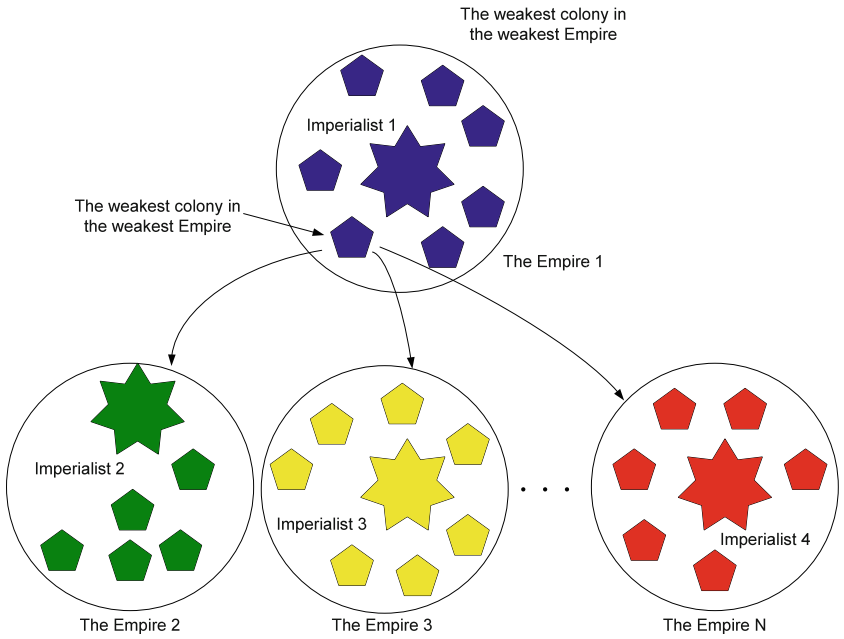

Fig. 7 Picking the weakest colony from the weakest empire and giving it to the empire that has the most likelihood to possess it

empires are more likely to possess them. To start the competition, first, the possession probability of each empire should be found based on its total power.

The normalized total cost is simply obtained by the following:

$$PTC_n = \max\{NC_i\} - TC_n$$

$$PP_n = \frac{NTC_n}{\sum_{i=1}^{N_{imp}} NTC_i}$$

where TC_n and NTC_n are the total cost and normalized total cost of the nth empire, respectively. Imperialist with the least cost has the most power. Then, the vector PP is formed as follows to divide the mentioned colonies among empires based on the possession probability of them.

$$PP_n = \left[PP_1, PP_2, PP_3, \ldots, PP_{\mathrm{N_{imp}}}\right]$$

The vector r, whose elements are uniformly distributed random numbers, is created with the same size as PP.

$$r_n - \left[r_1, r_2, r_3, \ldots, r_{N_{imp}}\right]$$
$$r_1, r_2, r_3, \ldots, r_{N_{imp}} \sim U(0, 1)$$

The vector D is formed as follows:

$$D = PP - r = \left[D_1, D_2, D_3, \ldots, D_{N_{imp}}\right]$$

An empire, whose relevant index in D is maximum, takes possession of mentioned colonies [1].

After assigning the weakest colony in the weakest empire in the real world, we may see some changes in that colony. We use this idea in the ICA, and let the colony change after this assignment. Let this colony (i.e., solution array) inherit some characteristics from itself, some other from its new relevant imperialist, and the remaining characteristics from revolution (i.e., new randomly generated numbers (characteristics)). For each characteristic in the colony (like gene in a chromosome in GA), we generate a random number r and follow this instruction for each characteristic, as depicted in the Fig. 8:

- If $r \leq \beta_1$, colony inherits the characteristic from itself.
- If $\beta_1 < r \leq \beta_2$, the same as the characteristic from the new relevant imperialist.
- If $r > \beta_2$, generate new randomly numbers in range [0, 1].

4.7 Eliminating the empire which has no colonies

Powerless empires will collapse in the imperialistic competition, and their colonies will be distributed among other empires. In modeling collapse mechanism, different factors can be defined for considering an empire powerless. In this paper, we assume an empire collapse when it loses all of its colonies.

4.8 Stopping criteria

In this paper, we consider two stopping criteria. The algorithm continues until no iteration is remaining or just one empire exists in the world.

The weakest colony:	0.83	0.81	0.25	0.49	0.16	0.61
The new relevant imperialist:	0.72	0.69	0.51	0.24	0.84	0.19
Random number (r): β_1=0.35, β_2=0.7	0.14	0.38	0.91	0.28	0.65	0.49
The changed and moved weakest colony:	0.83	0.69	0.15	0.49	0.84	0.19

Fig. 8 Changing in the structure of the moved colony in the new empire

Table 4 Test problem characteristics

Parameters	Levels
Number of jobs (N)	10, 20, 30, 50, 75, 100, 125, 150, 175 and 200
Processing time of jobs (P)	Uniform distributions [1, 10], [1, 20]
Size of jobs (S)	Uniform distributions [1, 10], [2, 4], [4, 8]
Earliness cost (E)	[1, 4]
Tardiness cost (T)	[5, 8]
Crisp due date of jobs (D)	Uniform distributions [round down $(1\text{-}L)\times BP$, round up $(1+H)\times BP$]
$d_{j,2}$	Uniform distributions $(0.9\times d_j, 0.95\times d_j)$
$d_{j,3}$	Uniform distributions $(1.05\times d_j, 1.1\times d_j)$
$d_{j,1}$	Uniform distributions $(0.4\times d_{j,2}, 0.6\times d_{j,2})$
$d_{j,4}$	Uniform distributions $(1.4\times d_{j,3}, 1.6\times d_{j,3})$

5 Computational experiments

5.1 Instances

To compare our algorithm and other solution methods, some test problems are needed. In this regard, we are going to generate the required data that can affect the performance of the algorithms including the number of jobs (n), range of processing time of jobs (p_j), size of jobs (s_j), range of earliness costs (α_i), tardiness costs (β_i), and due date of jobs (d_j) [11, 18, 34].

Ten different problem sizes for the number of jobs (i.e., $n \in \{10, 20, 30, 50, 75, 100, 125, 150, 175, 200\}$) are considered for the experimental study, which present different levels of difficulty for alternative solution methods. The processing times and size of jobs are generated randomly from uniform distributions between [1, 10] and [1, 20], respectively, and *Cap* considered 10 in all instances. Within each combination of N and P, three different types of job size (S) are employed from uniform distributions [1, 10], [2, 4], and [4, 8]. Hence, a number of problem configurations are equal to $10\times2\times3=60$.

Table 5 Factors and levels of proposed algorithms

Factors	No. of levels	MICA symbols	No. of levels	ICA symbols
Population size	3	A(1)—n A(2) —1.5 n A(3) —2 n	3	A(1)—n A(2)—1.5 n A(3) —2 n
Percentage of N_{imp}	3	B(1)—0.1 B(2)—0.15 B(3)—0.2	3	B(1)—0.1 B(2)—0.15 B(3)—0.2
$\alpha_1, \alpha_2, \alpha_3, \alpha_4$	13	C(1) —0.125, 0.125, 0.25, 0.5 C(2) —0.125, 0.125, 0.5, 0.25 C(3)—0.125, 0.25, 0.125, 0.5 C(4)—0.125, 0.25, 0.5, 0.125 C(5)—0.125, 0.5, 0.125, 0.25 C(6)—0.125, 0.5, 0.25, 0.125 C(7)—0.25, 0.125, 0.125, 0.5 C(8)—0.25, 0.125, 0.5, 0.125 C(9)—0.25, 0.25, 0.25, 0.25 C(10)—0.25, 0.5, 0.125, 0.125 C(11)—0.5, 0.125, 0.125, 0.25 C(12)—0.5, 0.125, 0.25, 0.125 C(13)—0.5, 0.25, 0.125, 0.125		—
λ	3	D(1)—0.25 D(2)—0.5 D(3)—0.75	3	C(1)—0.25 C(2)—0.5 C(3)—0.75
β_1, β_2	4	E(1)—0.25, 0. 5 E(2)—0.25, 0.75 E(3)—0.33, 0.66 E(4)—0.5, 0.75	3	—

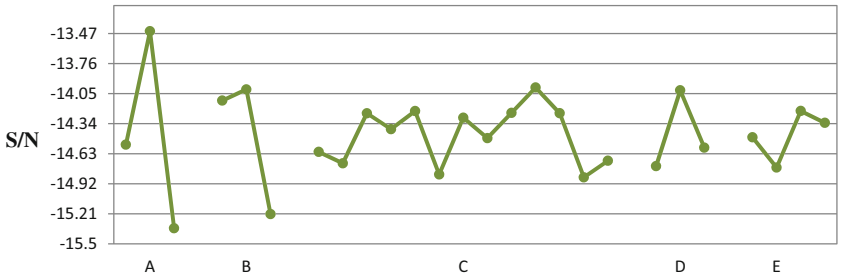

Fig. 9 Mean S/N ratio plot for each level of the factors in the MICA

The earliness cost and tardiness cost are assumed to be generated from uniform distribution over intervals [1, 4] and [4, 8], respectively. The crisp due dates in Tavakkoli-Moghaddam et al. [49] test problems are generated from a uniform distribution. We use such a procedure with some modifications to adapt the procedure for our problem as follows:

$$\overline{P} = \frac{\sum_{j=1}^{n} P_{ij}}{n} \tag{22}$$

$$\overline{B} = \frac{\sum_{j=1}^{n} s_j}{0.8 \times Cap} \tag{23}$$

$$BP = \overline{B} \times \overline{P} \tag{24}$$

First, the crisp due dates are generated from following distribution:

$$U \in (Round\ \ down(1-L) \times BP,\ \ Round\ \ up(1+H) \times BP) \tag{25}$$

where L and H are the lower and upper limits and are set to be 0.1 and 0.7, respectively. After generating the crisp due dates, each crisp due date is fuzzified in order to generate $d_{j,1}, d_{j,2}, d_{j,3}, d_{j,4}$ as explained in Table 4.

5.2 Parameter setting

It is known that the different levels of the parameters strongly affect the quality of the solutions obtained by a random search algorithm. Most users adjust parameters manually based on the reference values of the previous literature. Such trial-and-error method is time-consuming and ineffective, and often, it cannot locate the optimal combination. In this section, we investigate the behavior of the MICA and the ICA in different levels of parameters and find the optimal level of these parameters and operators.

The full factorial design of an experiment is a conventional statistical method used for calibration of parameters and operators. This method evaluates all the possible combinations of factors. Although the traditional full factorial method is the most widely used approach, this technique is not always efficient because its calculations become increasingly complex when the number of parameters is significantly high.

As it would be explained clearly later, there are three 3-level factors, one 4-level factors, and one 13-level factors for the MICA and three 3-level factors for the ICA. Moreover, there are 60 different test problems, and due to stochastic nature of the algorithms, ten replications are performed for each trial to achieve the more reliable results. Hence, the total number of running the problems for the MICA is $60 \times 3^3 \times 4 \times 13 \times 10 = 842{,}400$ trials. In this condition, to reduce the number of experiments and to be economic, several experimental design techniques are proposed.

The Taguchi optimization method is one of them that have been successfully applied to parameter tuning within recent years [36]. This method was first developed by Taguchi in 1960s as a system of cost-driven quality engineering that emphasizes on the effective application of engineering strategies. It uses an orthogonal array to organize the experimental results. To select the appropriate orthogonal array, it is necessary to compute the total degree of freedom.

In calibration of the MICA parameters, the proper array should contain a degree of freedom for the total mean, 2 degrees of freedom for each factor with three levels, 3 degrees of freedom for the factor with four levels, and 12 degrees of freedom for the factor with 13 levels. Thus, the sum of the required degrees of freedom is $1+2\times3+3\times1+12\times1=22$. Therefore, the appropriate array must have at least 22 rows. The selected orthogonal array should be able to accommodate the factor-level combinations in the experiment. Considering this, L64 (4^16 and16^1) is an appropriate array that satisfies these conditions. Since there are three factors with three levels and one

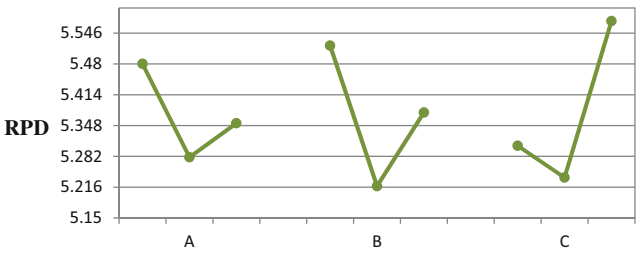

Fig. 10 Mean RPD plot for each level of the factors in the ICA

Table 6 Optimum parameters for the proposed ICA and MICA algorithms

Factors	ICA symbols	MICA symbols
Population size	1.5 n	1.5 n
Percentage of N_{imp}	0.15	0.15
α_1, α_2, α_3, α_4	–	0.25, 0.5, 0.125, 0.125
Λ	0.5	0.5
β_1, β_2	–	0.33, 0.66

factor with 13 levels and this scheme offers factors with four levels and 16 levels, respectively, we should adjust this array to the problem by means of adjustment techniques [38].

Using the dummy-level technique, we convert a 16-level column into a 13-level column and a 4-level column into a 3-level column. To assign the 4-level factor to the 3-level column and the 16-level factor to the 13-level column from the orthogonal array L64, some of these levels are required to be replicated twice. In this paper, the fourth level in a 4-level factor and three last levels in a 16-level factor are chosen to be replicated twice. The appropriate orthogonal array for the MICA is L64, respectively. It is clear that the total number of running for calibration of parameters of the MICA will be $60\times 64\times 10=38{,}400$ trials by the Taguchi method. On the contrary, the number of running in full factorial design equals to 842, 400, which is near to 22 times more.

Since there are three three-level factors for the ICA, the number of running in a full factorial design is only three times more and we use a full factorial design for the ICA. In the Taguchi method, the values of quality characteristics obtained through the experiments are transformed into a measure called signal-to-noise (S/N) ratio that is classified into three groups: the smaller-the-better, the larger-the-better, and nominal-is-best. Since the type of performance in this paper is the minimization, the corresponding smaller-the-better S/N ratio is as follows:

$$\mathrm{S}\Big/\mathrm{N\ Ratio} = -10\log_{10}(\mathrm{Performance\ Measure})^2$$

Table 8 Results of EDD on test problems

Problem	p1s1	p1s2	p1s3	p2s1	p2s2	p2s3
10j	109.72	62.02	101.38	78.99	74.65	117.39
20j	169.64	100.63	256.41	267.63	108.91	266.83
30j	293.04	180.87	308.66	319.6	186.96	462.12
50j	656.26	332.15	637.84	574.23	300.28	689.17
750j	1042.41	511.56	1024.06	1035.3	391.86	957.07
100j	1196.92	586.7	1333.85	1366	669.29	1475.62
125j	1621.03	771.61	1768.19	1462.93	693.85	1783.52
150j	1945.49	895.32	1996.06	2042.43	939.48	2126.96
175j	2476.1	1121.93	2334.82	2182.19	1063.96	2424.27
200j	2582.91	1308.25	2885.86	2554.33	1274.02	2725.09

The minimum variance of quality characteristics resulted from the S/N ratio is the optimal operator combination, which explains the reason why a parameter design is also called robust design. The parameters and operators of algorithms and their levels are depicted in Table 5.

As mentioned before, the experiments on the MICA and the ICA were based on the L64 orthogonal array and full factorial experiment, respectively. In order to be fair, the stopping criterion for all algorithms is equal to $6\times n$ milliseconds. This criterion is sensitive to the problem size. Using this stopping criterion, searching time increases according to the rise of number of jobs. The Taguchi experiments are performed for the MICA.

Because the scale of OFs in each instance is different, they cannot be used directly. In order to make the comparison easy and comprehensive, we use the relative percentage deviation (RPD) as a common and straightforward measure of comparing algorithms for each instance as follows.

$$\mathrm{RPD} = \frac{Alg_{sol} - Min_{sol}}{Min_{sol}} \times 100$$

where Alg_{sol} and Min_{sol} are the obtained objective value and the obtained best solution, respectively, for each replication of instance in a given test problem. Clearly, lower values of the RPD are preferable. The results of the experiments are transformed into the RPD. After converting the objective values to the RPDs,

Table 7 Best level of parameters for GA, SA, VNS, PSO, and ACO algorithms

Algorithm	Parameters and their best level				
GA	Population size=n,	Crossover=two-point,	Mutation=big swap,	Mutation probability (p_m)=0.1 Crossover percentage (p_c)=0.8	
SA	Initial temperature (T_0)=700,	n_{max}=300,	α=0.92		
VNS	Neighborhood structure=big swap, inversion, displacement			n_{max}=400	
PSO	Population size=1.5 n,	V_{min}=−4, V_{max}=+4, c_1=0.5, c_2=0. 5,	Decrement factor (α)=0.99,	Minimum of inertia weight=0.5 Maximum of inertia weight=0.8	
ACO	Population size=n,	Evaporation rate (ρ)=0.5,	Initial pheromone (τ_0)=1,	α=1,	β=3

Table 9 Improvements of algorithms in comparison with EDD as worst results (% equal to improvement %)

Problem	SA %	PSO %	GA %	ACO %	ICA %	MICA %	
10j	p1s1	58.24	38.3	35.83	46.54	*61.8*	56.55
	p1s2	63.13	55.85	55.77	65.93	69.77	*73.37*
	p1s3	51.26	45.41	*51.55*	48.07	43.06	47.98
	P2s1	27.72	27.67	28.27	37.22	32.4	*37.68*
	P2s2	61.80	52.01	55.32	60.13	57.63	*62.38*
	P2s3	*51.40*	43.4	46.31	41.59	40.2	41.79
20j	p1s1	29.48	36	33	46.97	42.55	*45.84*
	p1s2	56.14	57.24	53.39	52.88	57.11	*62.42*
	p1s3	28.82	24.37	13.32	42.59	45.31	*46.78*
	P2s1	*41.87*	29.76	40.61	36.38	34.72	39.85
	P2s2	41.22	43.32	47.31	50.29	53.13	*55.76*
	P2s3	*38.15*	24.35	28.81	34.19	35.54	35.41
30j	p1s1	41.12	26.54	40.27	41.14	55.13	*58.56*
	p1s2	66.39	65.79	*69.13*	68.06	66.22	63.92
	p1s3	24.41	32.07	39.3	38.95	37.71	*41.78*
	P2s1	30.63	29.77	42.05	*50.77*	48.03	50.63
	P2s2	47.24	42.94	53.07	55.34	52.76	*57.73*
	P2s3	38.52	40.02	*52.12*	48.2	45.23	46.36
50j	p1s1	25.64	27.8	27.84	27.59	27.53	*30.86*
	p1s2	30.78	31.11	26.51	27.85	*57.44*	51.87
	p1s3	26.41	32.79	42.27	40.88	41.66	*44.18*
	P2s1	*45.33*	45.2	40.74	39.69	40.1	38.26
	P2s2	31.16	40.6	36.28	40.26	47.33	*47.44*
	P2s3	28.28	31.18	35.56	*38.3*	31.33	36.84
75j	p1s1	13.79	17.73	16.47	19.44	29.38	*35.84*
	p1s2	12.88	10.73	21	30.5	32.53	*41.76*
	p1s3	26.96	27.29	28.62	25.66	29.17	*36.55*
	P2s1	20.94	26.52	23.08	31.15	28.8	*34*
	P2s2	31.93	35.09	36.95	35.81	39.96	*46.67*
	P2s3	23.4	25.33	26.86	27.23	38.14	*38.94*
100j	p1s1	12.74	8.66	9.85	24.33	29.81	*32.75*
	p1s2	17.49	23.48	36.44	27.64	*36.86*	38.31
	p1s3	28.82	27.69	31.39	*34.98*	30.58	32.29
	P2s1	12.89	19.21	16	15.3	28.85	*33.66*
	P2s2	25.19	27.72	32.29	31.04	35.02	*38.94*
	P2s3	5.03	14.59	12.09	27.64	31.62	*33.09*
125j	p1s1	17.59	20.27	18.95	29.07	27.01	*29.87*
	p1s2	20.63	22.29	30.61	*37.24*	35.94	34.97
	p1s3	−4.84	7.85	17.66	25.28	*34*	33.41
	P2s1	13.72	19.61	21.78	34.15	32.3	*35.53*
	P2s2	23.74	30.44	35.84	37.35	35.96	*39.29*
	P2s3	10.96	25.61	28.54	30.18	33.47	*33.76*
150j	p1s1	6.88	15.32	18.34	31.23	27.11	*30.28*
	p1s2	12.6	18.34	21.21	21.68	27.81	*32.8*
	p1s3	13.62	16.93	20.57	28	29.36	*33.16*
	P2s1	15.61	20.15	22.57	*31.71*	30.29	31.46
	P2s2	20.31	26.27	27.48	33.34	31.29	*36.04*
	P2s3	12.09	22.87	26.31	*36.49*	29.17	31.56
175j	p1s1	7.81	9.54	23.32	18.79	26.74	*27.06*

Table 9 (continued)

Problem	SA %	PSO %	GA %	ACO %	ICA %	MICA %	
	p1s2	7.36	16.7	30.94	28.85	34.47	*35.85*
	p1s3	16.45	17.97	24.58	17.92	26.27	*27.84*
	P2s1	2.35	5.33	11.92	21.16	24.98	*27.24*
	P2s2	16.17	21.17	21.56	22.74	30.51	*31.15*
	P2s3	10.09	15.39	18.37	19.75	25.87	*28.6*
200j	p1s1	11.37	19.25	18.67	26.39	27.51	*28.13*
	p1s2	20.76	31.17	32.38	31.9	32.24	*34.81*
	p1s3	11.12	16.84	21.45	24.08	*33.49*	33.14
	P2s1	9.97	4.46	16.2	20.56	19.74	*24.85*
	P2s2	17.35	22.65	28.39	31.65	31.46	*36.72*
	P2s3	8.54	15.38	18.16	20.24	24.83	*30.56*

the results are individually transformed into S/N ratios, and the ratios of trials are averaged in each level as shown in Fig. 9.

The full factorial experiments are performed for the ICA. The average of results is calculated and depicted in Fig. 10. The optimum levels of parameters and operators used in all algorithms are shown in Tables 6 and 7.

5.3 Experimental results

In this section, we present and compare the results of ICA and MICA with the SA, PSO, GA, and ACO algorithms as effective algorithms in BPM scheduling literature and with the EDD dispatching rule as a well-known heuristic related to due date. As mentioned above, we have 60 problem instances; each one includes ten performed replications to achieve the more reliable results. All instances are solved by all algorithms. Table 8 demonstrates the EDD results on test problems, in which the first column and first row represent the data set characteristics and the remaining columns show the results of the EDD on instances.

Since EDD showed the worst results in comparison to other metaheuristics, we calculated the improvement's percentage (i.e., improvement %) of each algorithm in comparison to EDD with a formula as follows:

$$\text{Improvement } \% = \frac{EDD_{sol} - Alg_{sol}}{EDD_{sol}} \times 100$$

Table 9 shows the improvements' percentage of each algorithm comparing to the EDD results in each instance. This table demonstrates the robust performance of the ICA and the MICA. Among algorithms, SA has smaller improvement. Other algorithms have better improvements, but in comparison to ACO, their improvements are small. Among SA, PSO, and GA, GA has better improvement and then PSO.

The averages of the RPD results are calculated for test problems (i.e., 60 data points per average for each algorithm) and are shown in Table 10. In this table, each row represents the average of results obtained for six test problems considered in each size of the large problem with ten replications for each algorithm.

Table 10 Average relative percentage deviation ($\overline{RPD}$) for the algorithms

Problem	SA	PSO	GA	ACO	ICA	MICA
10j	10.97	33.34	29.73	16.99	12.53	*6.81*
20j	19.37	24.63	24.92	10.23	7.94	*1.67*
30j	29.14	34.6	12.85	10.69	9.4	*4.86*
50j	24.86	18.57	19.07	17.35	5.06	*4.73*
75j	28.64	25.22	22.13	17.64	9.87	*0*
100j	28.11	23.13	18.57	13.08	4.88	*0.69*
125j	32.77	21.44	14.29	4.17	2.84	*0.75*
150j	30.2	20.41	16.26	4.73	6.67	*1.58*
175j	28.07	21.76	11.09	11.48	2.09	*0*
200j	26.76	19.02	12.91	8.26	4.68	*0.08*
$\overline{RPD}$	25.889	24.212	18.182	11.462	6.596	*2.117*

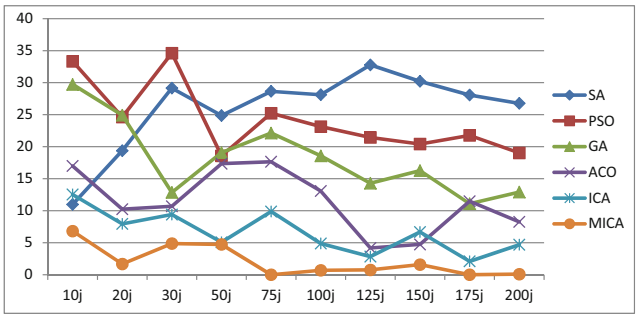

Fig. 11 Mean plot for the interaction among algorithms

According to the results, the average RPD obtained by the proposed MICA is 2.117, while for SA, PSO, GA, ACO, and ICA are 25.889, 24.212, 18.182, 11.462, and 6.596, respectively. The MICA and, after that, the ICA can find a better solution comparing to other algorithms for all the instances. Although SA, PSO, and GA have good results in comparison to EDD, the results of ACO are superior. Moreover, although the performance of the EDD rule is inferior in comparison to other algorithms, it produces tolerable solutions as a heuristic method.

To analyze the interaction between quality of the algorithms and different problem sizes, the average RPDs obtained by each algorithm are shown in Fig. 11. As it can be seen, the MICA and the ICA keep their robust performance in all the problem sizes. In the first problem size, SA has good results and better performance in comparison to even ICA, but with increasing size, gradually, other algorithms outperform it. In the first two sizes of problems, GA has not good results, but with increasing size, its RPD decreases and, in ninth size, outperforms ACO. In some sizes, ACO and ICA have been nearly competitive, but ACO has a superior performance in 125j and 150j. It is worth noting that because of studying a new and different problem from the previous works in this paper, a novel plan is utilized to generate the instances for the defined problem. So, here we cannot comprise the obtained results with other methods used in the literature.

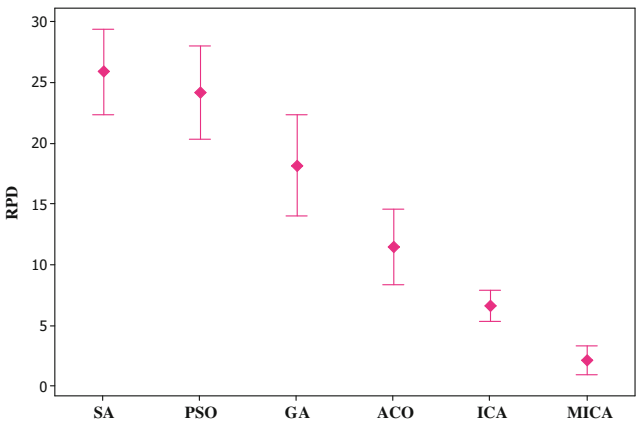

Fig. 12 LSD intervals among algorithms

Based on the results, we conclude that the proposed MICA and ICA can be used to effectively solve the BPM problems. Moreover, as it can be seen in the results, the performance of the MICA shows that modifying the colonies moving and imperialistic competition improves the performance of the ICA remarkably. In other words, embedding our suggestions and modifications in the ICA obviates mentioned drawbacks and makes balance between exploration and exploitation.

To verify the statistical validity of the results, we carry out the analysis of variance (ANOVA) technique to accurately analyze the results. The related results demonstrate that there is a clear statistically significant difference between performances of the algorithms. The mean plot and LSD intervals at the 95 % confidence level for all the algorithms are shown in Fig. 12. As it is shown, our proposed MICA provides statistically better results than ICA and others. SA and PSO both provide the statistically similar performance.

6 Conclusions and future research directions

In this paper, we modeled the SBPM scheduling problem considering a real-world application through fuzzy due dates to minimize the total weighted earliness and tardiness. At first, we converted the OF of the model to the total satisfaction or dissatisfaction degree, and then, a novel fuzzy mathematical model was developed. We also proposed a modified version of the ICA, named MICA, by introducing a new assimilation policy (moving of colonies) and imperialistic competition to balance between exploration and exploitation. Meanwhile, a new plan was presented to generate test data in a fuzzy environment. Then, in order to adjust the parameters and operators of the proposed algorithms, the relevant parameters are tuned. The computational results show that the proposed MICA is capable of obtaining better solutions compared to the ICA, SA, PSO, GA, and ACO in all problem sizes. Furthermore, the performance of the EDD, as a well-known dispatching rule related to the due date, was studied in SBPM scheduling problems.

Based on the results, we conclude that the proposing the new idea, by making balance between exploration and exploitation, can improve the performance of the ICA. So, we suggest to use the novel presented idea in ICA, named MICA, in the optimization areas. Besides, it can be extended to the case of scheduling with fuzzy processing times or other fuzzy parameters. In addition, considering multiple batch-processing machines (e.g., parallel machine, flow shop, open shop, and job shop) is encouraged.

Appendix 1 LINGO code of MILP model to solve the example.

```
model:
  sets:
  job/1..4/:p,s,c,d,e,t,alpha,beta;
  batch/1..4/:pbatch,cbatch;
  link1(job,batch):x;
  endsets
  data:
  p=2 2 2 2;
  s=1 1 1 1;
  d=3 3 5 5;
  alpha=1 1 1 1;
  beta=2 2 2 2;
  M=10000;
  cap=2;
  enddata
  min=@sum(job(j):(alpha(j)*e(j)+
beta(j)*t(j))*s(j));
  @for(job(j):@sum(batch(b):x(j,b))=1);
  @for(batch(b):@sum(job(j):s(j)*x(j,b))
<=cap);
  @for(batch(b):@for(job(j):pbatch(b)>=-
p(j)*x(j,b)));
  @for(batch(b):@sum(batch(i)|i#le#b:pb-
atch(i))=cbatch(b));
  @for(batch(b):@for(job(j):c(j)>=cbatc-
h(b)-M*(1-x(j,b))));
  @for(job(j):t(j)>=c(j)-d(j));
  @for(job(j):e(j)>=d(j)-c(j));
  @for(job(j):c(j)+e(j)-t(j)=d(j));
  @for(batch(b):@for(job(j):@bin(x(j,
b))));
  end
```

Appendix 2 LINGO code of MINLP model to solve the example.

```
model:
  sets:
  job/1..4/:p,s,c,d,e,t,alpha,beta;
  batch/1..4/:pbatch,cbatch;
  link1(job,batch):x;
  endsets
  data:
  p=2 2 2 2;
  s=1 1 1 1;
  d=3 3 5 5;
  alpha=1 1 1 1;
  beta=2 2 2 2;
  M=10000;
  cap=2;
  enddata
  min=@sum(job(j):(alpha(j)*e(j)+
beta(j)*t(j))*s(j));
  @for(job(j):@sum(batch(b):x(j,b))=1);
  @for(batch(b):@sum(job(j):s(j)*x(j,b))
<=cap);
  @for(batch(b):pbatch(b)=@max(job(j):-
p(j)*x(j,b)));
  @for(batch(b):@sum(batch(i)|i#le#b:pb-
atch(i))=cbatch(b));
  @for(job(j):@sum(batch(b):cbatch(b)*-
x(j,b))=c(j));
  @for(job(j):t(j)>=c(j)-d(j));
  @for(job(j):e(j)>=d(j)-c(j));
  @for(job(j):c(j)+e(j)-t(j)=d(j));
  @for(batch(b):@for(job(j):@bin(x(j,
b))));
  end
```

References

1. Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. IEEE Congr Evol Comput 4661–4667
2. Behnamian J, Zandieh M (2011) A discrete colonial competitive algorithm for hybrid flowshop scheduling to minimize earliness and quadratic tardiness penalties. Expert Syst Appl 38:14490–14498
3. Bellanger A, Oulamara A (2010) Minimizing total completion time on a batching machine with job processing time compatibilities. Electron Notes Discrete Math 36:1295–1302
4. Bellman R, Zadeh L (1970) Decision-making in a fuzzy environment. Manag Sci 17:141–164
5. Bezdek JC (1993) Fuzzy models—what are they, and why? IEEE Trans Fuzzy Syst 1:1–9
6. Chang PC, Wang HM (2004) A heuristic for a batch processing machine scheduled to minimise total completion time with non-identical job sizes. Int J Adv Manuf Technol 24:615–620
7. Chen H, Du B, Huang GQ (2011) Scheduling a batch processing machine with non-identical job sizes: a clustering perspective. Int J Prod Res 49:5755–5778
8. Cheng B, Li K, Chen B (2010) Scheduling a single batch-processing machine with non-identical job sizes in fuzzy environment using an improved ant colony optimization. J Manuf Syst 29:29–34
9. Chou FD, Wang HM (2008) A scheduling for a single semiconductor batch-processing machine to minimize total weighted tardiness. J Chin Inst Ind Eng 25:136–147
10. Chou FD, Chang PC, Wang HM (2006) A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem. Int J Adv Manuf Technol 31:350–359
11. Damodaran P, Manjeshwar PK, Srihari K (2006) Minimizing makespan on a batch processing machine with non-identical job sizes using genetic algorithms. Int J Prod Econ 103:882–891
12. Dupont L, Flipo CD (2002) Minimizing the makespan on a batch machine with nonidentical job sizes: an exact procedure. Comput Oper Res 29:807–819

13. Dupont L, Jolai Ghazvini F (1998) Minimizing makespan on a single batch processing machine with non-identical job sizes. Eur J Autom Syst 1998(32):431–440
14. Harikrishnan KK, Ishii H (2005) Single machine batch scheduling problem with resource dependent setup and processing time in the presence of fuzzy due date. Fuzzy Optim Decis Making 4:141–147
15. He CH, Lin Y, Yuan J (2007) Bicriteria scheduling on a batching machine to minimize maximum lateness and makespan. Theor Comput Sci 381:234–240
16. Hochbaum DS, Landy D (1997) Scheduling semiconductor burn-in operations to minimize total flowtime. Oper Res 45:874–885
17. Husseinzadeh Kashan A, Karimi B (2008) Scheduling a single batch-processing machine with arbitrary job sizes and incompatible job families: an ant colony framework. J Oper Res Soc 59:1269–1280
18. Husseinzadeh Kashan A, Karimi B, Jolai F (2006) Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with nonidentical job sizes. Int J Prod Res 44: 2337–2360
19. Husseinzadeh Kashan A, Karimi B, Jolai F (2010) An effective hybrid multi-objective genetic algorithm for bi-criteria scheduling on a single batch processing machine with non-identical job sizes. Eng Appl Artif Intell 23:911–922
20. Ikura Y, Gimple M (1986) Efficient scheduling algorithms for a single batch processing machine. Oper Res Lett 5:61–65
21. Ishii H, Tada M, Masuda T (1992) Two scheduling problems with fuzzy due-dates. Fuzzy Sets Syst 46:339–347
22. Jolai Ghazvini F (2005) Minimizing number of tardy jobs on a batch processing machine with incompatible job families. Eur J Oper Res 162:184–190
23. Jolai Ghazvini F, Dupont L (1998) Minimizing mean flow times criteria on a single batch processing machine with non-identical job sizes. Int J Prod Econ 55:273–280
24. Knutson K, Kempf K, Fowler J, Carlyle M (1999) Lot to order matching for a semiconductor assemble and test facility. IIE Trans 31:1103–1111
25. Koh SG, Koo PH, Kim DC, Hur WS (2005) Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. Int J Prod Econ 98:81–96
26. Kurz ME, Mason SJ (2008) Minimizing total weighted tardiness on a batch-processing machine with incompatible job families and job ready times. Int J Prod Res 46:131–151
27. Lee CY, Uzsoy R (1999) Minimizing makespan on a single batch processing machine with dynamic job arrivals. Int J Prod Res 37: 219–236
28. Lee CY, Uzsoy R, Martin-Vega LA (1992) Efficient algorithms for scheduling semiconductor burn-in operations. Oper Res 40:764–775
29. Li C, Lee C (1997) Scheduling with agreeable release times and due dates on a batch processing machine. Eur J Oper Res 96:564–569
30. Liu LL, Ng CT, Cheng TCE (2007) Scheduling jobs with agreeable processing times and due dates on a single batch processing machine. Theor Comput Sci 374:159–169
31. Liu LL, Ng CT, Cheng TCE (2010) On the complexity of bicriteria scheduling on a single batch processing machine. J Sched 13:629–638
32. Mathirajan M, Sivakumar AI (2006) Minimizing total weighted tardiness on heterogeneous batch processing machines with incompatible job families. Int J Adv Manuf Technol 28:1038–1047
33. Mathirajan M, Sivakumar AI (2006) A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. Int J Adv Manuf Technol 29:990–1001
34. Mathirajan M, Bhargav V, Ramachandran V (2010) Minimizing total weighted tardiness on a batch-processing machine with non-agreeable release times and due dates. Int J Adv Manuf Technol 48:1133–1148
35. Melouk S, Damodaran P, Chang PY (2004) Minimizing makespan for single batch processing machine with non-identical job sizes using simulated annealing. Int J Prod Econ 87:141–147
36. Molla-Alizadeh-Zavardehi S, Hajiaghaei-Keshteli M, Tavakkoli-Moghaddam R (2011) Solving a capacitated fixed-charge transportation problem by artificial immune and genetic algorithms with a Prüfer number representation. Expert Syst Appl 38:10462–10474
37. Mönch L, Unbehaun R, Choung YI (2006) Minimizing earliness–tardiness on a single burn-in oven with a common due date and maximum allowable tardiness constraint. OR Spect 28:177–198
38. Park SH (1995) Robust design and analysis for quality engineering. Chapman & Hall, London
39. Perez IC, Fowler JW, Carlyle WM (2005) Minimizing total weighted tardiness on a single batch process machine with incompatible job families. Comput Oper Res 32:327–341
40. Potts CN, Kovalyov MY (2000) Scheduling with batching: a review. Eur J Oper Res 120:228–249
41. RafieeParsa N, Husseinzadeh Kashan A, Karimi B (2010) A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes. Comput Oper Res 37:1720–1730
42. Rajabioun R, Atashpaz-Gargari E, Lucas C (2008) Colonial competitive algorithm as a tool for Nash equilibrium point achievement. Lect Notes Comput Sci 5073:680–695
43. Rajabioun R, Hashemzadeh F, Atashpaz-Gargari E, Mesgari B, Rajaei Salmasi F (2008b) Identification of a MIMO evaporator and its decentralized PID controller tuning using colonial competitive algorithm (pp. 11–12).IFAC World Congress
44. Sabouni MTY, Jolai F (2010) Optimal methods for batch processing problem with makespan and maximum lateness objectives. Appl Math Model 34:314–324
45. Sevaux M, Peres SD (2003) Genetic algorithms to minimize the weighted number of late jobs on a single machine. Eur J Oper Res 151:296–306
46. Sung CS, Choung YI (2000) Minimizing makespan on a single burn-in oven in semiconductor manufacturing. Eur J Oper Res 120:559–574
47. Sung CS, Choung YI, Hong JM, Kim YH (2002) Minimizing makespan on a single burn-in oven with job families and dynamic job arrivals. Comput Oper Res 29:995–1007
48. Tangudu SK, Kurz ME (2006) A branch and bound algorithm to minimise total weighted tardiness on a single batch processing machine with ready times and incompatible job families. Prod Plan Control 17:728–741
49. Tavakkoli-Moghaddam R, Rahimi-Vahed A, Mirzaei AH (2007) A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: weighted mean completion time and weighted mean tardiness. Inf Sci 177:5072–5090
50. Uzsoy R (1994) Scheduling a single batch-processing machine with non-identical job sizes. Int J Prod Res 32:1615–1635
51. Uzsoy R (1995) Scheduling batch processing machines with incompatible job families. Int J Prod Res 33:2685–2708
52. Uzsoy R, Yang Y (1997) Minimizing total weighted completion time on a single batch processing machine. Prod Oper Manag 6:57–73
53. Uzsoy R, Lee C, Martin-Vega L (1992) A review of production planning and scheduling models in the semiconductor industry, part I: system characteristics, performance evaluation and production planning. IIE Trans 24:47–60
54. Uzsoy R, Lee C, Martin-Vega L (1994) A review of prod planning and scheduling models in the semiconductor industry, part II: shop floor control. IIE Trans 26:44–55
55. Wang J (2004) A fuzzy robust scheduling approach for product development projects. Eur J Oper Res 152:180–194
56. Wang HM (2011) Solving single batch-processing machine problems using an iterated heuristic. Int J Prod Res 49:4245–4261

57. Wang CS, Uzsoy R (2002) A genetic algorithm to minimize maximum lateness on a batch processing machine. Comput Oper Res 29:1621–1640
58. Wang HM, Chang PC, Chou FD (2007) A hybrid forward/backward approach for single batch scheduling problems with non-identical job sizes. J Chin Inst Ind Eng 24:191–199
59. Yao S, Jiang Z, Li N (2012) A branch and bound algorithm for minimizing total completion time on a single batch machine within compatible job families and dynamic arrivals. Comput Oper Res 39:939–951
60. Yimer AD, Demirli K (2009) Fuzzy scheduling of job orders in a two-stage flowshop with batch-processing machines. Int J Approx Reason 50:117–137