International Journal of Engineering

Journal Homepage: www.ije.ir

Solving a New Multi-objective Unrelated Parallel Machines Scheduling Problem by Hybrid Teaching-learning Based Optimization

A. Sadati[a], R. Tavakkoli-Moghaddam*[b], B. Naderi[c], M. Mohammadi[c]

[a] *Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*
[b] *School of Industrial Engineering, South Tehran Brach, Islamic Azad University, Tehran, Iran*
[c] *Department of Industrial Engineering, Faculty of Engineering,* Kharazmi *University, Tehran, Iran*

PAPER INFO

Paper history:
Received 22 December 2016
Received in revised form 19 January 2017
Accepted 22 January 2017

Keywords:
Teaching–learning Based Optimization
Unrelated Parallel Machines
Makespan
Tardiness
Earliness

ABSTRACT

This paper considers a scheduling problem of a set of independent jobs on unrelated parallel machines (UPMs) that minimizes the maximum completion time (i.e., makespan or C_{max}), maximum earliness (E_{max}), and maximum tardiness (T_{max}) simultaneously. Jobs have non-identical due dates, sequence-dependent setup times and machine-dependent processing times. A multi-objective mixed-integer linear programming (MILP) is considered and then solved with the ε-constraint method in small-sized problems. The results are compared with those obtained by meta-heuristic algorithms. Furthermore, an effective hybrid multi-objective teaching–learning based optimization (HMOTLBO) algorithm is proposed, whose performance is compared with a non-dominated sorting genetic algorithm (NSGA-II) for test problems generated at random. The results show that the proposed HMOTLBO outperforms the NSGA-II in terms of different metrics.

***doi**: 10.5829/idosi.ije.2017.30.02b.09*

1. INTRODUCTION

This paper considers a parallel machines scheduling problem in which different machines perform the same function with different processing velocity, namely unrelated parallel machines (UPMs). In this study, each job has machine-dependent processing time, sequence-dependent setup time and due date.

In order to improve the performance of the production systems, we consider both manufacturer concerns, such as waiting time and WIP inventory and customer concerns, such as assuring on time receipt. For this purposes, a multi-objective problem to minimize makespan (i.e., C_{max}), maximum tardiness (i.e., T_{max}) and maximum earliness (i.e., E_{max}) is considered simultaneously. To make the problem applicable in real environment, it is sequence-dependent setup time [1]. Tavakkoli-Moghaddam et al. [2] presented a mathematical model for the UPMs scheduling problem, which minimizes the total earliness/tardiness penalties. They proposed a GA algorithm. Tavakkoli-Moghaddam et al. [3] presented the UPMs scheduling problem to minimize the total completion time and number of tardy jobs. They proposed a two-level mixed-integer programming (MIP) model for their problem. Tavakkoli-Moghaddam and Mehdizade [2] showed an integer linear programming (ILP) model for an identical PMs scheduling problem with family setups in order to minimize the total weighted flow time. They proposed a genetic algorithm (GA) to obtain good and near-optimal solutions. Tavakkoli-Moghaddam and Aramon-Bajestani [4] proposed lower and upper bounds for a UPMs scheduling problem to minimize the total weighted tardiness.

Torabi et al. [5] proposed a novel multi-objective model for a UPMs problem under uncertain processing times and due dates and proposed multi-objective particle swarm optimization (MOPSO) in order to find a Pareto frontier, in such a way that the total weighted flow time, total machine load variation and total weighted tardiness should be minimized. Lin and Ying

*Corresponding Author's Email: tavakoli@ut.ac.ir (R. Tavakkoli-Moghaddam)

[6] considered a UPMs scheduling problem with job sequence-dependent setup times and proposed a hybrid artificial bee colony algorithm in order to minimize the makespan. Rodriguez et al. [7] considered the UPMs scheduling problem that minimizes the total weighted completion time and proposed an iterated greedy meta-heuristic algorithm using destruction and construction phases in order to obtain a number of solutions. Bozorgirad and Logendran [8] addressed a UPMs scheduling problem with sequence-dependent group setup times that minimizes the total weighted completion time and total weighted tardiness and proposed tabu search to solve this problem. Lin et al. [9] considered a UPMs scheduling problem that minimizes the makespan, total weighted completion time and total weighted tardiness, and compared the performance of various heuristics. Nogueira et al. [10] considered a UPMs scheduling problem with machine and job sequence-dependent setup times and idle times that minimizes the total earliness/tardiness penalties. They proposed three different heuristics contained simple GRASP, path relinking and iterated local search.

Gharehgozli et al. [11] showed a new fuzzy mixed-integer goal programming (MIGP) model for a PMs scheduling problem with sequence-dependent setup times and release dates to minimize the total weighted flow time and the total weighted tardiness. Kayvanfar et al. [12] considered a PMs system with controllable processing times of jobs to minimize the makespan and total weighted tardiness/earliness penalties.

Tavakkoli-Moghaddam et al. [3] and [13] presented a novel, two-level MIP model for a UPMs scheduling problem with sequence-dependent setup times, non-identical due dates, ready times and precedence relations to minimize the number of tardy jobs and the total completion time.Gao [14] considered a multi-objective parallel machines scheduling problem to minimize the maximum completion time (i.e., makespan) and total earliness/tardiness penalties and proposed an artificial immune algorithm for solving this problem. Chyuand Shang [15] considered a bi-objective UPMs scheduling problem with job-sequence setup times and machine-dependent setup times to minimize the total weighted flow time and total weighted tardiness. They proposed a Pareto evolutionary algorithm to solve their problem. Lin and Lin [16] considered a UPMs scheduling problem to minimize the makespan and total weighted tardiness, and presented heuristic and tabu search algorithms to solve their problem. Salehi Mir and Rezaeian [17] considered a UPMs scheduling problem with sequence-dependent setup time, release dates, deteriorating jobs and learning effects to minimize the total machine load. They proposed the hybrid PSO-GA. Joo and Kim [18] presented a UPMs scheduling problem with sequence and machine-dependent setup times to minimize the total completion time. Additionally, they proposed a hybrid genetic algorithm with three dispatching rules.

A great number of meta-heuristic algorithms (i.e., ABC, PSO and DE) have been proposed in the last few decades. However, a teaching–learning based optimization (TLBO) algorithm is one of them proposed by Roa et al. [19]. In this paper, we hybridized this algorithm with hill climbing search for a new multi-objective UPMs scheduling problem. Furthermore, the ε-constraint method is used to solve this problem in small-sized problems and the results obtained by the hybrid multi-objective TLBO algorithm are compared with the results obtained by the non-dominated sorting genetic algorithm (NSGA-II).

2. PROBLEM FORMULATION

This paper presents the scheduling problem of a set of N independent jobs on M unrelated parallel machines to minimizethe makespan, maximum tardiness and maximum earliness simultaneously. It is assumed that each job can be processed by only one machine and each machine can process at most one job at a time. No job preemptions are allowed and each job becomes available at time zero. Jobs have sequence-dependent setup times with the same priority. The mentioned model is modified from the models presented in [3] and [13] as follow:

Notations:

N	Number of jobs
M	Number of machines
i	Job indices ($i = 1, \dots, N$)
m	Machine indices ($m = 1, \dots, M$)
K_m	Number of positions on machine m, $K_m \leq N$
k	Position indices ($k = 1, \dots, K_m$)
d_i	Due date of job$i$
p_{im}	Processing time of job i on machine$m$
s_{ijm}	Setup time to switch from job i to job j on machine m
x_{ikm}	Equals 1, if job i is scheduled in the kth position on machine m; and 0, otherwise
c_i	Completion time job i
t_i	Tardiness job$i$,$t_i = \max(0, c_i - d_i)$
e_i	Earliness job$i$,$e_i = \max(0, d_i - c_i)$
T_{max}	Maximum tardiness
E_{max}	Maximum earliness
C_{max}	Makespan

According to the above-mentioned assumptions and notations; the problem can be modelled by:

$$\text{Min}\, Z_1 = T_{max} \tag{1}$$

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

$$\text{Min}\, Z_2 = C_{max} \tag{3}$$

s.t.

$$\sum_{m=1}^{M}\sum_{k=1}^{K_m} x_{ikm} = 1; \quad \forall i \tag{4}$$

$$\sum_{i=1}^{N} x_{ikm} \le 1; \quad \forall m, k \tag{5}$$

$$\sum_{i=1}^{N} x_{ikm} - \sum_{j=1}^{N} x_{jk-1m} \le 0; \ \forall k \ge 2, m \tag{6}$$

$$c_i \ge p_{im} * x_{i1m}; \quad \forall m, i \tag{7}$$

$$c_i = \sum_{m=1}^{M}\sum_{k=2}^{K_m}\sum_{\substack{j=1\\ j\neq i}}^{N} x_{jk-1m} \times x_{ikm} \times (c_j + s_{jim}) + \sum_{m=1}^{M}\sum_{k=1}^{K_m} p_{im} \times x_{ikm}; \ \forall i \tag{8}$$

$$t_i \ge c_i - d_i; \quad \forall i \tag{9}$$

$$e_i \ge d_i - c_i; \quad \forall i \tag{10}$$

$$T_{max} \ge t_i; \quad \forall i \tag{11}$$

$$E_{max} \ge e_i; \quad \forall i \tag{12}$$

$$C_{max} \ge c_i; \quad \forall i \tag{13}$$

$$x_{ikm} = 0 \ or \ 1; \quad \forall m, k, i \tag{14}$$

$$t_i \ge 0, e_i \ge 0, c_i \ge 0; \quad \forall i \tag{15}$$

This model minimizes the maximum tardiness, maximum earliness and makespan stated by objective functions (1) to (3), respectively. Constraint (4) states that each job is assigned to exactly one position on one machine. Constraint (5) guarantees that assignment of at most one job to each position on each machine. Constraint (6) ensures that until one position on a machine is empty, jobs are not assigned to subsequent positions and jobs assigned to empty positions on each machines, respectively. Constraints (7) and (8) together ensure that only after starting the process by machine, no idle time could be inserted into the schedule, and no job preemption is allowed and should calculate the completion time of jobs. Constraint (9) is the definition of the tardiness of jobs. Constraint (10) is the definition of the earliness of jobs. Constraint (11) defines the maximum tardiness. Constraint (12) defines the maximum earliness. Constraint (13) defines the maximum completion time. Constraints (14) and (15) define the type of decision variable.

3. SOLUTION ALGORITHM

A solution representation in HMOTLBO and a chromosome representation in NSGA-II is an array consisting of $N + M - 1$ real values between (0, 1).

Coding: An array consisting of $N + M - 1$real values between (0, 1)

Decoding: Like a random key genetic algorithm (RKGA), values sorted in a descending order then according to the position of each value in the main representation, an integer between 1 to $N + M - 1$do it. These are integer numbers as the coding scheme for a multi-machines scheduling problem. The integer number representation all possible permutation of N jobs and $M - 1$ machines. Numbers that are smaller than and equal N represent jobs and numbers that are larger than N represent machines. For example, number $N + 1$ is $MACHINE_1$, $N + 2$ is $MACHINE_2$ and similarity $N + M - 1$ is $MACHINE_{M-1}$ and numbers prior to them are jobs allocated to them. Finally, for numbers Finally, numbers with smaller than and equal to N are assigned to $MACHINE_M$. Following is a simple example with nine jobs and three machines according Figure 1.

3. 1. Initial Population

For two proposed meta-heuristic algorithms, we produce solutions (learners)/ chromosomes to the number of the population size then compute objective functions according to Equations (11) to (13). For example, in the problem with nine jobs, three machines and population size 3, a sample of the population is shown in Table 1.

1. Producing 11 (3+9-1) real random in (0,1) and put them into the boxes:										
1	2	3	4	5	6	7	8	9	10	11
0.905	0.127	0.913	0.964	0.097	0.278	0.546	0.957	0.970	0.157	0.632
2. Sorting real numbers in descending order										
9	4	8	3	1	11	7	6	10	2	5
0.970	0.964	0.957	0.913	0.905	0.632	0.546	0.278	0.157	0.127	0.097
3. Decoding procedure										
			$MACHINE_1$	7,6						
			$MACHINE_2$:	9,4,8,3,1						
			$MACHINE_3$:	2,5						

Figure 1. Solution representation, encoding and decoding procedures

TABLE 1. Problem with 9 jobs, 3 machines and population size 3

													T_{max}	E_{max}	C_{max}
	L_1	0.498	0.945	0.340	0.585	0.223	0.751	0.255	0.506	0.699	0.890	0.959	127	6	137
Population=	L_2	0.547	0.138	0.149	0.257	0.840	0.254	0.814	0.243	0.929	0.350	0.196	36	11	46
	L_3	0.251	0.616	0.473	0.351	0.830	0.585	0.549	0.917	0.285	0.757	0.753	146	0	156

3. 2. Teaching–learning Based Optimization Roa et al. [19] proposed a teaching-learning based optimization (TLBO) algorithm. This algorithm is based on learning a group of learners of a teacher in a class, and this group is considered as population. The teacher in each population is considered as the best learner. The learning process in this algorithm includes two stages, the first one is named teacher stage and the second the learner stage as explained hereunder:

3. 2. 1. Teacher Stage In this stage, the learners' level of knowledge in iteration $t(x_{i,t})$transfers by using DM_t, i.e., difference between the teacher $x_{T,t}$ and mean result of learners M_t. Updated learner $(x_{i,t}^{'})$ is considered as follows:

$$x_{i,t}^{'} = x_{i,t} + DM_t \tag{16}$$

where, $DM_t = r_t(x_{T,t} - T_F M_t)$ (17)

T_F is the teaching factor [19] and $r_t$is the random number in the range [0, 1]. The detailed implementation for our problem is given as follows:

3. 2. 1. 1. Select Teacher In each iteration, the teacher is considered to be the best learner, so form the fronts and rank population using fast non-dominated sorting and compute the crowding distance [20]. Teacher selection based on select the solution with the lowest rank and in case of equality ranks, the solution with greater crowded distances is considered. The grade of the teacher is usually higher than the grade of the learner. Therefore, after selecting the best learner as the teacher by using hill-climbing search, the grade of the selected learner is increased and considered as teacher.

For example, learners' ranks of the above example are (1,1,1), Because of Equal ranks, we compute the crowding distances as (3, ∞, ∞). According to computed crowding distances, learner 2 or learner 3 is considered as teacher. We select learner 2 as a teacher, then using hill climbing search, the selected teacher is improved (Table 2).

3. 2. 1. 2. Mean Result of Learners, M_t For obtaining the mean result of learners, calculate the mean of population columns (Table 2).

3. 2. 1. 3. DM_t The difference between the teacher and the mean result at iteration *t* is computed using Equation (17).
For the above example, suppose that TF=1 and $r_t = 0.8$ (Table 2).

TABLE 2. Teacher stage

Teacher	0.547	0.138	0.149	0.257	0.840	0.254	0.814	0.243	0.929	0.350	0.196
M_t	0.432	0.566	0.321	0.398	0.631	0.530	0.539	0.555	0.638	0.666	0.636
DM_t	0.092	-0.342	-0.138	-0.113	0.167	-0.221	0.220	-0.250	0.233	-0.253	-0.352
$x_{1,t}^{'}$	0.590	0.603	0.202	0.472	0.390	0.530	0.475	0.256	0.932	0.637	0.607

TABLE 3. Learner stage

$x_{1,t}^{'}$	0.033	0.326	0.590	0.145	0.365	0.439	0.808	0.279	0.080	0.640	0.571

Current *solution*	0.970	0.957	0.485	0.800	0.141	0.421	0.915	0.792	0.959	0.655	0.035
Swap	0.970	0.421	0.485	0.792	0.141	0.957	0.915	0.800	0.959	0.655	0.035
Current	0.970	0.957	0.485	0.800	0.141	0.421	0.915	0.792	0.959	0.655	0.035
Inversion	0.970	0.957	0.915	0.421	0.141	0.800	0.485	0.792	0.959	0.655	0.035
Current *solution*	0.970	0.957	0.485	0.800	0.141	0.421	0.915	0.792	0.959	0.655	0.035
Insertion	0.970	0.957	0.959	0.485	0.800	0.141	0.421	0.915	0.792	0.655	0.035
Current *solution*	0.970	0.957	0.485	0.800	0.141	0.421	0.915	0.792	0.959	0.655	0.035
subtracting from 1	0.030	0.043	0.515	0.200	0.859	0.579	0.085	0.208	0.041	0.345	0.965

Figure 2. Types of neighborhood structures in hill climbing

3. 2. 1. 4. Update the Learners Each learner in population is updated by using Equation (16). If the updated number is negative, it is converted to positive, and if the obtained number is greater than 1, let (obtained number-1) instead of it. For the above example, for learner 1($x_{1,t}$), updated form is $x'_{1,t}$ (Table 2):

$$x'_{1,t} = x_{1,t} + DM_t \tag{18}$$

3. 2. 1. 5. Acceptance Updated Learner If$x'_{i,t}$ dominates$x_{i,t}$, then replaced$x_{i,t}$ by $x'_{i,t}$. This work done with Definition 1.

Definition 1. x_i dominates x_j if $f(x_i) \le f(x_j)$ for all objective functions and $f(x_i) < f(x_j)$ for at least one of objective functions [21].

For the above example, objective functions $x_{1,t}$ and $x'_{1,t}$ are (127 6 137) and (163 3 169) respectively; therefore, $x'_{1,t}$dose not dominate $x_{1,t}$, and$x_{1,t}$cannot be replaced by $x'_{1,t}$.

3. 2. 2. Learner Stage In this stage, the fronts are first formed and the population obtained from the teacher stage using fast non-dominated sorting is ranked as well, and the crowding distance is computed [20]. Then, the following steps in iteration *t* are carried out:

If $rank(x_{i,t}) < rank(x_{j,t})$ (if $rank(x_{i,t}) = rank(x_{j,t})$, then if crowding distance $(x_{i,t})>$ crowding distance$(x_{j,t})$), then

$$x'_{i,t} = x_{i,t} + r_t(x_{i,t} - x_{j,t}) \quad r_t \in (0,1) \tag{19}$$

Step 1. For learner $x_{i,t}$, randomly select another learner

If $rank(x_{j,t}) < rank(x_{i,t})$ (if $rank(x_{j,t}) = rank(x_{i,t})$ then-if crowding distance $(x_{j,t})>$ crowding distance$(x_{i,t})$), then

$$x'_{i,t} = x_{i,t} + r_t(x_{j,t} - x_{i,t}) \quad r_t \in (0,1) \tag{20}$$

$x_{j,t}$ $(i \ne j)$.

Step 2. Update learner $x_{i,t}$ $(x'_{i,t})$ by using Equations (19) or (20).

Step 3. If updated number is negative converted to positive and if obtained number is greater than 1, let (obtained number-1) instead of it.

Step 4. If $x'_{i,t}$ dominates $x_{i,t}$, then replace $x_{i,t}$ by $x'_{i,t}$.

Step 5. To make sure learning is done at the end of learner stage, if learner $x_{i,t}$ is not replaced by $x'_{i,t}$ using hill-climbing search, a better learner is replaced instead of $x_{i,t}$.

For the above example, suppose for learner 1 ($x_{1,t}$), learner 3 ($x_{3,t}$) is selected. According to Equation (20) and r_t =0.4, learner 1 is updated as follows (Table 3):

$$x'_{1,t} = x_{1,t} + r_t(x_{3,t} - x_{1,t}) \tag{21}$$

Objective functions $x_{1,t}$ and $x'_{1,t}$ are (127 6 137) and (129 2 139), respectively. Therefore, $x'_{1,t}$ dose not dominate $x_{1,t}$, and$x_{1,t}$ cannot be replaced by $x'_{1,t}$ and with hill climbing search $x_{1,t}$ is improved and inserted to replace it. Teacher and learner stages are repeated until the stopping criteria is met.

```
Consider current solution $x_{current}$
While (Termination criteria is not met) do
    $x_1 \leftarrow Swap$
    $x_2 \leftarrow Inversion$
    $x_3 \leftarrow Insertion$
    $x_4 \leftarrow subtracting\ from\ 1$
    $X = [x_{current}, x_1, x_2, x_3, x_4]$
    Calculate objective functions $X$ (Equations (11), (12), (13))
    Fast non-dominated sort $X$ [20, 184]
    $x'$ ←Select solution with lowest rank in $X$
    $x_{current} \leftarrow x'$
End while
```

Figure 3. Pseudo code of hill climbing

Parent$_1$	0.970	0.957	0.485	0.800	0.141	0.421	0.915	0.792	0.959	0.655	0.035
Parent$_2$	0.814	0.905	0.127	0.632	0.097	0.546	0.964	0.913	0.957	0.157	0.278
The coin results	Head	Head	Tail	Tail	Head	Tail	Head	Head	Tail	Head	Tail
Offspring	0.970	0.957	0.127	0.632	0.141	0.546	0.915	0.792	0.957	0.655	0.278

Figure 4. Uniform crossover operator

Parent	0.970	0.957	0.485	0.800	0.141	0.421	0.915	0.792	0.959	0.655	0.035
Offspring	0.970	0.421	0.485	0.800	0.141	0.957	0.915	0.792	0.959	0.655	0.035

Figure 5. Swapping mutation operator

3. 3. Hill-climbing Search In this paper, we use four types of neighborhood structures that contain swap, inversion, insertion and subtracting from 1 [22] (Figure 2). A pseudo code of the proposed hill-climbing method is shown in Figure 3.

3. 4. NSGA-II The NSGA-II proposed in this paper is described as follows:

Parameters:

Pop: Population

Npop: Population size

Pc: Percentage of the offspring population that completed with crossover operation

Pm: Percentage of the offspring population that completed with mutation operation

Npc: Numbers of offspring that created by crossover operation

Npm: Numbers of offspring that created by mutation operation

Tournament size: Number of individuals that are selected for tournament

Max-It: Maximum number of times to repeat the algorithm (termination condition)

Steps:

1. Create an initial population with *Npop* numbers by using Section 4 and calculate the value of objective functionsC_{max}, $T_{max}$and E_{max}(Equations (11) to (13)).
2. Form the fronts and rank population using fast non dominated sorting and compute crowding distance [20].
3. Create population of offspring with *Npop* numbers that included *Npc* (*Npop*×*Pc*) individuals that are obtained from crossover operator (Figure 4), *Npm* (*Npop*×*Pc*) obtained from mutation operator (Figure 5) and reset individuals are selected from the parent population (all of selections for crossover operator, mutation operator and reset individual doing in the tournament size, using the non-dominated sorting and crowding distance individuals in the parent population).
4. Combine the parent population and offspring population and create the population with 2×*Npop*, then compute the non-dominated sorting and crowding distance of individual of the current population and for creating a new population with *Npop* individual use the method offered by Deb et al. [20].
5. Repeat Steps 1 to 4 until *Max-It* is occurred.

3. 5. ε-constraint Method In this method, one of the objective functions is placed as objective function and optimized. The other objective functions are transferred into constraints as follows:

$$\text{Min} \quad f_j(x) \tag{22}$$

s.t.

$$f_h(x) \le \varepsilon_h; \quad h = 1, \dots, m, \quad h \neq j, \tag{23}$$

$$x \in S \tag{24}$$

where $j \in \{1, \dots, m\}$ and ε_h is upper bounds for the objective h ($h \neq j$). VariousPareto solutions can be found by changing the value of ε_h. We can let $f_h^{min} = f_h^*$ and $f_h^{max} = f_h^{nadir}$ by using payoff matrix [21]. We consider three sample problems in small sizes with 8, 6 and 4 jobs and 2 machines. These problems are solved with Lingo 8.0 and their results are shown in Table 10, in which the results of two proposed algorithms for the same problems are presented as well.

4. COMPUTATIONAL RESULTS

In order to test the effectiveness of the NSGA-II and HMOTLBO, we solve several test problems, (e.g., Naderi-Beni et al. [23]), and then compare their performances with a number of different metrics. The proposed meta-heuristics are coded in MATLAB R2016a software.

4. 1. Test Problem Instances Computational results given in [23] are conducted in medium and large-sized problems according to Tables 4 and 5. The processing times (p_{im}) are integers and are generated from a uniform distribution of U(1, 20), the due dates (d_i) are uniformly distributed in the interval $\left[P\left(1-t-\frac{r}{2}\right), P\left(1-t+\frac{r}{2}\right)\right]$, where $P = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} p_{ij}}{2M}$, t =0.8 , r =0.2 and the setup times are integers and are generated from a uniform distribution of U(1, 20).

4. 2. Evolution Metric Quality of the non-dominated solutions obtained from the proposed meta-heuristic algorithms is used to compare these algorithms. In this paper, three metrics (i.e., *N*-metric, *R*-metric and *S*-metric) are used [24].

4. 3. Parameter Tuning The quality of the solutions obtained from the proposed algorithms is affected by the values of their parameters. To set the parameters of proposed algorithms in this paper, the response surface methodology (RSM) [25] using Design Expert software is applied. The number of Pareto solutions created by proposed NSGA-II and HMOTLBO algorithms are used as responses and considered in order to ease comparisons. The levels of NSGA-II algorithm parameters are shown in Table 6. For the given parameters, RMS designs 30 experiments that contain 6 central and 24 axial points. According to our results, best combinations of parameters for medium and large problems are showed in Table 7.

For tuning the parameters of HMOTLBO, the levels of these parameters are shown in Table 8. For the given parameters, the RMS designs 20 experiments that contain six central and 14 axial points. According to the results, the best combinations of parameters for medium and large-sized problems are shown in Table 9.

TABLE 4. Medium-sized problems.

M	*N*	*M*	*N*	*M*	*N*	*M*	*N*
3	10	4	15	5	20	6	25
3	20	4	30	5	40	6	50
3	30	4	45	5	60	6	75
3	40	4	60	5	80	6	90

TABLE 5. Large-sized problems.

M	*N*	*M*	*N*	*M*	*N*	*M*	*N*
7	30	8	35	9	20	10	45
7	60	8	70	9	40	10	90
7	90	8	105	9	60	10	135
7	120	8	140	9	80	10	180

TABLE 6. The levels of NSGA-II parameters

	Levels	
Parameters	Lower	Upper
MaxIt	10	90
Npop	50	210
Pc	0.1	0.9
Pm	0.02	0.1

TABLE 7. Best combinations of NSGA-II parameters

		Parameters			
		Max-It	*Npop*	*Pc*	*Pm*
Value	M	60	150	0.6	0.07
	L	50	210	0.5	0.06

4. 4. Evaluating Results

Each proposed algorithm is run 3 times and each time includes 10 runs in 32 test problems including 16 medium problems and 16 large problems (Tables 4 and 5).The average results of 3 times are shown in Tables 11 and 12. The results of small problems are given in Table 10 that shows for small problems (N=4, 6, 8 and M=2), HMOTLBO and NSGA-II produce Pareto-optimal solutions similar to the ε-constraint method, although CPUT is reduced significantly. Tables 11 and 12 show that the HMOTLBO is better than the NSGA-II in terms of different metrics for medium and large-sized problems. The amount of changing the solving times in NSGA-II is less than HMOTLBO. Solving times in NASGA-II versus HMOTLBO has substantially reduced in larger size problems. Nvalues in both algorithms are relatively close; however, Rvalues are different. The results show that NSGA-II produces lots of Pareto solutions, but dominated by a few Pareto solutions produced by HMOTLBO.

TABLE 8. Levels of HMOTLBO parameters

	Levels	
Parameters	Lower	Upper
Max-It	5	25
Npop	10	30
TF	1	2

TABLE 9. Best combinations of HMOTLBO parameters

Parameters			
	MaxIt	*Npop*	*TF*
M	15	30	1
L	15	25	1.25

TABLE 10. Results for small-sized problem

	ε-Constraint				NSGA-II				HMOTLBO			
N	C_{max}	T_{max}	E_{max}	*CPUT*	C_{max}	T_{max}	E_{max}	*CPUT*	C_{max}	T_{max}	E_{max}	*CPUT*
4	26	15	2	120	26	15	2	8.46	26	15	2	7.62
	29	19	0		29	19	0		29	19	0	
6	39	24	11	1234	39	24	11	8.82	39	24	11	8.74
	41	27	0		41	27	0		41	27	0	
8	40	26	0	5346	40	26	0	9.21	40	26	0	8.98

TABLE 11. Results of NSGA-II and HMOTLBO algorithms for medium-sized problems

M	*N*	*CPUT*		*S*		*N*		*R*	
		HMOTLBO	NSGA-II	HMOTLBO	NSGA-II	HMOTLBO	NSGA-II	MOTLBOH	NSGA-II
3	10	15.17	23.57	3.11	1.77	3	3	0.77	0.83
3	20	16.64	24.24	4.12	4.97	4.67	3.33	0.96	0.6
3	30	19.36	24.81	12.45	77.85	6	6.67	0.84	0.57
3	40	22.18	25.40	156.91	18.75	5.33	9.67	0.69	0.76
4	15	16.52	24.32	2.57	2.48	4	3.33	1	0.4
4	30	20.33	25.54	25.88	94.49	7.33	4.67	1	0.29
4	45	24.15	25.87	18.96	45.40	6.67	5.67	0.87	0.49
4	60	27.09	26.79	22	142.81	6.67	8	0.81	0.59
5	20	18.90	24.71	14.70	7.33	7.67	1.67	0.95	0.16
5	40	23.38	26.48	4.07	127.91	6.33	3.67	0.97	0.31
5	60	29.64	27.40	18.72	29.22	9.33	4.33	0.85	0.38
5	80	35.95	29.46	38.25	7.15	6	9.67	0.96	0.53
6	25	21.56	26.64	7.55	25.79	8	5.33	1	0.34
6	50	26.12	26.78	30.86	23.40	9	6.67	0.86	0.44
6	75	34.32	28.55	111.63	59.39	8.33	6	0.81	0.47
6	90	42.14	30.72	52.92	32.51	4.67	10.33	0.68	0.68
Average		24.59	26.33	32.79	43.83	6.44	5.75	0.88	0.49

TABLE 12. Results of NSGAII and HMOTLBO algorithms for large-sized problems

M	*N*	*CPUT*		*S*		*N*		*R*	
		HMOTLBO	NSGA-II	HMOTLBO	NSGA-II	HMOTLBO	NSGA-II	HMOTLBO	NSGA-II
7	30	19.31	40.45	8.87	24.66	6.67	5.33	0.96	0.33
7	60	26.76	44.23	8.4	58.55	7.33	4.67	0.74	0.43
7	90	35.22	51.23	17.77	80.12	6.67	4.33	1	0.32
7	120	44.43	53.26	101.71	139.49	8.67	8.33	0.78	0.59
8	35	20.06	44.34	6.89	22.41	7.33	4.33	1	0.29
8	70	30.04	47.67	8.86	15.23	5.67	4	0.97	0.19
8	105	39.41	52.36	42.61	38.26	8.33	4.33	0.95	0.35
8	140	53.4	56.12	17.45	11.32	4.33	8.67	0.95	0.33
9	40	21.55	45.33	8.55	69.17	7.33	4	0.94	0.29
9	80	33.01	49.17	15.28	52.73	8.67	5.33	0.94	0.28
9	120	46.12	53.38	51.38	63.91	6	9.67	0.71	0.49
9	160	61.12	58.55	75.59	15.96	9.67	8	0.89	0.29
10	45	24.56	47.47	24.66	13.11	8.33	5.33	0.96	0.24
10	90	35.92	50.26	12.89	19.91	7	5.67	0.89	0.38
10	135	53.35	56.65	21.75	26.77	5.67	4	0.84	0.15
10	180	70.56	61.81	54.77	11.39	3.67	2.33	0.68	0.13
Average		38.43	50.77	29.84	41.44	6.96	5.52	0.89	0.32

5. CONCLUSION AND FUTURE RESEARCH

This paper has studied a multi-objective unrelated parallel machines scheduling problem in order to minimize the makespan, maximum tardiness and maximum earliness of jobs, in which sequence-dependent setup times are machine-dependent processing times have been considered. Additionally, a multi- objective mixed-integer linear programming (MILP) model has been formulated, and then solved by the ε-constraint method for small-sized problems. The results have been compared with the results obtained by the proposed meta-heuristic algorithms. Furthermore, an effective hybrid multi-objective teaching–learning based optimization (HMOTLBO) has been proposed. Its performance has been compared with a non-dominated sorting genetic algorithm (NSGA-II) on a number of test problems generated at random. The related results have shown that the proposed HMOTLBO is relatively better than the NSGA-II.

In this study, we have considered real conditions in an industrial environment. Of course, there are other conditions that help us to improve our research, such as taking into account pre-emption, precedence constrains and machine failures. Also, one can use other meta-heuristic algorithms and compare the results with our results.

6. REFERENCES

1. Chang, P. C. and Chen, S.-H., "Integrating dominance properties with genetic algorithms for parallel machine scheduling problems with setup times", ***Applied Soft Computing***, Vol. 11, No. 1, (2011), 1263-1274.
2. Tavakkoli-Moghaddam, R. and Mehdizadeh, E., "A new ILP model for identical parallel-machine scheduling with family setup times minimizing the total weighted flow time by a genetic algorithm", ***International Journal of Engineering Transactions A Basics***, Vol. 20, No. 2, (2007), 183-194.
3. Tavakkoli-Moghaddam, R., Taheri, F. and Bazzazi, M., "Multi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints", ***International Journal of Engineering, Transactions A: Basics***, Vol. 21, No. 3, (2008), 269-278.
4. Tavakkoli-Moghaddam, R. and Aramon-Bajestani, M., "A novel b and b algorithm for a unrelated parallel machine scheduling problem to minimize the total weighted tardiness", ***International Journal of Engineering***, Vol. 22, No. 3, (2009), 269-286.
5. Torabi, S. A., Sahebjamnia, N., Mansouri, S. A. and Bajestani, M. A., "A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem", ***Applied Soft Computing***, Vol. 13, No. 12, (2013), 4750-4762.
6. Lin, S.-W. and Ying, K.-C., "Abc-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times", ***Computers & Operations Research***, Vol. 51, (2014), 172-181.
7. Rodriguez, F. J., Lozano, M., Blum, C. and GarciA-MartiNez, C., "An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem", ***Computers & Operations Research***, Vol. 40, No. 7, (2013), 1829-1841.
8. Bozorgirad, M. A. and Logendran, R., "Sequence-dependent group scheduling problem on unrelated-parallel machines", ***Expert Systems with Applications***, Vol. 39, No. 10, (2012), 9021-9030.
9. Lin, Y.-K., Pfund, M. E. and Fowler, J. W., "Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems", ***Computers & Operations Research***, Vol. 38, No. 6, (2011), 901-916.
10. Nogueira, J. P., Arroyo, J. E. C., Villadiego, H. M. M. and Goncalves, L. B., "Hybrid grasp heuristics to solve an unrelated parallel machine scheduling problem with earliness and tardiness penalties", ***Electronic Notes in Theoretical Computer Science***, Vol. 302, (2014), 53-72.
11. Gharehgozli, A., Tavakkoli-Moghaddam, R. and Zaerpour, N., "A fuzzy-mixed-integer goal programming model for a parallel-machine scheduling problem with sequence-dependent setup times and release dates", ***Robotics and Computer-Integrated Manufacturing***, Vol. 25, No. 4, (2009), 853-859.
12. Kayvanfar, V., Komaki, G. M., Aalaei, A. and Zandieh, M., "Minimizing total tardiness and earliness on unrelated parallel machines with controllable processing times", ***Computers & Operations Research***, Vol. 41, (2014), 31-43.
13. Tavakkoli-Moghaddam, R., Taheri, F., Bazzazi, M., Izadi, M. and Sassani, F., "Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints", ***Computers & Operations Research***, Vol. 36, No. 12, (2009), 3224-3230.
14. Gao, J., "A novel artificial immune system for solving multiobjective scheduling problems subject to special process constraint", ***Computers & Industrial Engineering***, Vol. 58, No. 4, (2010), 602-609.
15. Chyu, C.-C. and Chang, W.-S., "A pareto evolutionary algorithm approach to bi-objective unrelated parallel machine scheduling problems", ***The International Journal of Advanced Manufacturing Technology***, Vol. 49, No. 5-8, (2010), 697-708.
16. Lin, Y.-K. and Lin, H.-C., "Bicriteria scheduling problem for unrelated parallel machines with release dates", ***Computers & Operations Research***, Vol. 64, (2015), 28-39.
17. Mir, M. S. S. and Rezaeian, J., "A robust hybrid approach based on particle swarm optimization and genetic algorithm to minimize the total machine load on unrelated parallel machines", ***Applied Soft Computing***, Vol. 41, (2016), 488-504.
18. Joo, C. M. and Kim, B. S., "Hybrid genetic algorithms with dispatching rules for unrelated parallel machine scheduling with setup time and production availability", ***Computers & Industrial Engineering***, Vol. 85, (2015), 102-109.
19. Rao, R. V., Savsani, V. J. and Vakharia, D., "Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems", ***Computer-Aided Design***, Vol. 43, No. 3, (2011), 303-315.
20. Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: NSGA-ii", ***IEEE Transactions on Evolutionary Computation***, Vol. 6, No. 2, (2002), 182-197.
21. Deb, K. and Miettinen, K., "Multiobjective optimization: Interactive and evolutionary approaches, Springer Science & Business Media, Vol. 5252, (2008).
22. Soni, N. and Kumar, T., "Study of various mutation operators in genetic algorithms", ***International Journal of Computer Science and Information Technologies (IJCSIT)***, Vol. 5, No. 3, (2014), 4519-4521.
23. Naderi-Beni, M., Ghobadian, E., Ebrahimnejad, S. and Tavakkoli-Moghaddam, R., "Fuzzy bi-objective formulation for

a parallel machine scheduling problem with machine eligibility restrictions and sequence-dependent setup times", ***International Journal of Production Research***, Vol. 52, No. 19, (2014), 5799-5822.

24. Han, Y.-Y., Gong, D.-w., Sun, X.-Y. and Pan, Q.-K., "An improved NSGA-ii algorithm for multi-objective lot-streaming flow shop scheduling problem", ***International Journal of Production Research***, Vol. 52, No. 8, (2014), 2211-2231.

25. Montgomery, D. C., "Design and analysis of experiments, John Wiley & Sons, (2008).

Solving a New Multi-objective Unrelated Parallel Machines Scheduling Problem by Hybrid Teaching-learning Based Optimization

A. Sadati[a], R. Tavakkoli-Moghaddam[b], B. Naderi[c], M. Mohammadi[c]

[a] *Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*
[b] *School of Industrial Engineering, South Tehran Brach, Islamic Azad University, Tehran, Iran*
[c] *Department of Industrial Engineering, Faculty of Engineering,* Kharazmi *University, Tehran, Iran*

P A P E R I N F O

Paper history:
Received 22 December 2016
Received in revised form 19 January 2017
Accepted 22 January 2017

Keywords:
Teaching–learning Based Optimization
Unrelated Parallel Machines
Makespan
Tardiness
Earliness

چکیده

در این مقاله، مسأله زمان‌بندی یک مجموعه از کارهای غیروابسته روی ماشین‌های موازی غیریکسان به منظور کمینه کردن بیشینه زمان تکمیل کارها (طول برنامه زمان‌بندی)، بیشینه دیرکرد و بیشینه زودکرد مورد مطالعه قرار می‌گیرد. کارها دارای موعدهای تحویل غیریکسان، زمان‌های راه‌اندازی وابسته به توالی و زمان‌های پردازش وابسته به ماشین می‌باشند. یک مدل برنامه‌ریزی خطی عدد صحیح مختلط چندهدفه مورد توجه قرار می‌گیرد که در مسائل با اندازه کوچک با روش محدویت–اپسیلون مورد حل شده است. سپس، نتایج با نتایج حاصل از الگوریتم‌های فراابتکاری بیان شده مقایسه می‌شوند. یک الگوریتم بهینه‌سازی چندهدفه تلفیقی بر اساس آموزش–یادگیری (HMOTLBO) پیشنهاد می‌شود و کارایی این الگوریتم پیشنهادی با الگوریتم ژنتیک مرتب شده غیرمغلوب(NSGA-II) روی تعدادی از مسائل نمونه که به صورت تصادفی تولید شده‌اند، مقایسه می‌شوند. نتایج نشان می‌دهد که الگوریتم پیشنهادی HMOTLBO نسبت به NSGA-II نتایج بهتری را ارائه می‌دهد.

doi*: 10.5829/idosi.ije.2017.30.02b.09*