Computers & Operations Research 41 (2014) 31–43

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Minimizing total tardiness and earliness on unrelated parallel machines with controllable processing times

Vahid Kayvanfar [a], GH.M. Komaki [b], Amin Aalaei [a], M. Zandieh [c,*]

[a] Department of Industrial Engineering, Amirkabir University of Technology, 424 Hafez Ave., 15916-34311 Tehran, Iran
[b] Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH44106-7070, USA
[c] Department of Industrial Management, Management and Accounting Faculty, Shahid Beheshti University, G.C., Tehran, Iran

ARTICLE INFO

Available online 11 August 2013

Keywords:
Total earliness and tardiness
Makespan
Just-in-Time
Controllable processing times
Unrelated parallel machines

ABSTRACT

Job scheduling has always been a challenging task in modern manufacturing and the most real life scheduling problems which involves multi-criteria and multi-machine environments. In this research our direction is largely motivated by the adoption of the Just-In-Time (JIT) philosophy in parallel machines system, where processing times of jobs are controllable. The goal of this paper is to minimize total weighted tardiness and earliness besides jobs compressing and expanding costs, depending on the amount of compression/expansion as well as maximum completion time called makespan simultaneously. Jobs due dates are distinct and no inserted idle time is allowed after starting machine processing. Also each machine is capable of processing only some predetermined jobs and operations with probably different speeds. A Mixed Integer Programming (MIP) model is proposed to formulate such a problem and is solved optimally in small size instances. A Parallel Net Benefit Compression-Net Benefit Expansion (PNBC–NBE) heuristic is then presented to acquire the optimal jobs set amount of compression and expansion processing times in a given sequence. To solve medium-to-large size cases, a proposed heuristic, two meta-heuristics and a hybrid technique are also employed. Experimental results demonstrate that our hybrid procedure is a proficient method and could efficiently solve such complicated problems.

1. Introduction

Most of research in the literature have treated tardiness and earliness as two criteria associated with completing a job at a time different from its given due date since both earliness and tardiness have affect on the system efficiency and must be taken into account. In other words, a major force of research in the scheduling field has been directed towards minimizing both tardiness and earliness penalties of scheduled jobs. Due to the extensive acceptance of Just-in-Time (JIT) philosophy in recent years, the due date requirements have been studied widely in scheduling problems, especially those with earliness–tardiness penalties. In fact, JIT philosophy seeks to identify and eliminate waste components as over production, waiting time, transportation, processing, inventory, movement and defective products [1]. Since earliness could represent manufacturer concerns and tardiness could embrace both customer and manufacturer concerns while none of them is desirable, we aim at minimizing weighted tardiness and earliness as well as makespan in parallel machines environment. A job in JIT scheduling environment that completes early must be held in finished goods inventory until its due date and may result in additional costs such as deterioration of perishable goods, while a tardy job which completes after its due date causes a tardiness penalty such as lost sales, backlogging cost, etc. So, an ideal schedule is one in which all jobs finish exactly on their assigned due dates [2]. Owing to their imposed additional costs to production systems, both earliness and tardiness must be minimized since neither of them is desirable. Baker and Scudder [3] presented the first survey on early/tardy (E/T) scheduling problems. Also this category of problems has been shown as NP-hard ones [4,5].

Machine scheduling problems fall into two main classes, single machine and multi-machine problems. Despite researches focused mostly on single machine E/T scheduling, since it is easier to solve, however parallel machine E/T scheduling problems are more practical in industrial production environments, such as mechanical industry, electronic industry and so on. The majority of earlier studies on parallel machine scheduling have dealt with performance criteria such as mean flowtime, mean tardiness, makespan and mean lateness. In accordance with increasing current trends towards JIT policy, traditional performance measures are no longer applicable. In its place, the emphasis has shifted towards E/T scheduling taking earliness in addition to tardiness into account [3]. Generally speaking,

* Corresponding author.
E-mail address: m_zandieh@sbu.ac.ir (M. Zandieh).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.08.003

readers interested in earliness–tardiness scheduling are referred to a survey conducted by Baker and Scudder [3] and the recent book by T'kindt and Billout [6]. Also readers especially interested in earliness and tardiness scheduling with setup considerations are referred to the survey article by Allahverdi et al. [7].

On the other hand, from among different types of parallel machines, scheduling of unrelated ones is one of the most important and yet complicated issues in the multi-machine manufacturing environments. Meanwhile, in spite of large amount of researches on parallel machines, few of them have surveyed unrelated parallel ones or sequence-dependent setups. In general, such a problem consists of simultaneous job allocation and job sequencing to the machines with similar, but not necessarily identical capabilities. Also in multi-machine manufacturing systems, the unit-time value of different processors may quite vary because of the technology differences, energy or labor requirements, tool usage and failure rates. Therefore, the job-machine assignment is very important and may affect the total processing costs. Of papers which are published in this field, one may refer to Li and Yang [8], Kim et al. [9] and Logendran et al. [10].

The rest of the paper is organized as follows: Section 2 gives related literature. In Section 3 problem description and basic properties will be described. Section 4 explains our proposed PNBC–NBE heuristic in detail. In Section 5 evolutionary algorithms consist of two meta-heuristics and a hybrid one are described. Section 6 discusses computational studies. Finally, Section 7 includes conclusions and future researches.

2. Literature survey

Widely used performance measures in due date-related scheduling problems include maximum tardiness, total or mean tardiness, total weighted tardiness/earliness and the number of tardy jobs [11,12]. Morton and Pentico [13] and Liaw et al. [14] indicated due date-related problems for multi-machine environments are usually computationally complex and therefore most existing consequences are typically for small size problems or simple settings. Cheng and Sin [15] studied a comprehensive review on parallel machine scheduling problems with conventional performance measures based on due date, completion time, and flow time. Rocha et al. [16] studied unrelated parallel machines considering sequence and machine-dependent setup times, due dates and weighted jobs. Also, Bank and Werner [17] considered unrelated parallel machine regarding release date as well as common due date where each of n jobs has to be processed without interruption on exactly one of m unrelated parallel machines.

Earliness and tardiness criteria were studied simultaneously on parallel machines in several papers. Of them, Sivrikaya and Ulusoy [18] developed a genetic algorithm (GA) approach to tackle the scheduling problem relevant to a set of independent jobs on parallel machines with earliness and tardiness penalties. Biskup and Cheng [19] studied scheduling of identical parallel machines with minimizing earliness, tardiness and completion time penalties goals. Ventura and Kim [20] considered parallel machines scheduling problem where jobs have noncommon due dates and may require, besides machines, certain additional limited resources for their handling and processing with the goal of minimizing total absolute deviation of job completion times about the corresponding due dates. Kedad-Sidhoum et al. [21] addressed the parallel machine scheduling problem in which the jobs have distinct due dates with earliness and tardiness costs.

Toksari and Guner [22] considered a parallel machine E/T scheduling problem with common due date and different penalties under the effects of position based learning and linear and nonlinear deterioration. Lin et al. [23] compared the performance of various heuristics and one meta-heuristic for unrelated parallel machine scheduling problems with the goal of minimizing makespan, total weighted completion time, and total weighted tardiness. Hsu et al. [24] studied unrelated parallel machine scheduling problem with setup time and learning effects simultaneously, in which the setup time is proportional to the length of the already processed jobs (i.e., the setup time of each job is past-sequence-dependent) with the objective of minimizing the total completion time. They showed that there exists a polynomial time solution for the proposed problem. In similar research, Kuo et al. [25] studied unrelated parallel machine scheduling problem with setup time and learning effects simultaneously in which the setup time is proportional to the length of the already processed jobs. Their objectives were to minimize the total absolute deviation of job completion times and the total load on all machines, respectively. They also showed that the proposed problem is polynomially solvable. Mor and Mosheiov [26] showed that minimizing total absolute deviation of job completion times (TADC) remains polynomial when position-dependent processing times are assumed (i) on uniform and unrelated machines and (ii) for a bicriteria objective consisting of a linear combination of total job completion times and TADC. Bozorgirad and Logendran [27] addressed a sequence-dependent group scheduling problem on a set of unrelated-parallel machines where the run time of each job differs on different machines. To benefit both producers and customers they tried to minimize a linear combination of total weighted completion time and total weighted tardiness. Vallada and Ruiz [28] studied unrelated parallel machine scheduling problem with machine and job-sequence dependent setup times with the objective of minimizing the total weighted earliness and tardiness. The idle time is allowed in their research. M′Hallah and Al-Khamis [29] addressed minimum weighted earliness–tardiness parallel machine scheduling problem with distinct deterministic known due dates regarding allowable machine idle time. They provided the exact solution for small or relatively easy instances.

In the face of real-life situation most classical scheduling models assume that job processing times are fixed, while the processing times depending on the amount of resources such as budgets, facilities capabilities, manpower. The controllable processing time means each job could process in a shorter or longer time depends on its efficacy on objective function by reducing or increasing the available resources such as equipment, energy, financial budget, subcontracting, overtime, fuel or human resources. When the processing times of jobs are controllable, selected processing times affect both the manufacturing cost and the scheduling performance. As an applicable case for such assumption, in chemical industry, the processing time of a job is increased by an inhibitor or reduced using catalyzer. An inhibitor is any agent that interferes with the activity of an enzyme. Actually, enzyme inhibitors are molecules that bind to enzymes and decrease their activity. More applications of such a substance could be found in Wang et al. [30] and Sørensen et al. [31]. CNC machines are other high usage examples in which job processing times could be controlled by setting the cutting speed and/or the feed rate on the machines.

There is a remarkable relation between E/T scheduling problems and controllable processing times concept, since by controlling the process time of jobs, earliness and tardiness could be decreased and consequently the scheduling environment gets closer to JIT philosophy. Almost certainly, Vickson [32] has studied one of the first researches on controllable processing time scheduling problems, with the objective of minimizing the total flow time and the total processing cost incurred due to job processing time compression. Researches on scheduling problem with controllable processing times and linear cost functions up to 1990 are surveyed by Nowicki and Zdrzalka [33]. Also Shabtay and Steiner [34] have conducted a complete survey on scheduling with controllable processing times.

Nowicki and Zdrzalka [35] considered a bicriterion approach to preemptive scheduling of m parallel machines for jobs having processing costs which are linear functions of variable processing times. In non-identical parallel machines environment, Alidaee and Ahmadian [36] surveyed this category of scheduling with linear compression cost functions to minimize: (1) the total compression cost and the total flow time; and (2) the total compression cost and the weighted sum of earliness and tardiness penalties. They formulated both problems as transportation problems. Cheng et al. [37] studied the unrelated parallel machine scheduling problem which processing times of jobs could only be compressed through incurring an extra cost, as this cost is a convex function of the amount of compression. They also assumed that due date is unrestrictively large. Jansen and Mastrolilli [38] studied the identical parallel machines makespan problem with controllable processing time. Job is allowed to compress its processing time in return for compression cost. Gurel and Akturk [39] surveyed the identical parallel CNC machines with controllable processing time which a time/cost trade-off consideration is conducted. As another work on non-identical parallel machine, Gurel et al. [40] considered a status in which processing times could be controlled at a certain compression cost. They used an anticipative approach to form an initial schedule so that the limited capacity of the production resources is utilized more effectively. Turkcan et al. [41] studied parallel CNC machines with controllable processing times with the minimization of manufacturing cost and total weighted earliness and tardiness objective. They assumed that parts have job-dependent earliness and tardiness penalties and also distinct due dates and allowable idle time was regarded, but they did not consider decompression costs. Aktürk et al. [42] considered a non-identical parallel machining where processing times of the jobs are only compressible at a certain manufacturing cost, which is a convex function of the compression on the processing time. They introduced alternative match-up scheduling problems for finding schedules on the efficient frontier of time/cost tradeoff. Also Leyvand et al. [43] studied scheduling of parallel machines regarding controllable processing times with the goal of maximizing the weighted number of jobs that are completed exactly at their due date and minimizing the total resource allocation cost. Li et al. [44] considered the identical parallel machine scheduling problem to minimize the makespan with controllable processing times, in which the processing times are linear decreasing functions of the consumed resource.

In addition to the mentioned studies, there are some ones which addressed the parallel processors with fuzzy processing times. Of them, one could refer to Peng and Liu [45] which developed a methodology for modeling parallel machine scheduling problems with fuzzy processing times. They presented three novel types of fuzzy scheduling models. Alcan and Başlıgil [46] presented a kind of GA based on machine code for minimizing the processing times in non-identical machine scheduling problem. They used triangular fuzzy processing times in order to adapt the GA to non-identical parallel machine scheduling problem. Also, Balin [47] addressed parallel machine scheduling problems with fuzzy processing times in which a robust GA approach embedded in a simulation model is proposed to minimize the maximum completion time (makespan). Chyu and Chang [48] presented two simulated annealing (SA) and a greedy randomized adaptive search procedure (GRASP) to solve unrelated parallel machine scheduling problems (UPMSPs) with two fuzzy optimization objectives – makespan and average tardiness. Kwong et al. [49] used GA and fuzzy-set theory to generate fault-tolerant fabric-cutting schedules in a JIT production environment. Their proposed method is demonstrated by two cases with production data collected from a Hong Kong-owned garment production plant in China. Mok et al. [50] proposed a fuzzification scheme to fuzzify the static standard time so as to incorporate some uncertainties, in terms of both job-specific and human related factors, into the fabric-cutting scheduling problem. A genetic optimization procedure is also proposed to search for fault-tolerant schedules using GAs, such that makespan and scheduling uncertainties are minimized.

GA as one of the most famous and well-known techniques and Imperialist Competitive Algorithm (ICA) as a novel global search heuristic are employed in this paper to acquire the reasonable solutions in large scale problems, since parallel machines were proven to be NP-hard. Successful application of GA in E/T parallel machines problems could be found in Cheng et al. [51], Funda and Gunduz [52] and Min and Cheng [53]. To the best of author's knowledge, no outstanding research is found in which both earliness and tardiness criteria as well as makespan has been studied on unrelated parallel processors wherein the processing times are controllable.

3. Problem description and basic properties

In this section, a Mixed Integer Programming (MIP) mathematical model is proposed for unrelated parallel machines with the objective of minimizing total earliness and tardiness in addition to makespan simultaneously, considering SDST assumption. In this context, a set of N jobs denoted by 1, 2,.., n has to be processed on a set of M unrelated parallel machines denoted by M_1, M_2, …, M_m. The setups are assumed to be simultaneously machine and job dependent, so when a given job i is processed immediately after job k on machine m, a setup time S_{kim} is incurred. The proposed model is constructed according to the following assumptions:

3.1. Assumptions

- All jobs and machines are available in time zero.
- Each machine could process only one operation at a time.
- The setup time for each job on each machine is sequence-dependent.
- After starting the process by machine, no idle time could be inserted into the schedule.
- No job operation preemption is allowed.
- All machines are unrelated and each job could be processed by a free machine.
- Each machine is capable of processing only some given jobs.
- Machines are available throughout the scheduling period (i.e., no breakdown).
- Transportation time between machines is negligible.
- The process time of each job on each machine differs.
- Each job has a distinct due date and must be processed only one time.
- All processing times and due dates are deterministic and pre-defined.
- Number of jobs and machines are fixed.

3.2. Notations

3.2.1. Subscripts

N	Number of jobs
M	Number of machines
i, j, k	Index for job (i, j, k=0,1,…,N)
m	Index for machine (m=1, 2,…,M)

3.2.2. Input parameters

p_{im}	Processing time of job i on machine m
p'_{im}	Crash (minimum allowable) processing time of job i on machine m
p''_{im}	Expansion (maximum allowable) processing time of job i on machine m
c_{im}	Compression unit cost of job i on machine m
c'_{im}	Expansion unit cost of job i on machine m
α_i	Earliness penalty for job i
β_i	Tardiness penalty for job i
d_i	Due date of job i
λ	Factory costs per time unit (including machines, labor and variable production costs and the costs dependent to the work time)
S_{kim}	Setup time for assigning job i after job k on machine m
b_{im}	1 if machine m is capable of processing job i; =0 otherwise
A	An arbitrary big positive number

3.2.3. Decision variables

C_i	Completion time of job i
C_{max}	Total completion time or makespan
E_i	Earliness of job i; $E_i=\max\{0, d_i-C_i\}$
T_i	Tardiness of job i; $T_i=\max\{0, C_i-d_i\}$
x_{im}	Compression amount of job i on machine m; $0\le x_{im}\le L_{im}$
x'_{im}	Expansion amount of job i on machine m; $0\le x'_{im}\le L'_{im}$
L_{im}	Maximum amount of job i compression on machine m; $L_{im}=p_{im}-p'_{im}$
L'_{im}	Maximum amount of job i expansion on machine m; $L'_{im}=p''_{im}-p_{im}$
y_{ikm}	1 if and only if job i and job k are done on machine m and job i is done earlier than job k.

3.3. The mathematical model

$$Min\ Z=\min\left(\lambda C_{\max}+\sum_{i=1}^{N}(\alpha_i E_i+\beta_i T_i)+\sum_{m=1}^{M}\sum_{i=1}^{N}(c_{im}x_{im}+c'_{im}x'_{im})\right) \tag{1}$$

$$\sum_{k=1}^{N} y_{0km}=1 \quad \forall m; \tag{2}$$

$$\sum_{k=1}^{N} y_{ikm}\le b_{im} \quad \forall m, i\in\{1,\ldots,N\}, i\ne k; \tag{3}$$

$$\sum_{m=1}^{M}\sum_{i=0}^{N} y_{ikm}\le 1 \quad \forall k\in\{1,\ldots,N\}, i\ne k; \tag{4}$$

$$\sum_{m=1}^{M}\sum_{i=0}^{N}\sum_{k=1}^{N} y_{ikm}=N \quad i\ne k; \tag{5}$$

$$\sum_{i=0}^{N} y_{ikm}\ge \sum_{j=1}^{N} y_{kjm}\ \forall m, k\in\{1,\ldots,N\}, \quad i\ne k, j\ne k; \tag{6}$$

$$C_i-d_i=T_i-E_i \quad \forall i; \tag{7}$$

$$C_k+p_{im}-x_{im}+x'_{im}+S_{kim} -A\times(1-y_{kim})\le C_i \quad \forall m, i, k\in\{1,\ldots,N\}, i\ne k; \tag{8}$$

$$C_k+p_{im}-x_{im}+x'_{im}+S_{kim}\le C_i \quad \forall m, i, k\in\{0\},\ i\ne k; \tag{9}$$

$$C_{\max}\ge C_i \quad \forall i; \tag{10}$$

$$p_{im}-p'_{im}\ge x_{im} \quad \forall i,\ m; \tag{11}$$

$$p''_{im}-p_{im}\ge x'_{im} \quad \forall i,\ m; \tag{12}$$

$$y_{ikm}\in\{0,1\} \quad \forall i, k, m; \tag{13}$$

$$T_i,\ E_i,\ C_i,\ x_{im},\ x'_{im}\ge 0 \quad \forall i,\ m; \tag{14}$$

Our multi-criteria objective function is stated in Eq. (1) which aims at minimizing maximum completion time called makespan and total weighted tardiness and earliness as well as jobs compression and expansion cost, depends on amount of compression/expansion. Eq. (2) guarantees that dummy job 0 must be processed earlier than any other job on each machine and subsequently the other jobs could be processed. Constraint (3) ensures that only one job after another utmost could be processed on each machine where this machine is capable of processing this new job. Constraint (4) makes sure there was utmost only one job before processing another job on the same machine. Actually Constraints (3) and (4) together ensure that there is only one job before and one job after the current given job on the same machine. Eq. (5) guarantees all jobs must be processed on machines. Equality (6) ensures that job i could be processed on only one machine. Eq. (7) defines the earliness and tardiness of job i. Since each job could only be tardy or early, if it could not be delivered timely, so it is obvious that T_i and E_i cannot take value simultaneously. Constraints (8) and (9) together ensure that only after starting the process by machine, no idle time could be inserted into the schedule, and no preemption of job is allowable. Also these constraints guarantee that only one job could be processed in each time on machine m. Set (10) defines the maximum completion time. Inequalities (11) and (12) limit the amount of compression and expansion of each job on each machine. Set (13) defines the binary variables and Set (14) identifies non-negativity constraints.

The considered properties of this problem, to a large extend, make it more realistic. Given the above system description, the objective is to determine the job-machine assignments and the jobs sequencing on each machine, simultaneously so that the total weighted earliness and tardiness besides cost of compressing and expanding of jobs as well as makespan is minimized.

4. The proposed PNBC–NBE

The proposed algorithm is expansion of net benefit compression–net benefit expansion (*NBC–NBE*) algorithm for single machine which is proposed by Kayvanfar et al. [54]. The concept of *NBC–NBE* algorithm is close to *NBC* one with a main idea similar to marginal cost analysis used in PERT/CPM with time/cost trade-off [55].

Theorem 1. *The NBC–NBE algorithm yields the optimal amount of compression/expansion for a given sequence.*

Proof. In order to prove the optimality amount of acquired job compression/expansion processing times, suffice it to say that the best selection is the job with maximum *NBC* or *NBE* to approach to such a goal. Choosing *NBC* or *NBE* has no difference in improving the objective function, since both of them is based on net benefit of compression or expansion. A backward approach is used to show this proof. First of all, it should be said that choosing the job with maximum *NBC* or *NBE* in last iteration results in its best consequence, since otherwise a better objective function value could be acquired in such a manner. Now, suppose that the best choice is the job with maximum *NBC* or *NBE* in iteration N+1.

Thereafter, we'll show that it is valid in iteration N. We use the contradiction philosophy to prove this claim. Suppose that job r be the job with maximum *NBC* or *NBE* in iteration N and the best selection in this iteration is another job called job u. If these jobs are selected in two consecutive steps, regardless of which job is selected at first, the objective function value will be the same. But, it must be mentioned that choosing job u firstly could be dominated by choosing job r firstly since one may occur that job u are negative in step N+1 and thereafter we could not select it in this iteration. Consequently, using aforementioned statements we proved that selecting job r in iteration N is the best choice. □

It is important to remind that the NBC–NBE algorithm could be applied on each single machine for a given sequence, separately. Applying NBC–NBE on parallel machine is actually an expansion version of single one. In order to improve the NBC–NBE performance on parallel machines we employ PNBC–NBE heuristic to ensure the best possible result of such algorithm. For scrutiny NBC–NBE performance, one could refer to Kayvanfar et al. [54]. As per our paper we aim at total tardiness and earliness criteria simultaneously in context of JIT approach, our PNBC–NBE must satisfy such a criterion. So, in the following subsection, we present another heuristic to assign the jobs on parallel machines.

4.1. Initial sequence on parallel machines considering JIT approach

Since tardiness and earliness are two reverse concepts, acquiring an algorithm which could minimize both simultaneously is obviously hard on a given machine, because trying to reduce one may result in an increase in the other and vice versa. In this context, owing to the importance of assigning jobs on parallel machines based on minimizing total tardiness and earliness at once we present such a heuristic to allocate the jobs on parallel machines. The steps of the presented heuristic called Initial Sequence based on Earliness–Tardiness criterion on Parallel machines (ISETP) are as follows:

Procedure 1. ISETP

(1) Sort the jobs according to Earliest Due Date (EDD) criterion and put them in *unscheduled* jobs category, *US*.
(2) Assign the first job to the first capable machine and set it in *scheduled* job category, *S*.

Do the following steps until no outstanding job is found:

(3) Assign next *unscheduled* job in EDD order to each capable machine separately and then calculate the C_{max} of each machine.
(4) Compute the difference between due date of this job and C_{max} of such machine ($C_{\max_j}$) called $U_j = |C_{\max_j} - d_j|$.
(5) Select the assignment which has led to minimum U_j and assign that job to such a machine. If two or more values are equal, select the machine with minimum index (however there is no difference in selecting from each).
(6) Transfer this job from *US* category into *scheduled* one, *S*.
(7) Update C_{max} of each machine and go to Step 3.

The term "capable machines" used in ISETP denotes those ones which are capable of processing some given jobs. To illustrate the ISETP performance, solving an example could be useful. It should be mentioned that for the sake of simplicity the jobs processing times on different machines are regarded equal and all machines are also capable of processing all jobs in this example, however in the main calculations in Section 6 all assumptions are exerted.

Table 1
Input data for an eight job problem with three machines.

J	1	2	3	4	5	6	7	8
p_i	4	6	5	7	5	6	4	6
d_i	5	5	11	6	13	13	11	20
α_i	0.5	1	1	1.25	1.5	1	1.5	0.5
β_i	0.5	0.5	1.25	0.5	0.5	1	3	0.5

Example 1. Consider the following problem with eight jobs and three parallel machines. Table 1

Iteration #1:

Step 1. Sort jobs according to EDD rule and set them in $US=\{J_1, J_2, J_4, J_3, J_7, J_5, J_6, J_8\}$.
Step 2. Assign the first job to the first machine and set it in *scheduled* job category, $S=\{J_1\}$.
Step 3. Assign the second job (J_2) to each machine separately,

$$C_{\max_1} = 4+6 = 10, \quad C_{\max_2} = C_{\max_3} = 6$$

Step 4. Compute all U_j as follows:

$$U_1 = |C_{\max_1} - d_1| = |10-5| = 5,\ U_2 = |C_{\max_2} - d_2| = |6-5| = 1,$$
$$U_3 = |C_{\max_3} - d_3| = |6-5| = 1.$$

Step 5. Select the min U_j and assign J_2 to such a machine. Since two values are equal, select the machine with minimum index, i.e., machine #2.
Step 6. Transfer this job from *US* category to *S* one, $US=\{J_4, J_3, J_7, J_5, J_6, J_8\}$ and $S=\{J_1, J_2\}$.
Step 7. Update C_{max} of each machine, $C_{\max_1} = 4, \quad C_{\max_2} = 6, \quad C_{\max_3} = 0$.

Iteration #2:

Step 3. Assign the third job (J_4) to each machine separately,

$$C_{\max_1} = 4+7 = 11, \quad C_{\max_2} = 6+7 = 13, \quad C_{\max_3} = 7.$$

Step 4. Compute all U_j as follows:

$$U_1 = |C_{\max_1} - d_1| = |11-6| = 5, \quad U_2 = |C_{\max_2} - d_2| = |13-6| = 7,$$
$$U_3 = |C_{\max_3} - d_3| = |7-6| = 1.$$

Step 5. Select the min U_j and assign J_4 to such machine, i.e., machine #3.
Step 6. Transfer this job from *US* category to *S* one, $US=\{J_3, J_7, J_5, J_6, J_8\}$ and $S=\{J_1, J_2, J_4\}$.
Step 7. Update C_{max} of each machine, $C_{\max_1} = 4, \quad C_{\max_2} = 6, \quad C_{\max_3} = 7$.

Iteration #3:

Step 3. Assign the forth job (J_3) to each machine separately,

$$C_{\max_1} = 4+5 = 9, \quad C_{\max_2} = 6+5 = 11, \quad C_{\max_3} = 7+5 = 12.$$

Step 4. Compute all U_j as follows:

$$U_1 = |C_{\max_1} - d_1| = |9-11| = 2,\ U_2 = |C_{\max_2} - d_2| = |11-11| = 0,$$
$$U_3 = |C_{\max_3} - d_3| = |12-11| = 1.$$

Step 5. Select the min U_j and assign J_3 to this machine, i.e., machine #2.

Step 6. Transfer this job from *US* category to *S* one, $US=\{J_7, J_5, J_6, J_8\}$ and $S=\{J_1, J_2, J_4, J_3\}$.
Step 7. Update C_{max} of each machine, $C_{\max_1}=4$, $C_{\max_2}=11$, $C_{\max_3}=7$.

Iteration #4:

Step 3. Assign the fifth job (J_7) to each machine separately,

$C_{\max_1}=4+4=8, \quad C_{\max_2}=11+4=15, \quad C_{\max_3}=7+4=11.$

Step 4. Compute all U_j as follows:

$U_1=|C_{\max_1}-d_1|=|8-11|=3, \quad U_2=|C_{\max_2}-d_2|=|15-11|=4,$

$U_3=|C_{\max_3}-d_3|=|11-11|=0.$

Step 5. Select the min U_j and assign J_7 to this machine, i.e., machine #3.
Step 6. Transfer this job from *US* category to *S* one, $US=\{J_5, J_6, J_8\}$ and $S=\{J_1, J_2, J_4, J_3, J_7\}$.
Step 7. Update C_{max} of each machine, $C_{\max_1}=4, \quad C_{\max_2}=11, \quad C_{\max_3}=11$.

Iteration #5:

Step 3. Assign the sixth job (J_5) to each machine separately,

$C_{\max_1}=4+5=9, \quad C_{\max_2}=11+5=16, \quad C_{\max_3}=11+5=16.$

Step 4. Compute all U_j as follows:

$U_1=|C_{\max_1}-d_1|=|9-13|=4, \quad U_2=|C_{\max_2}-d_2|=|16-13|=3,$

$U_3=|C_{\max_3}-d_3|=|16-13|=3.$

Step 5. Select the min U_j and assign J_5 to this machine. Since the two values are equal, select the machine with minimum index, i.e., machine #2.
Step 6. Transfer this job from *US* category to *S* one, $US=\{J_6, J_8\}$ and $S=\{J_1, J_2, J_4, J_3, J_7, J_5\}$.
Step 7. Update C_{max} of each machine, $C_{\max_1}=4$, $C_{\max_2}=16$, $C_{\max_3}=11$.

Iteration #6:

Step 3. Assign the seventh job (J_6) to each machine separately,

$C_{\max_1}=4+6=10, \quad C_{\max_2}=16+6=22, \quad C_{\max_3}=11+6=17.$

Step 4. Compute all U_j as follows:

$U_1=|C_{\max_1}-d_1|=|10-13|=3, \quad U_2=|C_{\max_2}-d_2|=|22-13|=9,$

$U_3=|C_{\max_3}-d_3|=|17-13|=4.$

Step 5. Select the min U_j and assign J_6 to this machine. i.e., machine #1.
Step 6. Transfer this job from *US* category to *S* one, $US=\{J_8\}$ and $S=\{J_1, J_2, J_4, J_3, J_7, J_5, J_6\}$.
Step 7. Update C_{max} of each machine, $C_{\max_1}=10, \quad C_{\max_2}=16, \quad C_{\max_3}=11$.

Iteration #7:

Step 3. Finally, assign the last job in EDD order (J_8) to each machine separately,

$C_{\max_1}=10+6=16, \quad C_{\max_2}=16+6=22, \quad C_{\max_3}=11+6=17.$

Step 4. Compute all U_j as follows:

$U_1=|C_{\max_1}-d_1|=|16-20|=4,$
$U_2=|C_{\max_2}-d_2|=|22-20|=2, \quad U_3=|C_{\max_3}-d_3|=|17-20|=3.$

Step 5. Select the min U_j and assign J_8 to this machine. i.e., machine #2.
Step 6. Transfer this job from *US* category to *S* one, $US=\{\ \}$ and $S=\{J_1, J_2, J_4, J_3, J_7, J_5, J_6, J_8\}$.
Step 7. Update C_{max} of each machine, $C_{\max_1}=10, \quad C_{\max_2}=22, \quad C_{\max_3}=11$.

Since there is no outstanding job in *US* category, the algorithm is terminated. For the sake of comparison of obtained solution with optimum one, this problem is solved by Lingo which the final solution comes as follows: Table 2

As pointed out earlier, our proposed PNBC–NBE takes advantage from the net benefit compression/expansion in parallel machines. The logic in our algorithm is based on calculating the difference between decreased total tardiness and the cost of compression and decreased total earliness and the cost of expansion of a job which is exerted by reducing or increasing of processing time of a job by one time unit. By compressing or expanding the jobs processing time, objective function reduces. Each job – depending on its tardiness or earliness on given machine – may compress or expand if this compression or expansion is economical i.e., the compression or expansion cost is smaller than its benefits (decreased tardiness or earliness). In such a way the objective function value will be decreased and such job will be compressed or expanded. The steps of our PNBC–NBE heuristic algorithm are stated as follows:

Procedure 2. PNBC–NBE

(1) Assign the jobs on parallel machines using ISETP heuristic technique, as an initial sequence.
(2) Employ NBC–NBE to determine the amount of reduction/expansion of jobs processing times.

Table 2
Comparison of ISETP and lingo in given example.

ISETP	Lingo
$\boldsymbol{Z=7}$	$Z^*=6$
$\boldsymbol{M(1)=J_1-J_6}$	$M(1)=J_2-J_7$
$\boldsymbol{M(2)=J_2-J_3-J_5-J_8}$	$M(2)=J_1-J_3-J_5$
$\boldsymbol{M(3)=J_4-J_7}$	$M(3)=J_4-J_6-J_8$

Table 3
Job information.

Job	p_i	p'_i	p''_i	d_i	α_i	β_i	c_i	c'_i
1	4	3	5	3	4	6	1	2
2	6	5	7	11	4	6	1	2
3	5	4	6	18	4	6	1	2
4	7	5	8	10	4	6	1	2
5	5	3	7	6	4	6	1	2
6	6	5	7	10	4	6	1	2
7	4	3	6	12	4	6	1	2
8	6	4	7	17	4	6	1	2
9	5	4	6	17	4	6	1	2
10	5	4	6	22	4	6	1	2

Do the three following steps (3–5) until stop criterion is met:

(3) Swap jobs and their corresponding capable machines randomly in order to reduce the earliness and tardiness values.
(4) Accept new sequence if objective function value is reduced, go to Step 5, else keep the previous sequence. Go to Step 3.
(5) Apply NBC–NBE to determine whether a given job could be further compressed or else expanded. Go to Step 3.
(6) Determine the ultimate sequence and compute the final amount of compression/expansion of jobs processing times.

In the aforementioned procedure, the stop criterion is defined as maximum number of non-improvement iterations which is set at 50.

Example 2. To validate the proposed model and illustrate its various features, numerical example with randomly generated data includes three machines and 10 jobs solved using branch and bound (B&B) method under Lingo 9.0 software on a PC with a 3.4 GHz Intel® Core™ i7-2600 processor and 4 GB RAM memory. Table 3 presents the values of all input parameters for each and every job. Since regarding different processing times for dissimilar machines in which each job has also a crash and expansion processing time (p'_{im} and p''_{im}) makes the reader confused, the jobs processing times are regarded equal on all machines in this example for the sake of straightforwardness. Moreover, assuming capability of processing all jobs by all machines in this example is also applied to get closer to such straightforwardness. However, in the main calculations in Section 6 all normal, crash and expansion processing times are regarded different on dissimilar machines and each machine could only process some given jobs.

Table 4 presents the obtained solution containing the completion time, tardiness/earliness penalty of each job as well as compression/expansion jobs processing times cost.

The objective function value (OFV) obtained after 1,092,056 iterations in CPU time 6:04′:15″ which is presented in Table 5 in addition to its cost components.

Fig. 1 shows the schema of JIT in parallel machines for this example. As could be seen, minimum earliness/tardiness and consequently minimum compression/expansion cost are acquired using such a sequence. In this example, completion time of job 4 is 8 but its due date is 10, which causes an earliness cost. Since the earliness penalty unit of job 4 is 4, the total cost is then $2 \times 4 = 8$.

Table 4
The obtained solution for each job.

Job	Completion time	Earliness penalty	Tardiness penalty	Compression penalty	Expansion penalty
1	3	0	0	1*1	0
2	11	0	0	1*1	0
3	18	0	0	0	1*2
4	8	2*4	0	0	0
5	6	0	0	0	0
6	10	0	0	0	1*2
7	12	0	0	0	0
8	17	0	0	0	1*2
9	17	0	0	0	1*2
10	22	0	0	0	0

Table 5
Objective function and its cost components.

OFV	Earliness	Tardiness	Compression	Expansion
18	8	0	2	8

5. Evolutionary algorithms

5.1. Imperialist competitive algorithm

Unlike the current evolutionary algorithms, such as GA and simulated annealing (SA), which are inspired from a natural process, ICA uses socio-political evolution processes, as source of inspiration [56]. Similar to other evolutionary algorithms ICA initiates with an initial population, like most evolutionary algorithms. Each individual of the population is called a '*country*' equivalent '*chromosome*' in GA. Some of the most powerful countries (in optimization terminology, countries with the least cost) are selected to be the imperialist states and the other countries constitute the colonies of these imperialists. All the colonies of initial countries are partitioned among the mentioned imperialists based on their power. Equivalent of fitness value in the GA, the power of each country, is conversely proportional to its cost. A set of one imperialist and its colonies is called an *empire*.

By constituting initial empires, each of their colonies begins progresses toward their related imperialist country. This is a simple kind of assimilation strategy which some of the imperialist states followed that. Afterward, the imperialistic competition starts among all the empires. Those empires which could not win this competition and are not capable of increasing their power or at least prevent decreasing its power will be removed from the struggle. The imperialistic competition will slowly however surely results in an enhancement in the power of powerful empires and a decrease in the power of weaker ones.

The total power of an empire depends on both the power of the imperialist country and the power of its colonies. This fact is modeled by defining the total power of an empire as the power of imperialist country plus a percentage of mean power of its colonies [56].

A thing causes all countries to converge into a state in which there exists only one empire in the world is colonies movement toward their related imperialists along with struggle among empires and also the collapse mechanism. In such a case, all other

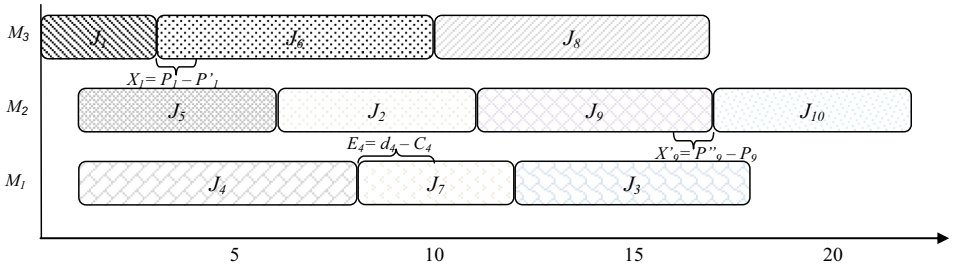

Fig. 1. Schema of just-in-time in parallel machines for the considered example.

countries will be considered colonies of that empire. In this ideal new world, colonies have the same position and power as the imperialist.

The major purpose of optimization is to acquire an optimal solution. In our problem an array of variable values which must be optimized is formed. Here, a country is a $1 \times N_{var}$ array which is defined by

$$country = (g_1, g_2, g_3, ..., g_{N_{var}})$$

where each g_i is a variable which should be optimized. Each of these variables could be interpreted as a socio-political characteristic of a *country*, such as religion, culture. From optimization perspective a solution with least cost value is the best one.

By evaluating the cost function f at variables $(g_1, g_2, g_3, \ldots, g_{N_{var}})$ the cost of a *country* is stated as follows:

$$Cost = f(country) = f(g_1, g_2, g_3, \ldots, g_{N_{var}})$$

In order to start the algorithm, first of all, we generate the initial population of size N_{pop}. We pick N_{imp} of the most powerful countries to constitute the empires. The leftover N_{col} of the population will be the colonies each of which belongs to an empire. We split the colonies among imperialists based on their power so as to constitute the initial empires.

All steps of ICA technique including initial number of colonies of an empire, movement of colonies toward the imperialist (assimilation), exchanging positions of the imperialist and a colony, total power of an empire, imperialistic competition, elimination of the powerless empires and finally convergence and stop criterion are applied according to Kayvanfar and Zandieh [57] which is ignored in this section owing to brevity.

5.2. Genetic algorithm

One of the most important merits of GA is considering a large number of sets of solutions in parallel. The GAs are bio-inspired optimization methods [58] which are extensively used to solve scheduling problems [59]. In contrast to other local search methods, such as SA or Tabu Search (TS) which are based on handling one feasible solution, GA utilizes a population of individuals in its search, giving it more resistance to premature convergence on local minima [60].

5.2.1. Representation of solutions

An array of jobs for each machine that represents the processing order of the jobs assigned to that machine is the most frequently used solution representation for the parallel machine scheduling problem. Since all machines are not capable of processing all jobs, the genes correspond to those jobs which could not be processed on a given machine (chromosome) are left blank. This action does not have any effect on jobs sequence. The GA is initially formed by a population of individuals (as *popsize*), where each individual consists of m arrays of jobs (one per machine).

5.2.2. Initial population and population size

The key decision in GA implementation is determining the appropriate population size. If the selected number is too small we may not be able to attain a good solution. On the contrary the large number takes too much computation time to achieve a better solution [61]. In this study the population size is taken experimentally 100. The population is randomly initialized, so that every job has an equally likely chance of starting out on any capable one machine.

5.2.3. Fitness function

The fitness evaluation goal is to calculate the goodness of the candidate individuals in the population with respect to the objective function and constraints of the proposed model. After generation of new population, fitness value of each chromosome is calculated (f_k). The higher the fitness value, the better the performance of the chromosome (i.e. parent). Therefore, parents with higher fitness values have more chances to survive. Among different criteria used as fitness value, as for our objective in this study, difference of objective function from a large positive number Γ (an upper bound) is considered as fitness value for each individual in this paper as follows:

$$\text{Fitness function} = \Gamma - \text{objective function} \quad (15)$$

According to Eq. (15) parents with higher fitness value are more eligible.

5.2.4. Selection mechanism

In the classical GAs, tournament and roulette wheel selection operators are common. These operators either require fitness and mapping calculations or the population to be continuously sorted. In this paper we make selection of parents according to the roulette wheel selection mechanism. In this technique, a probability of selection is assigned to each individual based on its fitness value. In spite of existing higher probability in the selection of better chromosomes, all individuals in the population will have a chance to be selected. We choose the parents randomly after ranking individuals based on their corresponding fitness and giving weighted probability of selection to each individual in the population.

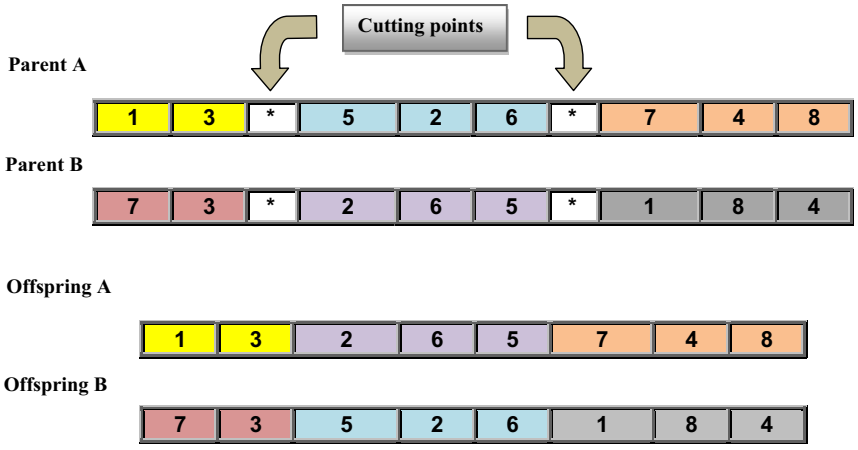

Fig. 2. Two point crossover operator .

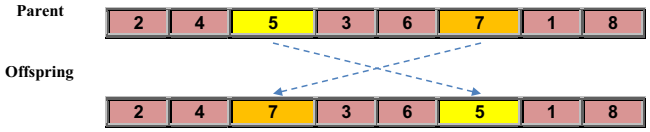

Fig. 3. Mutation operator .

5.2.5. Genetic operators

Reproduction is carried out by means of elitism process. In this method after ranking individuals based on their fitness, the best ones which have maximum fitness are directly copied into the next generation. The crossover operator is also applied according to a probability p_c by selected individuals that are obtained by roulette wheel technique. The aim of crossover operator is actually generating two good individuals (offsprings) from two selected progenitors (parents). In the literature, there are several common crossover operators for scheduling problems, such as one-point, two-point, uniform. A standard two-point crossover is used in this paper to produce two offsprings from two parent solutions. Having randomly chosen two points in a string, the sub-strings between the crossover points are interchanged. After such interchanging, if a job could not be processed on new chromosome, the corresponding gen left blank. Fig. 2 depicts applying the crossover operator on two selected parents.

A mutation operator according to a probability p_m could be applied once the offspring are obtained. By testing different mutation operators, the swap mutation yielded the best performance. This operator works by swapping two genes in the selected individual (chromosome) and keeps also away from getting stuck in the local suboptimal solutions and is very helpful to maintain the wealth of the population in dealing with large scale problems. Fig. 3 shows the mutation operator in which the offspring is produced by exchanging positions of two randomly selected genes, i.e., numbers 5 and 7. Generally, mutation operator is applied with the intention of diversifying the population in order that it does not prematurely converge to multiple copies of one solution.

5.2.6. Termination criterion

The number of generations (iterations) is used as the termination criterion. The lower number of generations, the lower probability of finding the best result would be and vice versa. On the other side when the number of generations is too high, the longer would be the iteration time. In this paper according to the testing different values experimentally, "50 non-improvement successive iterations" are regarded as stopping criterion.

5.3. The proposed hybrid IIGNN

Hybrid algorithms have received considerable interest in recent years and are being increasingly used to solve real-world problems. The deficiencies of existent meta-heuristics give rise to hybrid methods. Actually, thanks to the complementary properties of hybrid algorithms, the researchers persuade to employ these techniques, since hybrid approaches often outperform either method operating alone. However, they still require more computational effort.

The most common form of using hybrid methods is a two-level search technique in which a local search is employed inside population-based methods to promote either the entire or a fraction of the newly generated offspring. The time needed to attain the global optimum could be further reduced if local search techniques are employed to accelerate locating the most promising search space.

Since in ICA choosing an empire is analogous to the roulette wheel process which is used in selecting parents in GA and since calculation of the cumulative distribution function is not needed in ICA, i.e., the selection is based on only the values of probabilities of selection, this method is much faster than the conventional roulette wheel and consequently much faster than GA [57]. On the other hand, our proposed ISETP is a deterministic algorithm and could be implemented in a negligible computational time even in large size instances. So, these two techniques take advantage of high convergence speed while GA takes so longer, but GA as a well-known powerful meta-heuristic algorithm usually yields higher quality solutions compared with other techniques.

In this study, in order to improve solutions quality, we employ a hybrid heuristic in which our GA merges within ICA and ISETP. As pointed out earlier, such hybrid method benefits high speed of finding an initial solution using ISETP and ICA than GA and potential power of GA as a well-known population based technique in approaching good final solutions compared with ISETP and ICA. In better words, our ISETP and ICA firstly produce two initial solutions (chromosomes) in a small time and the other individuals are randomly generated by GA. Subsequently, the NBC–NBE is applied to minimize the objective function by decreasing/increasing jobs processing times for a given sequence on each machine. This hybrid heuristic is called IIGNN. The population size of IIGNN is regarded 100 same as in GA. It is also noticeable that the core component of this proposed hybrid method is genetic algorithm module that continuously searches the solution space.

The IIGNN is supposed to yield higher quality solutions with respect to the other used techniques in this study, since it takes benefits from relative advantages of all algorithms, i.e., speed and power of convergence within lesser computational time.

6. Computational study

The objective of the computational experiments described in this section is to evaluate the performance of the proposed algorithms. All instances are implemented in MATLAB 7.11.0 and run on a PC with a 3.4 GHz Intel® Core™ i7-2600 processor and 4 GB RAM memory.

6.1. Test problems

Two sets of experiments are performed in which the input parameters are as follows:

Normal, crash and expansion processing times (p_{im},p'_{im} and p''_{im}) are discretely uniformly distributed as follows, respectively: (3, 25), ($0.5\times p_i$, p_i) and (p_i, $1.5\times p_i$). Due dates follow a uniform distribution as [d_{min}, $d_{min}+\rho$P] where $d_{min}=max\ (0, P(\upsilon-\rho/2))$ and $P=1/m\sum_{m=1}^{M}\sum_{i=1}^{N}p_{im}$. The expression of P aims at satisfying the criteria of scale invariance and regularity described by Hall and Posner [5] for generating experimental scheduling instances. The two parameters υ and ρ are the tardiness and range parameters, respectively which have been assumed as follows in this paper: $\upsilon\in\{0.2,\ 0.5,\ 0.8\}$ and $\rho\in\{0.2,\ 0.5,\ 0.8\}$. Number of jobs ($N$) and number of machines (M) are assumed {10, 20 and 50} and {2, 3 and 5} respectively. Compression and expansion unit cost (c_{im} and c'_{im}) are uniformly distributed as (0.1, 2.5). Also, earliness and tardiness penalties (α_j and β_j) are uniformly distributed as (0.5, 2.5) and (0.5, 4.5) respectively. Finally setup times (S_{kim}) are generated from a discrete uniform distribution between 10 and 90.

For each quadruple (N, M, υ, ρ) three instances are generated in which each case has run five times in all methods so as to guarantee constancy of these techniques. For each combination of such quadruple totally $3^4\times3\times5=1215$ problems for each method are generated consequently.

Table 6
Computational results for small size problems.

M	J	υ	ρ	**Lingo**	**PNBC–NBE**		**INN**		**GNN**		**IIGNN**	
				MCPU time	MCPU time	PRE_{avg}	MCPU time	PRE_{avg}	mcpu time	PRE_{avg}	MCPU time	PRE_{avg}
2	10	0.2	0.2	53329.031	0.015	23.34	5.485	16.64	16.365	15.35	14.325	9.12
			0.5	65734.646	0.019	35.36	4.356	23.46	14.579	20.16	12.326	8.35
			0.8	74461.27	0.008	23.94	5.235	16.65	17.112	14.32	16.325	10.15
		0.5	0.2	56869.897	0.017	45.63	4.162	14.23	16.029	15.65	13.378	9.96
			0.5	69591.923	0.029	35.79	3.965	16.48	15.023	11.23	12.778	8.65
			0.8	44495.132	0.012	44.98	5.134	16.54	18.754	14.96	14.329	11.48
		0.8	0.2	50264.274	0.009	38.31	4.589	24.15	14.258	20.37	11.456	9.94
			0.5	46045.609	0.026	48.16	3.968	22.13	16.348	23.65	14.679	10.22
			0.8	34701.107	0.019	44.78	4.584	16.79	15.013	14.32	11.158	11.17
Mean				**55054.77**	**0.017**	**37.81**	**4.609**	**18.56**	**15.942**	**16.67**	**13.417**	**9.89**
3	10	0.2	0.2	7882.3639	0.016	25.19	7.385	21.31	18.329	24.65	16.236	12.36
			0.5	19082.645	0.021	34.31	6.471	25.39	19.467	20.15	16.365	10.98
			0.8	11821.907	0.024	44.15	4.265	16.35	18.743	13.42	16.321	10.12
		0.5	0.2	15936.675	0.019	30.82	4.965	22.78	17.321	16.78	14.701	9.98
			0.5	12344.04	0.011	29.34	7.418	19.23	19.395	18.32	16.365	12.36
			0.8	10103.467	0.017	36.18	6.018	15.36	18.368	14.97	17.369	11.08
		0.8	0.2	17419.811	0.018	20.84	7.281	14.48	19.114	11.19	16.132	9.36
			0.5	23077.393	0.019	21.93	6.114	19.76	16.357	19.34	16.325	12.65
			0.8	22781.849	0.026	23.78	6.187	27.65	17.158	19.31	16.321	10.71
Mean				**15605.57**	**0.019**	**29.62**	**6.234**	**20.26**	**18.250**	**17.57**	**16.237**	**11.07**
5	10	0.2	0.2	33430.647	0.024	34.15	6.325	24.48	19.325	20.35	16.465	14.36
			0.5	34072.351	0.042	29.97	4.365	22.78	17.326	17.39	15.465	13.12
			0.8	33149.912	0.012	22.34	6.987	23.48	19.589	18.65	19.645	12.79
		0.5	0.2	31026.56	0.016	22.17	5.778	20.98	18.369	20.39	17.326	11.37
			0.5	29023.875	0.031	22.09	7.719	20.45	20.415	16.34	16.125	10.31
			0.8	39783.892	0.028	34.48	6.981	24.09	18.499	19.33	19.354	12.34
		0.8	0.2	37544.497	0.043	22.36	6.134	19.65	15.656	18.14	16.265	14.72
			0.5	34808.227	0.017	31.65	6.947	27.98	18.326	21.07	16.458	15.55
			0.8	36737.806	0.017	24.36	7.195	19.07	19.158	17.65	19.413	12.26
Mean				**34397.53**	**0.026**	**27.06**	**6.492**	**22.55**	**18.518**	**18.81**	**17.391**	**12.98**
Mean of all				**35019.29**	**0.021**	**31.50**	**5.778**	**20.46**	**17.570**	**17.68**	**15.682**	**11.31**

6.2. Performance measures

In order to evaluate the performance of the proposed algorithms, two metrics are applied in this subsection. Since small sized problems (problems with 10 jobs on each machine) could be solved optimally by Lingo, the percentage relative error (*PRE*) is employed to assess the algorithms performance in such groups of instances. This metric is calculated as follows:

$$PRE = \frac{Alg_{sol} - O}{O} \times 100 \tag{16}$$

where Alg_{sol} is the objective value obtained by selected algorithm and O is the optimum value obtained by Lingo.

In medium-to-large cases, the relative percentage deviation (*RPD*) is employed to compare the efficiency of all heuristic methods with meta-heuristic techniques. The *RPD* computes in this fashion:

$$RPD = \frac{Alg_{sol} - Min_{sol}}{Min_{sol}} \times 100 \tag{17}$$

In which Alg_{sol} is the obtained objective value by the selected technique and Min_{sol} represents the best solution obtained for each case. Lower values of *PRE* and *RPD* are preferable perceptibly.

6.3. Comparative results

In small instances the obtained optimal solutions are compared with (1) PNBC–NBE, (2) ICA equipped with NBC–NBE called INN, (3) GA equipped with NBC–NBE called GNN and (4) a hybrid technique called IIGNN.

The proposed hybrid IIGNN consists of ISETP, ICA and GA which has been already described in previous section. Also, INN work out as follows: ICA and NBC–NBE are replaced with the mathematical model where ICA search the best possible sequence and NBC–NBE gives the optimal jobs amount set of compression/expansion on a given machine. The similar procedure is done for GNN, i.e., replacing GA and NBC–NBE with the mathematical model.

In medium-to-large size cases the PNBC–NBE, INN, GNN and IIGNN are compared with each other so as to determine the performance of all algorithms. The computational results for all small and medium-to-large size instances are shown in Tables 6 and 7 respectively.

As could be seen, in small cases, PNBC–NBE has the largest gap with the optimal solutions while our hybrid IIGNN has the smallest one in the average. Moreover, INN and GNN have similar performance (a 2.78% difference) however GNN outperforms INN in the average but by consuming more computational time. Also by increasing the number of machines the quality of PNBC–NBE solutions enhanced while all other algorithms do not follow such an improvement. In medium-to-large size problems, as could be seen, IIGNN yields again the best solutions among all other techniques while PNBC–NBE shows the weakest performance in the average. In this category, by increasing the number of machines, GNN has a little fluctuation in the range of 3.34% RPD while it could be said IIGNN follows on a constant trend to some extent (regardless of tiny decrease in RPD by increasing the number of machines). Also INN has a similar trend compared to GNN where varies in the range of 2.87% RPD. In fact, the GNN and INN quality undergo by a little fluctuation. Fig. 4 shows the average deviation of each algorithm from optimal solution in small size instances. Also in Fig. 5 the average deviation of all techniques with respect to the best obtained solutions in medium-to-large size problems is demonstrated.

It is noticeable that the computational time of all algorithms is incomparable with respect to Lingo in small instances, especially PNBC–NBE. This matter reveals the importance of presenting

Table 7
Computational results for medium-to-large size problems.

M	J	υ	ρ	**PNBC–NBE**		**INN**		**GNN**		**IIGNN**	
				MCPU time	RPD_{avg}	MCPU time	RPD_{avg}	MCPU time	RPD_{avg}	MCPU time	RPD_{avg}
2	20	0.2	0.2	0.036	30.89	16.465	7.57	40.653	4.25	35.265	2.42
			0.5	0.029	41.14	15.648	8.44	39.456	6.11	33.456	0.00
			0.8	0.03	33.09	14.465	22.09	45.258	5.53	40.154	2.53
		0.5	0.2	0.041	40.78	18.174	17.34	41.256	13.08	39.997	5.67
			0.5	0.036	56.09	16.435	18.84	39.654	9.82	36.564	1.08
			0.8	0.034	48.98	18.734	15.83	41.158	4.31	37.184	0.98
		0.8	0.2	0.029	30.03	13.369	13.98	39.445	5.23	31.255	1.03
			0.5	0.034	33.94	17.398	18.33	33.125	4.28	33.485	0.00
			0.8	0.028	40.25	17.958	20.07	39.648	11.34	30.256	3.42
	50	0.2	0.2	0.069	38.94	18.365	24.82	41.258	12.09	33.781	5.23
			0.5	0.056	45.81	18.212	16.24	39.658	6.34	34.154	1.06
			0.8	0.076	30.09	16.365	17.34	37.285	5.82	31.054	1.67
		0.5	0.2	0.066	36.32	16.458	13.49	41.256	9.22	36.609	3.14
			0.5	0.054	35.88	13.854	11.03	41.379	6.84	40.018	1.39
			0.8	0.057	37.31	18.952	9.51	34.558	4.24	31.975	0.00
		0.8	0.2	0.064	25.75	19.483	8.91	40.625	4.07	35.365	0.06
			0.5	0.069	49.81	17.354	13.93	41.654	5.85	38.156	1.14
			0.8	0.071	29.27	15.021	20.81	38.204	14.39	32.405	4.96
Mean				**0.049**	**38.02**	**16.817**	**15.48**	**39.752**	**7.38**	**35.063**	**1.99**
3	20	0.2	0.2	0.029	23.85	12.354	14.87	34.869	18.92	30.012	5.23
			0.5	0.023	38.24	14.265	21.04	32.654	15.24	31.256	6.12
			0.8	0.032	29.09	18.657	16.98	31.666	11.04	32.654	2.42
		0.5	0.2	0.027	31.95	15.445	20.54	40.012	14.85	34.857	2.09
			0.5	0.036	37.67	16.348	23.94	32.524	13.05	25.699	0.00
			0.8	0.037	33.82	17.454	22.46	34.154	8.43	30.214	1.85
		0.8	0.2	0.035	29.46	16.365	16.35	33.254	11.89	28.558	3.12
			0.5	0.029	40.82	15.332	17.93	36.482	10.33	32.154	1.42
			0.8	0.031	22.99	18.258	12.42	37.554	6.36	31.015	0.04
	50	0.2	0.2	0.054	33.78	18.348	20.01	46.778	9.53	40.145	0.65
			0.5	0.065	41.93	24.854	23.94	46.398	10.04	38.568	1.85
			0.8	0.048	30.86	26.352	16.34	35.415	9.32	30.145	3.12
		0.5	0.2	0.039	38.63	19.124	16.93	46.337	6.45	39.654	0.00
			0.5	0.058	29.82	22.14	12.53	44.654	5.93	39.078	0.19
			0.8	0.054	41.67	20.367	15.93	43.113	9.32	37.456	1.66
		0.8	0.2	0.064	42.78	19.341	20.43	42.154	11.53	32.002	1.39
			0.5	0.061	36.89	25.789	18.25	41.351	9.34	33.096	3.22
			0.8	0.049	31.09	16.369	19.42	40.021	11.31	35.235	1.04
Mean				**0.043**	**34.19**	**18.731**	**18.35**	**38.855**	**10.72**	**33.433**	**1.97**
5	20	0.2	0.2	0.024	20.39	13.458	10.75	49.258	6.32	39.325	0.32
			0.5	0.023	34.24	16.354	19.49	46.345	12.42	41.201	2.09
			0.8	0.021	39.09	14.784	23.95	50.289	14.12	41.074	1.48
		0.5	0.2	0.019	40.24	18.742	23.69	45.658	12.42	35.365	0.00
			0.5	0.019	29.73	13.118	19.75	47.458	8.45	41.154	1.88
			0.8	0.023	32.99	16.468	19.42	45.669	9.35	35.465	3.12
		0.8	0.2	0.02	29.16	16.754	18.49	51.205	10.53	45.987	2.65
			0.5	0.021	34.23	13.456	16.36	48.154	6.32	38.598	0.91
			0.8	0.027	21.42	17.447	12.42	47.845	4.53	36.45	0.00
	50	0.2	0.2	0.046	31.04	19.668	19.44	49.254	11.03	40.015	2.42
			0.5	0.039	40.43	20.185	20.23	56.344	9.34	49.124	4.23
			0.8	0.034	29.53	18.359	15.98	49.756	6.43	40.105	3.42
		0.5	0.2	0.039	39.34	20.369	19.01	47.458	9.53	46.326	2.94
			0.5	0.041	33.53	19.158	11.53	58.358	5.23	42.654	0.53
			0.8	0.045	29.54	16.224	13.23	56.655	6.43	50.125	1.92
		0.8	0.2	0.043	34.59	15.486	12.48	62.771	5.34	50.045	2.21
			0.5	0.039	26.83	19.899	9.32	56.109	3.22	39.108	0.00
			0.8	0.036	31.03	19.341	20.21	49.022	12.93	39.691	3.42
Mean				**0.031**	**32.08**	**17.182**	**16.99**	**50.978**	**8.55**	**41.767**	**1.86**
Mean of all				**0.041**	**34.76**	**17.577**	**16.94**	**43.195**	**8.88**	**36.755**	**1.94**

heuristics and then hybrid algorithm with an approximately 11% gap in obtained solutions with the optimal ones within very economical computational time.

7. Conclusions and future researches

In this study we have successfully implemented minimizing the problem of total weighted earliness and tardiness, makespan as well as jobs cost compressing and expanding depends on the amount of compression/expansion on unrelated parallel machines environment in which jobs processing times are controllable. Jobs due dates are distinct, no preemption of operations of each job is allowed and the setup time for each job on each machine is sequence-dependent. Also after starting the process by machine, no idle time could be inserted into the schedule and each machine could only process some predetermined jobs. A mixed integer programming (MIP) is proposed for the considered problem and solved via Lingo optimally in small size cases. Also a heuristic for obtaining initial sequence on parallel machines based on Just-In-Time (JIT) approach called ISETP is proposed and subsequently a parallel net benefit compression-net benefit expansion

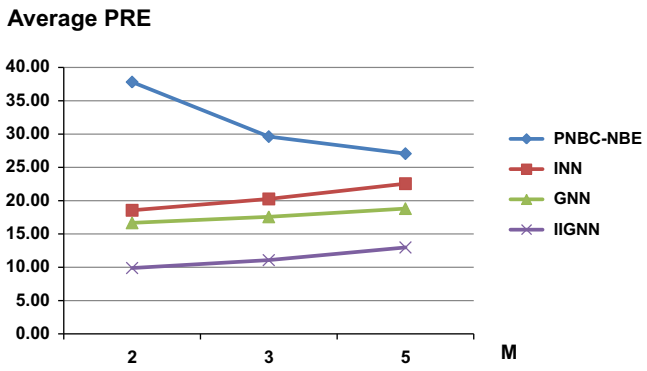

Fig. 4. The average *PRE* of different techniques in small problems.

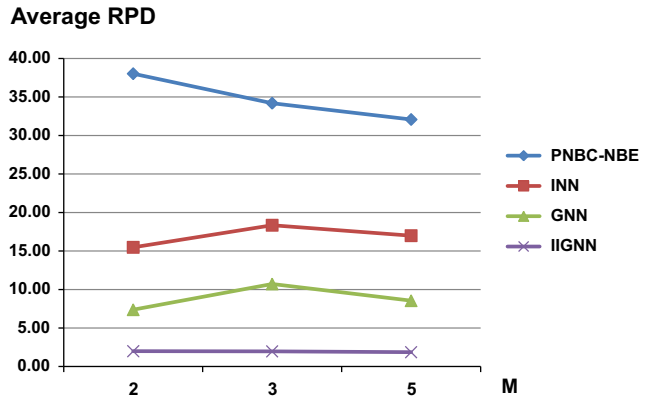

Fig. 5. The average *RPD* for different methods in medium-to-large problems.

(PNBC–NBE) heuristic for acquiring an optimal set of jobs compression/expansion processing times in a given sequence on each machine is then presented.

Two different types of instances are considered to be tested on this problem. In all categories, four algorithms including PNBC–NBE, ICA equipped with NBC–NBE called INN, GA equipped with NBC–NBE called GNN and a hybrid one so-called IIGNN are conducted. In small size cases which could be solved optimally, percentage relative error (*PRE*) and in medium-to-large size ones, relative percentage deviation (*RPD*) are employed as the performance measure. Computational results demonstrate the power and the high quality of our proposed hybrid heuristic for solving such a complex and applicable problem.

Extending the multi-objective model of the above-mentioned problem could be regarded as a direction for future research in order to observe more features in approaching to JIT policy. Also research efforts should be made in the future to implement the considered problem with the same approach on other environments such as job shop.

References

[1] Wang JB. Single machine scheduling with common due date and controllable processing times. Applied Mathematics and Computation 2006;174:1245–54.
[2] Baker KR. Elements of sequencing and scheduling.Hanover: HN; 1994.
[3] Baker KR, Scudder GD. Sequencing with earliness and tardiness penalties: a review. Operations Research 1990;38:22–36.
[4] Sun H, Wang G. Parallel machine earliness and tardiness scheduling with proportional weights. Computers & Operations Research 2003;30(5):801–8.
[5] Hall NG, Posner ME. Generating experimental date for computational testing with machine scheduling applications. Operations Research 2001;49:854–65.
[6] T'kindt V, Billaut J-C. Scheduling: Theory Models and Algorithms.Berlin: Springer; 2000.
[7] Allahverdi A, Gupta JND, Aldowaisan T. A review of scheduling research involving setup considerations, OMEGA. International Journal of Management Science 1999;27:219–39.
[8] Li K, Yang S. Non-identical parallel-machine scheduling research with minimizing total weighted completion times: models, relaxations and algorithms. Applied Mathematical Modeling 2009;33:2145–58.
[9] Kim E, Sung C, Lee I. Scheduling of parallel machines to minimize total completion time subject to s-precedence constraints. Computers & Operations Research 2009;36:698–710.
[10] Logendran R, McDonell B, Smucker B. Scheduling unrelated parallel machines with sequence-dependent setups. Computers & Operations Research 2007;34(11):3420–38.
[11] Koulamas C. The total tardiness problem: review and extensions. Operations Research 1994;42(6):1025–41.
[12] Sen T, Sulek JM, Dileepan P. Static scheduling research to minimize weighted and unweighted tardiness: a state-of-the-art survey. International Journal Production Economics 2002;83:1–12.
[13] Morton TE, Pentico DW. Heuristic Scheduling Systems.NY: John Wiley & Sons; 1993.
[14] Liaw CF, Lin YK, Cheng CY, Chen M. Scheduling unrelated parallel machines to minimize total weighted tardiness. Computers & Operations Research 2003;30:1777–89.
[15] Cheng TCE, Sin CCS. A state-of-the-art review of parallel-machine scheduling research. European Journal of Operational Research 1990;47:271–92.
[16] Rocha PL, Ravetti MG, Mateus GR, Pardalos PM. Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. Computers & Operations Research 2008;35(4):1250–64.
[17] Bank J, Werner F. Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties. Mathematical and Computer Modeling 2001;33(4–5):363–83.
[18] Sivrikaya F, Ulusoy G. Parallel machine scheduling with earliness and tardiness penalties. Computers & Operations Research 1999;26:773–87.
[19] Biskup D, Edwin Cheng TC. Multiple-machine scheduling with earliness, tardiness and completion time penalties. Computers & Operations Research 1999;26(1):45–57.
[20] Ventura JA, Kim D. Parallel machine scheduling with earliness-tardiness penalties and additional resource constraints. Computers & Operations Research 2003;30(13):1945–58.
[21] Kedad-Sidhoum S, Solis YR, Sourd F. Lower bounds for the earliness-tardiness scheduling problem on parallel machines with distinct due dates. European Journal of Operational Research 2008;189(3):1305–16.
[22] Toksari MD, Güner E. Parallel machine earliness/tardiness scheduling problem under the effects of position based learning and linear/nonlinear deterioration. Computers & Operations Research 2009;36(8):2394–417.
[23] Lin YK, Pfund ME, Fowler JW. Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. Computers & Operations Research 2011;38(6):901–16.
[24] Hsu CJ, Kuo WH, Yang DL. Unrelated parallel machine scheduling with past-sequence-dependent setup time and learning effects. Applied Mathematical Modeling 2011;35(3):1492–6.
[25] Kuo WH, Hsu CJ, Yang DL. Some unrelated parallel machine scheduling problems with past-sequence-dependent setup time and learning effects. Computers & Industrial Engineering 2011;61(1):179–83.
[26] Mor B, Mosheiov G. Total absolute deviation of job completion times on uniform and unrelated machines. Computers & Operations Research 2011;38(3):660–5.
[27] Bozorgirad MA, Logendran R. Sequence-dependent group scheduling problem on unrelated-parallel machines. Expert Systems with Applications 2012;39(10):9021–30.
[28] Vallada E, Ruiz R. Scheduling unrelated parallel machines with sequence dependent setup times and weighted earliness–tardiness minimization. In: Ríos-Mercado RZZ, Ríos-Solís YAA, editors. Just-in-Time Systems, Vol. 60. New York: Springer; 2012. p. 67–90.
[29] M'Hallah R, Al-Khamis T. Minimising total weighted earliness and tardiness on parallel machines using a hybrid heuristic. International Journal of Production Research 2011:1–26.
[30] Wang A, Huang Y, Taunk P, Magnin DR, Ghosh K, Robertson JG. Application of robotics to steady state enzyme kinetics: analysis of tight-binding inhibitors of dipeptidyl peptidase IV. Analytical Biochemistry 2003;321(2):157–66.
[31] Sørensen JF, Kragh KM, Sibbesen O, Delcour J, Goesaert H, Svensson B, et al. Potential role of glycosidase inhibitors in industrial biotechnological applications. Biochimica et Biophysica Acta (BBA) – Proteins & Proteomics 2004;1696(2):275–87.
[32] Vickson RG. Two single machine sequencing problems involving controllable job processing times. IIE Transactions 1980;12:258–62.
[33] Nowicki E, Zdrzalka S. A survey of results for sequencing problems with controllable processing times. Discrete Applied Mathematics 1990;26:271–87.
[34] Shabtay D, Steiner G. A survey of scheduling with controllable processing times. Discrete Applied Mathematics 2007;155:1643–66.
[35] Nowicki E, Zdrzałka S. A bicriterion approach to preemptive scheduling of parallel machines with controllable job processing times. Discrete Applied Mathematics 1995;63(3):237–56.
[36] Alidaee B, Ahmadian A. Two parallel machine sequencing problems involving controllable job processing times. European Journal of Operational Research 1993;70(3):335–41.
[37] Cheng TCE, Chen ZL, Li CL. Parallel-machine scheduling with controllable processing times. IIE Transactions 1996;28:177–80.

[38] Jansen K, Mastrolilli M. Approximation schemes for parallel machine scheduling problems with controllable processing times. Computers & Operations Research 2004;31(10):1565–81.
[39] Gurel S, Akturk MS. Scheduling parallel CNC machines with time/cost trade-off considerations. Computers & Operations Research 2007;34(9):2774–89.
[40] Gürel S, Körpeoğlu E, Aktürk MS. An anticipative scheduling approach with controllable processing times. Computers & Operations Research 2010;37 (6):1002–13.
[41] Turkcan A, Akturk MS, Storer RH. Predictive/reactive scheduling with controllable processing times and earliness-tardiness penalties. IIE Transactions 2009;41(12):1080–95.
[42] Aktürk M, Atamtürk A, Gürel S. Parallel machine match-up scheduling with manufacturing cost considerations. Journal of Scheduling 2010;13(1):95–110.
[43] Leyvand Y, Shabtay D, Steiner G, Yedidsion L. Just-in-time scheduling with controllable processing times on parallel machines. Journal of Combinatorial Optimization 2010;19(3):347–68.
[44] Li K, Shi Y, Sl Yang, Cheng By. Parallel machine scheduling problem to minimize the makespan with resource dependent processing times. Applied Soft Computing 2011;11(8):5551–7.
[45] Peng J, Liu B. Parallel machine scheduling models with fuzzy processing times. Information Sciences 2004;166(1–4):49–66.
[46] Alcan P, Başlıgil H. A genetic algorithm application using fuzzy processing times in non-identical parallel machine scheduling problem. Advances in Engineering Software 2012;45(1):272–80.
[47] Balin S. Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation. Information Sciences 2011;181 (17):3551–69.
[48] Chyu CC, Chang WS. Optimizing fuzzy makespan and tardiness for unrelated parallel machine scheduling with archived metaheuristics. The International Journal of Advanced Manufacturing Technology 2011;57(5–8):763–76.
[49] Kwong CK, Mok PY, Wong WK. Determination of fault-tolerant fabric-cutting schedules in a just-in-time apparel manufacturing environment. International Journal of Production Research 2006;44(21):4465–90.
[50] Mok PY, Kwong CK, Wong WK. Optimisation of fault-tolerant fabric-cutting schedules using genetic algorithms and fuzzy set theory. European Journal of Operational Research 2007;177(3):1876–93.
[51] Cheng RW, Gen MS, Tozawa T. Minmax earliness/tardiness scheduling in identical parallel machine system using genetic algorithms. Computers & Industrial Engineering 1995;29(1–4):513–7.
[52] Funda SS, Gunduz U. Parallel machine scheduling with earliness and tardiness penalties. Computers & Operations Research 1999;26(8):773–87.
[53] Min L, Cheng W. Hybrid genetic algorithm method for identical parallel machine earlines/tardiness scheduling problem. Acta Autom Sin 2000;26 (2):258–62.
[54] Kayvanfar V, Mahdavi I, Komaki GM. Single machine scheduling with controllable processing times to minimize total tardiness and earliness. Computers & Industrial Engineering 2013 65(1): 166–75.
[55] Hillier FS, Lieberman GJ. Introduction to operations research. 7th ed.New York: McGraw-Hill; 2001.
[56] Khabbazi A, Atashpaz E, Lucas C. Imperialist competitive algorithm for minimum bit error rate beamforming. International Journal of Bio-Inspired Computation 2009;1:125–33.
[57] Kayvanfar V, Zandieh M. The economic lot scheduling problem with deteriorating items and shortage: an imperialist competitive algorithm. The International Journal of Advanced Manufacturing Technology 2012;62(5):759–73.
[58] Holland JH. Adaptation in natural and artificial systems.Ann Arbor: The University of Michigan Press; 1975.
[59] Goldberg DE. Genetic algorithms in search, optimization and machine learning.Reading: Addison-Wesley; 1989.
[60] Park BJ, Choi HR, Kim HS. A hybrid genetic algorithm for the job shop scheduling problems. Computers & Industrial Engineering 2003;45:597–613.
[61] Back T, Fogel D, Michalawecz Z. Handbook of evolutionary computation. Oxford: Oxford University Press; 1997.