Int J Adv Manuf Technol (2015) 77:545–563
DOI 10.1007/s00170-014-6461-8

ORIGINAL ARTICLE

A bi-objective identical parallel machine scheduling problem with controllable processing times: a just-in-time approach

M. H. Fazel Zarandi · Vahid Kayvanfar

Received: 9 December 2013 / Accepted: 1 October 2014 / Published online: 21 October 2014

Abstract In this research, a bi-objective scheduling problem with controllable processing times on identical parallel machines is investigated. The direction of this paper is mainly motivated by the adoption of the just-in-time (JIT) philosophy on identical parallel machines in terms of bi-objective approach, where the job processing times are controllable. The aim of this study is to simultaneously minimize (1) total cost of tardiness, earliness as well as compression and expansion costs of job processing times and (2) maximum completion time or makespan. Also, the best possible set amount of compression/expansion of processing times on each machine is acquired via the proposed "bi-objective parallel net benefit compression-net benefit expansion" (BPNBC-NBE) heuristic. Besides that, a sequence of jobs on each machine, with capability of processing all jobs, is determined. In this area, no inserted idle time is allowed after starting machine processing. For solving such bi-objective problem, two multi-objective meta-heuristic algorithms, i.e., non-dominated sorting genetic algorithm II (NSGAII) and non-dominated ranking genetic algorithm (NRGA) are applied. Also, three measurement factors are then employed to evaluate the algorithms' performance. Experimental results reveal that NRGA has better convergence near the true Pareto-optimal front as compared to NSGAII, while NSGAII finds a better spread in the entire Pareto-optimal region.

Keywords Just-in-time · Makespan · Controllable processing times · Multi-objective · Identical parallel machines

M. H. F. Zarandi (✉) · V. Kayvanfar
Department of Industrial Engineering, Amirkabir University of Technology, 424 Hafez Ave, 15875-4413 Tehran, Iran
e-mail: zarandi@aut.ac.ir

V. Kayvanfar
e-mail: v.kayvanfar@aut.ac.ir

1 Introduction

Tardiness and earliness have been considered as two criteria associated with completing a job at a time different from its given due date in a large amount of researches since both earliness and tardiness affect on the system efficiency and must be taken into account. In other words, a major force of researches 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 (E/T) penalties. In fact, JIT philosophy tries to recognize and remove waste elements as over transportation, production environment, waiting time (either in production or services), inventory/stock, faulty goods, processing, and movement. Since earliness could represent manufacturer concerns and tardiness could embrace both customer and manufacturer concerns while none of them is desirable, we are going to simultaneously minimize weighted tardiness and earliness as well as makespan in terms of a practical bi-objective problem on parallel machine 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 [1]. Owing to their imposed additional costs to production systems, both earliness and tardiness must be minimized since neither of them is desirable. This category of problems has been shown as non-deterministic polynomial-time (NP)-hard ones [2, 3].

The greater part of the researches on E/T scheduling problems deals with the single machine; however, in the past years, many researchers have investigated multi-criteria parallel

machine scheduling problems with two or more criteria that apply simultaneously or hierarchically in the objective function. The majority of earlier studies on parallel machine scheduling have dealt with performance criteria such as mean flow time, mean tardiness, makespan, and mean lateness. In accordance with increasing current trends toward JIT policy, the traditional measures of performance are no longer applicable. In its place, the emphasis has shifted towards E/T scheduling taking earliness in addition to tardiness into account [4]. Baker and Scudder [4] presented the first survey on E/T scheduling problems. Also, the E/T problem has proven to be NP-hard [2, 5].

Most of the real-world scheduling problems are naturally multi-objective. These objectives are often in conflict with each other. In such cases, managers try to find the best solution that satisfies all the considerations simultaneously. Multi-objective optimization with such conflicting objective functions gives rise to a set of optimal solutions, in place of one optimal solution. The main reason for the optimality of many solutions is that no solution could be alone considered to be better than any other with respect to all objective functions. These optimal solutions are called "Pareto-optimal solutions." There is a large range of industries with trade-offs in their objectives such as building construction, aircraft, etc.

With respect to the real-life situation, most of the classical scheduling models assume that job processing times are fixed, while the processing times depend on amount of resources such as budgets, capabilities of facilities, manpower, etc. Obviously, this assumption neglects some realistic constraints. The controllable processing time means that each job could process in a shorter or longer time depending 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. The chosen processing times have an effect on both the scheduling performance and manufacturing cost, whenever the job processing times are controllable. As an applicable case, in chemical industry, the processing time of a job is increased by an inhibitor or reduced by means of a catalyzer. An inhibitor is any agent that interferes with the activity of an enzyme. As a matter of fact, 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. [6] and Sørensen et al. [7].

In this paper, in order to exhibit real-world situation, we formulate a multi-objective scheduling problem as a bi-objective one which simultaneously minimizes (1) total cost of tardiness, earliness as well as compression and expansion costs of job processing times and (2) maximum completion time so-called makespan. In practice, the usage of both objectives is well-justified, whereas the first objective actually focuses on the make-to-order (MTO) philosophy in supply chain management and production theory: an item should be delivered exactly when it is required by the customer while makespan minimization implies the maximization of the throughput. In order to consider more complexity of real-world situations, in this paper, we investigate the non-preemptive scheduling problem with n jobs on m identical parallel machines in a bi-objective approach in which job processing times are controllable. The main contributions of this research could be mentioned as follows:

- To the best of the authors' knowledge, there exists no accomplished research in which such criteria have been investigated on identical parallel machine environment in terms of bi-objective approach where the job processing times are controllable and no preemption is allowed.
- In this research, "controllable processing times" mean the jobs could be either compressed or expanded up to a certain limit, while in almost all of the already done researches, the controllable processing times signify only reduction in processing time value. This matter, despite simplicity in appearance, causes the essential differences in calculations scheme and complexity. In better words, the problem transforms to a more complex one with regard to such a concept which has itself a considerable novelty.
- The best possible set amount of compression/expansion of processing times on each machine is achieved via the proposed "bi-objective parallel net benefit compression-net benefit expansion" (BPNBC-NBE) heuristic for the first time. Also, owing to the importance of assigning jobs on parallel machines based on minimizing total tardiness and earliness as well as makespan at once, a heuristic is proposed in this study so as to allocate the jobs on parallel machines considering such criteria which is noteworthy.
- Two well-known multi-objective meta-heuristic algorithms, i.e., non-dominated sorting genetic algorithm II (NSGAII) and non-dominated ranking genetic algorithm (NRGA) are customized to the studied problem in this research for solving such a bi-objective problem. A considerable amount of efforts have been accomplished for doing so.

The rest of the paper is organized as follows: Section 2 explains related literature. In Section 3, the problem formulation including assumptions, notations, and mathematical model are described. In Section 4, the proposed bi-objective parallel net benefit compression/net benefit expansion (BPNBC-NBE) is described. Section 5 explains the employed multi-objective algorithms. Section 6 consists of computational results, and finally, Section 7 includes conclusions and future works.

2 Literature review

There are several papers in which earliness and tardiness criteria are simultaneously studied on parallel machines. Of

them, Sivrikaya and Ulusoy [8] developed a genetic algorithm (GA) approach to attack the scheduling problem of a set of independent jobs on parallel machines with earliness and tardiness penalties. Sun and Wang [2] studied the problem of scheduling n jobs with a common due date and proportional early and tardy penalties on m identical parallel machines. They showed that this problem is NP-hard and proposed a dynamic programming algorithm to solve it. Cheng et al. [9] addressed the E/T scheduling problem in identical parallel machine system with the objective of minimizing the maximum weighted absolute lateness. Kedad-Sidhoum et al. [10] addressed the parallel machine scheduling problem in which the jobs have distinct due dates with earliness and tardiness costs. They proposed new lower bounds for the considered problem. Su [11] addressed the identical parallel machine scheduling problem in which the total earliness and tardiness about a common due date are minimized subject to minimum total flow time. Drobouchevitch and Sidney [12] considered a scheduling problem of n identical non-preemptive jobs with a common due date on m uniform parallel machines. Biskup et al. [13] studied scheduling a given number of jobs on a specified number of identical parallel machines so as to minimize total tardiness. Xi and Jang [14] considered the performances of apparent tardiness cost-based (ATC-based) dispatching rules with the goal of minimizing total weighted tardiness on identical parallel machines with unequal future ready time and sequence-dependent setup times.

In the last few decades, a large amount of researchers have studied multi-objective parallel machine scheduling problems. Of them, one could mention to Shmoys and Tardos [15] which studied unrelated parallel machine scheduling to minimize the makespan and total cost of the schedule. Coffman and Sethi [16] proposed two heuristics for identical parallel machine scheduling problem so as to minimize the makespan subject to the minimum total flow time. Lin and Liao [17] considered makespan minimization for identical parallel machines subject to minimum total flow time. Ruiz-Torres and Lopez [18] proposed four heuristics for identical parallel machine scheduling problem so as to minimize the makespan as well as the number of tardy jobs. Suresh and Chaudhuri [19] proposed a tabu search (TS) method to minimize both maximum tardiness and maximum completion time for unrelated parallel machines. Ruiz-Torres et al. [20] proposed heuristics based on a simulated annealing (SA) and a neighborhood search to minimize the average flow time as well as the number of tardy jobs on identical parallel machines. Chang et al. [21] proposed a two-phase sub-population GA (TPSPGA) so as to minimize the makespan in addition to total tardiness on parallel machines. Gao et al. [22] presented a multi-objective scheduling model (MOSP) on non-identical parallel machines in order to minimize the maximum completion time among all the machines (makespan) and the total earliness/tardiness penalty of all the jobs. In another similar work, Gao [23] considered jobs on non-identical parallel machines with processing constraints and presented GAs to minimize the makespan and total earliness/tardiness penalties. Tavakkoli-Moghaddam et al. [24] presented a two-level mixed integer programming model of scheduling N jobs on M parallel machines that minimize two objectives, namely the number of tardy jobs and the total completion time of all the jobs. Radhakrishnan and Ventura [25] and Sivrikaya and Ulusoy [8] minimized total earliness and tardiness cost in JIT production environments and solved them via GA and mixed integer programming.

There is a significant relevance between early/tardy (E/T) scheduling problems and concept of controllable processing times since by controlling the job processing times as far as possible, earliness and tardiness as well as makespan could be decreased, i.e., by compressing/expanding the job processing times, earliness, and tardiness and also makespan may be reduced. Almost certainly, Vickson [26] has studied one of the first researches on controllable processing time in scheduling environment, with the goal of minimizing the total processing cost incurred due to job processing time compression as well as the total flow time. Researches on scheduling problem with controllable processing times and linear cost functions up to 1990 are surveyed by Nowicki and Zdrzalka [27]. Lee [28] studied single machine scheduling problem regarding controllable processing times with the goal of minimizing total job processing cost plus the average flow cost. The tolerance ranges of job processing times were determined in this research so that the optimal sequence remains unchanged. Liman et al. [29] considered a single machine common due window scheduling problem in which the job processing times could be reduced up to a certain limit. Their objective was composed of costs associated with the window location, its size, processing time reduction as well as job earliness and tardiness. Shabtay and Steiner [30] have accomplished a complete survey on scheduling with controllable processing times. Gurel and Akturk [31] surveyed the identical parallel CNC machines with controllable processing time in which a time/cost trade-off consideration is conducted. In non-identical parallel machine environment, Alidaee and Ahmadian [32] surveyed this category of scheduling area with linear compression cost functions to minimize (1) the total compression cost plus the total flow time and (2) the total compression cost and the weighted sum of earliness and tardiness penalties. Wan [33] studied a non-preemptive single machine common due window scheduling problem where the job processing times are controllable with linear costs and the due window is movable. The objective of this research was to find a job sequence, a processing time for each job as well as a position of the common due window so as to minimize the total cost of weighted earliness/tardiness and processing time compression. Shakhlevich and Strusevich [34]

proposed a unified approach so as to solve preemptive uniform parallel machine scheduling problems in which the job processing times are controllable. They also showed that the total compression cost minimization problem in which all due dates should be met could be formulated in terms of maximizing a linear function over a generalized polymatroid. Shabtay [35] studied a batch delivery single machine scheduling problem in which the due dates are controllable. His objective was to minimize holding, tardiness, earliness, due date assignment as well as delivery costs. Aktürk et al. [36] 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. Leyvand et al. [37] studied the JIT scheduling problem on parallel machine in which the job processing times are controllable. Their goal was (1) maximizing the weighted number of jobs which are exactly completed at their due date and (2) minimizing the total resource allocation cost. They considered four models for treating the two criteria. Yin and Wang [38] investigated single machine scheduling problem with learning effect where the jobs have controllable processing times. They focused on two objectives separately: (1) minimizing a cost function comprising makespan, total absolute differences in completion times, total compression cost, and total completion time, (2) minimizing a cost function comprising total waiting time, makespan, total compression cost, and total absolute differences in waiting times. Li et al. [39] 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. Jansen and Mastrolilli [40] studied the identical parallel machine makespan problem with controllable processing time where each job is allowed to compress its processing time in return for compression cost.

Niu et al. [41] studied two decompositions for bi-criteria job shop scheduling problem with discrete controllable processing times. To tackle the problem, they proposed assignment-first decomposition (AFD) as well as sequencing-first decomposition (SFD) procedures. Renna [42] addressed the policy so as to manage job shop scheduling area in which the jobs are controllable. His proposed policy concerned the evaluation of resource workload. Also, he applied two approaches so as to allocate the resources to the machines. Low et al. [43] studied the unrelated parallel machine scheduling problem with eligibility constraints in which the job processing times are controllable through the assignment of a non-renewable common resource. Their goal was to allocate the jobs onto the machines and to assign the resource so as to minimize the makespan. Kayvanfar et al. [44] investigated a single machine with controllable processing times where each job could be either compressed or expanded to a given extent, while almost certainly in most of the other researches, theretofore, compressing the jobs was only regarded in the modeling. They tried to simultaneously minimize total earliness/tardiness as well as job compression/expansion cost. Kayvanfar et al. [45] also proposed a drastic hybrid heuristic algorithm besides two meta-heuristics so as to tackle the single machine with controllable processing times. In a more complete recently done research, Kayvanfar et al. [46] addressed unrelated parallel machines with controllable processing times with the single goal of minimizing 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.

3 Problem definition and modeling

A bi-objective non-linear mathematical model for identical parallel machine scheduling problem is proposed in this section to simultaneously minimize (1) the total tardiness and earliness penalties as well as job compression/expansion costs and (2) maximum completion time (makespan). All processing times, due dates, and maximum amount of job compression/expansion are assumed to be integer. The considered problem is formulated according to the following assumptions.

3.1 Assumptions

- All jobs are available in time zero and they could be processed only by one machine eventually.
- The normal processing time could be compressed by an amount of x_j or expanded by an amount of x'_j which necessitates a unit cost of compression or expansion, respectively.
- Processing a job at its normal processing time will arise no additional processing cost.
- All machines are capable of processing all jobs and each machine could work only on one job at a time.
- The jobs are independent of each other.
- After starting the process by machine, no idle time could be inserted into the schedule.
- All machines are identical and a job could be processed by any free machine.
- Jobs preemption is not allowed.
- Transportation time between machines and machine setup times are negligible.
- The process time of each job on each machine is the same.
- Number of jobs and machines are fixed.
- No breakdown is allowed, i.e., all machines are available throughout the scheduling period.

3.2 Notations

3.2.1 Subscripts

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

3.2.2 Input parameters

$_{pj}$	Normal processing time of job j
p'_j	Crash (minimum allowable) processing time of job j
p''_j	Expansion (maximum allowable) processing time of job j
c_j	Compression unit cost of job j
c'_j	Expansion unit cost of job j
α_j	The earliness unit penalty of job j
β_j	The tardiness unit penalty of job j
d_j	Due date of job j
L_j	Maximum amount of job j compression, $L_j=p_j-p_j'$
L'_j	Maximum amount of job j expansion, $L_j'=p_j''-p_j$

3.2.3 Decision variables

C_j	Completion time of job j
$C_{\max}$	Maximum completion time of all jobs (makespan)
y_{jkm}	1 if job j is assigned on machine m in priority k; otherwise, it is zero
E_j	Earliness of job j; $E_j=\max\{0, d_j-C_j\}$
T_j	Tardiness of job j; $T_j=\max\{0, C_j-d_j\}$
x_j	Amount of job j compression, $0\le x_j\le L_j$
x'_j	Amount of job j expansion, $0\le x'_j\le L'_j$

A non-preemptive parallel machine scheduling problem with N jobs on M ($N>M$) identical parallel machines is simultaneously available at time zero. A compression or expansion unit cost (c_j or c'_j) is occurred, if the processing time is reduced or increased by one time unit, respectively. It is evident that each job could only be compressed or expanded. The goal is to determine the job sequence as well as an optimal set amount of compression/expansion of processing times on each machine simultaneously so that both considered objectives, i.e., (1) total weighted earliness and tardiness penalties as well as job compression/expansion costs and (2) makespan are minimized.

3.3 The mathematical model

$$\text{Min}\, Z_1 = \sum_{j=1}^{N} \left(\alpha_j E_j + \beta_j T_j + c_j x_j + c'_j x'_j\right) \tag{1}$$

$$\text{Min}\, Z_2 = C_{\max} \tag{2}$$

The first objective (Eq. (1)) includes earliness penalties, tardiness penalties, and cost of jobs compressing and expanding processing times, depending on the amount of compression/expansion. Equation (2) minimizes the maximum completion time so-called makespan. Makespan is equivalent to the completion time of the last job leaving the system. Pinedo [47] showed that machine utilization could be increased if the makespan be minimum. The utilization for bottleneck or near bottleneck equipment is closely related to the throughput rate of the system. Consequently, reducing makespan should also lead to a higher throughput rate. This problem is subjected to the following constraints:

$$\sum_{m=1}^{M} \sum_{k=1}^{K} y_{jkm} = 1 \quad \forall j; \tag{3}$$

Equality (3) ensures that job j could be processed only in one priority k and onto one machine.

$$\sum_{j=1}^{N} y_{jkm} \le 1 \quad \forall k, \; m; \tag{4}$$

Inequality (4) guarantees that only one job could be processed in priority k on machine m.

$$C_j - d_j = T_j - E_j \quad \forall j; \tag{5}$$

Equation (5) defines the earliness and tardiness of job j. Since each job could only be tardy or early, if it could not be delivered timely, so it is obvious that T_j and E_j cannot take value simultaneously.

$$y_{j1m}\left(p_j - x_j + x_j^{'}\right) \le C_j \quad \forall j, m; \tag{6}$$

$$\left(\sum_{m=1}^{M}\sum_{k\ge 2}^{K}\sum_{i\ne j}^{N} y_{ik-1m}\, y_{jkm}\, C_i\right) + p_j - x_j + x_j^{'} = C_j \quad \forall j; \tag{7}$$

The first job on each machine is determined by means of constraint (6). Also, constraint (7) is employed so as to identify the rest of job sequence on all machines. As a matter of fact, constraints (6) and (7) jointly ensure that only after starting the process by machine, no idle time could be inserted into the schedule and no job preemption is also allowed.

$$C_{\max} \ge C_j \forall j; \tag{8}$$

Eq. (8) ensures that the makespan is greater than any of job completion times.

$$L_j = p_j - p_j^{'} \ge x_j \quad \forall j; \tag{9}$$

$$L_j^{'} = p_j^{''} - p_j \ge x_j^{'} \quad \forall j; \tag{10}$$

Inequalities (9) and (10) limit the amount of compression and expansion of each job.

$$y_{jkm} \in \{0, 1\} \quad \forall j, k, m; \tag{11}$$

$$T_j, E_j, x_j, x_j^{'} \ge 0 \quad \forall j; \tag{12}$$

Constraints (11) and (12) provide the logical binary and non-negativity integer necessities for the decision variables, respectively.

4 The proposed BPNBC-NBE

This section proposes a heuristic algorithm to calculate the best possible set amount of compression/expansion of processing times on each machine in a bi-objective approach which is called "bi-objective parallel net benefit compression/net benefit expansion (BPNBC-NBE)" and its expanded version of PNBC-NBE algorithm for single-objective parallel machines proposed by Kayvanfar et al. [46]. The PNBC-NBE algorithm is also an expanded version of NBC-NBE technique which is proposed by [44] for a single machine. The concept of NBC-NBE algorithm is close to the *NBC* one with a main idea similar to marginal cost analysis used in PERT/CPM with time/cost trade-off [48].

Since in this study earliness and tardiness as the first objective besides makespan as the second objective should be minimized, the proposed BPNBC-NBE heuristic should minimize these objectives. Accordingly, a heuristic called ISETMP is presented in the following subsection in order to generate initial solution via allocating the jobs on the parallel machines which is utilized within the BPNBC-NBE algorithm. It is assumed that applying such heuristic for finding the initial sequences (solutions) may enhance the final obtained solutions' quality.

4.1 Initial sequence on parallel machines considering E/T and makespan minimization

Presenting a heuristic technique to be able to minimize earliness, tardiness as well as makespan simultaneously is difficult on a given machine since earliness and tardiness are two concepts which have reverse meaning. In better words, by reducing each of them, the other one increases and vice versa. In this context, owing to the importance of assigning jobs on parallel machines based on minimizing total tardiness and earliness as well as makespan at once, we present such a heuristic method to allocate the jobs on parallel machines considering such criteria. The proposed heuristic called initial sequence based on earliness-tardiness-makespan criteria on parallel machine (ISETMP) is described as follows:

Procedure 1. ISETMP

1. Sort jobs according to earliest due date (EDD) criterion and put them in *unscheduled* jobs category called "US."
2. Assign the first job to the first machine and set it in *scheduled* job category called "*S*."

   Do the following steps until no job is found in the US category:
3. Assign next *unscheduled* job in EDD order to each 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 ($C_{\max_m}$) and add the result with $C_{\max_m}$, called $U_m = \left|C_{\max_m} - d_j\right| + C_{\max_m}$. The term $\left|C_{\max_m} - d_j\right|$ tries to minimize the first objective and the

second term ($C_{\max_m}$) makes an effort to minimize the second objective. If more than one job have the same due dates, select one job randomly at first.

5. Select the assignment which has resulted in the minimum U_m and assign that job to such a machine. If two or more U_m values are equal, select one machine randomly.
6. Transfer this job from the US category into the scheduled one, S.
7. Update $C_{\max}$ of each machine and go to step 3.

An example is here solved to illustrate the ISETMP performance.

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

Iteration #1:

Step 1 Sort jobs according 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 the 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_m as follows:

$$U_1 = |C_{\max_1} - d_2| + C_{\max_1} = |10-5| + 10 = 15$$
$$U_2 = |C_{\max_2} - d_2| + C_{\max 2} = |6-5| + 6 = 7$$
$$U_3 = |C_{\max_3} - d_2| + C_{\max 3} = |6-5| + 6 = 7$$

Step 5 Select the min U_m and assign J_2 to such a machine. Since two values are equal, machine #2 is chosen randomly.

Step 6 Transfer this job from the US category to the 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, $C_{\max_1} = 4,\ C_{\max_2} = 6,\ C_{\max_3} = 0$.

Iteration #2:

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

J	1	2	3	4	5	6	7	8
P_j	4	6	5	7	5	6	4	6
d_j	5	5	11	6	13	13	11	20
α_j	0.5	1	1	1.25	1.5	1	1.5	0.5
β_j	0.5	0.5	1.25	0.5	0.5	1	3	0.5

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

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

Step 4 Compute all U_m as follows:

$$U_1 = |C_{\max_1} - d_4| + C_{\max_1} = |11-6| + 11 = 16$$
$$U_2 = |C_{\max_2} - d_4| + C_{\max_2} = |13-6| + 13 = 20$$
$$U_3 = |C_{\max_3} - d_4| + C_{\max_3} = |7-6| + 7 = 8$$

Step 5 Select the min U_m and assign J_4 to such machine, i.e., machine #3.

Step 6 Transfer this job from the US category to the 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, $C_{\max_1} = 4, C_{\max_2} = 6, C_{\max_3} = 7$.

Iteration #3:

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

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

Step 4 Compute all U_m as follows:

$$U_1 = |C_{\max_1} - d_3| + C_{\max_1} = |9-11| + 9 = 11$$
$$U_2 = |C_{\max_2} - d_3| + C_{\max_2} = |11-11| + 11 = 11$$
$$U_3 = |C_{\max_3} - d_3| + C_{\max_3} = |12-11| + 12 = 13$$

Step 5 Select the min U_m. Since two values are equal, machine #2 is selected by chance.

Step 6 Transfer this job from the US category to the 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,

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

Step 4 Compute all U_m as follows:

$$U_1 = |C_{\max_1} - d_7| + C_{\max_1} = |8-11| + 8 = 11$$
$$U_2 = |C_{\max_2} - d_7| + C_{\max_2} = |15-11| + 15 = 16$$
$$U_3 = |C_{\max_3} - d_7| + C_{\max_3} = |11-11| + 11 = 11$$

Step 5 Select the min U_m. Since two values are equal, machine #3 is selected randomly.

Step 6 Transfer this job from the US category to the 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, $C_{\max_1} = 4,\ C_{\max_2} = 11,\ C_{\max_3} = 11$.

Iteration #5:

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

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

Step 4 Compute all U_m as follows:

$$U_1 = |C_{\max_1} - d_5| + C_{\max_1} = |9-13| + 9 = 13$$
$$U_2 = |C_{\max_2} - d_5| + C_{\max_2} = |16-13| + 16 = 19$$
$$U_3 = |C_{\max_3} - d_5| + C_{\max_3} = |16-13| + 16 = 19$$

Step 5 Select the min U_m and assign J_5 to such machine, i.e., machine #1.

Step 6 Transfer this job from the US category to the 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, $C_{\max_1} = 9,\ C_{\max_2} = 11,\ C_{\max_3} = 11$.

Iteration #6:

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

$$C_{\max_1} = 9 + 6 = 15,\ C_{\max_2} = 11 + 6 = 17,\ C_{\max_3} = 11 + 6 = 17.$$

Step 4 Compute all U_m as follows:

$$U_1 = |C_{\max_1} - d_6| + C_{\max_1} = |15-13| + 15 = 17$$
$$U_2 = |C_{\max_2} - d_6| + C_{\max_2} = |17-13| + 17 = 21$$
$$U_3 = |C_{\max_3} - d_6| + C_{\max_3} = |17-13| + 17 = 21$$

Step 5 Select the min U_m and assign J_6 to this machine, i.e., machine #1.

Step 6 Transfer this job from the US category to the 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, $C_{\max_1} = 15,\ C_{\max_2} = 11,\ C_{\max_3} = 11$.

Iteration #7:

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

$$C_{\max_1} = 15 + 6 = 21,\ C_{\max_2} = 11 + 6 = 17,\ C_{\max_3} = 11 + 6 = 17.$$

Step 4 Compute all U_m as follows:

$$U_1 = |C_{\max_1} - d_8| + C_{\max_1} = |21-20| + 21 = 22$$
$$U_2 = |C_{\max_2} - d_8| + C_{\max_2} = |17-20| + 17 = 20$$
$$U_3 = |C_{\max_3} - d_8| + C_{\max_3} = |17-20| + 17 = 20$$

Step 5 Select the min U_m and assign J_8 to this machine, i.e., machine #2.

Step 6 Transfer this job from the US category to the 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, $C_{\max_1} = 15,\ C_{\max_2} = 17,\ C_{\max_3} = 11$.

As could be seen, there is no job in the US category and therefore the algorithm is terminated. Also, the $C_{\max}$ of system equals to 17 (Fig. 1).

Now, we introduce the proposed BPNBC-NBE which takes advantage from the net benefit compression/expansion concept on parallel machines in a bi-objective approach which is described as follows:

Procedure 2. BPNBC-NBE

1. Assign the jobs on parallel machines using ISETMP heuristic, as initial sequence.
2. Apply NBC-NBE so as to determine the amount of reduction/expansion of job processing times on each machine.

Do the two following steps (3–4) until stop criterion is met:

3. Swap jobs and their corresponding machines randomly in order to reduce the sum of total "earliness and tardiness as well as makespan" values.

1. Accept the new sequence, if both objective values reduce. Go to step 4.

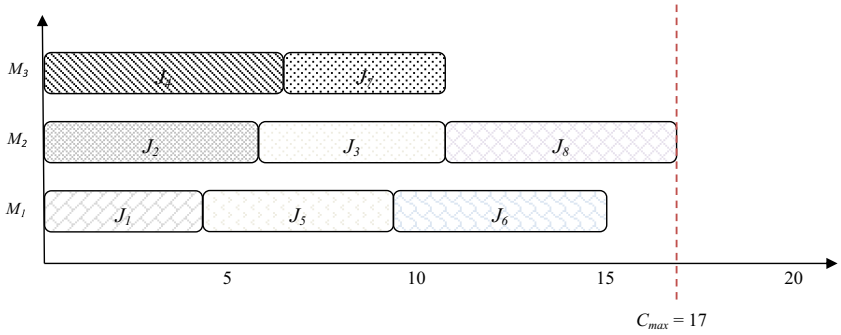

Fig. 1 Gantt chart of assigned jobs on parallel machines

2. Reject the new sequence, if both objective values increase. Go to step 3.
3. If one objective reduces and the other one increases, accept the new sequence if total reduction is more than total increase and go to step 4, or else keep the previous sequence. Go to step 3.

4. Apply NBC-NBE to determine whether a given job could be further compressed or expanded. Go to step 3.
5. Determine the final sequence and calculate the final amount of compression/expansion of job processing times.

In the aforementioned procedure, the stop criterion is defined as maximum number of non-improvement iterations which is set at 30.

5 Resolution methods

A mono-objective optimization algorithm will be stopped upon finding an optimal solution; however, it would be an ideal case to find only a single solution for a multi-objective problem in terms of non-dominance criterion and most often the process of optimization causes more than one solution. Evolutionary algorithms (EAs) are potent stochastic search methods which mimic the Darwinian principles of natural selection (survival of the fittest) and are well suited for solving optimization problems with difficult search landscapes (e.g., multimodal search spaces, multiple objectives, large solution spaces, constraints, and non-linear and non-differentiable functions) [49]. To date, a large number of different multi-objective evolutionary algorithms (MOEAs) have been suggested. Of them are strength Pareto evolutionary algorithm II (SPEAII) [50], niched Pareto genetic algorithm (NPGA) [51], non-dominated sorting genetic algorithm II (NSGAII) [52], and non-dominated ranking genetic algorithm (NRGA) [53]. Comprehensive information about other MOEA algorithms could be found in Coello et al. [54] and Deb [55]. These MOEAs employ Pareto dominance notion to guide the search and afterward return the Pareto-optimal set as the best result. These algorithms have two objectives: (1) convergence to the Pareto-optimal set and (2) maintenance of diversity in solutions of the Pareto-optimal set [52]. In better words, two common features on all operators were at first allocating fitness to population members based on non-dominated sorting and afterward preserving diversity among solutions of the same non-dominated front. In order to solve the sample generated instances, two well-known MOEAs, i.e., NSGAII and NRGA are employed in this study which are of elite multi-objective EAs and execute based on the non-dominance concept. The NRGA takes advantage of the sorting algorithm in NSGAII. Moreover, for the rank-based roulette wheel selection, Al Jadaan et al. [53] utilized a revised roulette wheel selection algorithm where each member of population is assigned a fitness value equal to its rank in the population; the highest rank has the highest probability to be selected. The probability is calculated as follows [53]:

$$P_i = \frac{2 \times \text{Rank}}{N \times (N+1)} \tag{13}$$

Where N is the number of individuals in the population. Producing a set of Pareto-optimal solutions, besides being capacitated decision-maker to select one of them in order to optimize the model, is the main advantage of these techniques.

5.1 Solution representation

In this research, an encoding scheme proposed by Cheng et al. [9] is employed so as to represent a solution (chromosome) for the considered problem. In this encoding scheme, integers are applied to represent all job sequences while / is used to represent the jobs partitioning on parallel machines. As an instance, supposing there are ten jobs and three machines, so the chromosome could be presented as [3 9 2/4 6 8/5 1 7 10]. This schedule means that jobs 3, 9, and 2 on machine 1; jobs 4, 6, and 8 on machine 2; and jobs 5, 1, 7, and 10 on machine 3 will be processed. Also, an initial set of solutions are generated so as to make up the initial population.

5.2 Fitness ranking (non-dominated sorting algorithm)

In order to sort a population of size N' according to the level of non-domination, each solution (chromosome) must be compared with every other solution in the population to realize if it is dominated. In fact, the thought behind the non-dominated sorting process is that a ranking selection scheme is employed to give emphasis to good points and a niche method is utilized to keep steady sub-populations of good points. All individuals in the first non-dominated front are found in this step. In order to find the individuals in the second and higher non-dominated levels, the solutions of the first front are discounted temporarily and the above procedure is repeated [52]. In fact, all solutions which are non-dominated with respect to each other are assigned as rank 1 and then removed from competition. For the remaining individuals, the next set of non-dominated solutions are assigned as rank 2 and then removed from competition. This procedure continues until no chromosome is found. It is obvious that solutions in the lower rank dominate the ones in the higher ranks.

5.3 Diversity mechanism

Along with convergence to the Pareto-optimal set, it is preferred in which an EA preserves good solutions spread in the obtained set of solutions. To get an estimate of the density of solutions surrounding a particular solution in the population, the average distance of two points on either side of this point along with each of the objectives are calculated. This quantity i'_{distance} which is named crowding distance serves as an approximation of the size of the largest cuboid including the point i' without any other point in the population. Actually, overall crowding distance value is calculated as the sum of individual distance values corresponding to each objective. A solution located in a less dense cuboid is allowed to have a higher probability to survive in the next generation. By assigning a crowding distance to all population members, the crowded -comparison operator ($\prec_n$) must be used in order to compare two solutions for their extent of proximity with other solutions which guides the selection process at the various stages of the algorithm toward a uniformly spread-out Pareto-optimal front [52].

In Fig. 2, crowding distance computation procedure is shown where $LL\ [i']_{m'}$ refers to the m'th objective value of the i'th individual in the non-dominated set LL. The complexity of this procedure is governed by the sorting algorithm. When all solutions are in only one front, i.e., in the worst case, the sorting requires O ($M'N'$ log N') computations [52] (Fig. 3).

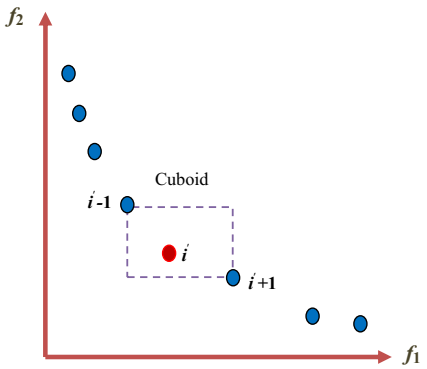

Fig. 3 Crowding distance calculation

5.4 Selection mechanism

Selection is a major operator in GA, which itself does not produce new solutions, instead, it selects better solutions to be transferred to the next generation and also chooses individuals on which genetic operator would operate. Consider a population P_0 (parent population) which is produced randomly at first. This population P_0 is sorted based on the non-domination. Also, each solution is assigned a rank (or fitness) equals to its non-domination level. Obviously, minimization of fitness is assumed. Selection operator selects a set $P' \subseteq P_0$ of the chromosomes (better members of the population with lower fitness) that will be given the chance to be mated and mutated. The usual binary tournament selection operator based on the crowded-comparison operator $\prec_n$ is used in NSGAII while the usual rank-based roulette wheel selection operator is employed in NRGA. As crowded-comparison operator needs both the rank and crowded distance of each solution in the population, we compute these quantities as the same time as forming the population P_{t+1}. The crossover and mutation operators besides explained selection mechanisms are used to create a child population Q_0' of size N' in both NSGAII and NRGA. Both crossover and mutation operator are explained in the following subsections. Consequently, a combined population $R_t' = P_t \cup Q_t' R_t'$ is formed and is then sorted according

Fig. 2 Pseudo-code of the crowding distance method in a non-dominated set

```
Crowding-distance-assignment (LL)
l = |LL|                                   number of solutions in LL
For each i', set LL[i']_distance = 0        initialize distance
For each objective m' = 1 to M'
    LL = sort (LL, m')                     sort using each objective value
    LL[1]_distance = LL[l]_distance = ∞     so that boundary points are always selected
    For i' = 2 to (l -1)                   for all other points
        LL[i']_distance = LL[i']_distance + (LL[i' + 1]_m' − LL[i' − 1]_m')
```

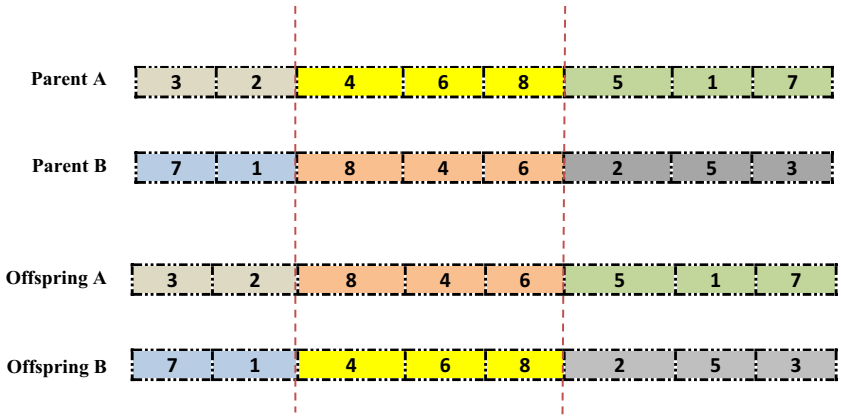

Fig. 4 Applied two-point crossover operator

to non-domination. Also, since all previous and current population members are included in R_t', elitism is ensured. The diversity among non-dominated solutions is introduced by using the crowding comparison procedure, which is used during the population reduction phase [52].

5.4.1 Crossover operator

Crossover is the process of taking two parent chromosomes from the mating pool and producing offspring by combining them, aiming to find better solutions. The crossover operator is applied according to a probability pc by selected individuals that are obtained by roulette wheel technique. Among several common crossover operators for scheduling problems such as one-point, two-point, uniform, etc., a standard two-point crossover is employed in this research so as to produce two offspring from two parent solutions. Having randomly chosen two points in a string, the sub-strings between the crossover points are interchanged. Figure 4 depicts applying the crossover operator on the two selected parents.

5.4.2 Mutation operator

Mutation operator is generally applied with the intention of diversifying the population in order that it does not prematurely converge to multiple copies of one solution. Similar to the used crossover operator, mutation operator is also carried out on two parts of chromosomes. A mutation operator according to a probability pm could be applied, once the offspring are obtained. Among different tested mutation operators, the swap mutation yielded the best performance. This operator is worked by swapping two genes in the selected chromosomes and also keeping away from getting stuck in local suboptimal solutions and is very helpful to maintain the wealth of the population in dealing with large-scale problems. Figure 5 shows the mutation operator.

6 Tests and results

In this section, some experiments are conducted for the purpose of assessment of effectiveness and competitiveness of applied algorithms. All test problems 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 Data generation and settings

To demonstrate the employed algorithms' performance, two categories of test examples in terms of small and medium-to-large-sized instances are generated in this study. Table 2 shows how these sample instances are generated.

Where in due date calculation $d_{min}=max\,(0, P\,(\upsilon-\rho/2))$ and $P=1/m\sum_{i=1}^{n}p_j$. 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 are regarded as $\upsilon \in \{0.2, 0.5, 0.8\}$ and $\rho \in \{0.2, 0.5, 0.8\}$. In order to ensure constancy of employed algorithms, five instances are generated for each quadruple (N, M, υ, ρ) where each instance has run five times in all methods. Consequently, 2025, i.e., $3^4\times5\times5$ sample instances are generated for each algorithm.

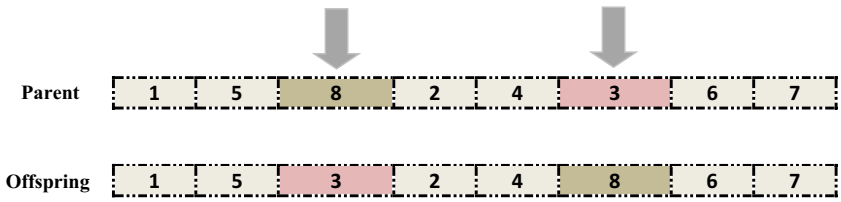

Fig. 5 Applied mutation operator

Table 2 Data set distribution

Input variables	Distribution
Number of jobs (n)	10, 20, 50
Number of machines (m)	2, 3, 5
Normal processing time (p_j)	~DU (10, 40)
Crash processing time (p'_j)	~DU (0.5×p_i, p_i)
Expansion processing time (p''_j)	~DU (p_i, 1.5×p_i)
Due dates (d_j)	~DU [d_{min}, d_{min}+ρP]
Unit cost of compression (c_j)	~U (0.5, 4.5)
Unit cost of expansion (c'_j)	~U (0.5, 4.5)
Earliness penalty (α_j)	~U (1.5, 5.0)
Tardiness penalty (β_j)	~U (2.5, 5.0)

6.2 Performance measures

In multi-objective problems, since a solution could be the best for some objectives however not for the others, it is not rational to arrange a set of solutions. Most of the multi-objective optimization methods estimate the Pareto-optimal front by a set of non-dominated solutions. Because of the incommensurable and conflicting nature of Pareto archive's solutions which make this process more complex, how to evaluate the quality of these solutions is a significant decision. Frankly speaking, comparing the solutions of two different Pareto approximations coming from two algorithms is not straightforward. To do so, unfortunately, different features of non-dominated fronts cannot be taken into consideration by one numerical value.

Table 3 Evaluation of non-dominated solutions for small problems

M	N	υ	ρ	Δ		ds		sp	
				NSGAII	NRGA	NSGAII	NRGA	NSGAII	NRGA
2	10	0.2	0.2	0.1048	0.1348	623.04	590.01	0.0123	0.0166
			0.5	0.2215	0.2579	112.90	100.47	0.2422	0.2829
			0.8	0.0905	0.1663	298.13	306.54	0.0266	0.0146
		0.5	0.2	0.1327	0.1204	151.34	135.11	0.0419	0.0536
			0.5	0.2383	0.2232	112.11	112.15	0.1157	0.1195
			0.8	0.1382	0.1887	287.74	232.04	0.0924	0.0765
		0.8	0.2	0.1067	0.1282	465.49	416.38	0.2008	0.3031
			0.5	0.1627	0.1712	529.30	595.02	0.1268	0.1410
			0.8	0.3695	0.3088	175.00	120.80	0.0922	0.0999
Mean				0.1739	0.1888	306.12	289.84	0.1056	0.1231
3	10	0.2	0.2	0.1093	0.1395	309.35	301.45	0.3038	0.3062
			0.5	0.1522	0.1698	882.99	856.38	0.0494	0.0716
			0.8	0.0578	0.0320	769.09	703.72	0.0986	0.1172
		0.5	0.2	0.1286	0.1213	419.36	370.81	0.1558	0.1663
			0.5	0.0162	0.0092	642.00	608.54	0.0198	0.0180
			0.8	0.1268	0.1005	835.06	657.23	0.0715	0.0770
		0.8	0.2	0.0925	0.1157	100.54	111.67	0.4043	0.4112
			0.5	0.0725	0.0844	144.63	131.11	0.0304	0.0287
			0.8	0.1095	0.1109	192.25	192.58	0.0903	0.1005
Mean				0.0961	0.0981	477.25	437.05	0.1360	0.1441
5	10	0.2	0.2	0.1967	0.2808	159.76	146.73	0.1454	0.1597
			0.5	0.1418	0.1780	181.15	155.11	0.0231	0.0287
			0.8	0.1084	0.1426	965.76	734.73	0.0922	0.1168
		0.5	0.2	0.2742	0.3573	168.30	171.09	0.4489	0.4016
			0.5	0.2117	0.2056	378.58	419.95	0.0698	0.0178
			0.8	0.0413	0.0207	284.57	240.99	0.1842	0.1706
		0.8	0.2	0.1274	0.1316	1195.08	982.78	0.0236	0.0252
			0.5	0.3205	0.3070	876.98	710.14	0.0997	0.1088
			0.8	0.2822	0.2362	1596.84	1016.14	0.0897	0.0918
Mean				0.1893	0.2067	645.22	508.63	0.1307	0.1245
Mean of all				0.1531	0.1645	476.20	411.84	0.1241	0.1306

Table 4 Evaluation of non-dominated solutions for medium-to-large problems

M	*N*	*υ*	*ρ*	*Δ*		*ds*		*sp*	
				NSGAII	NRGA	NSGAII	NRGA	NSGAII	NRGA
2	20	0.2	0.2	0.2192	0.12795	1139.03	904.24	0.0848	0.1017
			0.5	0.2869	0.34112	882.41	766.14	0.1131	0.1355
			0.8	0.1752	0.21255	1652.20	1306.11	0.0167	0.0109
		0.5	0.2	0.2119	0.2015	804.58	717.46	0.0995	0.1219
			0.5	0.1163	0.1430	597.80	573.67	0.1485	0.1777
			0.8	0.2414	0.2437	307.56	314.06	0.0613	0.0589
		0.8	0.2	0.2611	0.2516	1131.32	913.77	0.0841	0.0874
			0.5	0.2514	0.3699	585.35	417.27	0.0795	0.1109
			0.8	0.1445	0.3445	902.14	821.89	0.1023	0.1231
	50	0.2	0.2	0.1451	0.3268	1344.53	1303.69	0.1231	0.1308
			0.5	0.2041	0.2059	1875.01	1571.11	0.2305	0.1405
			0.8	0.2241	0.3240	1442.72	1075.62	0.1394	0.1508
		0.5	0.2	0.2756	0.2731	1422.91	1028.44	0.0903	0.1135
			0.5	0.2450	0.3303	374.37	326.20	0.1386	0.1759
			0.8	0.2827	0.2733	2530.12	2124.50	0.0692	0.0697
		0.8	0.2	0.3577	0.4360	1565.49	1442.68	0.1660	0.1992
			0.5	0.3232	0.5480	2893.01	2920.62	0.0812	0.1166
			0.8	0.2398	0.1895	1592.28	1581.46	0.1049	0.1174
Mean				0.2336	0.2857	1280.16	1117.16	0.1074	0.1190
3	20	0.2	0.2	0.2613	0.2789	589.61	517.20	0.0848	0.0888
			0.5	0.2676	0.3186	735.15	601.07	0.1055	0.1161
			0.8	0.3577	0.4486	1572.20	1239.54	0.0305	0.0329
		0.5	0.2	0.2520	0.2900	1758.08	1484.94	0.0526	0.1109
			0.5	0.2136	0.1840	514.65	443.24	0.1294	0.1616
			0.8	0.1328	0.1236	1367.89	1161.45	0.0197	0.0381
		0.8	0.2	0.2606	0.1669	2018.11	2111.77	0.0287	0.0199
			0.5	0.2159	0.1645	900.08	638.24	0.1131	0.1551
			0.8	0.2327	0.1886	1082.63	1148.59	0.2143	0.3059
	50	0.2	0.2	0.3550	0.3927	1759.47	1643.69	0.0985	0.1265
			0.5	0.4305	0.4173	2240.59	2077.74	0.1523	0.1721
			0.8	0.2252	0.2566	2460.15	2148.05	0.0731	0.0836
		0.5	0.2	0.3374	0.2880	1007.88	1058.49	0.1237	0.2257
			0.5	0.4175	0.6219	2929.52	2461.73	0.0617	0.0766
			0.8	0.3747	0.3927	1442.28	1225.02	0.0098	0.1217
		0.8	0.2	0.1027	0.1079	3565.12	3280.52	0.1446	0.2051
			0.5	0.2218	0.2287	1904.96	1687.83	0.1669	0.1205
			0.8	0.0839	0.1203	3049.77	2856.54	0.0536	0.0555
Mean				0.2635	0.2772	1716.56	1543.65	0.0924	0.1231
5	20	0.2	0.2	0.2247	0.2627	680.28	487.81	0.1129	0.1201
			0.5	0.1539	0.1494	1030.12	854.48	0.0522	0.0991
			0.8	0.4710	0.4566	589.20	409.99	0.2687	0.2135
		0.5	0.2	0.5159	0.5881	810.48	771.99	0.0834	0.0751
			0.5	0.3286	0.2763	1654.02	1629.17	0.1677	0.1415
			0.8	0.3962	0.3576	712.45	622.78	0.0445	0.0724
		0.8	0.2	0.2120	0.1500	811.44	608.34	0.0813	0.0553
			0.5	0.3037	0.3229	775.45	560.57	0.1277	0.1469
			0.8	0.2829	0.2792	159.97	128.34	0.0781	0.1026

Table 4 (continued)

M	*N*	*υ*	*ρ*	*Δ*		*ds*		*sp*	
				NSGAII	NRGA	NSGAII	NRGA	NSGAII	NRGA
	50	0.2	0.2	0.3014	0.4748	1798.36	1478.58	0.1012	0.1485
			0.5	0.0890	0.1554	2401.21	2130.35	0.0634	0.0736
			0.8	0.3601	0.3699	4287.57	3626.02	0.1100	0.1537
		0.5	0.2	0.2613	0.3717	1676.26	1212.05	0.0932	0.1470
			0.5	0.4894	0.5143	2924.31	2200.70	0.0310	0.0316
			0.8	0.3590	0.2982	4737.59	5018.90	0.0764	0.0685
		0.8	0.2	0.1857	0.1494	1201.55	937.26	0.0886	0.0896
			0.5	0.2680	0.2763	1962.15	1816.57	0.0294	0.0306
			0.8	0.1397	0.1377	3185.80	2932.16	0.1268	0.1265
Mean				0.2968	0.3106	1744.34	1523.67	0.0965	0.1053
Mean of all				0.2646	0.2912	1580.35	1394.83	0.0987	0.1158

There are two objectives in a multi-objective optimization, not like in a single-objective one: (1) convergence to the Pareto-optimal set and (2) preservation of diversity among solutions of the Pareto-optimal set. As pointed out earlier, these instances cannot be measured satisfactorily with one performance metric. Many performance metrics have been suggested by Coello et al. [54], Deb and Sachin [56], Deb [55], and Zitzler [57].

The performance of our proposed algorithms is analyzed on a set of generated sample instances. Notwithstanding the existence of a number of metrics to find the diversity among obtained non-dominated solutions, two measurement factors are applied in this study so as to assess the algorithms' performance, i.e., spacing (sp) and spread (Δ). One more metric is proposed in this research which calculates the distance from origin coordinates (ds). Diversity is simply measured by the number of disjoint solutions in the locally non-dominated frontier and distance is measured by Euclidean distance rule.

6.2.1 Spacing

The first employed metric is called *spacing* which has been introduced by Schott [58] and is calculated with a relative distance measure between consecutive solutions in the obtained non-dominated set as follows:

$$\mathrm{sp} = \sqrt{\frac{1}{\left|E'\right|}\sum_{i'=1}^{\left|E'\right|}\left(\mathrm{dist}_{i'} - \overline{\mathrm{dist}}\right)^2} \tag{14}$$

Where $\mathrm{dist}_i = \min_{k\in E'\,\Lambda\, k\neq i'}\sum_{m'=1}^{M'}\left|f_{m'}^{i'} - f_{m'}^{k}\right|$ and $\overline{dist}$ $= \sum_{i'=1}^{\left|E'\right|}\frac{\mathrm{dist}_{i'}}{E'}$. The distance measure is the minimum value of the sum of the absolute difference in objective function values between the i'th solution and any other solution in the obtained non-dominated set. It is noticeable that this distance measure is different from the minimum Euclidean distance between two solutions. A zero value for this metric signifies all members of the Pareto front currently available are equidistantly spaced.

This metric measures the standard deviation of different $\mathrm{dist}_{i'}$ values. When the solutions are near uniformly spread, the corresponding distance $\mathrm{dist}_{i'}$ measure will be small. Accordingly, the algorithm finding a set of non-dominated solutions having a smaller sp is preferable. In addition, the above-mentioned metric offers helpful information about the spread of the attained non-dominated solutions, but does not take into consideration the extent of the spread. As long as the spread is uniform within the range of obtained solutions, the metric sp produces a small value.

6.2.2 Spread

The second metric used in this research is *spread* which was suggested by Deb et al. [59] so as to conquer the aforementioned difficulty. This metric measures the extent of spread achieved among the obtained solutions as follows:

$$\Delta = \frac{\sum_{m'=1}^{M'}\mathrm{dist}_{m'}^{e'} + \sum_{i'=1}^{\left|E'\right|}\left|\mathrm{dist}_{i'} - \overline{\mathrm{dist}}\right|}{\sum_{m'=1}^{M'}\mathrm{dist}_{m'}^{e'} + \left|E'\right|\overline{\mathrm{dist}}} \tag{15}$$

Where any distance measure between neighboring solutions and $\overline{\mathrm{dist}}$ could be the mean value of these distance

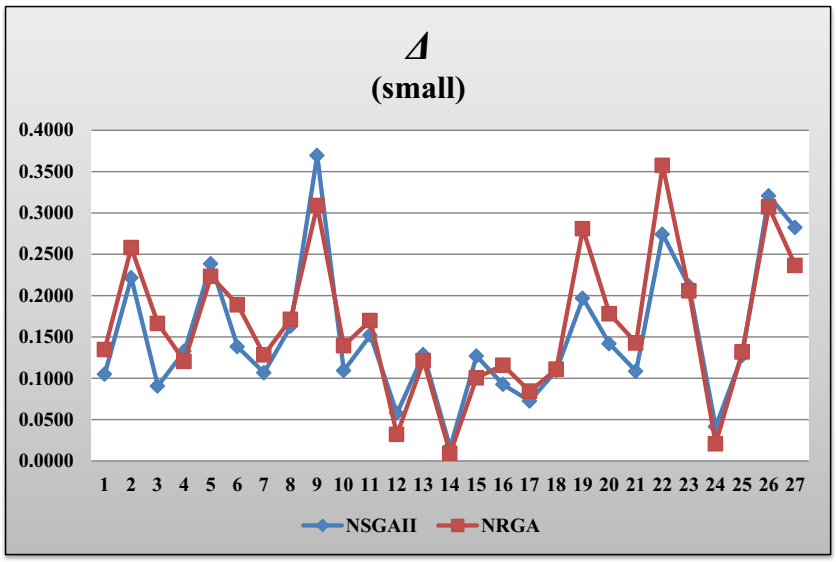

Fig. 6 Comparing Δ metric for small-sized problems

measures. The sum of the absolute differences in objective values, the crowding distance, or Euclidean distance could be used to calculate $\text{dist}_{i'}$. The parameter $\text{dist}^{e'}_{m'}$ is the distance between the extreme solutions of P^* and E' corresponding to m'th objective function. For an ideal distribution of solutions, $\Delta=0$. Generally, it could be said that the smaller the value of Δ, the higher the algorithm capability in finding non-dominated solutions with good diversity.

6.2.3 Distance

The last metric is employed with the intention of measuring the solutions' convergence to the origin coordinates (point [0,0]) which is desirable in a bi-objective space (in minimization case). "Euclidean distance" metric is used for doing so.

$$\text{ds} = \sqrt{\sum_{i'=1}^{n} \sum_{m'=1}^{M'} f_{m'}^{i'^2}} \tag{16}$$

The lower ds values are obviously preferable. Those Pareto fronts which are closer to the origin coordinates have better convergence.

Each problem in all categories has run five times on each algorithm in order to ensure constancy of the employed algorithms. Also, all aforementioned metrics are calculated in each run.

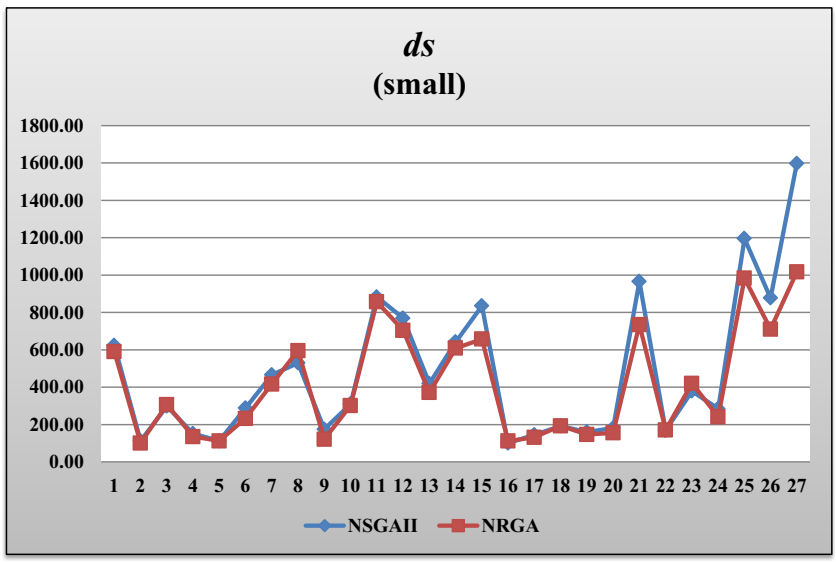

Fig. 7 Comparing *ds* metric for small-sized problems

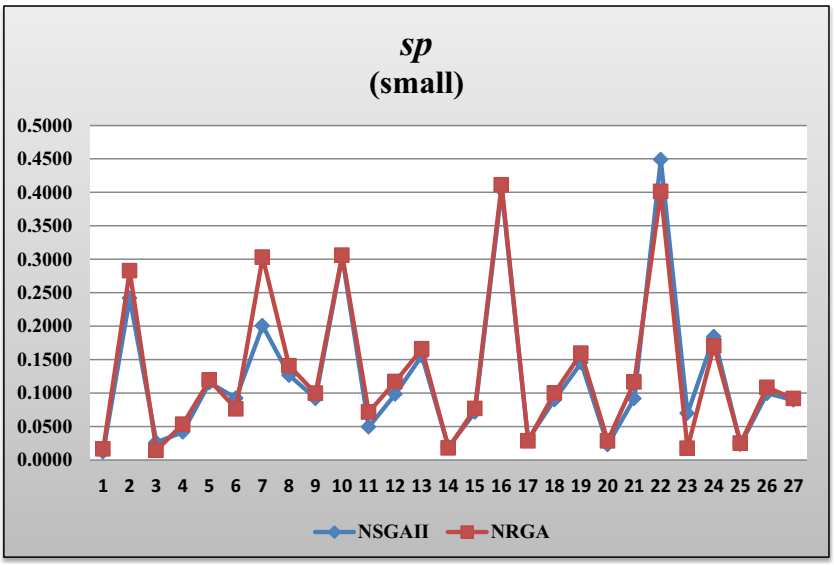

Fig. 8 Comparing sp metric for small-sized problems

6.3 Computational results

The performance of the employed algorithms is studied in this subsection through different measures on two categories of sample instances, i.e., small- and medium-to-large-sized ones. Tables 3 and 4 present the results of employed measures, including sp, Δ, and ds where the average results of performance indexes for these cases are calculated.

As pointed out earlier, a smaller value of the employed performance measure is preferable. According to Table 3, in small-sized problems, NSGAII yields better results in Δ and sp criteria, but NRGA outperforms NSGAII in ds criterion in "mean of all." However, in both Δ and sp, there is no significant difference between consequences of employed NSGAII and NRGA. Based on the obtained results, as it could be seen, in few instances, NRGA outperforms NSGAII in Δ and sp, while NSGAII surpasses NRGA in ds.

The mean performance of applied metrics for both NSGAII and NRGA in 27 small set instances is depicted in Figs. 6, 7, and 8.

As it could be seen in Table 4, the similar consequence is obtained, i.e., NSGAII outperforms NRGA in Δ and sp metrics in "mean of all." However, NRGA surpasses NSGAII in ds criterion.

The attained outcomes in the considered problem reveal the ability of NRGA in converging to the true front and capabilities of NSGAII in finding diverse solutions in the front. As a matter of fact, NRGA gets closer to the true Pareto-optimal front than NSGAII, while NSGAII finds a better spread in the entire Pareto-optimal region than NRGA.

In Figs. 9, 10, and 11, the obtained results of NSGAII and NRGA in the three employed measurement factors are demonstrated in 54 medium-to-large set instances.

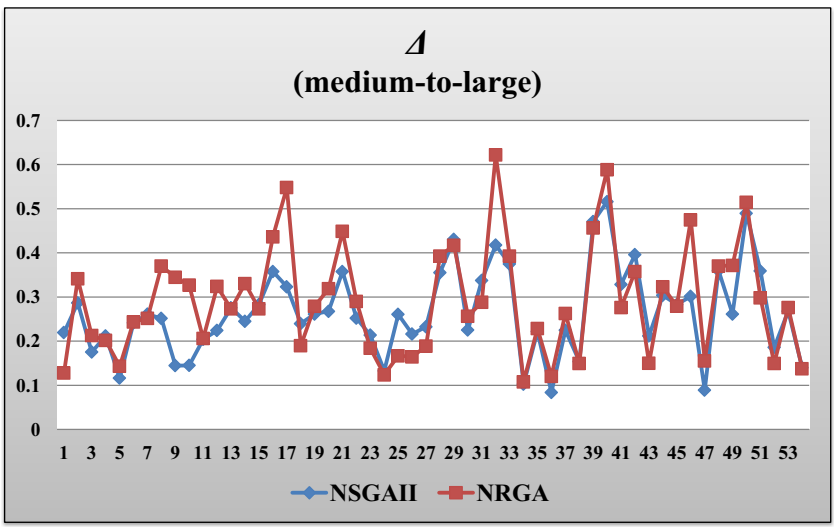

Fig. 9 Comparing Δ metric for medium-to-large-sized problems

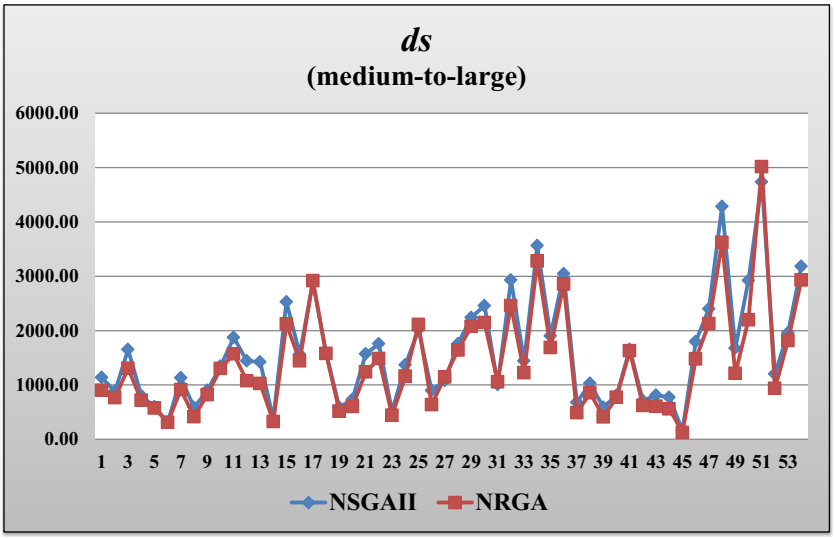

Fig. 10 Comparing *ds* metric for medium-to-large-sized problems

7 Conclusions and future studies

In the main, according to multi-objective nature of real-world problems, it is more realistic to investigate optimization problems within a multi-objective environment. In this study, we have successfully solved a bi-objective identical parallel machine problem considering just-in-time (JIT) philosophy where job processing times are controllable. Each job has a distinct due date and no job preemption is allowed. The aim of this study was to simultaneously minimize (1) total cost of tardiness, earliness as well as compression and expansion costs of job processing times and (2) maximum completion time so-called makespan. Besides determining a sequence of jobs on each machine with capability of processing all jobs, the best possible set amount of compression/expansion of processing times on each machine was also acquired via the proposed "bi-objective parallel net benefit compression-net benefit expansion" (BPNBC-NBE) heuristic.

Two well-known multi-objective evolutionary algorithms, i.e., non-dominated sorting genetic algorithm II (NSGAII) and non-dominated ranking genetic algorithm (NRGA) were employed so as to solve such bi-objective problem. In order to compare these methods, three measurement factors were also applied to assess the algorithms' performance. Computational results demonstrated that NSGAII outperforms NRGA in terms of diversity while NRGA surpasses NSGAII in converging to the true front in all small- and medium-to-large-sized problems.

As a direction for future research, it would be interesting to develop such a bi-objective problem on other multi-machine environment, such as flowshop or different types of parallel machines. Employing different performance measures could also be tested so as to consider different aspects of comparison. For the sake of enhancing the goodness-of-fit of the proposed techniques, further analysis could be made through advanced statistical analysis such as

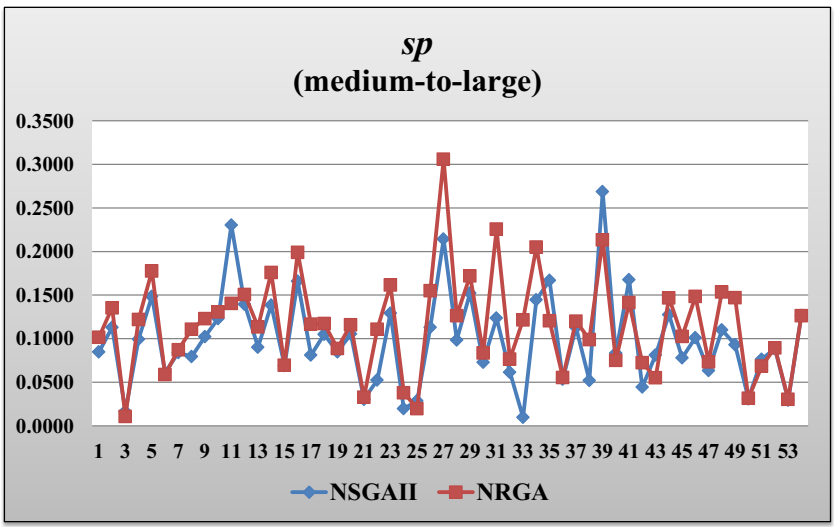

Fig. 11 Comparing sp metric for medium-to-large-sized problems

design of experiments, factorial design, and Taguchi method for parameters' tuning. Another improvement could also be the hybridization of the proposed algorithms in order to increase the quality of the obtained results. Also, regarding multi-resource manufacturing system in order to compare with the proposed methodology in this paper is another interesting future research.

References

1. Baker, K.R., 1994. Elements of sequencing and scheduling, Hanover: HN
2. Sun H, Wang G (2003) Parallel machine earliness and tardiness scheduling with proportional weights. Comput Oper Res 30(5): 801–808
3. Hall NG, Posner ME (2001) Generating experimental date for computational testing with machine scheduling applications. Oper Res 49:854–865
4. Baker KR, Scudder GD (1990) Sequencing with earliness and tardiness penalties: a review. Oper Res 38:22–36
5. Hall NG, Posner ME (1991) Earliness–tardiness scheduling problems. I: weighted deviation of completion times about a common due date. Oper Res 39:836–846
6. Wang A, Huang Y, Taunk P, Magnin DR, Ghosh K, Robertson JG (2003) Application of robotics to steady state enzyme kinetics: analysis of tight-binding inhibitors of dipeptidyl peptidase IV. Anal Biochem 321(2):157–166
7. Sørensen, J.F., Kragh, K.M., Sibbesen, O., Delcour, J., Goesaert, H., Svensson, B., et al., 2004. Potential role of glycosidase inhibitors in industrial biotechnological applications. Biochimica et Biophysica Acta (BBA)-Proteins & amp; Proteomics 1696(2), 275–287
8. Sivrikaya F, Ulusoy G (1999) Parallel machine scheduling with earliness and tardiness penalties. Comput Oper Res 26: 773–787
9. Cheng R, Gen M, Tosawa T (1995) Minmax earliness/tardiness scheduling in identical parallel machine system using genetic algorithms. Comput Ind Eng 29:513–517
10. Kedad-Sidhoum S, Solis YR, Sourd F (2008) Lower bounds for the earliness-tardiness scheduling problem on parallel machines with distinct due dates. Eur J Oper Res 189(3):1305–1316
11. Su LH (2009) Minimizing earliness and tardiness subject to total completion time in an identical parallel machine system. Comput Oper Res 36(2):461–471
12. Drobouchevitch IG, Sidney JB (2012) Minimization of earliness, tardiness and due date penalties on uniform parallel machines with identical jobs. Comput Oper Res 39(9):1919–1926
13. Biskup D, Herrmann J, Gupta JND (2008) Scheduling identical parallel machines to minimize total tardiness. Int J Prod Econ 115(1):134–142
14. Xi Y, Jang J (2012) Scheduling jobs on identical parallel machines with unequal future ready time and sequence dependent setup: an experimental study. Int J Prod Econ 137(1):1–10
15. Shmoys, D.B., Tardos, E., 1993. Scheduling unrelated machines with costs, In: Proceedings of the 4th Annual ACM-SIAM Symposium, Austin, TX, January 25–27, pp. 448–454
16. Coffman EG, Sethi R (1976) Algorithms minimizing mean flow time: schedule length properties. Acta Informatica 6:1–14
17. Lin CH, Liao CJ (2004) Makespan minimization subject to flowtime optimality on identical parallel machines. Comput Oper Res 31: 1655–1666
18. Ruiz-Torres AJ, Lopez FJ (2004) Using the FDH formulation of DEA to evaluate a multi-criteria problem in parallel machine scheduling. Comput Ind Eng 47:107–121
19. Suresh V, Chaudhuri D (1996) Bicriteria scheduling problem for unrelated parallel machines. Comput Ind Eng 30:77–82
20. Ruiz-Torres AJ, Enscore EE, Barton RR (1997) Simulated annealing heuristics for the average flow-time and the number of tardy jobs bicriteria identical parallel machine problem. Comput Ind Eng 33:257–260
21. Chang PC, Chen SH, Lin KL (2005) Two-phase sub population genetic algorithm for parallel machine-scheduling problem. Expert Syst Appl 29:705–712
22. Gao J, He G, Wang Y (2009) A new parallel genetic algorithm for solving multi objective scheduling problems subjected to special process constraint. Int J Adv Manuf Technol 43:151–160
23. Gao J (2010) A novel artificial immune system for solving multiobjective scheduling problems subject to special process constraint. Comput Ind Eng 58:602–609
24. Tavakkoli-Moghaddam R, Taheri F, Bazzazi M, Izadi M, Sassani F (2009) Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints. Comput Oper Res 36(12):3224–3230
25. Radhakrishnan S, Ventura JA (2000) Simulated annealing for parallel machine scheduling with earliness/tardiness penalties and sequence-dependent setup times. Int J Prod Res 38(10):2233–2252
26. Vickson RG (1980) Two single machine sequencing problems involving controllable job processing times. AIIE Trans 12:258–262
27. Nowicki E, Zdrzalka S (1990) A survey of results for sequencing problems with controllable processing times. Discret Appl Math 26: 271–287
28. Lee IS (1991) Single machine scheduling with controllable processing times: a parametric study. Int J Prod Econ 22(2):105–110
29. Liman S, Panwalkar S, Thongmee S (1997) A single machine scheduling problem with common due window and controllable processing times. Ann Oper Res 70:145–154
30. Shabtay D, Steiner G (2007) A survey of scheduling with controllable processing times. Discret Appl Math 155:1643–1666
31. Gurel S, Akturk MS (2007) Scheduling parallel CNC machines with time/cost trade-off considerations. Comput Oper Res 34(9):2774–2789
32. Alidaee B, Ahmadian A (1993) Two parallel machine sequencing problems involving controllable job processing times. Eur J Oper Res 70(3):335–341
33. Wan, G., 2007. Single machine common due window scheduling with controllable job processing times, combinatorial optimization and applications, A. Dress, Y. Xu, and B. Zhu, Editors. Springer Berlin Heidelberg. pp. 279–290
34. Shakhlevich N, Strusevich V (2008) Preemptive scheduling on uniform parallel machines with controllable job processing times. Algorithmica 51(4):451–473
35. Shabtay D (2010) Scheduling and due date assignment to minimize earliness, tardiness, holding, due date assignment and batch delivery costs. Int J Prod Econ 123(1):235–242
36. Aktürk M, Atamtürk A, Gürel S (2010) Parallel machine match-up scheduling with manufacturing cost considerations. J Sched 13(1): 95–110
37. Leyvand Y, Shabtay D, Steiner G, Yedidsion L (2010) Just-in-time scheduling with controllable processing times on parallel machines. J Comb Optim 19(3):347–368
38. Yin N, Wang X-Y (2011) Single-machine scheduling with controllable processing times and learning effect. Int J Adv Manuf Technol 54(5–8):743–748

39. Li K, Shi Y, Yang S, Cheng BY (2011) Parallel machine scheduling problem to minimize the makespan with resource dependent processing times. Appl Soft Comput 11(8):5551–5557
40. Jansen K, Mastrolilli M (2004) Approximation schemes for parallel machine scheduling problems with controllable processing times. Comput Oper Res 31(10):1565–1581
41. Niu, G., Sun, S., Lafon, P., Zhang, Y., & Wang, J. (2012). Two decompositions for the bicriteria job-shop scheduling problem with discretely controllable processing times. International Journal of Production Research, 50(24):7415–7427
42. Renna P (2013) Controllable processing time policies for job shop manufacturing system. Int J Adv Manuf Technol 67(9–12):2127–2136
43. Low C, Li R-K, Wu G-H (2013) Ant colony optimization algorithms for unrelated parallel machine scheduling with controllable processing times and eligibility constraints. Proceedings of the Institute of Industrial Engineers Asian Conference 2013. Springer, Singapore, pp 79–87
44. Kayvanfar V, Mahdavi I, Komaki GM (2013) Single machine scheduling with controllable processing times to minimize total tardiness and earliness. Comput Ind Eng 65(1):166–175
45. Kayvanfar V, Mahdavi I, Komaki GM (2013) A drastic hybrid heuristic algorithm to approach to JIT policy considering controllable processing times. Int J Adv Manuf Technol 69:257–267
46. Kayvanfar V, Komaki GHM, Aalaei A, Zandieh M (2014) Minimizing total tardiness and earliness on unrelated parallel machines with controllable processing times. Comput Oper Res 41:31–43
47. Pinedo, M., 1995. Scheduling: theory, algorithms, and systems. Englewood Cli (s, NJ: Prentice-Hall
48. Hillier FS, Lieberman GJ (2001) Introduction to operations research, 7th edn. McGraw-Hill, New York
49. Anagnostopoulos KP, Mamanis G (2010) A portfolio optimization model with three objectives and discrete variables. Comput Oper Res 37(7):1285–1297
50. Zitzler, E., Laumanns, M., Thiele, L., 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35: CH-8092 Zurich, Switzerland
51. Horn, J., Nafploitis, N., Goldberg, D.E., 1994. A niched Pareto genetic algorithm for multiobjective optimization. In: Proceeding of the first IEEE Conference on Evolutionary Computation, IEEE Press, pp. 82–87
52. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
53. Al Jadaan O, Rajamani L, Rao CR (2008) Non-dominated ranked genetic algorithm for solving multi-objective optimisation problems: NRGA. J Theor Appl Inf Technol 2:60–67
54. Coello, C.C., Lamont, G.B., Veldhuizen, D.A., 2007. Evolutionary algorithms for solving multi-objective problems. 2nd ed. Springer
55. Deb K (2001) Multi objective optimization using evolutionary algorithms. Wiley, Chichester
56. Deb, K., Sachin, J., 2004. Running performance metrics for evolutionary multi-objective optimization. Kanpur Genetic Algorithm Laboratory (KanGAL), Report No.2002004
57. Zitzler, E., 1999. Evolutionary algorithms for multiobjective optimization: methods and applications. Ph.D. dissertation ETH 13398, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland
58. Schott, J.R., 1995. Fault tolerant design using single and multicriteria genetic algorithms. Master's Thesis, Boston MA: Department of Aeronautics and Astronautics, Massachusetts Institute of Technology
59. Deb K, Agarwal S, Pratap A, Meyarivan T (2000) Technical report 200001, Indian Institute of Technology. Kanpur Genetic Algorithms Laboratory (KanGAL), Kanpur, A fast and elitist multi objective genetic algorithm: NSGA-II