BI-OBJECTIVE UNRELATED PARALLEL MACHINES
SCHEDULING PROBLEM WITH WORKER ALLOCATION AND
SEQUENCE DEPENDENT SETUP TIMES CONSIDERING
MACHINE ELIGIBILITY AND PRECEDENCE CONSTRAINTS

REZA ALIZADEH FOROUTAN, JAVAD REZAEIAN* AND MILAD SHAFIPOUR

Department of Industrial Engineering and Management
Mazandaran University of Science and Technology
Babol, Iran

(Communicated by Ruhul Sarker)

Abstract. In today's competitive world, scheduling problems are one of the
most important and vital issues. In this study, a bi-objective unrelated par-
allel machine scheduling problem with worker allocation, sequence dependent
setup times, precedence constraints, and machine eligibility is presented. The
objective functions are to minimize the costs of tardiness and hiring workers.
In order to formulate the proposed problem, a mixed-integer quadratic pro-
gramming model is presented. A strategy called repair is also proposed to
implement the precedence constraints. Because the problem is NP-hard, two
metaheuristic algorithms, a multi-objective tabu search (MOTS) and a multi-
objective simulated annealing (MOSA), are presented to tackle the problem.
Furthermore, a hybrid metaheuristic algorithm is also developed. Finally, com-
putational experiments are carried out to evaluate different test problems, and
analysis of variance is done to compare the performance of the proposed al-
gorithms. The results show that MOTS is doing better in terms of objective
values and mean ideal distance (MID) metric, while the proposed hybrid al-
gorithm outperforms in most cases, considering other employed comparison
metrics.

1. Introduction and literature review. Unrelated parallel machines are abun-
dant in various manufacturing industries such as textile, electronics, chemical, plas-
tics forming as well as service industries [35]. Today, with the successful application
of the concept of just-in-time production in production management and warehous-
ing, the need to complete the processing of jobs in time has become a key issue in
industrial environments. In other words, delivering jobs in time is as important as
being late in completing them. The costs associated with tardiness of jobs motivate
researchers to use them in the form of various optimization criteria in scheduling
problems. The unrelated parallel machines scheduling problem with human resource
allocation proposed in this study is described as follows: a set of \( n \) different jobs,
\( N = \{1, 2, \ldots, n\} \) is to be processed in one stage on a set of \( m \) parallel machines,
\( M = \{1, 2, \ldots, m\} \). The objectives are to minimize the cost of tardy jobs and the
cost of hiring workers at the same time. The machines are unrelated ($R_m$) and processing times of jobs are not the same on different machines. Each job is processed by only one machine and each machine is not capable of processing more than one job at a time. Some jobs have predecessors. Processing of each job can be done on a subset of machines, which are called processing sets. All jobs are available at the beginning of the scheduling horizon. Before starting the processing of each job on each machine, an operation is performed to prepare the machine for processing that job which is referred to as the setup operation on the machine. The time required for the machine preparation operation is called setup time. This time depends on the type of current and previous job and the type of machine on which the processing is performed. Each job must be completed on its delivery time, otherwise it imposes a tardiness penalty to the objective function. Furthermore, the setup operations are done by a set of $w$ workers, $W = \{1, 2, \ldots, w\}$, each of which has a determined skill level. The preparation operation or setup of each job on each machine must be performed by a worker before processing the job on that machine. Setup time depends on the sequence of jobs and machines. Therefore, a group of $W(W \leq M)$ workers is considered for the setup operations. In addition, there is a pre-determined skill level for each worker (a random number between $[0.5, 1.5]$) according to which a worker can perform the preparation operation faster or slower than their colleagues. Accordingly, skilled workers have a skill level of less than 1, while novice workers have a skill level of more than 1.

Just-in-time (JIT) production was first introduced in Australia in the 1950s by the British Motor Corporation (Australia) [20]. JIT scheduling has emerged as a response to the necessity of fulfilling each customer’s order at their most desired time [27]. The idea then migrated to Japan and developed by Toyota automotive manufacturer in the early 1970s [25], where it was found as one of the most effective strategies of production management to produce the right quantity of parts at the right location and at the right time within the minimum production costs [28]. Shortened lead time, maximum utilization of equipment and workers, and reduced inventories are some of the promising advantages of the successful JIT implementation [9]. One of the best-known objective functions of just-in-time scheduling, which is investigated in the present study is to minimize the total costs of tardiness. It reduces the costs of inventory and warehousing and at the same time, responds to customer demands by delivering the product in time.

Despite many studies on identical parallel machines, the scheduling problems of unrelated parallel machines have received less attention. This is all the more apparent given the setup times restrictions. A large number of scheduling researches have focused on parallel machines problems. Most of these studies have been done in the field of identical ($P_m$) and uniform parallel ($Q_m$) machines problems, while unrelated ($R_m$) parallel machines problems have been studied far less [32, 30]. Vallada and Ruiz proposed a genetic algorithm including a quick local search and an enhanced local search crossover operator to tackle the unrelated parallel machine scheduling problem with machine and job sequence dependent setup times. The objective is to minimize the maximum completion time (or makespan). They evaluated the presented method using a comprehensive benchmark set which showed an excellent performance outperforming the other evaluated methods [30]. Hamzadayi and Yıldız solved identical parallel machines scheduling problem in which common server and sequence dependent setup times were considered. They developed
a mixed integer linear programming model and metaheuristic-based solution approaches to minimize the makespan [18]. Lately, Lei et al. studied unrelated parallel machines scheduling problem in the heterogeneous production network to minimize the makespan and proposed a novel imperialist competitive algorithm with memory to solve it [24].

Whenever a theoretical scheduling model is to be applied in a real production environment, the human factor must be considered to implement practicable scheduling. Potential capacity constraints imposed by machines and workers is referred to as a Dual-Resource Constrained (DRC) [8]. DRC environment has attracted a great deal of attention over the years. In 2011, Xu et al. provided a complete literature review of research on DRC systems over the past two decades [34]. Recently, Dhiuffaoui el al. presented a state-of-the-art review about DRC in classical and flexible job shop problems [8]. In general, workers constraints in DRC systems are discussed under two different subsections, entitled worker flexibility and worker allocation. Worker flexibility is about considering different skills for each worker and worker allocation is about assigning workers to jobs and machines. Gong et al. studied an energy-efficient flexible flow shop scheduling with worker and machine flexibility considering processing time, energy consumption and worker cost related factors. A hybrid evolutionary algorithm was presented to solve the proposed problem [17]. Gong et al. presented a mathematical model of an extended flexible job shop scheduling problem considering worker flexibility and proposed a hybrid artificial bee colony algorithm to solve the problem [16].

With the successful application of the concepts of JIT production in manufacturing systems, research on earliness and tardiness of jobs has been come under the spotlight. Because of the growing importance of JIT policies, many manufacturing companies need to schedule jobs to be completed on time and delivered to customers. Lost sales costs, opportunity costs, and costs of spoilage can be examples of tardiness costs. Chen studied unrelated parallel machines problem with sequence-dependent setup times and unequal ready times to minimize the weighted number of tardy jobs and proposed iterated hybrid metaheuristic algorithms to address the problem [6]. Zeidi and MohammadHosseini dealt with unrelated parallel machines with sequence dependent setup times, taking due date constraints into consideration to minimize the total cost of tardiness and earliness. They designed a metaheuristic algorithm consists of genetic algorithm as the core and simulated annealing as local search procedure to tackle their proposed mathematical model [37]. Kim et al. focused on an identical parallel machine scheduling problem with sequence-dependent setup times and jobs which can be split to minimize total tardiness of jobs. A new mathematical model with metaheuristic approaches was proposed to solve the problem. The results indicate that the suggested algorithm provides better solution quality and less computational time compare to optimization solvers [22].

Scheduling problems with limited access to machines are presented under names such as eligibility constraints, processing set restrictions, as well as machines eligibility. In these problems, each machine is only able to process some of the jobs. Afzalirad and Rezaeian addressed an unrelated parallel machine scheduling problem with different release dates, sequence-dependent setup times, and resource constrains considering machine eligibility and precedence constraints. They proposed a novel integer mathematical modeling and makespan was employed as the objective function. Two new metaheuristic algorithms were developed to find optimal or
near optimal solutions [1]. Wang and Wang studied the order acceptance and unrelated parallel machines scheduling problem with machine eligibility constraints to maximize total net profit and minimize the makespan. A mathematical model was formulated as multi-objective mixed integer linear programming [31]. Afzalirad and Shafipour designed an efficient genetic algorithm to deal with resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions. An integer mathematical programming model was developed to minimize makespan [2].

The setup time of a machine includes all the operations that are performed in order to prepare the machine for processing a job. Generally, scheduling problems with setup times are divided into two general categories. In the first one, setup time depends only on the type of job being processed on the machine, which is called sequence independent setup time; and in the second one, in addition to the type of job that is to be processed on the machine, the setup time also depends on the previous job that has been processed on the machine, which is referred to as sequence dependent setup time, e.g., in chemical industry, paper production, glass industry, plastic industry, etc. Ezugwu and Akutsah studied unrelated parallel machines scheduling problem including sequence dependent setup times and presented a firefly algorithm that was improved with a local search solution mechanism and measured the effectiveness of the proposed algorithm using a comparison with three different popular metaheuristic algorithms [12].

Trying to update a capacity constraint to the mixed integer programming model of the unrelated parallel machines scheduling problem with sequence dependent setup times, Khanh Van and Van Hop developed a new algorithm based on the combination between the genetic algorithm with the ISETP (initial sequence based on earliness-tardiness criterion on parallel machine) heuristics [21]. Ezugwu et al. addressed the makespan minimization of the unrelated parallel machines scheduling problem with sequence dependent setup times and developed an improved Symbiotic Organisms Search (SOS) algorithm to deal with the problem [11].

Precedence constraint or the dependency of starting some jobs on doing other jobs has many uses in real-world problems. Çakar et al. suggested identical parallel robots scheduling problem to minimize mean tardiness with unequal release dates and precedence constraints, and proposed a hybrid solution approach consisting of genetic algorithm and simulated annealing to tackle the problem [5]. Zhu and Zhou concentrated on multi-objective flexible job shop scheduling problem with hierarchical precedence constraints and proposed a novel evolutionary multi-objective grey wolf optimizer to minimize the objectives of makespan, maximum machine workload and total machine workload [39]. Zhang et al. introduced the new concept of soft precedence constraint (SPC) in machine scheduling problems. They developed an integer programming formulation to balance the tradeoff between the SPC violation penalty and other criteria relative to job completion times, and designed approximation algorithms [36].

Multi-objective parallel machine scheduling problems have been less studied than single-objective ones, some of which are reviewed here. Cota et al. dealt with the unrelated parallel machine scheduling problem with setup times to minimize makespan and total energy consumption. Multi-objective extensions of the adaptive large neighborhood search (ALNS) metaheuristic with learning automata (LA) were proposed. Two new algorithms were also developed and compared [7]. Zhang et al. studied unrelated parallel machine scheduling problem considering processing
speed and processing time with the objectives of optimizing total energy consumption and makespan and proposed a novel effective heuristic evolutionary algorithm to solve the presented problem [38]. Munoz-Villamizar et al. assessed the effectiveness of parallel machine scheduling problem with earliness and tardiness costs and variable setup times. A mixed-integer linear programming model was defined and four different objective functions were used to compare different scheduling configurations [26]. A multi-objective parallel machine scheduling problem under fully fuzzy environment was investigated by Arık and Toksarı, in which fuzzy job deterioration effect, fuzzy learning effect and fuzzy processing times were taken into consideration. Minimizing total tardiness penalty cost, earliness penalty cost, and cost of setting due dates were the objectives. Different approaches for modelling fuzzy mathematical programming models were compared [3]. Finally, Wang et al. addressed a bi-objective identical parallel machine scheduling problem to minimize the total energy consumption and the makespan simultaneously. An augmented $\varepsilon$-constraint method was applied to obtain an optimal Pareto front to tackle small scale instances and a constructive heuristic method with a local search strategy was proposed to deal with larger problems. NSGA-II was applied to obtain good approximate Pareto fronts [33]. A summary of several most recent studies in the literature is given in Table 1.

In this research, the unrelated parallel machines scheduling problem with jobs delivery time and worker skills considering sequence dependent setup times, precedence constraints, and machine eligibility is investigated to minimize costs of tardiness and hiring workers. To the best of authors’ knowledge, the scheduling problem presented in this study has not been investigated yet. An innovative hybrid multi-objective metaheuristic algorithm, which is a combination of multi-objective tabu search and multi-objective simulated annealing, is developed to solve the problem. Furthermore, in this study, a new method is used to represent the solutions.

The rest of the paper is organized as follows: in section 2, first the proposed problem is defined and a new mixed-integer quadratic programming (MIQP) model is presented. The proposed model is then validated with two numerical examples. After that in section 3, two different metaheuristic algorithms used in this research including multi-objective tabu search and multi-objective simulated annealing algorithm are described and the proposed hybrid algorithm is introduced. Then in section 4, the processes of data generation and parameters setting are explained. In addition, the computational results of the proposed algorithms are presented. The results are evaluated using five different metrics and convergence of the proposed algorithms are discussed. Analysis of variance is also carried out to investigate the differences in the performance of the algorithms. Finally, conclusions and proposed ideas for future study are given in section 5.

2. Problem definition. In this section, first, the problem assumptions, indices, parameters and decision variables are introduced; then, the proposed mixed-integer quadratic programming model is presented and eventually, the presented model is validated using a randomly generated test problem.

2.1. Assumptions.

- $n$ jobs are to be processed by $m$ machines.
- The skill level of each worker is pre-determined.
- Each worker, depending on their skill level, performs only one preparation operation at a time.
Table 1. A comparison of some most recent studies existing in the literature.

| Reference | Unrelated Parallel Machines | Worker Flexibility Allocation | Machine Eligibility | Sequence Dependent Setup Times | Precedence Constraints | Objective Function(s) | Solution Approach(es) |
|-----------|-----------------------------|-------------------------------|---------------------|-------------------------------|------------------------|-----------------------|----------------------|
| Cota et al. [7] | Yes | No | No | Yes | No | Makespan; Total energy consumption | ALNS, LA, MO-ALNS, MO-ALNS/D |
| Zhu and Zhou [19] | No | No | No | No | Yes | Makespan; Total workload; Maximum machine workload | EMOGWO |
| Munoz-Villamizar et al. [26] | No | No | No | Yes | No | Total tardiness penalty cost; Earliness penalty cost; Cost of setting due dates | GAMS solver |
| Arık and Toksarı [3] | No | No | No | No | No | Makespan; Total worker cost; Green production indicator | A fuzzy local search algorithm |
| Gong et al. [17] | No | Yes | No | No | No | Makespan; Total worker cost; Green production indicator | A hybrid evolutionary algorithm |
| Zhang et al. [36] | No | No | No | No | Yes | Total penalties; Makespan; Total completion time | Approximation algorithms |
| Gong et al. [16] | No | Yes | No | No | No | Makespan | A hybrid artificial bee colony |
| Wang et al. [33] | No | No | No | No | No | Total energy consumption; Makespan | An augmented ε-constraint; A heuristic method; NSGA-II |
| Zhang et al. [38] | Yes | No | No | No | No | Total energy consumption; Makespan | A heuristic evolutionary algorithm |
| Lei et al. [24] | Yes | No | No | No | No | Makespan | An imperialist competitive algorithm |
| Khanh Van and Van Hop [21] | Yes | No | No | Yes | No | Total weighted earliness and tardiness; Makespan | A hybrid algorithm based on GA and ISETP |
| Kim et al. [22] | No | No | No | Yes | Yes | Total tardiness; Makespan | SA, GA |
| This paper | Yes | Yes | Yes | Yes | Yes | Cost of tardiness; Cost of hiring workers | MOTS, MOSA, A hybrid evolutionary algorithm |

- Setup times are dependent on the sequence of jobs and type of machines.
- Machines are parallel and each machine can perform only one operation at a time.
- Each job can be processed by a maximum of one machine at a given time and preemptions are not allowed.
- Performing any operation requires selecting a machine from the machines that are capable of doing the job.
- Setup times and processing times are pre-determined and positive.
Jobs are not independent of each other and there are precedence relationships between them.

2.2. Notations. Indices, parameters, and decision variables of the proposed mathematical model are as follows.

Indices:
- \( h, j, l \): indices of jobs (\( h, j, l = 0, 1, \ldots, n \))
- \( i, i' \): index of machines (\( i, i' = 1, 2, \ldots, m \))
- \( k \): index of workers (\( k = 1, 2, \ldots, w \))

Parameters:
- \( N \): set of jobs
- \( M \): set of machines
- \( W \): set of workers
- \( A \): set of ordered pairs of jobs with a precedence relationship, such that if job \( j \) is a precedence for job \( l \), then \((j, l) \in A\)
- \( T_{il} \): processing time of job \( l \) on machine \( i \)
- \( Y_{ijl} \): if machine \( i \) is capable of processing job \( l \) is 1, otherwise 0
- \( S_{ikjl} \): setup time of job \( l \) on machine \( i \) that is performed by worker \( k \) after the processing of job \( j \) on machine \( i \)
- \( D_j \): delivery time of job \( j \)
- \( \beta_j \): tardiness penalty of job \( j \)
- \( C_{worker} \): cost of hiring each worker
- \( B \): an arbitrary large number

Decision variables:
- \( X_{ikjl} \): if job \( l \) is processed after job \( j \) on machine \( i \), and the setup is done by worker \( k \) is 1, otherwise 0
- \( Q_{ijl} \): if job \( l \) is processed after job \( j \) is 1, otherwise 0
- \( CS_l \): completion time of the setup of job \( l \)
- \( C_l \): completion time of the processing of job \( l \)
- \( T_l \): tardiness of job \( l \)

2.3. The mathematical model. The proposed MIQP model is formulated as follows.

Objective functions:

\[
f_1 : \min \sum_{i=1}^{m} \sum_{k=1}^{w} \sum_{j=0}^{n} \sum_{l=1}^{n} X_{ikjl} \beta_j T_j
\]

\[
f_2 : \min \sum_{i=1}^{m} \sum_{k=1}^{w} \sum_{j=0}^{n} \sum_{l=1}^{n} X_{ikjl} (2 - \pi_k) C_{worker}
\]

Subject to:

\[
S_{ikjl} = X_{ikjl} \pi_k S_{ijl} \quad \forall i = 1, 2, \ldots, m; k = 1, 2, \ldots, w; j, l = 1, 2, \ldots, n
\]

\[
\sum_{i=1}^{m} \sum_{k=1}^{w} \sum_{j=0}^{n} X_{ikjl} Y_{ijl} Y_{il} = 1 \quad \forall l = 1, 2, \ldots, n
\]

\[
\sum_{k=1}^{w} \sum_{j=0}^{n} X_{ikjl} Y_{ijl} Y_{il} + \sum_{k=1}^{w} \sum_{h=1}^{m} X_{i'klh} Y_{i'l} Y_{i'h} \leq 1
\]
that job  

\[
\forall i, i' = 1, 2, \ldots, m; i \neq i'; l = 1, 2, \ldots, n
\]

\[
\sum_{i=1}^{m} \sum_{k=1}^{w} \sum_{l=1}^{n} X_{ikjl} Y_{ij} Y_{il} \leq 1 \quad \forall j = 1, 2, \ldots, n  
\]  

(6)

\[
\sum_{k=1}^{w} \sum_{l=1}^{n} Y_{ij} Y_{il} = 1 \quad \forall i = 1, 2, \ldots, m
\]

(7)

\[
\sum_{k=1}^{w} \sum_{l=1}^{n} X_{ikjl} = 0
\]

(8)

\[
C_l - CS_l \geq \sum_{i=1}^{m} \sum_{k=1}^{w} \sum_{l=0}^{n-1} T_{il} X_{ikjl} Y_{ij} Y_{il} \quad \forall l = 1, 2, \ldots, n
\]

(9)

\[
CS_l - C_j \geq \sum_{i=1}^{m} \sum_{k=1}^{w} X_{ikjl} Y_{ij} Y_{il} - B(1 - \sum_{i=1}^{m} \sum_{k=1}^{w} X_{ikjl} Y_{ij} Y_{il})
\]

\[
\forall j = 0, 1, 2, \ldots, n; l = 1, 2, \ldots, n
\]

(10)

\[
C_l - C_j \geq \sum_{i=1}^{m} \sum_{k=1}^{w} T_{il} X_{ikjl} Y_{ij} Y_{il} \quad \forall (j,l) \in A
\]

(11)

\[
CS_l - CS_j \geq \sum_{i=1}^{m} \sum_{h=0}^{n} S_{ikhl} X_{ikhl} Y_{ih} Y_{il}
\]

(12)

\[
- B(2 - \sum_{i=1}^{m} \sum_{h=0}^{n} X_{ikhl} Y_{ih} Y_{il} + X_{ikhl} Y_{ih} Y_{ij}) + Q_{jl}
\]

\[
\forall i = 1, 2, \ldots, w; j = 1, 2, \ldots, n; l = j + 1, j + 2, \ldots, n
\]

(13)

\[
C_{ijo} = 0
\]

(14)

\[
T_l = C_l - D_l \quad \forall l = 1, 2, \ldots, n
\]

(15)

\[
CS_l, C_i, T_l \geq 0 \quad \forall l = 1, 2, \ldots, n
\]

(16)

\[
X_{ikjl} \in \{0, 1\}
\]

(17)

\[
\forall i = 1, 2, \ldots, m; k = 1, 2, \ldots, w; j = 0, 1, 2, \ldots, n; l = 1, 2, \ldots, n
\]

(18)

The first objective function 1 represents the total cost of tardiness. The second objective function 2 calculates the total cost of hiring workers. Constraint 3 determines the impact of each worker’s skill on the setup time. Constraint 4 shows that job l can only be processed on one machine by only one worker after job j. Constraint 5 ensures the sequence of jobs on each machine and says that if job l is processed after job j on machine i, then job h cannot be processed after job l on any other machine than i. Constraints 4 and 5 together ensure that each job is processed by only one machine (a machine capable of processing that job) and its preparation operation is done only by one worker. Constraint 6 shows that each job must be at most ahead of one other job. Constraint 7 indicates that dummy job (j = 0) must be done before any job on each machine. Constraint 8 implies...
that each job cannot be done before itself. Constraint 9 indicates the minimum float between completion time of the setup and completion time of the processing of a job. Constraint 10 ensures that if job \( l \) is performed immediately after job \( j \) on a machine, the end of processing time of job \( j \) and the end of completion time of the setup of job \( l \) are separated by at least an interval, equal to the setup time of job \( l \). Constraint 11 observes the precedence relationships between jobs. Constraint 12 and 13 together guarantee the constraint of workers, such that, if the setup operation of jobs \( j \) and \( l \) is performed by worker \( k \), then the setup of job \( j \) must be completed before the beginning of the setup of job \( l \). Constraint 14 sets the completion time of dummy job to zero. Constraint 15 computes the tardiness of job \( j \). Finally, constraints 16-18 specify the range of decision variables.

2.4. Model validation. In order to validate the proposed mathematical model, an instance of 8 jobs, 2 machines, and 2 workers, is investigated. A weighting method is applied to turn the objective functions into a single-objective, such that, a weight of 0.6 and a weight of 0.4 are considered for the first and second objective functions respectively. Processing times, delivery times and tardiness penalties of the test problem are listed in Tables 2 and 3. Note that, in the following tables, \( J_j \) represents job \( j \) and \( M_i \) represents machine \( i \).

### Table 2. Processing time \((T_{il})\) of the jobs.

|       | \( J_1 \) | \( J_2 \) | \( J_3 \) | \( J_4 \) | \( J_5 \) | \( J_6 \) | \( J_7 \) | \( J_8 \) |
|-------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| \( M_1 \) | 0         | 4         | 2         | 1         | 5         | 3         | 5         | 4         |
| \( M_2 \) | 0         | 3         | 5         | 8         | 2         | 2         | 6         | 3         |
| \( M_3 \) | 0         | 1         | 5         | 4         | 4         | 2         | 3         | 3         |

### Table 3. Delivery time and tardiness penalty of the jobs.

|       | \( J_1 \) | \( J_2 \) | \( J_3 \) | \( J_4 \) | \( J_5 \) | \( J_6 \) | \( J_7 \) | \( J_8 \) |
|-------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| \( D_j \) | 10        | 13        | 8         | 15        | 6         | 16        | 14        | 18        |
| \( \beta_j \) | 4         | 3         | 7         | 5         | 9         | 8         | 6         | 2         |

In addition, setup time of each job, according to the type of machine and the sequence of jobs, is shown in Table 4, and the availability of \( Y_{ils} \) is given in Table 5. The skill level of the first and second workers are considered 0.67 and 1.2, respectively. Moreover, the cost of hiring each worker is equal to 200 units. The precedence constraints are as follows: \( A = \{(2,4),(6,8)\} \).

The test problem is solved using LINGO 18.0 and results are represented in Table 6 and Figure 1. The results show that all jobs are assigned to machines that are capable of processing them. According to the results presented, all constraints are met; both precedence constraints are considered and there is no overlap between jobs and subsequent setup times on a machine. Therefore, it can be concluded that the proposed model is well-formulated. Furthermore, to validate the quality of solutions, the single objective solutions are compared with extreme solutions on the Pareto frontier calculated by LINGO 18.0. Test problems (listed in Table 7) are small sized ones consisted of different combinations of jobs \((n = 4,\ldots,8)\), machines \((m = 2,3)\), and workers \((w = 2,3)\) with/without precedence constraints which can be solved within 2 hours. The results are depicted in Figure 2.
Table 4. Setup time for each machine.

| (j, l) ∈ A | l=1 | l=2 | l=3 | l=4 | l=5 | l=6 | l=7 | l=8 |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|
| J_{j=1}   | 3   | 3   | 1   | 3   | 3   | 1   | 3   | 1   |
| J_2       | 3   | 2   | 3   | 1   | 3   | 1   | 3   | 2   |
| J_3       | 3   | 1   | 1   | 2   | 3   | 3   | 2   | 1   |
| J_4       | 3   | 2   | 2   | 3   | 3   | 3   | 1   | 3   |
| M_1       | J_5 | 2   | 2   | 1   | 2   | 3   | 1   | 1   |
|           | J_6 | 1   | 3   | 1   | 3   | 1   | 1   | 2   |
|           | J_7 | 2   | 1   | 2   | 3   | 2   | 1   | 1   |
|           | J_8 | 2   | 2   | 1   | 3   | 2   | 1   | 2   |

| (j, l) ∈ A | l=1 | l=2 | l=3 | l=4 | l=5 | l=6 | l=7 | l=8 |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|
| J_1       | 3   | 2   | 2   | 1   | 2   | 2   | 1   | 3   |
| J_2       | 3   | 2   | 1   | 1   | 1   | 2   | 3   | 2   |
| J_3       | 3   | 2   | 2   | 2   | 1   | 3   | 1   | 3   |
| J_4       | 1   | 1   | 2   | 2   | 1   | 2   | 1   | 3   |
| M_2       | J_5 | 2   | 1   | 2   | 1   | 3   | 1   | 3   |
|           | J_6 | 3   | 3   | 2   | 2   | 1   | 3   | 1   |
|           | J_7 | 1   | 1   | 1   | 2   | 3   | 3   | 2   |
|           | J_8 | 1   | 3   | 1   | 2   | 1   | 3   | 2   |

| (j, l) ∈ A | l=1 | l=2 | l=3 | l=4 | l=5 | l=6 | l=7 | l=8 |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|
| J_1       | 1   | 1   | 3   | 3   | 1   | 3   | 2   | 3   |
| J_2       | 3   | 1   | 1   | 3   | 3   | 2   | 1   | 2   |
| J_3       | 1   | 1   | 3   | 1   | 2   | 3   | 1   | 1   |
| J_4       | 3   | 1   | 2   | 2   | 3   | 2   | 2   | 1   |
| M_3       | J_5 | 1   | 3   | 1   | 3   | 1   | 2   | 3   |
|           | J_6 | 1   | 3   | 1   | 3   | 2   | 2   | 1   |
|           | J_7 | 1   | 1   | 3   | 2   | 2   | 1   | 3   |
|           | J_8 | 1   | 1   | 2   | 2   | 3   | 1   | 3   |

Table 5. The availability of $Y_{il}$s.

| J_i | J_2 | J_3 | J_4 | J_5 | J_6 | J_7 | J_8 |
|-----|-----|-----|-----|-----|-----|-----|-----|
| M_1 | *   | *   | *   | *   | *   | *   | *   |
| M_2 | *   | *   | *   | *   | *   | *   | *   |
| M_3 | *   | *   | *   | *   | *   | *   | *   |

3. Metaheuristic approaches. Real-world scheduling problems are generally complex, large-sized and multi-objective with several constraints. Hence, exact algorithms and classical methods of operations research are often not capable of achieving the optimal solution of such problems. To solve large size optimization problems in a reasonable computational time, like the NP-hard scheduling problem presented in this study, heuristics and metaheuristic approaches, known as approximate algorithms, are usually applied. Unlike heuristic methods, which are generally suitable for finding local optimal, in metaheuristics, mechanisms are designed to direct the search process so that global-optimum or near-optimal solutions can be achieved. Hence, in this paper a multi-objective tabu search (MOTS),
Table 6. Results of the first test problem.

| Job | $S_{ikj}$ | $C_{ikj}$ | $C_l$ | $T_{ikj}$ | $T_l$ | $X_{ikj} = 1$ |
|-----|-----------|-----------|-------|-----------|-------|----------------|
| 1   | 0         | 0         | 0     | 0         | 0     | $X_{1213}$    |
| 2   | 1.2       | 10        | 14    | 4         | 1     | $X_{1228}$    |
| 3   | 1.2       | 6         | 8     | 2         | 0     | $X_{1232}$    |
| 4   | 2.4       | 17        | 25    | 8         | 10    | $X_{2217}$    |
| 5   | 1.2       | 2         | 6     | 4         | 0     | $X_{2274}$    |
| 6   | 3.6       | 14        | 16    | 2         | 0     | $X_{3215}$    |
| 7   | 1.2       | 4         | 14    | 6         | 0     | $X_{3256}$    |
| 8   | 2.4       | 20        | 24    | 4         | 6     |                |

$F = 0.6 \times f_1 + 0.4 \times f_2 = 39 + 448 = 487$

Figure 1. Results of the test problem with $n = 8$, $m = 2$, and $w = 2$.

Table 7. Test problems used to validate the quality of solutions.

| Test Problem | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  |
|--------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| $(n, m, w)$  | $(4, 2, 2)$ | $(5, 3, 2)$ | $(5, 2, 2)$ | $(6, 3, 2)$ | $(6, 2, 2)$ | $(7, 2, 2)$ | $(7, 2, 3)$ | $(7, 2, 3)$ | $(8, 2, 2)$ |

with constraints ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
without constraints ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

a multi-objective simulated annealing (MOSA) and a proposed hybrid evolutionary algorithm are employed.

3.1. **Tabu search algorithm.** First introduced by Glover in 1986, tabu search is a metaheuristic which offers a heuristic local search process to find the solution space away from the local optimal [14]. One of the fundamental concepts of this algorithm is the use of external memory which provides more flexibility to search.
Using an operator called Move, this algorithm moves frequently from one solution to another to search the neighborhood until the algorithm stop criterion is reached.

3.1.1. Solution space. Solution space is a discrete or continuous set of all possible solutions to the problem. The neighborhood structure includes all possible solutions that are obtained with a small local change in the current solution. In order to optimize the current solution, the appropriate definition of the solution space and the neighborhood structure is one of the most important steps in defining a Tabu search algorithm.

3.1.2. Tabu search. Prohibited search or tabu is one of the features that distinguishes the Tabu search algorithm from other local search algorithms. Tabus are primarily intended to prevent loops in solving the algorithm. They avoid engaging in iterations that do not improve the current objective value and return to local optimal points after getting away from them. Tabus or unauthorized movements are a list of movements that have been performed in recent iterations of the algorithm. They are stored in short-term memory called Tabu list. This list is usually fixed and limited in size, and while the list is filled, the initial moves entered into the list are removed.

3.2. Multi-objective tabu search algorithm. Due to the fine performance of the Tabu search algorithm in finding optimal and near-optimal solutions, researchers have made great efforts to apply this algorithm in multi-objective problems. Multi-objective tabu search (MOTS) method was first presented by Hansen. The algorithm is used heuristically to generate non-dominated alternatives for multi-objective combinatorial optimization problems [19]. In 2006, Baykasoglu introduced multi-objective tabu search algorithm for continuous optimization problems with a simple neighborhood strategy [4]. The difference between this method and the basic algorithm is in selection and updating steps. Two more lists are added to the tabu
list. The first one is the Pareto list in which non-dominated solutions (the seed solutions) found by the algorithm are collected, and the second one is the candidate list in which the solutions from the neighborhood, that are not dominated by any of the solutions of neighborhood, and the Pareto list are included in the candidate list. Eventually, one of the solutions from the candidate list is considered as the new current solution. The general flowchart of the MOTS algorithm is depicted in Figure 3. The major steps of the algorithm are as follows.

3.2.1. **The seed solution.** In order to select the best neighborhood solution as the new current seed solution (seed solution) based on Pareto optimality logic, first, the values of the objective functions for each neighborhood solution are calculated. The current candidate solutions are then identified from the neighborhood solutions. Finally, one of the solutions in the candidate list is randomly selected as the seed solution.

3.2.2. **Updating lists.** Initially, the first feasible solution is stored as the first Pareto solution. The solutions that are dominated by any neighborhood solution, are removed from both Pareto and candidate lists in each iteration. Then seed solution is added to the Pareto list and the rest of the candidate solutions are added to the candidate list.

3.2.3. **Aspiration criteria.** Aspiration criteria are applied so that not all the tabus are put in the tabu list to let the algorithm reach better solutions and to prevent it from getting stuck in loops. Several methods have been proposed to establish the aspiration criterion. As an example, the most common and simplest aspiration criterion is to put a number of available solutions on the tabu list and to avoid them up to a certain number of algorithm executions.

3.3. **Simulated annealing algorithm.** Inspired by the heat treatment of annealing, the simulated annealing algorithm was first introduced by Kirkpatrick et al. in 1983 [23]. It is a metaheuristic approach for solving unconstrained and bound-constrained optimization problems. The technique models the process of melting a solid material and then slowly reduces the temperature to decrease defects and minimizes the system energy. The principal idea of this method is a random search procedure using Markov chain. A new solution called the neighbor solution is randomly generated in each iteration of the algorithm.

3.3.1. **Selecting algorithm parameters.** The SA algorithm starts the process of solving with an initial solution, the initial temperature and the maximum number of iterations. Temperature controls the probability of choosing a worse answer, and the number of iterations determines the number of repetitions of the algorithm to reach a global optimum or a near-optimal solution. Initial temperature. Choosing the proper initial temperature is one of the most important steps of the SA algorithm. If the temperature is too high, almost all new solutions are accepted. Therefore, the minimum is not easily obtained. If the temperature is too low, the probability of accepting a new solution tends to zero, since inappropriate solutions are rarely be accepted. Hence, the variety of solutions will be limited and the system will get stuck in a local minimum. As a result, there must be a proper initial temperature, so at first, the system can search almost the entire solution space; then proportional to the increase of number of iterations, the system temperature is reduced to achieve a proper convergence.
Temperature reduction. Another important parameter of the annealing process is how the temperature changes during the execution of the algorithm. One method is to use one of the temperature-reduction functions. In this study, the geometric annealing function (cooling function) is used to reduce the temperature of the algorithm:

$$T_k = \alpha T_{k-1}$$  \hspace{1cm} (19)

In which $T_k$ is the temperature of current stage, $T_{k-1}$ is the temperature of next stage and $\alpha$ ($\alpha < 1$) is the temperature reduction rate [15].

Equilibrium state. It is necessary to check the equilibrium state after a series of iterations at a certain temperature, whether the annealing process continues or stops at the same temperature. There are several methods for determining the number of moves required to check the equilibrium state ($N$) at one temperature. The one applied in this research is using a constant value ($N=\text{constant}$). If the equilibrium condition is met, the temperature is reduced and the annealing process is repeated at the next temperature; otherwise, new moves are made at the same temperature and the equilibrium state is checked again.
Stopping conditions. Whenever one of the following criteria is met, the algorithm terminates and reports the last solution which improved the objective function as the final output: (1) Reaching a final temperature, (2) Achieving a predetermined number for successive temperature values with no improvement in solution quality, (3) Reaching a fixed amount of CPU time, and (4) when the objective reaches a pre-specified threshold value.

3.4. Multi-objective simulated annealing algorithm. Since the proposed problem is a bi-objective, a multi-objective version of the algorithm is needed to solve the problem. Due to the efficiency of the SA algorithm in achieving global optimum, researchers have developed this algorithm in several ways to tackle multi-objective problems. Among the various versions introduced for multi-objective simulated annealing (MOSA) algorithm based on Pareto concept, Engrand’s proposed approach with an archived population is used in this research [10]. In this method, every time a solution is accepted, it is compared to other archived solutions using the non-dominance principle. At the end of the optimization search, there is an archived population which actually represents the trade-off surface between all the objective functions of interest, among which the best solution can be chosen according to priorities. Engrand presents his proposed algorithm by making changes in the step of selecting a new solution.

MOSA is introduced by defining a new function called $G$ (or the energy function) to be used for calculating the probability of acceptance of a new solution, as follows:

$$G(X) = \sum_{j=1}^{P} \ln f_j(X)$$  \hspace{1cm} (20)

Where $f_1, f_2, \ldots, f_P$ are the objective functions that must be minimized, such that in each iteration, all these functions are calculated and stored in $X$. The probability of acceptance of the new solution $X_{n+1}$ is equal to:

$$p = \exp\left(-\frac{\Delta G}{T_k}\right)$$  \hspace{1cm} (21)

Any solution that is accepted is nominated to enter the archive of non-dominated solutions. The archiving strategy can be seen in [10]. To create diversity and avoid convergence to the local optimal pareto, MOSA performs the search using the strategy of return to base at each step based on the best solutions found in the archive.

3.4.1. Implementation of MOSA. Consider $X$ as an element of the search space, $F=(f_1, \ldots, f_i, \ldots, f_P)$ as the vector of objective functions and $n$ as the number of accepted individuals. Using the same notation, Algorithm 1 shows the steps of MOSA [10].

3.5. Proposed hybrid algorithm. Hybrid metaheuristics are combinations of metaheuristic algorithms with other optimization techniques to increase the efficiency and flexibility of the algorithms tackling complex and large size real-world problems. Hybrid metaheuristic algorithms are divided into two categories: (1) the collaborative ones, which are based on the sequential or parallel implementation of several optimization techniques and the exchange of information between them, like parallel implementation of several local search algorithms and even with a population-based evolutionary algorithm (e.g. [8]); and (2) the integrative ones in which a part of one algorithm is embedded in another algorithm. Sometimes,
Algorithm 1

Initiate $X_0$, $F_0$
Randomly perturb $X_n$ to generate $X_{n+1}$
evaluate $F_{n+1}$
evaluate $G_{n+1}(X) = \sum_{j=1}^{p} \ln f_j(X_{n+1})$ and $p = \exp\left(-\frac{\Delta G}{T_k}\right)$, accept $X_{n+1}$ with the probability $p$
if $X_{n+1}$ accepted then
   Update the archived population:
      Do not archive $X_{n+1}$ if dominated by one archived individual,
      Archive $X_{n+1}$ if not dominated by any archived individual,
      Remove archived individual if dominated by $X_{n+1}$
   $X_{n+1} = X_{n+1}$
end if
if necessary, then
   Return to base from any archived individual, i.e. $X_n = Y_m^*$
   Go to line 2
end if
if necessary, then
   $T_{k+1} = \alpha T_k$
end if
if convergence not reached, then
   Go to line 2
end if

$Y_m^*$ is a randomly chosen archived individual.

classical methods of combining algorithms are replaced with precise techniques such as branch and bound or integer linear programming to improve new solutions. To combine algorithms, the main characteristics of them must be known well, e.g., the main features of the SA algorithm are solution acceptance criteria and temperature changes. Also, for the TS algorithm, the tabu list and the aspiration criteria can be mentioned. To improve the final solution, many researchers have also used methods of combining algorithms in multi-objective problems. In this study, a combination of MOTS and MOSA is developed to deal with the proposed bi-objective unrelated parallel machines scheduling problem. The steps for implementing the proposed hybrid algorithm are as follows (Algorithm 2).

3.5.1. Components of the proposed hybrid algorithm. In this section, the components of the proposed hybrid metaheuristic algorithm are introduced. Solutions representation. The first step in relation to metaheuristics is to choose a method for representation of the solutions or so-called chromosomes. Choosing an appropriate solution representation method is one of the most important parts of algorithm design. In a scheduling problem with workers allocation, a chromosome must simultaneously represent the assignment of jobs to machines and the sequence of jobs assigned to each machine, as well as the assignment of workers to machines. In this study, since the machine eligibility constraint is considered, the chromosome representation used is a two-dimensional array in which the number of rows is equal to 3 and the number of columns is equal to the number of jobs. In the first row, the numbers of jobs are shown and in the second row the numbers of the machines assigned to them are placed. In the third line, workers who perform setup operation
Algorithm 2

1. Set the initial temperature $T_0$ and create an initial solution $x_0$ as the current solution and evaluate it ($F_0$).

2. Create a new solution in the neighborhood of the current solution using hybrid neighborhood solution generation operators.

3. The new solution should not be dominated by the current solution and be in the tabu list. In this case, it is evaluated ($x_n \rightarrow x_{n+1}$).

4. The new neighborhood solution ($x_{n+1}$) is accepted with probability $p = \exp(-\frac{\Delta G}{T_k})$; If the new solution ($x_{n+1}$) is accepted, the archive population is updated using the archive storage strategy.

5. Execute the strategy of return to base and randomly select a member of the archive population as the current solution.

6. Reduce the temperature using the temperature function $T_{k+1} = \alpha T_k, \alpha < 1$.

7. If the termination condition is met, the solutions in the archive population are reported as the final.

Generating initial solution. According to Figure 4, in order to generate an initial solution, first of all, jobs are permutationally placed next to each other in the first row; then, to satisfy the precedence constraint, the generated row is modified using a correcting algorithm. Afterwards, to create the second row, due to the limited access to the machines, according to the sequence of jobs in the first row, for each job a machine is randomly selected from the set of machines that can process that job. Eventually, in the third row, a worker is randomly assigned to each job. Algorithm 3 shows the proposed correcting method.

To clarify how the correcting algorithm operates, suppose a problem with 8 jobs and the precedence constraints as shown in Figure 5. Assume that the random permutation obtained for the first row is $p = \{5, 3, 1, 4, 2, 6, 8, 7\}$. Figure 6 shows the steps of implementing the algorithm. The modified row obtained by the algorithm is $p_{new} = \{1, 4, 5, 3, 2, 6, 8, 7\}$. 

![Figure 4](image-url)
Algorithm 3

while number of elements of new permutation = number of elements of current permutation do
    for each element of current permutation do
        if its predecessor(s) is conducted then
            get it on new permutation and go to next element
        else go to next element
        end if
    end for
end while

Figure 5. An example of the precedence constraints.

Fitness function. The role of the fitness function is to indicate the degree of fitness or the value of objective function of each chromosome. This function forms the basis of the selection phase. This function is the basis of the selection phase. In this study, the fitness function is formulated according to the sequence of jobs on machines, the allocation of workers to the jobs to perform the setup operations on machines, the completion time of the processing of jobs and the cost of hiring workers.

Neighbor solutions and operators. Depending on the type of problem, there are several operators to generate a random neighbor solution. Since the proposed problem is discrete, the related operators are employed to generate neighbor solutions. Also, after generating each neighbor solution, due to the precedence constraints between jobs, each solution must be converted into a feasible solution using the proposed correcting algorithm. The operators used are described in the following.

Swap operator. Two columns are randomly selected from a two-dimensional chromosome and replaced with each other (Figure 7).

Reversion operator. Two columns of genes are selected and the columns between them are reversed (Figure 8).

Relocation operator. A column from the chromosome is randomly selected and the worker assigned to the corresponding gene in the third row is randomly changed (Figure 9).
4. **Experiments and results.** In this section, some numerical experiments are carried out to evaluate the performance of the proposed algorithms and compare them to each other according to objective values and five other different metrics. To do so, a set of test problems is considered and the input data are randomly generated. Then, the parameters of all algorithms are set using Taguchi method.
Afterwards, the experimental results from the execution of test problems are presented. Finally, the algorithms are evaluated and one-way analysis of variance (ANOVA) tests for problems of all sizes are performed.

4.1. **Data generation.** The input data used for test problems are generated according to functions listed in Table 8. The values of jobs processing time, jobs delivery time, setup times and jobs tardiness penalties are randomly generated using uniform distributions and for workers skill, a random function is applied. In addition, for precedence constraints, a number of jobs are randomly assigned to some other jobs as predecessors. Also, to generate machine accessibility restrictions, all jobs are randomly assigned to a number of machines.

| Parameter             | Generating function |
|-----------------------|---------------------|
| Jobs processing time  | $T_{it} \sim U[5 \ 15]$ |
| Jobs delivery time    | $D_j \sim [20 \ 40]$ |
| Setup times           | $S_{itl} \sim [1 \ 7]$ |
| Jobs tardiness penalties | $\beta_j \sim [1 \ 10]$ |
| Worker’s skill        | $\pi_k \sim rand(0.5, 1.5)$ |
4.2. **Parameters setting.** Parameters setting plays an important role in the performance of metaheuristic algorithms. Hence, finding the best values for the controller parameters of the proposed algorithms is an essential step of their implementation. For this purpose, a series of calibration experiments are usually carried out to find the optimal combination of different values of the algorithm controller parameters. Considering the fact that the time and cost of the experiment grow exponentially as the number of experiment levels and the number of parameters increase, a design of experiments (DOE) based on Taguchi’s method is employed [29]. Controller parameters of the two metaheuristics and the hybrid algorithm presented in this study are as follows: maximum number of iterations ($max_{it}$), tabu list ($Tabu_{list}$) and number of generated neighbor solutions ($N_{new}$) for the MOTS; maximum number of iterations ($max_{it}$), number of iterations per temperature ($Sub_{it}$), initial temperature ($T_0$) and cooling rate ($\alpha$) for the MOSA; and tabu list ($Tabu_{list}$), maximum number of iterations ($max_{it}$), number of iterations per temperature ($Sub_{it}$), initial temperature ($T_0$) and cooling rate ($\alpha$) for the proposed hybrid algorithm. Each of the controller parameters has a significant impact on the quality of solutions and computational times. According to several experiments and their results, the effective range of each of the controller parameters of the algorithms are obtained.

A number of experiments are performed to achieve the effective range of parameters and the results are as the following: For MOTS, the effective range of $Tabu_{list}$, $N_{new}$ and $max_{it}$ are the intervals $[2, 7]$, $[10, 40]$ and $[30, 70]$ respectively. For MOSA, the best solution quality is obtained when $max_{it}$, $Sub_{it}$, $T_0$ and $\alpha$ are between $[80, 200]$, $[20, 30]$, $[60, 150]$ and $[0.93, 0.99]$ respectively; and the hybrid algorithm performs its best when $Tabu_{list}$, $max_{it}$, $Sub_{it}$, $T_0$ and $\alpha$ are taking values from the intervals $[2, 5]$, $[70, 250]$, $[20, 30]$, $[60, 150]$ and $[0.93, 0.99]$ respectively.

After determining the effective range of controller parameters, to investigate the impact of the interaction of these parameters on the performance of the proposed algorithms and to achieve the optimal combination of them, a DOE based on Taguchi method is set. Each of the parameters introduced before is tested at three levels (Table 9). The important point here is that the problem under study is a bi-objective, so each algorithm results in several solutions that do not dominate each other after one run and these solutions are evaluated by several metrics. On the other hand, the Taguchi experiment deals with only one output metric, so to put the Taguchi method into practice, different solutions have to be converted into one. To do so, a method for measuring multi-objective algorithms is needed. In this research, mean ideal distance (MID) and computational time are used as output metric. There are 9, 9 and 27 different combinations of parameters examined for MOTS, MOSA and the hybrid algorithm respectively. The generated combinations are analyzed using MINITAB 16.

First, correlation coefficients of parameters for $SN$ ratios are calculated to determine the order of importance of the parameters. Results show that $max_{it} = 30$ for MOTS, $max_{it} = 200$ for MOSA and $Sub_{it} = 25$ for the hybrid algorithm is more important than other parameters. Then, analysis of variance for $SN$ ratios is used to show the relative impact of each factor on the response level value. Eventually, in order to prioritize the importance of each factor, the response levels are ranked according to the indices of mean of responses and $SN$ ratios of factors. Furthermore, the interactions of factors are analyzed to find the best values of each. Taguchi’s
Table 9. Levels of the controller parameters.

| Algorithm | Parameter      | Levels         |
|-----------|----------------|----------------|
| MOTS      | Tabu$_{list}$  | 2, 4, 6        |
|           | $N_{new}$      | 10, 20, 40     |
|           | max$_{it}$     | 30, 50, 70     |
| MOSA      | max$_{it}$     | 100, 200, 300  |
|           | Sub$_{it}$     | 20, 25, 30     |
|           | $T_0$          | 60, 100, 150   |
|           | $\alpha$      | 0.93, 0.95, 0.99 |
| Hybrid    | Tabu$_{list}$  | 2, 4, 5        |
|           | max$_{it}$     | 70, 150, 200   |
|           | Sub$_{it}$     | 20, 25, 30     |
|           | $T_0$          | 80, 100, 150   |
|           | $\alpha$      | 0.93, 0.95, 0.99 |

DOE is executed separately for each algorithm and the results are reported in Table 10.

Table 10. Ranking of factors (based on S/N ratios and mean of responses) and their best values for each algorithm.

| Algorithm | Parameter      | Rank (S/N ratios) | Rank (Means) | Best value |
|-----------|----------------|-------------------|--------------|------------|
| MOTS      | Tabu$_{list}$  | 3                 | 3            | 4          |
|           | $N_{new}$      | 2                 | 2            | 40         |
|           | max$_{it}$     | 1                 | 1            | 70         |
| MOSA      | max$_{it}$     | 1                 | 1            | 200        |
|           | Sub$_{it}$     | 2                 | 2            | 25         |
|           | $T_0$          | 4                 | 4            | 100        |
|           | $\alpha$      | 3                 | 3            | 0.93       |
| Hybrid    | Tabu$_{list}$  | 1                 | 2            | 5          |
|           | max$_{it}$     | 3                 | 1            | 70         |
|           | Sub$_{it}$     | 2                 | 3            | 25         |
|           | $T_0$          | 4                 | 4            | 150        |
|           | $\alpha$      | 5                 | 5            | 0.93       |

4.3. Experimental results. In order to compare the performance of the proposed algorithms, there are 20 test problems considered in three scales of small, medium, and large. The results of solving test problems using the proposed algorithms are shown in Tables 11 and 12. The presented algorithms are coded using MATLAB R2021a and run on a computer Intel(R) Core(TM) 2 Duo, 4.4 GHz CPU with 1 GB RAM.

For ease of understanding, sample Pareto frontiers of all three proposed metaheuristics are compared in Figures 10 and 11 for a medium and a large size problem.

In addition, to compare the objective values, the hypervolume indicator (or S-metric) is also employed. The unary hypervolume indicator is a measure of the quality of a set $P = \{p^{(1)}, p^{(2)}, \ldots, p^{(n)}\}$ of $n$ nondominated objective vectors produced in a run of a multi-objective optimizer, such as a multi-objective evolutionary algorithm. Assuming a minimization problem involving $d$ objectives, this indicator
Table 11. Computational results for the problems of all sizes: MOTS & MOSA.

| Problem (n, m, w) | Obj. functions | MOTS | MOSA |
|------------------|----------------|------|------|
|                  | P1* P2 P3 P4   | P1  | P2  | P3  |
| (8, 2, 2)        | f1 f2          | 710 | 826 | 1661| 1968|
|                  |                | 4853| 4769| 4769| 4769|
| (8, 3, 2)        | f1 f2          | 856 | 3656| 1550| 1418|
|                  |                | 1063| 4418| 4418| 4418|
| (10, 3, 2)       | f1 f2          | 1063| 3662| 1434| 4880|
|                  |                | 771 | 4418| 4418| 4418|
| (10, 4, 2)       | f1 f2          | 996 | 898 | 1550| 1968|
|                  |                | 5719| 5314| 2055| 2229|
| (12, 3, 2)       | f1 f2          | 1460| 6352| 5557| 5557|
| (12, 3, 3)       | f1 f2          | 1207| 4418| 4418| 4418|
| (12, 4, 2)       | f1 f2          | 480 | 6445| 5399| 5399|
| (15, 4, 3)       | f1 f2          | 10760| 4880| 5557| 5557|
| (15, 6, 3)       | f1 f2          | 2456| 9484| 9475| 9475|
| (20, 4, 3)       | f1 f2          | 6317| 6943| 9658| 9658|
| (20, 6, 5)       | f1 f2          | 2894| 8410| 9826| 9826|
| (25, 4, 3)       | f1 f2          | 12610| 2055| 2055| 2055|
| (25, 6, 5)       | f1 f2          | 3715| 8823| 12725|12725|
| (30, 8, 4)       | f1 f2          | 4133| 11685|19537|19537|
| (30, 10, 6)      | f1 f2          | 14996| 15081|22217|22217|
| (40, 10, 6)      | f1 f2          | 8993| 15081|22217|22217|
| (40, 12, 8)      | f1 f2          | 6334| 14996|15081|15081|
| (50, 12, 8)      | f1 f2          | 13801| 14996|15081|15081|
| (50, 15, 10)     | f1 f2          | 11148| 19918|19905|19905|
|                  |                | 14555| 19450|19905|19905|
|                  |                | 14555| 19450|19905|19905|

Pz (z = 1, 2, ..., Z) denotes the zth pareto solution.

Consists of the measure of the region which is simultaneously dominated by \( P \) and bounded above by a reference point \( r \in \mathbb{R}^d \) such that \( r \geq (\max_{p \in P} p_1, ..., \max_{p \in P} p_d) \), where \( p = (p_1, ..., p_d) \in P \subset \mathbb{R}^d \), and the relation \( \geq \) applies component-wise. This region consists of an orthogonal polytope, and may be seen as the union of \( n \) axis-aligned hyper-rectangles with one common vertex (the reference point, \( r \)) [13]. The larger the hypervolume indicator, the better the algorithm performance. The results are indicated in Table 13 and Figure 12, and illustrated in Figures 13 and 14 for two sample test problems.

From Table 13 and Figure 12, it can be seen that MOTS generally outperforms in terms of achieving better objective values, comparing to the proposed hybrid algorithm and MOSA.

Because of the conflict between the objective functions of a multi-objective problem, the multi-criteria optimization approach is employed to tackle the problem. The main goal in such problems is to find a set of points (solutions) that are dominant over other points. Whenever non-dominated points are global optimum, they are called Pareto points. Unlike single-objective problems, multi-objectives are not limited to a single optimal solution, but contain a set of optimal solutions. None of the solutions can be considered superior to the others unless the decision-maker’s preferences are defined. The set of all Pareto optimal solutions in a multi-objective...
Table 12. Computational results for the problems of all sizes: The proposed hybrid.

| Problem (n, m, w) | Obj. functions | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | Hybrid |
|------------------|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--------|
| (8, 2, 2)        | f1             | 1216 | 1400 | 1624 | 1961 |
|                  | f2             | 4266 | 4266 | 4113 | 3961 |
| (8, 3, 2)        | f1             | 989  | 1465 | 1667 | 1791 | 1791 | 1867 | 1870 | 1896 |
|                  | f2             | 6997 | 5793 | 5793 | 5488 | 5184 | 5184 | 4880 | 4855 |
| (10, 3, 3)       | f1             | 2433 | 2586 | 2671 | 2696 | 2702 | 2702 | 2702 | 2704 |
|                  | f2             | 3295 | 3295 | 3208 | 3152 | 3152 | 3065 | 3065 | 3065 |
| (10, 4, 2)       | f1             | 6534 | 6533 | 6328 | 6125 | 5922 | 5719 |
|                  | f2             | 2109 | 2090 | 2491 | 2346 |
| (12, 3, 2)       | f1             | 1429 | 2342 | 2497 | 2535 | 2583 | 2583 | 2583 | 2583 |
|                  | f2             | 8268 | 7790 | 7313 | 7046 | 6564 | 6358 | 6091 | 5889 | 5613 |
| (15, 4, 3)       | f1             | 6835 | 6835 | 6589 | 6310 | 6147 | 6147 | 6147 | 6147 |
|                  | f2             | 2100 | 2100 | 2078 | 2078 | 2078 | 2078 | 2078 | 2078 |
| (20, 4, 3)       | f1             | 6897 | 6999 | 7519 |
|                  | f2             | 1120 | 1078 | 9755 |
| (20, 6, 5)       | f1             | 9712 | 7385 | 7496 | 7445 | 9537 | 9537 | 9537 | 9537 |
|                  | f2             | 9718 | 9718 | 9587 | 9518 | 9518 | 9386 | 9309 | 9253 | 9044 | 8782 | 8779 |
| (25, 4, 3)       | f1             | 11769| 11826| 12126| 13470| 17224| 17224| 17224| 17224|
|                  | f2             | 10769| 10769| 10598| 10598| 10598| 10598| 10598| 10598|
| (25, 6, 5)       | f1             | 9584 | 6531 | 6684 | 6501 | 8534 | 8415 | 9107 | 9122 |
|                  | f2             | 10561| 10561| 10389| 10266| 10240| 10117| 10114| 9999 |
| (30, 4, 4)       | f1             | 13191| 13505| 13512| 14068| 15569| 15569| 15569| 15569|
|                  | f2             | 16578| 16411| 16391| 16211| 16191| 16071| 15895| 15408| 15408|
| (30, 6, 6)       | f1             | 10892| 10951| 10968| 13671| 17288| 17288| 17288| 17288|
|                  | f2             | 15176| 15039| 14762| 14762| 14762| 14762| 14762| 14762|
| (40, 10, 6)      | f1             | 21192| 22416| 23254| 24128| 24128| 24128| 24128| 24128|
|                  | f2             | 25930| 25229| 24604| 24604| 24604| 24604| 24604| 24604|
| (40, 12, 8)      | f1             | 47230| 47348| 47348| 52019| 52076| 52076| 52076| 52076|
|                  | f2             | 23776| 23729| 23729| 23729| 23729| 23729| 23729| 23729|
| (50, 15, 10)     | f1             | 18791| 23480| 26830| 28710| 28254| 29460| 29460| 29460|
|                  | f2             | 28935| 28895| 28637| 28376| 28302| 28202| 28202| 28202|

Figure 10. Pareto fronts of all three proposed metaheuristics, a medium test problem (n = 20, m = 6, w = 5).
problem, is called the Pareto optimal set and the corresponding objective vectors are called the optimal Pareto edge. Therefore, it can be concluded that the main goal of the multi-criteria optimization approach is to achieve more global optimums (Pareto points). Since there are solutions that do not have priority over each other, in order to evaluate the performance, quality, and variance of the proposed multi-objective metaheuristic algorithms and compare them with each other, the comparison metrics used are different from the comparative metrics for single-objective metaheuristics. Having different comparison metrics is essential to identify the best set of non-dominated solutions when visual interpretation of results is very difficult. Among the various comparison metrics, the following five metrics are considered in this study:

- Number of Pareto solutions (NPS)
  This metric calculates the number of non-dominated optimal solutions that
Figure 12. Comparison of the hypervolume indicator for the problems of all sizes.

Figure 13. Hypervolumes comparison, a medium test problem ($n = 20, m = 6, w = 5$).

Figure 14. Hypervolumes comparison, a large test problem ($n = 30, m = 10, w = 6$).

an algorithm obtains per execution. According to this metric, the more the
number of non-dominated solutions, the better the performance of the algorithm.

- Mean ideal distance (MID)
  Mean ideal distance measures the proximity of the Pareto set obtained by the algorithm to the optimal Pareto edge. Since finding the optimal Pareto edge is not possible for many problems, MID calculates the average of distances of Pareto points from an ideal point, which is considered (0, 0) for a minimization problem with two objectives. It is calculated through Eq. (22):

\[ MID = \frac{\sum_{i=1}^{n} C_i}{n} \]  

(22)

where, \( n \) is the number of Pareto solutions and \( C_i \) is the distance of \( i \)th Pareto solution from ideal point \( C_i \) is calculated via Eq. (23):

\[ C_i = \sqrt{\sum_{k=1}^{m} \left( \frac{f_{k, total}^{max} - f_{k, total}^{min}}{f_{k, total}^{max} - f_{k, total}^{min}} \right)^2} \]  

(23)

\( f_{k, total}^{max}, f_{k, total}^{min} \) denote the maximum and minimum values of \( k \)th objective functions, respectively. For maximizing/minimizing objectives, \( f_{k}^{best} \) is equivalent to \( f_{k, total}^{max}, f_{k, total}^{min} \), respectively. Also, \( f_{ki} \) indicates the value of \( k \)th objective function for \( i \)th Pareto optimal solution. The lower the MID of an algorithm, the shorter the distance from the Pareto edge.

- Diversification metric (DM)
  This criterion shows the diversity of solutions obtained by metaheuristic algorithms. Eq. (24) is used to calculate this comparison metric:

\[ DM = \sqrt{\sum_{k=1}^{m} (f_{k, total}^{max} - f_{k, total}^{min})^2} \]  

(24)

Higher DM values are preferable.

- Spread of non-dominance solution (SNS)
  This metric measures the variance of the set of non-dominant solutions obtained by the algorithm, which means the diversity of Pareto solutions. The SNS metric is calculated by the following equation:

\[ SNS = \sqrt{\frac{\sum_{i=1}^{n} (MID - c_i)^2}{n - 1}} \]  

(25)

The higher value of SNS indicates the more diversity and better performance of the algorithm in terms of this comparison metric.

- Computational time (Time)
  Computational time or CPU time measures the running time of an algorithm. The less the execution time of the algorithm, the better. The results of comparing the performance of the proposed algorithms, using the introduced metrics for the problems of all sizes are given in Table 14.

4.4 Algorithms evaluation. According to the results listed in Tables 13 and 14, the proposed MOTS generally obtains better objective values (larger S-metrics) for problems of all sizes. For small size problems, it can be seen that the proposed hybrid algorithm, in addition to generating more Pareto (non-dominated) points, spends much less computational time to achieve a near-optimal solution.
Table 14. Comparisons results for the problems of all sizes.

| Problem size | n | m | w | NFS | MID | DM | SNS | Time |
|--------------|---|---|---|-----|-----|----|-----|------|
| Small        | 8 | 2 | 2 | 4 | 487 | 510 | 500 | 14  |
|              | 3 | 2 | 2 | 4 | 477 | 495 | 444 | 0   |
|              | 10| 3 | 1 | 7 | 315 | 587 | 550 | 0   |
|              | 4 | 2 | 2 | 7 | 580 | 726 | 542 | 174 |
| Medium       | 15| 4 | 2 | 1 | 10| 104 | 107 | 313 | 756 |
|              | 6 | 3 | 1 | 2 | 937 | 109 | 990 | 888 |
|              | 20| 4 | 1 | 7 | 104 | 188 | 512 | 0   |
| Large        | 30| 8 | 4 | 3 | 145 | 214 | 152 | 413 |
|              | 10| 6 | 3 | 2 | 135 | 280 | 711 | 624 |
|              | 40| 10| 6 | 2 | 3  | 265 | 446 | 809 |
|              | 50| 12| 8 | 3 | 2  | 242 | 600 | 378 |
|              | 15| 10| 2 | 3 | 125 | 593 | 531 | 370 |

For medium size problems, although the MOTS algorithm performs better than the MOSA and hybrid algorithms in some problems according to the MID criterion, the hybrid algorithm performs better than the other two algorithms considering other performance evaluation metrics. Similar to the medium size problems, the number of near-optimal Pareto points obtained by the hybrid algorithm is higher than the other two algorithms for all large size problems. Furthermore, although the MOTS algorithm has a relatively better performance according to the MID criterion compared to the other two algorithms, the proposed hybrid algorithm outperforms in most cases considering other metrics.

To investigate the convergence of the proposed algorithms and also their behaviors during previous iterations, further analysis is performed as follows. A large problem with \( n = 40, m = 10 \) and \( w = 6 \) is considered and solved using all three proposed algorithms. The pareto fronts of MOTS, MOSA and the hybrid algorithm after some number of replications, as well as the pareto front of their final iteration are depicted in Figures 15-17.

It can generally be seen that the number of Pareto solutions increases as the number of iterations grows and this increase in the number of Pareto solutions is considerable for the proposed hybrid algorithm. In addition, the displacement of the Pareto solutions on the length and width axes during different iterations of each algorithm shows the trend of change in the values of \( f_1 \) and \( f_2 \) well. The convergence curves of the proposed algorithms are shown in Figures 18-20. Pareto solutions in each iteration are sorted from small to large based on the values of \( f_1 \). In this order, \( P_z, (z = 1, 2, \ldots, Z) \) denotes the zth pareto solution. Note that \( Z \) varies per iteration and algorithm.

It can be observed that the final step of convergence obtained from MOTS is achieved in the 66th iteration. For MOSA, the algorithm converges after 137 iterations, and for the proposed hybrid algorithm, the final significant changes in the pareto front are occurred in the 54th replication as illustrated in Figure 21.
4.4.1. **Statistical test.** In order to compare the algorithms more precisely, using statistical approaches, three separate one-way ANOVA tests for small, medium and large size problems are performed according to the metrics used. The ANOVA analysis tests the differences between the means of the results achieved by the proposed algorithms to find out whether they are statistically significant or not. The one-way ANOVA is carried out for all test problems using SPSS 16.0 as follows:

\[
\begin{align*}
H_0 & : \mu_{MOTS} = \mu_{MOSA} = \mu_{Hybrid} \\
H_1 & : Means \ are \ not \ are \ equal
\end{align*}
\]
where, the confidence interval for the mean difference between the values is considered 95%. The results are displayed in Table 15. Note that, for each sample problem, the mean of solutions on its Pareto front is used to analyze the S-metric. For each pareto point, the objective value is calculated via: \( F = 0.6 \times f_1 + 0.4 \times f_2 \).

According to Table 15. It can be observed that except for the MID metric for small size problems and the SNS metric for medium and large size problems, in other
cases $\text{Sig. (p-value)} \leq 0.05$ which means the null hypothesis ($H_0$) is rejected. Hence, there is at least one statistically significant difference between the groups tested. Post hoc pairwise multiple comparisons should be used to determine which two or more groups are significantly different. For this purpose, it is first referred to the test
Figure 21. The last significant change in the Pareto front of the proposed hybrid algorithm.

Table 15. One-way ANOVA for all test problems (between groups).

| Metric | Problem size | Sum of Squares | df | Mean Square | F    | Sig |
|--------|--------------|----------------|----|-------------|------|-----|
| S-metric | Small | 8758922.903 | 2  | 4379461.452 | 5.519 | 0.012 |
|         | Medium | 8.541E7 | 2  | 4.271E7     | 5.519 | 0.012 |
|         | Large  | 1.348E9 | 2  | 6.739E8     | 6.716 | 0.008 |
| NP      | Small  | 130.083  | 2  | 65.042      | 24.500 | 0.000 |
|         | Medium | 8.541E7 | 2  | 4.271E7     | 4.538 | 0.029 |
|         | Large  | 42.333   | 2  | 21.167      | 15.744 | 0.000 |
| MID     | Small  | 6748717.583 | 2 | 3374358.792 | 2.312 | 0.124 |
|         | Medium | 1.932E8  | 2  | 9.662E7     | 5.348 | 0.029 |
|         | Large  | 2.856E9  | 2  | 1.428E9     | 11.064 | 0.001 |
| DM      | Small  | 8430960.250 | 2 | 4215480.125 | 10.368 | 0.001 |
|         | Medium | 7.217E7  | 2  | 3.608E7     | 7.737 | 0.005 |
|         | Large  | 1.954E8  | 2  | 9.768E7     | 5.840 | 0.013 |
| SNS     | Small  | 122196.583 | 2 | 61098.292 | 5.999 | 0.013 |
|         | Medium | 1.564E7  | 2  | 7817526.167 | 2.349 | 0.130 |
|         | Large  | 1.236E8  | 2  | 6179140.222 | 0.994 | 0.911 |
| Time    | Small  | 20371.750 | 2 | 10185.875 | 21.221 | 0.000 |
|         | Medium | 43694.111 | 2 | 21847.056 | 6.425 | 0.010 |
|         | Large  | 177652.111 | 2 | 88826.056 | 10.736 | 0.001 |

of homogeneity of variances. If $\text{Sig} \leq 0.05$, there is a significant difference between the variances. Hence the Dunnett T3 comparative test is applied; otherwise, equal variances are assumed and since samples size is equal, Tukey comparison is used. The results of post-hoc multiple comparisons are presented in Table 16.

The results show that the proposed hybrid algorithm improves the performance of the MOSA and even outperforms the MOTS in several cases. Significant differences in computational time of the proposed hybrid algorithm with the other two algorithms may be due to the fact that the hybrid algorithm avoids storing duplicate solutions in the archive of non-dominated solutions as much as possible, which remarkably decreases the computational time of the proposed hybrid algorithm compared to the other two algorithms.

5. Conclusion and future work. Unrelated parallel machines scheduling problem is an important class of scheduling models that is of great theoretical and practical importance. In this study, bi-objective unrelated parallel machines scheduling problem with workers allocation and sequence dependent setup times considering machine eligibility and precedence constraints was investigated. This is a highly practical problem in the real world and can be used in various environments such as plants, banks, and stations. The proposed problem was formulated as a new mixed-integer quadratic programming model. It was validated with the global solver of
LINGO 18.0 using a random test problem. Because the problem is NP-hard, to find near-optimal solutions in a reasonable computational time, a multi-objective tabu search and a multi-objective simulated annealing were employed. Furthermore, a hybrid algorithm, which is a combination of MOTS and MOSA approaches, was developed. Afterwards, using a set of test problems, performances of the proposed metaheuristics were evaluated according to different comparative metrics. Results show that the proposed MOTS performs better with respect to the S-metric for the problems of all sizes and to the MID metric for medium and large size examples, which do not statistically imply significant differences. Besides, the suggested hybrid algorithm mostly outperforms MOTS and MOSA with regards to the other employed metrics. Hence, it can be concluded that the proposed hybrid algorithm performs satisfactorily. Finally, one-way ANOVA indicates that the presented hybrid algorithm is doing more efficiently as the size of problems grow, so it can be employed for real world large sized problems.

For future research, non-deterministic conditions like fuzzy processing times can be assumed. A multi-objective optimization including other objective functions like minimizing total completion time, production costs, or total tardiness or earliness can be investigated. Different constraints such as machine breakdown or the possibility of defective jobs can be considered. Some population-based metaheuristics such as genetic algorithm can be used for further comparisons with the proposed algorithms. In addition, applying the proposed hybrid algorithm to solve optimization problems in different areas is suggested.

REFERENCES

[1] M. Afzalirad and J. Rezaeian, Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions, Computers & Industrial Engineering, 98 (2016), 40–52.
[2] M. Afzalirad and M. Shafipour, Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions, Journal of Intelligent Manufacturing, 29 (2018), 423–437.
[3] O. A. Arik and M. D. Toksari, Multi-objective fuzzy parallel machine scheduling problems under fuzzy job deterioration and learning effects, International Journal of Production Research, 56 (2018), 2488–2505.
A. Baykasoglu, Applying multiple objective tabu search to continuous optimization problems with a simple neighbourhood strategy, *International Journal for Numerical Methods in Engineering*, 65 (2006), 406–424.

T. Çağar, R. Köker and Y. Sari, Parallel robot scheduling to minimize mean tardiness with unequal release date and precedence constraints using a hybrid intelligent system, *International Journal of Advanced Robot Systems*, 9 (2012), 252.

C. L. Chen, Iterated hybrid metaheuristic algorithms for unrelated parallel machines problem with unequal ready times and sequence-dependent setup times, *The International Journal of Advanced Manufacturing Technology*, 60 (2012), 693–705.

L. P. Cota, F. G. Guimarães, R. G. Ribeiro, I. R. Meneghini, F. B. de Oliveira, M. J. Souza and P. Siarry, An adaptive multi-objective algorithm based on decomposition and large neighborhood search for a green machine scheduling problem, *Swarm and Evolutionary Computation*, 51 (2019), 100601.

M. Dhillaoui, H. E. Nouri and O. B. Driss, Dual-resource constraints in classical and flexible job shop problems: A state-of-the-art review, *Procedia Computer Science*, 126 (2018), 1507–1515.

E. C. H. Dorion, J. C. F. Guimarães, E. A. Severo, Z. C. Reis and P. M. Oleva, Innovation and production management through a just in sequence strategy in a multinational brazilian metal-mechanic industry, 2014 IEEE International Conference on Management of Innovation and Technology, (2014), 54–60.

P. Engrand, A multi-objective optimization approach based on simulated annealing and its application to nuclear fuel management, 1998.

A. E. Ezugwu, O. J. Adeleke and S. Viriri, Symbiotic organisms search algorithm for the unrelated parallel machines scheduling with sequence-dependent setup times, *PloS One*, 13 (2018), e0200030.

A. E. Ezugwu and F. Akutsah, An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times, *IEEE Access*, 6 (2018), 54459–54478.

C. M. Fonseca, L. Paquete and M. López-Ibáñez, An improved dimension-sweep algorithm for the hypervolume indicator, 2006 IEEE International Conference on Evolutionary Computation, (2006), 1157–1163.

F. Glover, Future paths for integer programming and links to artificial intelligence, *Comput. Oper. Res.*, 13 (1986), 533–549.

F. W. Glover and G. A. Kochenberger, (Eds.), *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, 57. Kluwer Academic Publishers, Boston, MA, 2003.

G. Gong, R. Chiong, Q. Deng and X. Gong, A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility, *International Journal of Production Research*, 58 (2020), 4406–4420.

G. Gong, R. Chiong, Q. Deng, W. Han, L. Zhang, W. Lin and K. Li, Energy-efficient flexible flow shop scheduling with worker flexibility, *Expert Systems with Applications*, 141 (2020), 112902.

A. Hamzadaiy and G. Yildiz, Modeling and solving static m identical parallel machines scheduling problem with a common server and sequence dependent setup times, *Computers & Industrial Engineering*, 106 (2017), 287–298.

M. P. Hansen, Tabu search for multiobjective optimization: MOTS, *Proceedings of the 13th International Conference on Multiple Criteria Decision Making*, (1997), 574–586.

O. Hurdies, Just in time (JIT) production: An effective approach to efficiency, *Logistics & Supply Chain Review*, 1 (2020), 32–39.

B. Khanh Van and N. Van Hop, Genetic algorithm with initial sequence for parallel machines scheduling with sequence dependent setup times based on earliness-tardiness, *Journal of Industrial and Production Engineering*, 38 (2021), 18–28.

J. G. Kim, S. Song and B. Jeong, Minimising total tardiness for the identical parallel machine scheduling problem with splitting jobs and sequence-dependent setup times, *International Journal of Production Research*, 58 (2020), 1628–1643.

S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, *Science*, 220 (1983), 671–680.
[24] D. Lei, Y. Yuan, J. Cai and D. Bai, An imperialist competitive algorithm with memory for distributed unrelated parallel machines scheduling, *International Journal of Production Research*, 58 (2020), 597–614.

[25] H. M. Md, Lead time reduction and process cycle improvement of an ice-cream manufacturing factory in Bangladesh by using value stream map and kanban board, *Australian Journal Of Basic and Applied Sciences*, (2016).

[26] A. Munoz-Villamizar, J. Santos, J. Montoya-Torres and M. Alvaréz, Improving effectiveness of parallel machine scheduling with earliness and tardiness costs: A case study, *International Journal of Industrial Engineering Computations*, 10 (2019), 375–392.

[27] J. Rezaeian, N. Derakhshan, I. Mahdavi and R. A. Foroutan, Due date assignment and JIT scheduling problem in blocking hybrid flow shop robotic cells with multiple robots and batch delivery cost, *International Journal of Industrial Mathematics*, 13 (2021), 145–162.

[28] M. Savsar, Simulation analysis of a pull-push system for an electronic assembly line, *International Journal of Production Economics*, 51 (1997), 205–214.

[29] G. Taguchi, Introduction to quality engineering: Designing quality into products and processes, No. 658.562 T3 (1986).

[30] E. Vallada and R. Ruiz, A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times, *European J. Oper. Res.*, 211 (2011), 612–622.

[31] B. Wang and H. Wang, Multiobjective order acceptance and scheduling on unrelated parallel machines with machine eligibility constraints, *Math. Probl. Eng.*, 2018 (2018), 12pp.

[32] I. L. Wang, Y. C. Wang and C. W. Chen, Scheduling unrelated parallel machines in semiconductor manufacturing by problem reduction and local search heuristics, *Flexible Services and Manufacturing Journal*, 25 (2013), 343–366.

[33] S. Wang, X. Wang, J. Yu, S. Ma and M. Liu, Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan, *Journal of Cleaner Production*, 193 (2018), 424–440.

[34] J. Xu, X. Xu and S. Q. Xie, Recent developments in Dual Resource Constrained (DRC) system research, *European Journal of Operational Research*, 215 (2011), 309–318.

[35] K. C. Ying and S. W. Lin, Unrelated parallel machine scheduling with sequence-and machine-dependent setup times and due date constraints, *International Journal of Innovative Computing*, 8 (2012), 3279–3297.

[36] A. Zhang, X. Qi and G. Li, Machine scheduling with soft precedence constraints, *European J. Oper. Res.*, 282 (2020), 491–505.

[37] J. R. Zeidi and S. MohammadHosseini, Scheduling unrelated parallel machines with sequence-dependent setup times, *The International Journal of Advanced Manufacturing Technology*, 81 (2015), 1487–1496.

[38] L. Zhang, Q. Deng, G. Gong and W. Han, A new unrelated parallel machine scheduling problem with tool changes to minimise the total energy consumption, *International Journal of Production Research*, 58 (2020), 6826–6845.

[39] Z. Zhu and X. Zhou, An efficient evolutionary grey wolf optimizer for multi-objective flexible job shop scheduling problem with hierarchical job precedence constraints, *Computers & Industrial Engineering*, 140 (2020), 106280.

Received January 2021; 1st revision June 2021; final revision September 2021; early access November 2021.

E-mail address: reza.a.foroutan@gmail.com
E-mail address: j.rezaeian@ustmb.ac.ir
E-mail address: mi.shafipour@gmail.com