RAIRO-Oper. Res. 55 (2021) 1501–1522
https://doi.org/10.1051/ro/2021062

RAIRO Operations Research
www.rairo-ro.org

EFFICIENT ALGORITHMS TO MINIMIZE MAKESPAN OF THE UNRELATED PARALLEL BATCH-PROCESSING MACHINES SCHEDULING PROBLEM WITH UNEQUAL JOB READY TIMES

Yaser Zarook[1,*], Javad Rezaeian[1], Iraj Mahdavi[1] and Masoud Yaghini[2]

Abstract. This paper considers the minimization of makespan in the unrelated parallel batch processing machines scheduling problem with considering non-identical job size and dynamic job ready time. The considered unrelated machines have different capacity and different processing speed. Each machine processes a number of the jobs as a batch at the same time so that the machine's capacity is not exceeded. The batch processing time and the batch ready time are equal to the largest processing time and the largest ready time of jobs in the same batch, respectively. In this paper, a Mixed Integer Linear Programming (MILP) model, two categories of the heuristic procedures (six heuristics) and a meta-heuristic algorithm are proposed to solve the problem. A lower bound is also presented by relaxing of the original problem to evaluate the quality of the proposed algorithms. The computational experiments show the performance of the proposed algorithms under the considered measures.

Mathematics Subject Classification. 90B35, 90C11, 68W25.

Received April 16, 2020. Accepted April 17, 2021.

1. Introduction & literature review

In the recent years, Batch-Processing (BP) operation has been a critical solution to eliminate of the production bottlenecks in the most industries. The BP by preventing of the setup time repetition leads to the elimination of the bottlenecks and the faster production [1]. There are two categories of BP: Serial batching (S-batching) and parallel batching (P-batching). In S-batching, the processing time of a batch is equivalent to the sum of the processing times of jobs within the same batch, and in P-batching; several jobs are processed in a batch on a processor simultaneously, such that the processing time of the batch is equivalent to the largest processing time of jobs within the same batch. P-batching problem has attracted many investigators in the most industries such as test electronic parts to detect early failures in semiconductor manufacturing, the chemical, food and mineral processing, metalworking industries and etc. [15,17,19,21,22]. This paper is placed in P-batching category, and in the rest of paper, means of the batch is P-batching. The motivation of this study is stem from elimination of the bottleneck in the cutting stations of the metal industries.

Keywords. Unrelated parallel machines, batch-processing, heuristic, dynamic job ready times, Makespan, mathematical modeling.

[1] Department of Industrial Engineering, Mazandaran University of Science and Technology Babol, Mazandaran, Iran.

[2] Department of Rail Transportation Engineering, School of Railway Engineering Iran University of Science and Technology, Tehran, Iran.

*Corresponding author: y.zaruk89.ac@gmail.com

By studying in the literature of batch processing, the various machine's environments are observable such as; Dauzère-Pérès and Mönch [7] studied a single batch processing machine with incompatible job families to minimize the weighted number of tardy jobs objective, they proposed two different mixed integer linear programming formulations and a random key genetic algorithm. Koh *et al.* [13] studied a single batch processing machine with arbitrary job sizes and incompatible job families, they obtained optimal solution for the small size problem by integer programming model and efficient solution for the large size problem in the reasonable computational time by heuristic algorithms. Zarook *et al.* [28] suggested a new mathematical model to minimize makespan on the single batch-processing machine scheduling problem with considering machine's aging effect and maintenance activities, they proposed two meta-heuristic algorithms (GA & ICA) to solve the real size problem. Zhou *et al.* [30] presented a multi-objective model for a single batch processing machine scheduling problem with dynamic job arrival times, due to computational complexity, they proposed a hybrid multi-objective meta-heuristic algorithm to find the Pareto front. Another machine environment in the BP is identical batch processing in the parallel machines that machines have similarity characters. Chang *et al.* [3] proposed a simulated annealing algorithm to minimize makespan in identical parallel batch-processing machines setting with arbitrary job size. Damodaran and Chang [4] presented the heuristic algorithms to minimize Makespan on the identical parallel batch processing machines; they compared the performance of the proposed heuristics with a simulated annealing approach. Kashan *et al.* [11] and Damodaran *et al.* [6] proposed the heuristics and meta-heuristic algorithms to minimize makespan in the identical parallel batch processing machines with arbitrary job sizes. Shengchao *et al.* [23] proposed distance matrix based heuristics to minimize makespan in the identical parallel-batch processing machines with arbitrary job sizes and dynamic job release times. Muter [20] proposed the exact algorithms to minimize Makespan on the single and identical-parallel batch processing machines. In the real production systems, new machinery and old ones are often used together; newer machines usually have the larger capacities and the shorter processing times rather than the older ones; this machinery environment is called the unrelated parallel machines. José Elias *et al.* [9] studied the minimization of makespan in the unrelated parallel batch processing machines with non-identical job sizes and unequal job ready times where the machines have same capacity; they proposed several efficient heuristics and also a lower bound to evaluate the quality of the proposed heuristics. Zhao-Hong *et al.* [29] studied the effective heuristics for minimizing makespan in the parallel batch machines with the different machine's capacity and the dynamic job release times while the speed of the machines was considered identical. José Elias *et al.* [10] considered the unrelated parallel batch processing machine scheduling problem with different speed and capacity of the machines and unequal job ready time. They proposed an Iterated Greedy algorithm to minimize total flow time as the objective function. Table 1 shows a brief description of the previous studies:

According to Table 1, this study considers the unrelated parallel batch-processing machines with different speed and capacity. Each job has arbitrary size and dynamic ready time. The proposed problem minimizes the makespan, or, the completion time of the last batch therefore leads to the upper throughput level in the bottleneck station and the upper productivity, finally. The motivation of this paper is stem from elimination of the bottleneck in the production lines such as the cutting stations in the metal industries; where there are the several cutting machines with different speed and capacity (unrelated P-batching). The cutting machines have the capability of cutting the batches of the sheets metal with different thicknesses (job size) at the same time, that leads to eliminate production bottlenecks and higher productivity. This problem is shown as $R_m|p\text{-batch}, p_{jk}, s_j, r_j, B_k|C_{\max}$ in the classical symbolic of the scheduling literature, where first section of the symbolization shows machinery environment (R_m indicates m machines in the unrelated parallel environment). The second section of the symbolization explains conditions and constraints of the proposed scheduling problem (p_{jk} denotes the processing time of job j on the machine k, s_j and r_j denote size and ready time of job j, respectively, B_k denotes the capacity of machine k, p-batch shows parallel batch processing machines). In the third section of symbolization objective function of scheduling problem is shown ($C_{\max}$ or makespan: the completion time of the last batch).

Uzsoy [25] showed that minimization of the Makespan on a single BPM with non-identical job sizes and identical release times to be NP-hard, therefore our problem is also NP-hard. On the other hand, Pinedo [21]

TABLE 1. Classification of the previous works.

Authors	Year	Objective ($C_{\max}$)	Assumptions					Solution approaches			
								Exact	Approximate		
			PBPM*	Different capacity	Unequal job size	Dynamic job ready time	Different speed	Mathematical model	Heuristic**	Meta-heuristic***	Lower bound****
Uzsoy	1994	✓	✗	✗	✓	✗	✗	✓	✓	✗	✓
Koh *et al.*	2004	✓	✓	✗	✓	✗	✗	✗	✓	✓	✗
Chang *et al.*	2004	✓	✓	✗	✓	✗	✗	✓	✗	✓	✗
Xu and Bean	2007	✓	✓	✓	✓	✗	✗	✓	✗	✓	✗
Rui *et al.*	2012	✓	✗	✗	✓	✓	✗	✓	✓	✓	✓
Xu *et al.*	2013	✓	✓	✗	✓	✓	✗	✓	✗	✓	✓
Zarook *et al.*	2014	✓	✗	✗	✓	✓	✗	✓	✗	✓	✓
Shengchao *et al.*	2016	✓	✓	✗	✓	✓	✗	✓	✓	✗	✗
José Elias *et al.*	2016	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓
Lars and Roob	2017	✗	✓	✗	✓	✗	✗	✓	✗	✓	✗
Zhao-Hong *et al.*	2017	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓
José Elias *et al.*	2019	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓
Ibrahim Muter	2020	✓	✓	✗	✓	✗	✗	✓	✗	✗	✓
Zhou *et al.*	2020	✓	✗	✗	✓	✓	✗	✓	✓	✓	✗
This study	2020	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Notes. (*)Parallel Batch Processing Machines. (**)Heuristics: are the problem dependent approximate techniques those are usually adapted to the problem at hand. (***)Meta-heuristics: are the problem independent techniques that can be applied to solve a wide range of problems approximately. (****)Lower bound: is a type of heuristic approach that is adapted to the relaxation problem.

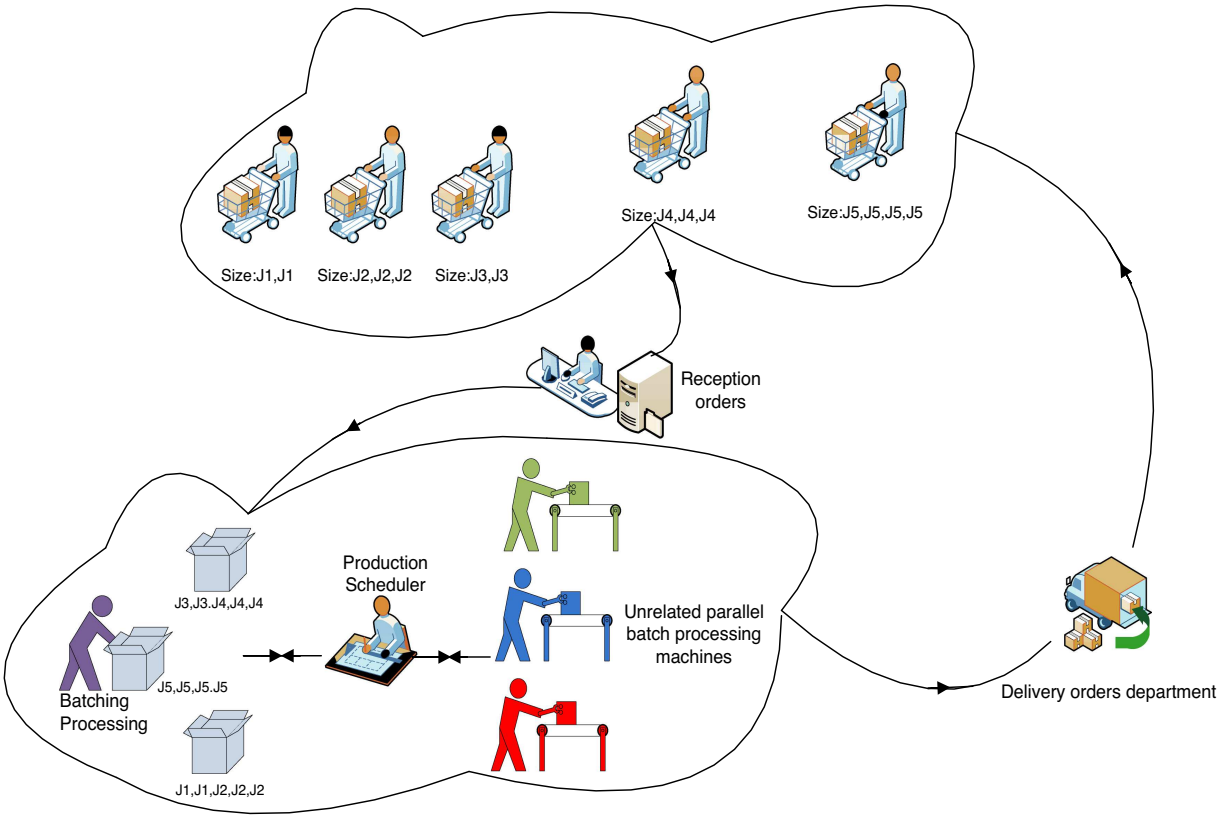

FIGURE 1. Illustration of the problem.

shows the relaxation of the proposed problem $(R_m||C_{\max})$ is strongly NP-hard, therefore, considered problem is also NP-hard in strong sense. For this reason, an MILP model is only used to solve small size of the problem exactly in this paper, since that to exactly solve real size of the NP-hard problem in the reasonable time is impossible, so in the most of literature studies, researchers have used of the approximate methods to find a better solution in a reasonable amount of time [5,10,16,18,22,27]. The proposed scheduling problem can be partitioned in two sub-problems; *i.e.*, grouping the jobs in the batches and scheduling of the batches on the machines, based on this strategic, this paper considers two categories of the heuristics algorithms; in first category, batching of the jobs is done by a Modified Full Batch Largest Process Time (FBLPT) role and then the allocation of the batches on the machines takes place with considering various the machine's scenarios; Earliest Idle time (EI), Shortest Completion time (SC) and Shortest Processing time (SP). In second category, the allocation of the jobs on the machines takes place with considering various machine's scenarios: Earliest Idle time (EI), Shortest Completion time (SC) and Shortest Processing time (SP) and then batching of the jobs on each machine is done by Full Batch Largest Process Time-Shortest Release Time (FBLPT-SRT) role. Therefore, in this paper, six efficient heuristics and an effective meta-heuristic algorithm are proposed to solve of the real size problems with the little computational effort, which is valuable in some of the practical applications. A lower bound is provided to evaluate the performance of the proposed algorithms. Figure 1 shows scheme of the proposed problem as graphically.

The rest of the paper is organized as follows: Section 2 includes a problem definition and mathematical modeling. The proposed heuristics and RKGA meta-heuristic are presented in Section 3. Design of experiments, parameters setting and a lower bound are presented to evaluate the quality of the proposed algorithms as

computational experiments in Section 4. Finally, Section 5 presents the conclusion and the suggestions for the future researches.

2. Problem formulation

2.1. Problem description

This paper considers the unrelated parallel batch processing machines scheduling problem with the following assumptions:

- There are N jobs with arbitrary job size, dynamic job arrival, and different processing time on the machines.
- There are M unrelated parallel machines that have different speed and capacity.
- Total job sizes within each batch should not be exceeded than allocated machine's capacity.
- No job is allowed to split on the different batches.
- Each batch must be processed without interruption (Once a batch is being processed, the batch-processing machine cannot be interrupted; no jobs can be removed from the machine until the processing of the batch is completed).
- The machine is assumed to be available continuously and breakdown is not allowed.
- The machines cannot be idle when be existed at least one non-completed batch.
- The size of the jobs does not exceed of the largest machine capacity

$$Q_{\max} = \max_{k \epsilon M} \{B_k\},\ (s_j \leq Q_{\max}, \forall j \epsilon J)\,.$$

- All of the parameters are non-negative integer number.

2.2. Mathematical description of the problem

This subsection provides a mathematical description of the problem, a set of N jobs $J = \{1, 2, 3, \ldots, n\}$ must be processed on a set of M machines $M = \{1, 2, 3, \ldots, m\}$. The following indices, parameters and decision variables are used in the mathematical model:

Indices

j Index for jobs ($j = 1, 2, \ldots, n$).
i Index for batche ($i = 1, 2, \ldots, n$).
k Index for machine ($k = 1, 2, \ldots, m$).

Parameters (non-negative integer number)

s_j Size of job j.
r_j Release date of job j.
p_{jk} Processing time of job j on machine k.
B_k Capacity of machine k.

Decision variables

Independent decision variables (binary)

$X_{jik} = 1$, if job j is assigned to batch i on machine k; 0, otherwise.

Dependent decision variables (non-negative integer number)

P_{ik} = processing time of batch i on machine k.
ST_{ik} = start time of batch i on machine k.
C_{ik} = completion time of batch i on machine k.

2.3. The mathematical modeling

In this subsection, the objective function and the constraints are formulated as follows:

$$\min Z = C_{\max} \tag{2.1}$$

s.t.:

$$\sum_{k=1}^{m}\sum_{i=1}^{n} X_{jik} = 1 \qquad \forall j = 1,\ldots,n \tag{2.2}$$

$$X_{j'i+1k} \leq X_{jik} \qquad \forall, j = 1,\ldots,n,\ j' \neq j,\ k = 1,\ldots,m \tag{2.3}$$

$$\sum_{j=1}^{n} X_{jik} * s_j \leq B_k \qquad i = 1,\ldots,n,\ \forall k = 1,\ldots,m \tag{2.4}$$

$$X_{jik} * r_j \leq \mathrm{ST}_{ik} \qquad \forall i, j = 1,\ldots,n,\ \forall k = 1,\ldots,m \tag{2.5}$$

$$C_{\mathrm{i-1k}} \leq \mathrm{ST}_{ik} \qquad \forall i = 2,\ldots,n,\ \forall k = 1,\ldots,m \tag{2.6}$$

$$X_{jik} * p_{jk} \leq P_{ik} \qquad \forall i, j = 1,\ldots,n,\ \forall k = 1,\ldots,m \tag{2.7}$$

$$\mathrm{ST}_{ik} + P_{ik} \leq C_{ik} \qquad \forall i = 1,\ldots,n,\ \forall k = 1,\ldots,m \tag{2.8}$$

$$C_{ik} \leq C_{\max} \qquad \forall i = 1,\ldots,n,\ \forall k = 1,\ldots,m \tag{2.9}$$

$$P_{ik} \geq 0\,(\text{integer}), C_{ik} \geq 0\,(\text{integer}), \mathrm{ST}_{ik} \geq 0\,(\text{integer}) \qquad \forall i = 1,\ldots,n,\ \forall k = 1,\ldots,m \tag{2.10}$$

$$X_{jik} = 0 \text{ or } 1 \qquad \forall i, j = 1,\ldots,n,\ \forall k = 1,\ldots,m. \tag{2.11}$$

In the proposed mathematical model, objective function (2.1) is minimization of Makespan. Constraint set (2.2) ensures that each job should be assigned to the one batch and be just processed on the one machine. Constraint set (2.3) guaranties the sequence of the batches on each machine, it means that batch $i+1$ on machine k can be processed when batch i has been processed on machine k. Constraint set (2.4) ensures that the total size of the jobs in batch i that is assigned to machine k does not exceed of the capacity of machine k. Constraint sets (2.5) and (2.6) indicate relationship of the start time of batch i on the machine k with the largest release time of the jobs in same batch and completion time of previous batch $i-1$ on the machine k, respectively. These constraints guarantee that each batch can be processed when it is ready and the previous one on the machine has been processed. Constraint sets (2.7) and (2.8) calculate the processing time and the completion time of batch i on the machine k, respectively. In the other word, process time of each batch is equal to largest job processing time in same batch and completion time each batch is equal to the start time of the batch in addition to the processing time of same batch. Constraint set (2.9) defines objective function of the model (Makespan; which is equal to the largest batch completion time among all the batches). Constraint sets (2.10) and (2.11) represent non-negative and binary decision variables.

The proposed mathematical model coded in commercial solver (Lingo 11), on an hp 4520 s laptop with 4 GB of RAM and a 3 GHz processor running in windows 7. Since that the proposed problem is Np-Hard, the proposed model can be solved small size instances optimality and it expends an exponential CPU time with ascending problem size and unable to achievement optimal solution for large problem instances in the reasonable time. Therefore, the proposed heuristics and meta-heuristic algorithm are described to approximately solve large-sized problem instances in reasonable time as the following.

3. Approximate solution approaches

3.1. Heuristics

The proposed scheduling problem can be partitioned in two sub-problems; *i.e.*, grouping the jobs in the batches and scheduling the batches on the machines Based on this strategic this paper proposes two categories of heuristic algorithms as following:

TABLE 2. Data of numerical example.

Jobs	1	2	3	4	5	6	7	8	9	10
s_j	4	2	2	1	2	3	4	2	3	3
r_j	1	3	0	5	8	7	12	2	10	2
P_{j1}	3	2	1	5	8	9	2	6	7	3
P_{j2}	9	5	8	2	1	6	5	8	3	4
P_{j3}	5	8	2	2	6	4	3	2	1	1
Average$_{p_j}$	5.6	5.0	3.6	3.0	5.0	6.3	3.3	5.3	3.6	2.6

TABLE 3. Structure of proposed heuristics.

	Stage 1	Stage 2	Proposed Heuristics
Category 1	FBLPT-SRT (batching of jobs)	EI, SC, SP (schedule batches on the machines)	FBLPT-SRT|EI FBLPT-SRT|SC FBLPT-SRT|SP
Category 2	JSRT-EI, JSRT-SC, JSRT-SP (allocation of jobs on the machines)	FBLPT-SRT (batching of jobs on the machines)	JSRT-EI|FBLPT-SRT JSRT-SC|FBLPT-SRT JSRT-SP|FBLPT-SRT

First category includes two stages as follows:

Stage 1: grouping the jobs in the batches.
Stage 2: scheduling of the batches on the machines.

Second category considers two stages as follows:

Stage 1: allocating of the jobs to the machines.
Stage 2: batching of the jobs on each batch processing machine.

A numerical example is provided to better understanding of the proposed heuristics, consider 3 machines with capacity; $B_1 = 6$, $B_2 = 4$ and $B_3 = 5$ and 10 jobs with the parameters of Table 2.

Table 3 shows the structure of proposed heuristics which are described in more.

3.1.1. First category heuristics

Procedures of this category are described by provided numerical example as following:

Stage 1. Batching the jobs; since that each job has m processing times on the machines and also each machine has different capacity, Full Batch Largest Processing Time (FBLPT) role cannot be used directly [29]. Thus, we Modified FBLPT role as follows:

Step 1. Arrange the jobs in the decreasing order of $\sum_{k\in M} P_{jk}/|M|$.
In the proposed numerical example job order be: J_6, J_1,J_8,J_2,J_5,J_3,J_9,J_7,J_4,J_{10}.

Step 2. Select the job at head of the above list and place it in a feasible batch with the smallest residual capacity. If the job fits in no existing batches, place the job in a new batch. Repeat step 2 until all of the jobs have been assigned to the batches. Note that the capacity of batches is equal to the smallest machine's capacity.
In the proposed numerical example the created batches be: $\{J_6\}$, $\{J_1\}$, $\{J_8, J_2\}$, $\{J_5, J_3\}$, $\{J_9\}$, $\{J_7\}$, $\{J_4, J_{10}\}$.

Step 3. Sort the created batches in the previous step according to the Shortest Ready Time (SRT) role and call it *Sort Batch list*.
In the proposed numerical example *Sort Batch list* be: $\{J_1\}$, $\{J_8, J_2\}$, $\{J_4, J_{10}\}$, $\{J_6\}$, $\{J_5, J_3\}$, $\{J_9\}$, $\{J_7\}$.

Stage 2. Schedule of *Sort Batch list* on the machines; three methods are considered to assignment the batches on the machines as follows:

(A) *Earliest Idle time (EI)*: select batch i at head of *Sort Batch list* and then allocate it to the earliest idle machine. If more than one machine can be idle at the same time then it is allocated to the machine with the shortest processing time. Repeat until all of the batches have been scheduled.

(B) *Shortest Completion time (SC)*: select batch i at head of the *Sort Batch list* and then allocate it to the machine that completes it in the shortest completion time. Repeat until all of the batches have been scheduled.

(C) *Shortest Processing time (SP)*: select batch i at head of the *Sort Batch list* and then allocate it to the machine k which has the shortest processing time. It means, $K = \arg\min\{P_k^i | k \in M\}$. If K can be shown more than one machine then it is allocated to the machine with the less completion time. Repeat until all of the batches have been scheduled.

Note: At the end of this subsection, the batches on each machine must be merged together so that the size of each batch is not exceeded of the machine's capacity.

Three heuristics are extracted in this subsection, which are named FBLPT-SRT|EI, FBLPT-SRT|SC, and FBLPT-SRT|SP, respectively. Results of proposed heuristics on the numerical example are shown in Figure 2.

Stage 1. Grouping jobs into batches.

FBLPT-SRT	B_1	B_2	B_3	B_4	B_5	B_6	B_7
JOB	{1}	{8, 2}	{4, 10}	{6}	{5, 3}	{9}	{7}
Batch ready time	1	3	5	7	8	10	12
P_{i1}	3	6	7	9	8	7	2
P_{i2}	9	8	3	6	8	3	5
P_{i3}	5	8	2	4	6	1	3

Stage 2. Allocate batches to machines.

3.1.2. Second category heuristics

Procedures of this category are described by provided numerical example as following:

Stage 1. Allocating of the jobs to the machines as following:

Step 1. Arrange the jobs in Shortest Ready Times role and named JSRT list.

In the proposed numerical example JSRT list is equal to: $J_3, J_1, J_8, J_{10}, J_2, J_4, J_6, J_5, J_9, J_7$.

Step 2. Three methods are presented to assign to JSRT list to the machines as follows:

(A) *Earliest Idle Time role*: allocate job j at head of JSRT list to the earliest idle machine. If more than one machine can be idle at the same time then job j is allocated to the machine with the shortest processing time. Repeat until all of JSRT list have been assigned to the machines. In the proposed numerical example allocation of the jobs on the machines be: $M_1 = [J_3, J_1, J_2, J_7]$, $M_2 = [J_{10}, J_5, J_9]$ and $M_3 = [J_8, J_4, J_6]$.

(B) *Shortest Completion Time role*: allocate job j at head of JSRT list to the machine that completes it in the shortest completion time. Repeat until all of JSRT list have been assigned to the machines. In the proposed numerical example allocation of the jobs on the machines be: $M_1 = [J_3, J_1, J_{10}, J_6, J_7]$, $M_2 = [J_4, J_5]$ and $M_3 = [J_8, J_2, J_9]$.

(C) *Shortest processing time role*: allocate job j at head of JSRT list to the machine that processes it with the shortest processing time. If job j has identical processing time on the more than one machine then it is allocated to the machine with the shortest completion time. Repeat until all of JSRT list have been assigned to the machines. In the proposed numerical example allocation of the jobs on the machines be: $M_1 = [J_3, J_1, J_2, J_7]$, $M_2 = [J_4, J_5]$ and $M_3 = [J_8, J_{10}, J_6, J_9]$.

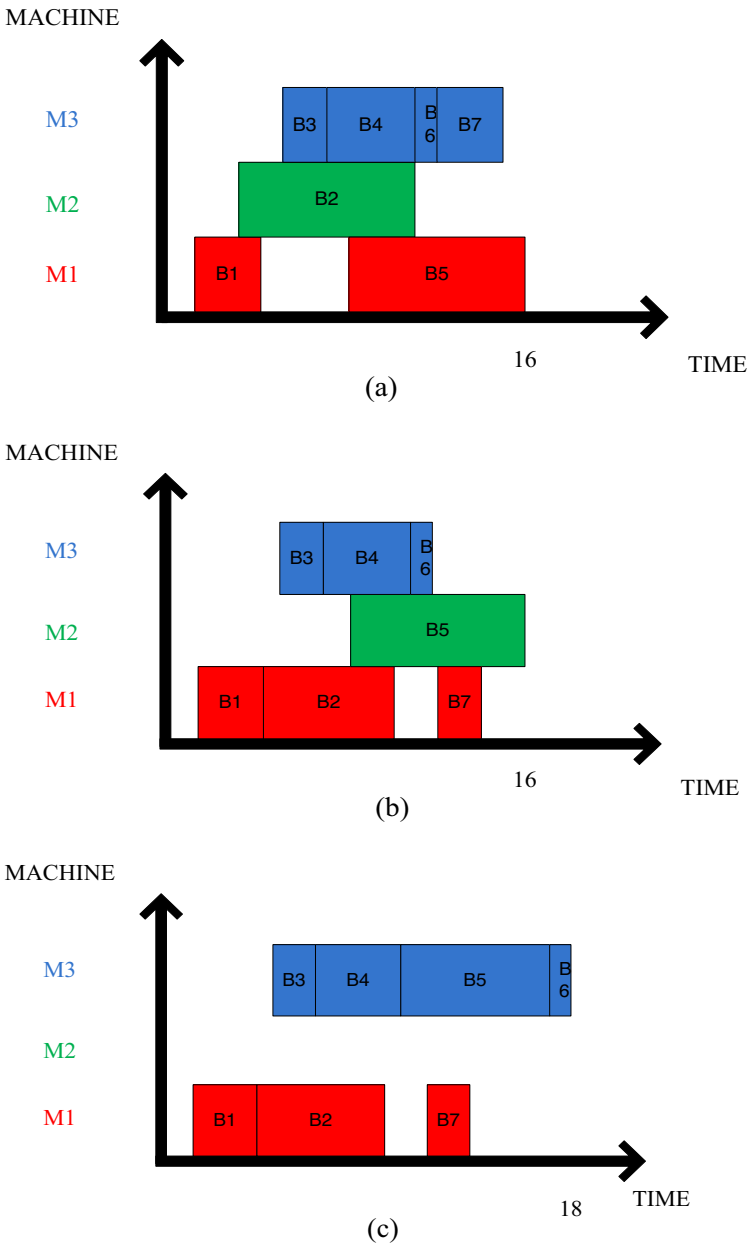

FIGURE 2. Results of first category heuristics on the Numerical example. (a) FBLPT-SRT|EI. (b) FBLPT-SRT|SC. (c) FBLPT-SRT|SP.

Stage 2. Batching the jobs on each machine; the assigned jobs to the machine k must be batched according to Full Batch Largest Processing Time (FBLPT) role, such that size of the batches on the machine k don't exceed of machine's capacity (B_k). Let L_k be a list of the created batches on machine k. The batches of L_k must be ordered according to the Shortest Ready Time (SRT) role. Finally, three heuristics are extracted in this subsection, which are named JSRT-EI|FBLPT-SRT, JSRT-SC|FBLPT-SRT and JSRT-SP|FBLPT-SRT, respectively. Results of proposed heuristics on the numerical example are shown in Figure 3.

Stage 1. Allocate jobs to machines.

Machines	M_1	M_2	M_3
JSRT-EI	3, 1, 2, 7	10, 5, 9	8, 4, 6
JSRT-SC	3, 1, 10, 6, 7	4, 5	8, 2, 9
JSRT-SP	3, 1, 2, 7	4, 5	8, 10, 6, 9

Stage 2. Grouping jobs into batches on each machine by FBLPT-SRT.

3.2. RKGA meta-heuristic

Genetic Algorithm (GA) has been presented by Holland [8] at the first time which has been used to solve combinational optimization problems. RKGA is a special type of GA which is introduced by Bean [2] for the first time and it has been used in the literature [12, 14, 26]. This paper proposes RKGA to solve of the $R_m|p\text{-batch}, p_{jk}, s_j, r_j, B_k|C_{\max}$ problem in the real size instance. The proposed RKGA is described in more detail as following.

3.2.1. Encoding & Decoding

In this subsection, random key representation scheme proposed by Bean [2] is developed to generate initial population as much as the *population size*. The main advantage of the considered representation is covering of the entire solution space, *i.e.*, there is a unique string associated with the every feasible solution for the problem. The proposed chromosome included a matrix $m \times n$ where all of the elements are 0 or 1, so that in each column, number 1 is only repeated at the once, after allocation of the jobs on the machines, a FBLPT-SRT role is used for batching of the jobs and scheduling of the batches on the machines. Figure 4 shows an encoding and decoding chromosome on the provided numerical example.

In Figure 4, random key chromosome allocates the jobs to the machines as: $M_1 = [J_1, J_2, J_3, J_6, J_7]$, $M_2 = [J_4, J_5]$ and $M_3 = [J_8, J_9, J_{10}]$. The assigned jobs to the machine k must be batched according to Full Batch Largest Processing Time (FBLPT) role, such that the size of the batches on the machine k don't exceed of the machine's capacity (B_k) and then the created batches on the machine k must be arranged according to the Shortest Ready Time (SRT) role which in the numerical example, the sequence of the batch on the machines is equal to: $M_1 = \{J_1, J_2\}, \{J_6\}, \{J_3, J_7\}$, $M_2 = \{J_4, J_5\}$, and $M_3 = \{J_{10}\}, \{J_8, J_9\}$.

3.2.2. Evaluation & selection strategies

The fitness evaluation function assigns a value to each member of the population which is reflecting their relative superiority. In the proposed algorithm, the evaluation of the solutions is based on objective function value (Makespan), in the other word; a solution with the lower Makespan is preferable, therefore, $1/C_{\max}$ (the inversed Makespan) is as the proposed fitness function. The proposed selection strategy is based on the roulette wheel proposed by Holland [8].

3.2.3. Proposed operators for reproduction

Since of the proposed RKGA is a population-based approach, at first a random population is generated much as *population size*. Afterward, to reproduction from one generation to the next one is done with an elitist strategy, the mutation and the crossover operators which the mechanisms of the operators are shown in Figures 5 and 6.

3.2.4. Termination criterion

A population based algorithm repeats iterations until to reach a termination condition such as max-time, max-generations, and/or the generations without improvement. In this paper, RKGA is stopped, if the maximum number of generations (max_gen) and/or the pre-determined iteration without improvement in the best solution (max_no_improve) to be happened. Figure 7 shows structure of the proposed RKGA. The more details of the proposed RKGA are provided by a Pseudo-code in the appendix section.

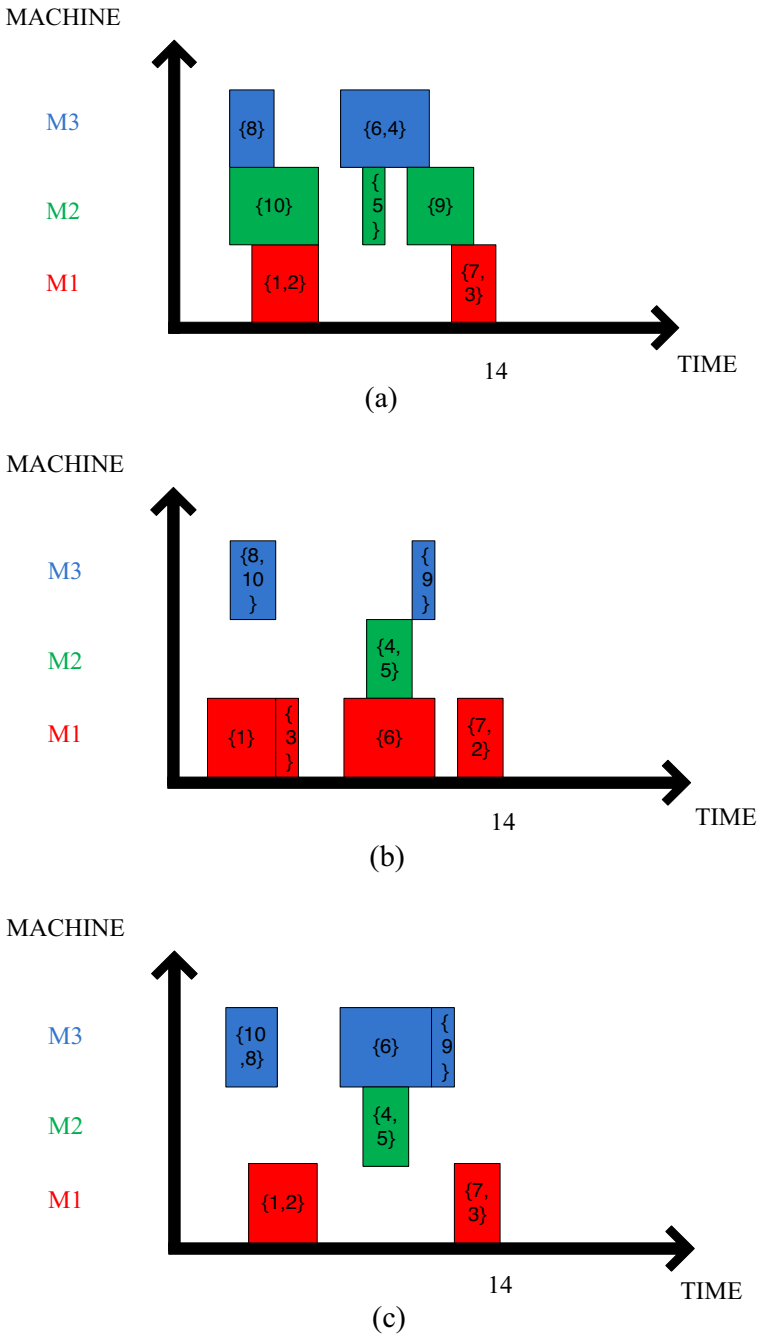

FIGURE 3. Results of second category heuristics on the Numerical example. (a) JSRT-EI|FBLPT-SRT. (b) JSRT-SC|FBLPT-SRT. (c) JSRT-SP|FBLPT-SRT.

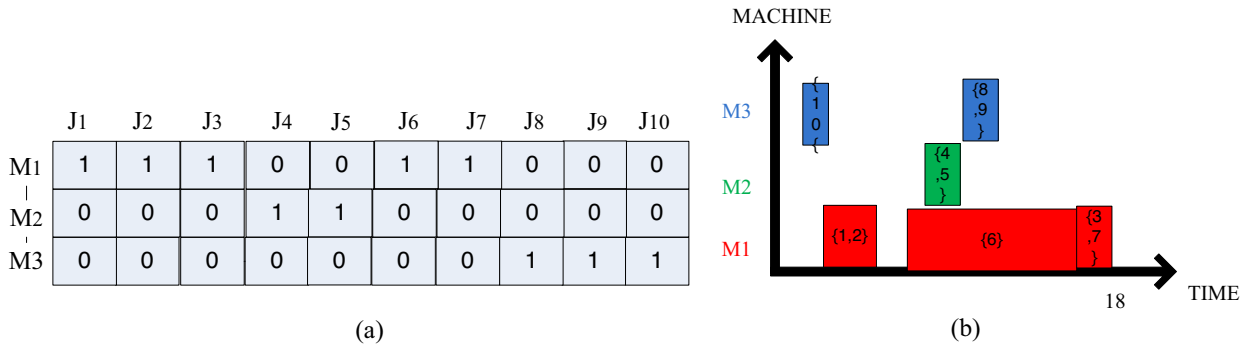

FIGURE 4. A Encoding & Decoding on the numrical example. (a) Encoding of chromosome. (b) Decoding of random key chromosome.

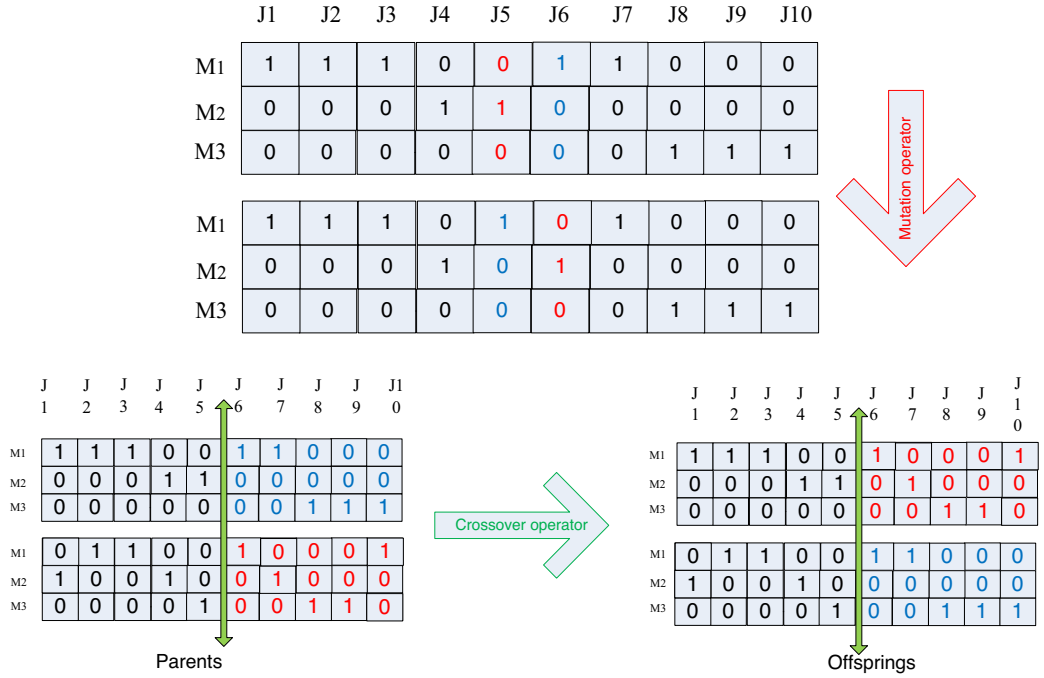

FIGURE 5. Operators.

4. COMPUTATIONAL EXPERIMENTS

4.1. Data generation & parameters setting

Random test problem instances with number of the jobs equal to $N = 10, 20, 50, 100$, and 200 and the number of the machines equal to, $M = 2, 3, 4$, and 5 are considered to evaluate the efficiency of the proposed algorithms. The parameters of machine's capacity, job processing, job ready time, and job size are generated with discrete uniform distributions in intervals $[4, 10]$, $[1, 50]$, $[0, 50]$, and $[1, \max_k \{B_k\}]$, respectively. The parameters of the considered problem instances and their levels are shown in Table 4 which 20 test problem instances are extracted from it. The problem instances are symbolized as $N_j M_k$ for example, $N_1 M_2$ *i.e.*, the problem instance with

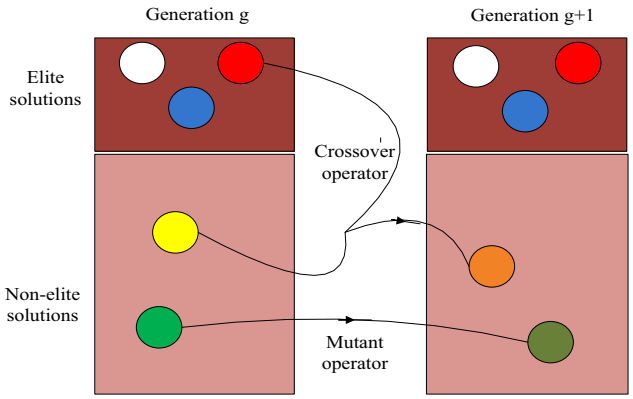

FIGURE 6. Evolutionary structure.

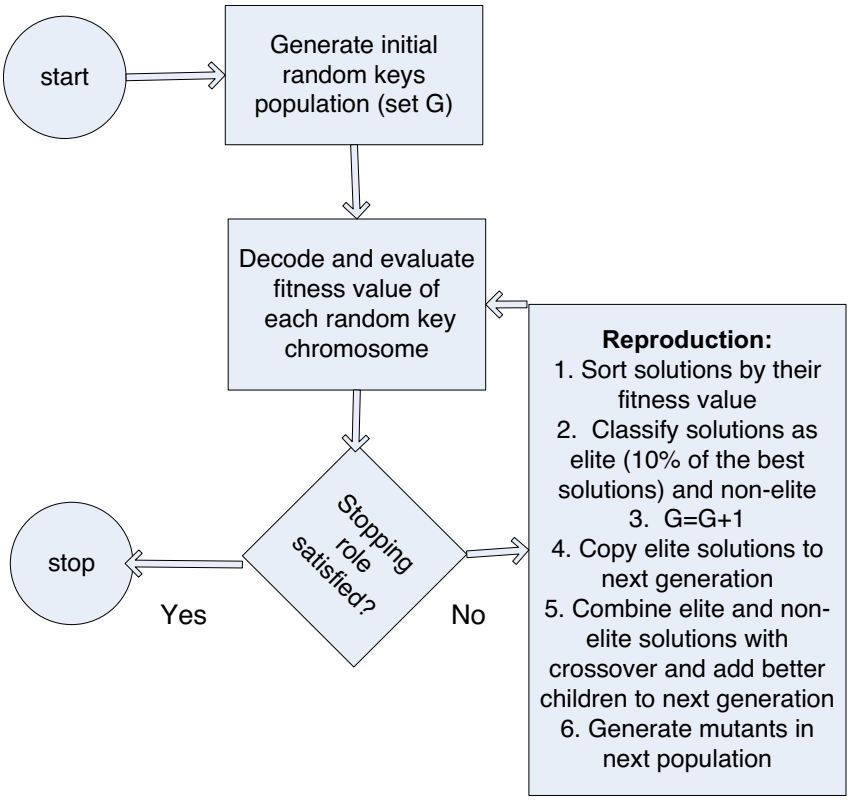

FIGURE 7. Structure of the proposed RKGA.

TABLE 4. Parameters and their levels.

Parameters	Levels
Number of jobs	$N_1 = 10, N_2 = 20, N_3 = 50, N_4 = 100, N_5 = 200$
Number of machines	$M_1 = 2, M_2 = 3, M_3 = 4, M_4 = 5$
Machine's capacity	$B_K \sim \cup [4, 10]$
Processing time	$P_{jK} \sim \cup [1, 50]$
Ready time	$R_j \sim \cup [0, 50]$
Size of jobs	$S_j \sim \cup [1, \max_k \{B_k\}]$

TABLE 5. Factors of RKGA and their levels.

Factors	Levels	Outputs of Taguchi (best level)
Max_gen	100–150–200-250–300	150
Max_no_improve	5–10–15–20	10
Population size	150–200–250–300	250
Crossover rate	0.1–0.15–0.2–0.25	0.15
Mutate rate	0.02–0.04–0.05–0.07	0.02

$N = 10$ and $M = 3$ is considered. There are several effective factors on the performance of the proposed RKGA such as: Max_generation, Max_no_improvement, Population size, Crossover rate, and Mutate rate. In this study, Taguchi experimental design is applied to analyze affect of the parameters of RKGA [24], the appropriate orthogonal array will be $L_{17}\left(4^4, 5^1\right)$ for the considered factor's levels of Table 5 which the readers can be refer to Zarook *et al.* [28] in more details of parameters setting. All of the possible combinations of the factors were executed on the random test problems for the run five times. Finally, the results of parameters setting are tuned in the last column of Table 5.

4.2. Lower bound

In this subsection, a Lower Bound (LB) is presented for the problem so that the comparison of between the proposed algorithms can be possible. Since of the proposed problem is strongly NP-hard, it is computationally infeasible to obtain optimal solutions for large-scale instances, this is another reason why a lower bound is needed. The proposed algorithms are compared with the MILP model in the small size instances and with the proposed lower bound in the large size instances, respectively. The proposed LB is achieved by relaxing of $R_m|\text{batch}, p_{jk}, s_j, r_j, B_k|C_{\max}$ (original problem) to $P_m|\text{batch}, prmp, r_j, s_j, p_j, B'|C_{\max}$ problem, it means that the unrelated parallel machine is relaxed to the identical parallel machine with $p_j = \min_{k\in M}\{p_{jk}\}$ and machine's capacity equal to $B' = \max_{k\in M}\{B_k\}$ which the proposed LB is provided on the relaxed problem as following:

(1) Let J be the set of all jobs and J_1 be the subset of J so that satisfies equation (4.1). Let each job from J_1 is assigned to a separate batch.

$$J_1 = \left\{ j \in J | B' - s_j < \min_{i\in J}\{s_i\} \right\}. \tag{4.1}$$

(2) Let $J_2 = \{J \backslash J_1\}$, convert each job $j \in J_2$ to s_j unit size jobs with processing time p_j and ready time r_j. Arrange the created jobs in descending order of their processing times and then assign them into the batches with maximum capacity B', in the other words, the minimum number of batches is equal to $\left\lceil \frac{1}{B'} \sum_{j\in J_2} s_j \right\rceil$.

(3) Let B be set of all the created batches from two previous steps and P_b be the processing time of batch $b \in B$. Arrange B in descending order of their processing times such as $P_1 \geq P_2 \geq \ldots \geq P_{|B|}$. Therefore, a lower

bound on the relaxation problem is calculated as equation (4.2).

$$C_{\max}^{\mathrm{LB}} = \max\left\{\max_{j\in J}\{r_j + p_j\}, \left\lceil \frac{\sum_{b\in B} P_b}{|M|} \right\rceil + \min_{j\in J}\{r_j\}, P_{|B|-1} + P_{|B|} + \min_{j\in J}\{r_j\}\right\}. \tag{4.2}$$

For example consider the provided numerical example in Section 3 which $B' = 6$, $|M| = 3$ and $P_j = \{3, 2, 1, 2, 1, 4, 2, 2, 1, 1\}$.

Step 1. $J_1 = \{\emptyset\}$.
Step 2. $J_2 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.
Step 3. Let $|B| = \left\lceil \frac{1}{B'} \sum_{j\in J_2} s_j \right\rceil = \lceil \frac{26}{6} = 5 \rceil$, so that five created batches are formed as following:

$$\{6, 1\}, \{1, 2, 4, 7\}, \{7, 8, 3\}, \{5, 9, 10\}, \{10\}$$
$$C_{\max}^{\mathrm{LB}} = \max\left\{14, \left\lceil \frac{11}{3} \right\rceil + 0, 1 + 1 + 0\right\} = 14.$$

In this numerical example, the value of LB, second category of heuristics and mathematical model are equal to $C_{\max}^{*} = 14$.

4.3. Comparison of the proposed algorithms

In this section, all of the extracted problem instances of Table 4, are solved by the proposed approaches (mathematical model, LB, heuristics and RKGA meta-heuristic). The proposed mathematical model is programmed in Lingo11 (commercial solver) and the proposed heuristics, the proposed lower bound, and RKGA meta-heuristic have been coded with C++ language (in Matlab7 software) on an HP 4520 s laptop with 4 GB of RAM and a 3 GHz processor running on Windows 7. The results of the solutions (Makespan) are reported at Table 6. The results of Table 6 show MILP model solves problem instances 1–6 as optimality and MILP is unable to solve problem instances 7 until 14 as optimality, therefore, the best of obtained solution in 1800 s is considered as local optimal solution. Also MILP obtains any solution for large instance 15 until 20 in 1800 s whereas the proposed heuristics, LB and meta-heuristic reach to the solution in reasonable computational time for these problem instances, approximately.

One of the measures to evaluate of the proposed algorithms is Relative Percentage Deviation (RPD) measure which is calculated according to the equation (4.3).

$$\mathrm{RPD}_i^A = \frac{C_{\max_i}^A - C_{\max_i}^{\mathrm{LB}}}{C_{\max_i}^{\mathrm{LB}}}. \tag{4.3}$$

In this equation, RPD_i^A is RPD measure of algorithm A on the problem instance i and $C_{\max_i}^A$, $C_{\max_i}^{\mathrm{LB}}$ are equal to $C_{\max}$ of algorithm A and the proposed LB on the problem instance i, respectively. In the other word, RPD_i^A value shows Gap between algorithm A and LB which the smaller value of RPD_i^A shows a better algorithm A. Table 7 reports RPD measure of the proposed algorithms on the all of the problem instances.

In equation (4.4), X_A^{RPD} is the mean and S_A^{RPD} is the standard deviation of the RPD of the proposed algorithm A which are calculated on the RPD values of the Table 7, t is the t-distribution in the uncertainty level (α) is equal to 5% and $n = 20$ is degrees of freedom ($t_{0.025,19} = 1.48$). Table 8 is calculation report of Minitab software according to equation (4.4) and Figure 8 shows interval plot of RPD measure according to Table 8 and also the GAP between of the proposed algorithms is observable in this figure.

$$\begin{aligned}\left(L_A^{\mathrm{RPD}}, U_A^{\mathrm{RPD}}\right) &= \left(\bar{X}_A^{\mathrm{RPD}} - \mathrm{LSD}_A^{\mathrm{RPD}}, X_A^{\mathrm{RPD}} + \mathrm{LSD}_A^{\mathrm{RPD}}\right) \\ &= \left(\bar{X}_A^{\mathrm{RPD}} - \left(t_{\frac{\alpha}{2},n-1} \cdot S_A^{\mathrm{RPD}} \sqrt{\frac{2}{n}}\right), \bar{X}_A^{\mathrm{RPD}} + \left(t_{\frac{\alpha}{2},n-1} \cdot S_A^{\mathrm{RPD}} \sqrt{\frac{2}{n}}\right)\right).\end{aligned} \tag{4.4}$$

Table 6. Makespan value of experiments.

Run code	First category heuristics			Second category heuristics			Meta-heuristic RKGA	MILP (Lingo)	LB
	FBLPT-SRT|EI	FBLPT-SRT|SC	FBLPT-SRT|SP	JSRT-EI|FBLPT-SRT	JSRT-SC|FBLPT-SRT	JSRT-SP|FBLPT-SRT			
N_1M_1	50.4	40.6	59.3	51.6	40.6	56.9	45.8	40.6*	36.9
N_1M_2	48.9	38.6	62.5	50.4	39.8	60.7	50.2	35.9*	31.2
N_1M_3	41.9	35.9	59.6	49.7	35.4	55.9	43.9	30.9*	25.3
N_1M_4	36.9	31.1	50.1	45.6	32.3	50.6	45.7	23.6*	19.8
N_2M_1	185.9	170.5	191.3	180.4	167.6	185.8	170.1	165.0*	153.6
N_2M_2	180.6	165.5	185.2	179.8	166.6	184.5	172.4	161.3*	148.9
N_2M_3	168.8	160.1	174.6	167.7	159.8	170.2	155.5	151.2[a]	141.4
N_2M_4	158.8	149.9	169.7	156.6	144.4	166.8	164.8	130.8[a]	130.8
N_3M_1	290.0	285.2	291.3	292.3	288.8	294.5	308.3	271.1[a]	260.4
N_3M_2	281.7	276.8	288.5	274.2	266.6	281.7	300.5	260.5[a]	256.7
N_3M_3	272.3	252.1	280.9	271.6	248.7	277.9	275.9	245.3[a]	245.0
N_3M_4	270.0	249.8	278.4	269.6	245.5	271.5	267.7	241.0[a]	232.5
N_4M_1	515.5	501.4	532.8	510.7	495.5	515.4	550.4	492.0[a]	465.8
N_4M_2	509.6	496.8	526.7	506.7	490.1	511.8	536.7	484.4[a]	460.1
N_4M_3	503.9	487.6	519.8	498.2	478.4	505.5	509.9	–	452.7
N_4M_4	496.6	470.7	509.1	489.5	461.8	495.2	489.3	–	443.8
N_5M_1	1238.7	1006.8	1368.8	1201.5	991.9	1329.4	1480.1	–	872.9
N_5M_2	1195.4	968.9	1306.9	1115.6	913.7	1250.2	1350.0	–	864.6
N_5M_3	1108.8	935.5	1263.3	1090.5	901.0	1100.8	1307.8	–	852.9
N_5M_4	1070.9	905.7	1150.8	1000.6	884.6	1085.1	1279.6	–	842.1

Notes. *Optimal (exact) solution in 1800″. [a]Local optimum solution (best solution in 1800″). –Commercial solver unable to achieve any solution in time 1800″ (s).

According to Figure 8, the priority of proposed algorithms in aspect of minimum objective value (or mean RPD) and maximum stability (or minimum deviation RPD) as follows: JSRT-SC|FBLPT-SRT ≻ FBLPT-SRT|SC ≻ FBLPT-SRT|EI ≻ JSRT-EI|FBLPT-SRT ≻ RKGA ≻ JSRT-SP|FBLPT-SRT ≻ FBLPT-SRT|SP.

Since that, another measure to compare the proposed algorithms is CPU time; Figure 9 shows the behavior of the proposed algorithms based on CPU time measure on the large size problem instances (problems 11–20) According to this measure, RKGA has the shortest CPU time and the priority of the proposed algorithms in terms of minimum CPU time on the large size instance problems as follows: RKGA ≻ FBLPT-SRT|SP ≻ JSRT-SP|FBLPT-SRT ≻ JSRT-EI|FBLPT-SRT ≻ FBLPT-SRT|EI ≻ FBLPT-SRT|SC ≻ JSRT-SC|FBLPT-SRT.

5. Conclusions and future research

In this paper, the minimization of Makespan in the unrelated parallel batch processing machine environment studied with considering non-identical job sizes and dynamic job ready times which the unrelated parallel batch-processing machines have different capacity and speed. At first, a MILP model is presented for this problem. Since of the problem is NP-hard, 6 heuristics and a meta-heuristic algorithm are proposed to solve the problem in the real size instances. Heuristics classified in two categories; category1 allocate the jobs into the batches and then the batches are scheduled on the machines. On the contrary, category2 allocate the jobs on the machines and then each machine is considered as a single batch processing machine scheduling problem. Finally, a lower bound is proposed to evaluate the quality of the proposed algorithms.

The computational results showed the priority of the proposed algorithms in aspect of minimum objective value (or mean RPD) and maximum stability (or minimum deviation RPD) as follows: JSRT-SC|FBLPT-SRT ≻ FBLPT-SRT|SC ≻ FBLPT-SRT|EI ≻ JSRT-EI|FBLPT-SRT ≻ RKGA ≻ JSRT-SP|FBLPT-SRT ≻ FBLPT-SRT|SP and the priority of the proposed algorithms in terms of minimum CPU time on the large size

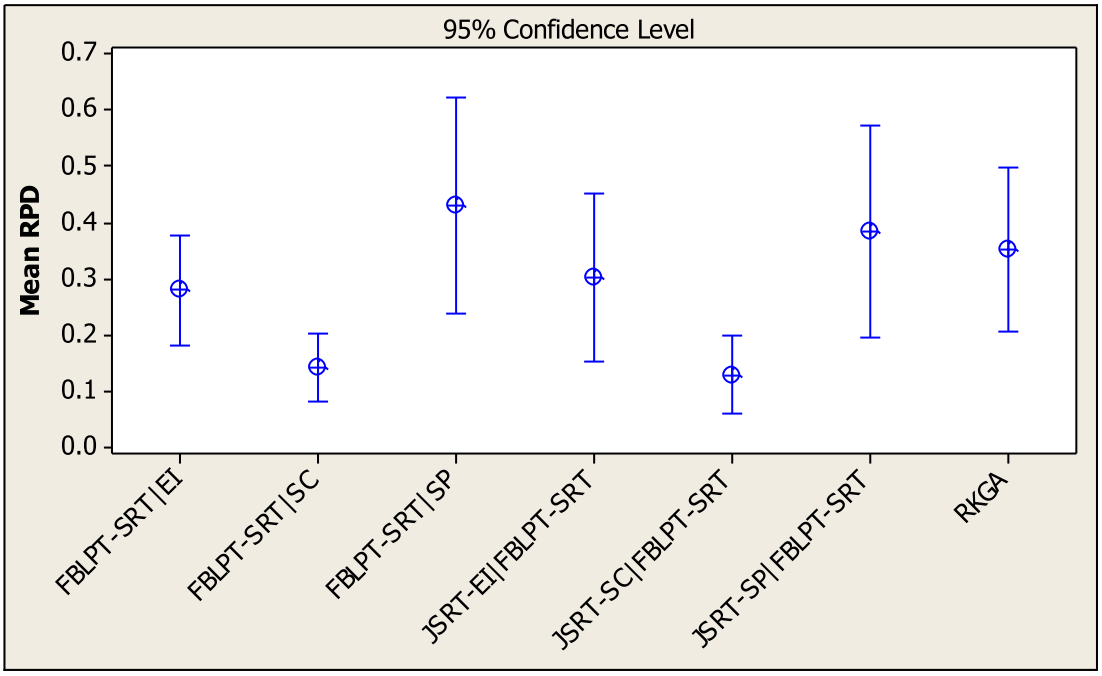

FIGURE 8. Interval Plot of proposed algorithms on the all instance problem.

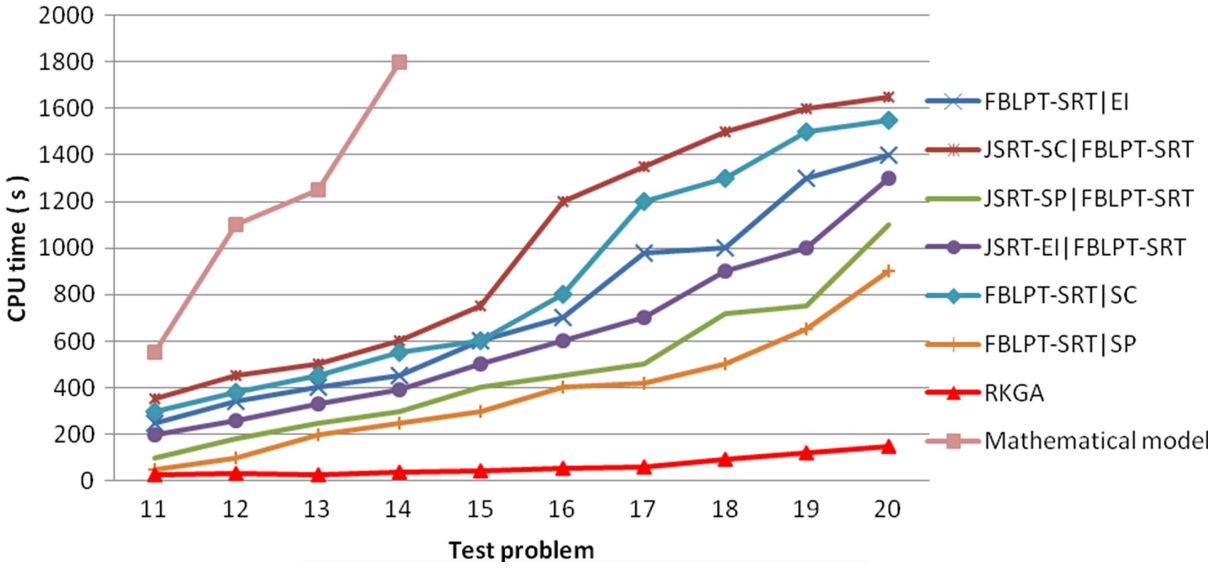

FIGURE 9. CPU time of proposed algorithms.

Table 7. RPD measure of the proposed algorithms.

Run code	Heuristics 1			Heuristics 2			Meta-heuristic RKGA
	FBLPT-SRT|EI	FBLPT-SRT|SC	FBLPT-SRT|SP	JSRT-EI|FBLPT-SRT	JSRT-SC|FBLPT-SRT	JSRT-SP|FBLPT-SRT	
N_1M_1	0.365	0.100	0.607	0.398	0.100	0.542	0.241
N_1M_2	0.567	0.237	1.003	0.615	0.275	0.945	0.608
N_1M_3	0.656	0.418	1.355	0.964	0.399	1.209	0.735
N_1M_4	0.863	0.570	1.530	1.303	0.631	1.555	1.308
N_2M_1	0.210	0.110	0.245	0.174	0.091	0.209	0.107
N_2M_2	0.212	0.111	0.243	0.207	0.118	0.239	0.157
N_2M_3	0.193	0.132	0.234	0.185	0.130	0.203	0.099
N_2M_4	0.214	0.146	0.297	0.197	0.103	0.275	0.259
N_3M_1	0.113	0.095	0.118	0.122	0.109	0.130	0.183
N_3M_2	0.097	0.078	0.123	0.068	0.038	0.097	0.170
N_3M_3	0.111	0.028	0.146	0.108	0.015	0.134	0.126
N_3M_4	0.161	0.074	0.197	0.159	0.055	0.167	0.151
N_4M_1	0.106	0.076	0.143	0.096	0.063	0.106	0.181
N_4M_2	0.107	0.079	0.144	0.101	0.065	0.112	0.166
N_4M_3	0.113	0.077	0.148	0.100	0.056	0.116	0.126
N_4M_4	0.118	0.060	0.147	0.102	0.040	0.115	0.102
N_5M_1	0.419	0.153	0.568	0.376	0.136	0.522	0.695
N_5M_2	0.382	0.120	0.511	0.290	0.056	0.445	0.561
N_5M_3	0.300	0.096	0.481	0.278	0.056	0.290	0.533
N_5M_4	0.271	0.075	0.366	0.188	0.050	0.288	0.519

Table 8. LSD implementation for RPD measure.

Algorithm	Mean ($\bar{X}$)	Standard deviation (S)	LSD	Lower limit	Upper limit
FBLPT-SRT|EI	0.27890	0.210379	0.09846	0.18044	0.37736
FBLPT-SRT|SC	0.14175	0.130315	0.06098	0.08077	0.20273
FBLPT-SRT|SP	0.43030	0.412515	0.19306	0.23723	0.62336
JSRT-EI|FBLPT-SRT	0.30155	0.318353	0.14899	0.15255	0.45054
JSRT-SC|FBLPT-SRT	0.12930	0.147809	0.06917	0.06012	0.19847
JSRT-SP|FBLPT-SRT	0.38495	0.402660	0.18845	0.19491	0.57341
RKGA	0.35135	0.313027	0.14651	0.20481	0.49781

instance problems as follows: RKGA $\succ$ FBLPT-SRT|SP $\succ$ JSRT-SP|FBLPT-SRT $\succ$ JSRT-EI|FBLPT-SRT $\succ$ FBLPT-SRT|EI $\succ$ FBLPT-SRT|SC $\succ$ JSRT-SC|FBLPT-SRT.

Several areas exist for future research, such as; considering other objective functions, multiobjective optimization and fuzzy parameters for this problem. Extension of the proposed model to the case of scheduling other shop systems, *e.g.*, flow shop, job shop machines, is worthwhile.

Appendix A. Pseudo code of proposed algorithms

Algorithm FBLPT-SRT|EI.

Stage 1:

1: Arrange the jobs in decreasing order of $\sum_{k\in M} P_{jk}/\,|M|$.
2: **for** $j = 1$ to n of above list **do**
3: Assign the job j to the feasible batch with the smallest residual capacity. Note that batches capacity is equal to the smallest machine capacity.
4: If there is not a batch or the job j fits in no existing batches, place the job in a new batch.
5: **end for**
6: Sort the batches of, in non-decreasing order of their ready times (batch ready times) and call it *Sort Batch list*.

Stage 2:

7: **for** $i = 1$ to $|$*Sort Batch list*$|$ **do**
8: Allocate batch i to the machine with the earliest idle time.
9: If more than one machine can be idle at the same time then it is allocated to the machine with the shortest processing time.
10: **end for**
11: At the end of this subsection, the batches on the each machine must be merged together so that the machine capacity is not exceeded.

Algorithm FBLPT-SRT|SC.

Stage 1:

1: Arrange the jobs in decreasing order of $\sum_{k\in M} P_{jk}/\,|M|$.
2: **for** $j = 1$ to n of above list **do**
3: Assign the job j to the feasible batch with the smallest residual capacity. Note that batches capacity is equal to the smallest machine capacity.
4: If there is not a batch or the job j fits in no existing batches, place the job in a new batch.
5: **end for**
6: Sort the batches of, in non-decreasing order of their ready times (batch ready times) and call it *Sort Batch list*.

Stage 2:

7: **for** $i = 1$ to $|$*Sort Batch list*$|$ **do**
8: Allocate batch i to the machine that completes it in the shortest completion time.
9: **end for**
10: At the end of this subsection, the batches on the each machine must be merged together so that the machine capacity is not exceeded.

Algorithm FBLPT-SRT|SP.

Stage 1:

1: Arrange the jobs in decreasing order of $\sum_{k\in M} P_{jk}/\,|M|$.
2: **for** $j = 1$ to n of above list **do**
3: Assign the job j to the feasible batch with the smallest residual capacity. Note that batches capacity is equal to the smallest machine capacity.
4: If there is not a batch or the job j fits in no existing batches, place the job in a new batch.
5: **end for**
6: Sort the batches of, in non-decreasing order of their ready times (batch ready times) and call it *Sort Batch list*.

Stage 2:

7: **for** $i = 1$ to $|$*Sort Batch list*$|$ **do**
8: Allocate batch i to the machine k which has the shortest processing time. That means, $K = \arg\min\{P_k^i | k \in M\}$.
9: If K can be shown more than one machine then it is allocated to the machine with less completion time.
10: **end for**
11: At the end of this subsection, the batches on the each machine must be merged together so that the machine capacity is not exceeded.

Algorithm JSRT-EI|FBLPT-SRT.

Stage 1:

1: Arrange the Jobs in Shortest Ready Times role and named JSRT list.
2: **for** $j = 1$ to n of JSRT LIST **do**
3: Assign job j to the machine that would have the earliest idle time.
4. If more than one machine can be idle at the same time then job j is allocated to the machine with the shortest processing time.
5: **end for**
6: A subset of jobs M_k for each machine k is determined.

Phase 2:

7: **for** each machine k **do**
8: Group the jobs of set M_k into batches by applying the FBLPT role such that size of the batches on the machine k don't exceed of capacity of machine $k\,(B_k)$.
9: If the job fits in no existing batches on machine k, place the job in a new batch.
10: **end for**
11: Let L_k be a list of the created batches on machine k. The batches of L_k must be ordered according to the Shortest Ready Time (SRT) role.

Algorithm JSRT-SC|FBLPT-SRT.

Stage 1:

1: Arrange the Jobs in Shortest Ready Times role and named JSRT list.
2: **for** $j = 1$ to n of JSRT LIST **do**
3: Assign job j to the machine that to the machine that completes it in the shortest completion time.
4: **end for**
5: A subset of jobs M_k for each machine k is determined.

Phase 2:

6: **for** each machine k **do**
7: Group the jobs of set M_k into batches by applying the FBLPT role such that size of the batches on the machine k don't exceed of capacity of machine $k\,(B_k)$.
8: If the job fits in no existing batches on machine k, place the job in a new batch.
9: **end for**
10: Let L_k be a list of the created batches on machine k. The batches of L_k must be ordered according to the Shortest Ready Time (SRT) role.

Algorithm JSRT-SP|FBLPT-SRT

Stage 1:
1: Arrange the Jobs in Shortest Ready Times role and named JSRT list.
2: **for** $j = 1$ to n of JSRT LIST **do**
3: Assign job j to the machine that processes it with the shortest processing time.
4. If job j has identical processing time on the more than one machine then it is allocated to the machine with the shortest completion time.
5: **end for**
6: A subset of jobs M_k for each machine k is determined.
Phase 2:
7: **for** each machine k **do**
8: Group the jobs of set M_k into batches by applying the FBLPT role such that size of the batches on the machine k don't exceed of capacity of machine $k\,(B_k)$.
9: If the job fits in no existing batches on machine k, place the job in a new batch.
10: **end for**
11: Let L_k be a list of the created batches on machine k. The batches of L_k must be ordered according to the Shortest Ready Time (SRT) role.

Algorithm RKGA.
1: Initialize parameters *Max_gen, Max_no_improve, Population size, Crossover rate* and *Mutate rate*;
2: Create an initial population of chromosomes and evaluate the fitness value of each chromosome;
3: Set $G = 0$;
4: **While** stopping conditions are not satisfied **do**
5: Update $G = G + 1$;
6: Migrate 10% of best chromosomes in the current population to the next population;
7: **for** $k = 1, \ldots, 80\%$ * *Population size* **do**
8: Select two parents from the current population;
9: Apply crossover to the two parents and generate two offspring;
10: Apply mutation to each offspring with probability;
11: Evaluate the fitness of each offspring;
12: Put the better offspring in the next population;
13: **end for**
14: **end while**
15: Report the best fitness found so far;

Acknowledgements. The authors would thank the referees whose with constructive suggestions have greatly improved the presentation of the paper.

References

[1] R. Baker, Principles of Sequencing and Scheduling. Wiley, New Jersey (1943).
[2] J.C. Bean, Genetic algorithms and random keys for sequencing and optimization. *ORSA J. Comput.* **6** (1994) 154–160.
[3] P.Y. Chang, P. Damodaran and S. Melouk, Minimizing makespan on parallel batch processing machines. *Int. J. Prod. Res.* **42** (2004) 4211–4220.
[4] P. Damodaran and P.Y. Chang, Heuristics to minimize makespan of parallel batch processing machines. *Int. J. Adv. Manuf. Technol.* **37** (2008) 1005–1013.
[5] P. Damodaran, P.K. Manjeshwar and K. Srihari, Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms. *Int. J. Prod. Econ.* **103** (2006) 882–891.
[6] P. Damodaran, N.S. Hirani and M.C. Velez-Gallego, Scheduling identical parallel batch processing machines to minimize makespan using genetic algorithms. *Eur. J. Ind. Eng.* **3** (2009) 187–206.
[7] S. Dauzère-Pérès and L. Mönch, Scheduling jobs on a single batch processing machine with incompatible job families and weighted number of tardy jobs objective. *Comput. Oper. Res.* **40** (2013) 1224–1233.
[8] J. Holland, Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975).
[9] C.A. José Elias and Y.-T.L. Joseph, Scheduling unrelated parallel batch processing machines with non-identical job sizes and unequal ready times. *Comput. Oper. Res.* **78** (2017) 117–128.

[10] C.A. José Elias, Y.-T.L. Joseph and T. Ricardo Gonçalves, An iterated greedy algorithm for total flow time minimization in unrelated parallel batch machines with unequal job release times. *Eng. App. Artif. Intell.* **77** (2019) 239–254.

[11] A.H. Kashan, B. Karimi and M. Jenabi, A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. *Comput. Oper. Res.* **35** (2008) 1084–1098.

[12] S.-G. Koh, P.-H. Koo, J.-W. Ha and W.S. Lee, Scheduling parallel batch processing machines with arbitrary job sizes and incompatible job families. *Int. J. Prod. Res.* **42** (2004) 4091–4107.

[13] S. Koh, P.H. Koo, D.C. Kim and W.S. Hur, Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. *Int. J. Prod. Econ.* **98** (2005) 81–96.

[14] M. Lars and S. Roob, A Mata heuristic framework for batch machine scheduling problems with incompatible job families and regular sum objective. *Appl. Soft Comput. J.* **68** (2018) 835–846.

[15] C.-Y. Lee, R. Uzsoy and L.-A. Martin-Vega, Efficient algorithms for scheduling semi-conductor burn-in operations. *Oper. Res.* **40** (1992) 764–775.

[16] S. Malvea and R. Uzsoy, A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families *Computers & Operations Research* **34** (2007) 3016–3028.

[17] M. Mathirajan and A.I. Sivakumar, A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *Int. J. Adv. Manuf. Technol.* **29** (2006) 990–1001.

[18] S. Melouk, P. Damodaran and P.Y. Chang, Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. *Int. J. Prod. Econ.* **87** (2004) 141–147.

[19] L. Monch, J.W. Flower, S. Dauzere-Peres, S.J. Mason and O. Rose, A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *J. Scheduling* **14** (2011) 583–599.

[20] I. Muter, Exact algorithms to minimize makespan on single and parallel batch processing machines. *Eur. J. Oper. Res.* **285** (2020) 470–483.

[21] M.L. Pinedo, Scheduling Theory, Algorithms, and Systems, 3 edition. Springer, New York (2008).

[22] X. Rui, C. Huaping and L. Xueping, Makespan minimization on single batch-processing machine via ant colony optimization. *Comput. Oper. Res.* **39** (2012) 582–593.

[23] Z. Shengchao, C. Huaping and L. Xueping, Distance matrix based heuristics to minimize makespan of parallel-batch processing machines with arbitrary job sizes and release times. *Appl. Soft Comput.* **52** (2017) 630–641.

[24] G. Taguchi, Introduction to Quality Engineering. Asian Productivity Organization/UNIPUB White Plains (1986).

[25] R. Uzsoy, Scheduling a single batch processing machine with non-identical job sizes. *Int. J. Prod. Res.* **32** (1994) 1615–1635.

[26] S. Xu and J.C. Bean, A genetic algorithm for scheduling parallel non-identical batch processing machines. In: IEEE Symposium on Computational Intelligence in Scheduling (2007) 143–150.

[27] R. Xu, H.P. Chen and X.P. Li, A bi-objective scheduling problem on batch machines via a pareto-based ant colony system. *Int. J. Prod. Econ.* **145** (2013) 371–386.

[28] Y. Zarook, J. Rezaeian, R. Tavakkoli-Moghaddam, I. Mahdavi and N. Javadian, Minimization of makespan for the single batch-processing machine scheduling problem with considering aging effect and multi-maintenance activities. *Int. J. Adv. Manuf. Technol.* **76** (2015) 1879–1892.

[29] J. Zhao-Hong, W. Ting-Ting, Y.-T. Joseph and K.L. Leung, Effective heuristics for makespan minimization in parallel batch machines with non-identical capacities and job release times. *J. Ind. Manage. Optim.* **13** (2017) 977-993.

[30] S. Zhou, M. Jin and N. Du, Energy-efficient scheduling of a single batch processing machine with dynamic job arrival times. *Energy* **209** (2020) 118420.