Computers and Operations Research 103 (2019) 46–63

Contents lists available at ScienceDirect

Computers and Operations Research

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

A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server

Gulcin Bektur*, Tugba Saraç

Industrial Engineering Department, Eskişehir Osmangazi University, Eskişehir, Turkey

ARTICLE INFO

Article history:
Received 9 April 2018
Revised 11 October 2018
Accepted 11 October 2018
Available online 12 October 2018

Keywords:
Unrelated parallel machine scheduling
Scheduling with a common server
Simulated annealing
Tabu search
Sequence-dependent setup times
Dispatching rule

ABSTRACT

Parallel machine scheduling problems with common servers have many industrial applications. In this paper, we study a generalized problem of scheduling with a common server, which is the unrelated parallel machine scheduling problem with sequence-dependent setup times and machine eligibility restrictions. The objective function involves the minimization of the total weighted tardiness. A mixed integer linear programming (MILP) model is proposed to solve this complex problem. Due to the NP hardness of the problem, tabu search (TS) and simulated annealing (SA) algorithms are proposed. The initial solutions of the algorithms are obtained by a modified apparent tardiness cost with setups (ATCS) dispatching rule. The proposed algorithms are compared using a randomly generated data set.

1. Introduction

In this paper, we consider an unrelated parallel machine scheduling problem with a common server. In most cases, the necessary resources are limited in modern production systems. Studies that consider the servers used in the setup operation of jobs are called scheduling with common server problems (Hamzadayı and Yıldız, 2017). When there is a single server for setup operations, only one job is set up at a time (Kim and Lee, 2012). Common servers are used in many industrial applications. One of these applications is flexible manufacturing systems. In a flexible manufacturing system, a robot or AGV (Automated Guided Vehicle) is shared by machines during the setup operation (Hall et.al., 2000). Another application is in the printing industry. Specifically, when copier machines are set up according to an order, the staff team is considered a common server, and the server is shared by the machines during the setup phase (Huang et.al., 2010).

We investigate an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server. The objective is to minimize the total weighted tardiness. According to the standard three-field notation described by Graham et al. (1979), our problem could be denoted as $R_m|ST_{sd}, S_1, M_j|\Sigma w_j T_j$.

The scheduling notations used in the literature review are summarized in Table 1.

In previous studies of unrelated machine scheduling, a common server has not been examined. Chen and Chen (2009) and Kayvanfar and Teymourian (2014) proposed a hybrid heuristic algorithm that considers unrelated parallel machines and sequence-dependent setup times to solve this problem. Chen (2012) considered unrelated parallel machines, sequence-dependent setup times, and different arrival times, and a hybrid metaheuristic algorithm was proposed. Afzalirad and Rezaeian (2017) considered the problem of scheduling unrelated parallel machines with sequence-dependent setup times. They proposed a multiobjective ant colony optimization approach.

Only identical parallel machines are discussed in the literature for scheduling problems with a common server. A few studies of common servers have considered sequence-dependent setups. Among the studies that consider sequence-dependent setup times and identical parallel machines, Huang et al. (2010) suggested a hybrid genetic algorithm (GA) for the problem of $PD_m|ST_{sd}, S_1|C_{max}$ and applied the solution methodology to a printing industry case study. Türker and Sel (2011) proposed a hybrid heuristic approach for the problem of $P_2|ST_{sd}, S_1|C_{max}$. Bektur and Saraç (2016) addressed the problem of $P_2|ST_{sd}, S_1|C_{max}$ and developed a mathematical model and a GA for the problem. Hamzadayı and Yıldız (2016a) proposed SA and dispatching rules for the problem of $P_m|ST_{sd}, S_1, r_j, brkdwn, prmp|C_{max}$. Hamzadayı and Yıldız (2016b) proposed an SA and seven dispatching rule-based

* Corresponding author.
E-mail addresses: gcol@ogu.edu.tr (G. Bektur), tsarac@ogu.edu.tr (T. Saraç).

https://doi.org/10.1016/j.cor.2018.10.010
0305-0548/

Table 1
The scheduling notations used in the literature review.

Notation	Description
P_m, S_1	m identical parallel machines with a common server
PD_m, S_1	m dedicated parallel machines with a common server
ST_{sd}	Sequence-dependent setup time
r_j	Release dates
brkdwn	Breakdowns
S_{m-1}	(m-1) units of the common server
C_{max}	Makespan
ΣC_j	Total completion time
prmp	Preemptions

complete rescheduling approaches for the dynamic m identical parallel machines scheduling problem with a common server to generate new schedules depending on the hybrid rescheduling policy. Hamzadayı and Yıldız (2017) proposed a mathematical model, SA and GA, for the problem of $P_m|\ ST_{sd},\ S_1\ |C_{max}$.

In the literature on scheduling with a common server, the objective function is often considered the makespan. In most studies, two parallel machines are considered, or setup and processing times are equally considered. Koulamas (1996) considered two identical parallel machines with one server problem; the objective function was minimizing machine idle times, and a beam search algorithm was proposed. Kravchenko and Werner (1997) showed that the problem of $P_2|S_1,\ s_j=1|C_{max}$ is NP hard. They proposed a list scheduling approach for m machines. Kravchenko and Werner (1998) considered the problem of $P_m|S_{m-1},\ s_j=1|C_{max}$. They assumed that the setup times of all jobs are equal to one and proposed a polynomial time algorithm. Glass et.al. (2000) showed that the problem of $PD_2|S_1,\ s_{ij}=s,\ p_j=p|C_{max}$ is NP hard and proposed a greedy algorithm for the $PD_2|S_1\ |C_{max}$ problem. Hall et al. (2000) obtained new complexity results for the problem of identical parallel machine scheduling with one server. Kravchenko and Werner (2001) proposed a metaheuristic algorithm for the problem of $P_m|S_1,\ s_j=1|C_j$, under the assumption that the job setup times are equal. Abdekhodaee and Wirth (2002) considered the problem of $P_2|S_1|C_{max}$. Brucker et.al. (2002) discussed the complexity of the parallel machine scheduling problem with a common server. A mathematical model and two different heuristic algorithms were proposed for two identical parallel machine scheduling problems with a common server, and the objective function was minimizing the makespan. Abdekhodaee et.al (2004) showed that the $P_2|S_1,\ p_j=p,\ s_j=s|C_{max}$ problem is NP hard. They assumed that the processing time and job setup times were equal and found the lower bounds of the problem. Guirchoun et al. (2005) presented new complexity results for identical parallel machines with a single server. Abdekhodaee et al. (2006) proposed a GA for the problem of $P_2|S_1,\ p_j=p,\ s_j=s|C_{max}$. Zhang and Wirth (2009) proposed two different heuristic approaches for $P_2|S_1|C_{max}$ under the assumption that the release dates of the jobs are dynamic. Ou et.al. (2010) considered the problem of $P_m|S_1|C_j$ and proposed short processing time (SPT) first and branch and bound algorithm to solve the problem. Werner and Kravchenko (2010) considered the problem of $P_m|S_1|C_{max}$ and proposed a polynomial time algorithm for the problem with equal processing and setup times. Su (2011) proposed a dynamic longest processing time (LPT) first algorithm for the problem of $P_2|S_1,\ r_j|C_{max}$. Kim and Lee (2012) proposed two different mathematical models and a hybrid heuristic algorithm for the problem of $P_m|S_1|C_{max}$. Gan et.al. (2012) proposed a branch-and-price algorithm for the scheduling problem of two identical parallel machines with a common server; the objective function was minimizing the makespan. Jiang et.al. (2013) considered the problem of $P_2|S_1,\ prmp,\ s_{ij}=s,\ p_j=p\ |C_{max}$ and proposed a heuristic approach. Hasani et.al. (2014a) suggested a mathematical model for the problem of $P_2|\ S_1|C_{max}$. In another study by Hasani et.al. (2014b), a GA and SA were proposed for the same problem. Jiang et.al. (2015) proposed list scheduling and an LPT approach for the problem of $P_2|S_1,\ s_i=1\ |C_{max}$. Zhang et.al. (2016) proposed an SPT algorithm for the problem of $P_m|S_1,\ p_j=p|C_j$.

In this study, a mixed integer linear programming (MILP) model is proposed for the problem. The problem considered in this study is NP hard because the single machine scheduling problem that minimizes the total tardiness is NP hard (Du and Leung, 1990). Because the problem is NP hard, simulated annealing (SA) and tabu search (TS) algorithms are proposed to solve the problem, and the performances of the MILP model, SA and TS are compared. Initial solutions of the algorithms are obtained based on the modified ATCS dispatching rule.

In previous studies of machine scheduling with common server problems, only identical parallel machines were examined. In this study, unrelated parallel machines, which are generalizations of other parallel machine scheduling problems, were studied. In unrelated parallel machine scheduling, the processing times of jobs vary according to the machine (Yang- Kueui and Chi- Wei, 2013). Unrelated parallel machines constitute the most difficult parallel machine environment (Vallada and Ruben, 2011). Moreover, the objective of minimizing the total weighted tardiness has never been examined for parallel machine scheduling with a common server. Only complexity analyses with the objective of minimizing total tardiness have been performed. This study is the first to use the TS algorithm for scheduling problems with a common server and to modify the apparent tardiness cost with setups (ATCS) dispatching rule for problems with a common server.

The remainder of this paper is organized as follows. In the next section, the problem is described, and an MILP model is proposed. In Section 3, the modified ATCS, SA and TS algorithms are presented. In Section 4, we analyze the computational performances of the algorithms based on test problems. Finally, Section 5 provides conclusions and suggests avenues for future work.

2. Problem description

The problem in this study was defined by considering plastic injection machines. In the plastic part production industry, a factory receives orders with different colors, sizes and other specifications. Machines must be cleaned and prepared for the next order to be processed. The molds used to produce plastic parts are positioned on machines before production. These setup processes involve a team who must work together to set up a machine. Therefore, the team is considered a common server. All machines share this server, and the server can set up only one machine at any time. If more than one machine requests the server, machine idle times occur. The setup operation must be completed before a job can be processed on the machine. The server cannot perform setup operations for a machine if the machine is processing a job, but once the setup is completed, the machine can process a job without the presence of the server. Plastic parts are produced by an injection machine and a mold. A plastic part cannot be produced on a machine if the corresponding mold is not positioned on the machine. Therefore, machine eligibility restrictions are taken into account. For each job j, there is a set of machines M_j $(j=1,\ldots,n)$ capable of processing the job. In addition, unrelated parallel machines, which is the most realistic case of parallel machine scheduling problems, are addressed. Job setup times are sequence dependent because the setup time is shorter when similar jobs are produced in succession on the same machine, and the setup time is longer if the previous job was dissimilar. Setup times are not dependent on the machine because a common server per-

forms sequence-dependent setup regardless of the machine. The objective is the minimization of the total weighted tardiness.

It is assumed that the machines are not malfunctioning and that jobs are available at time zero. The machines are ready at the beginning of the scheduling period. No setup is required for the first jobs of each machine. If we consider the setup times for the first jobs, the server consecutively completes all first jobs, and the last machine must wait for a period equal to the total setup time of the first jobs. However, this is not the case in real life because machines operate without interruption. In other words, the machines run without interruption, and during the scheduling period, there is almost no requirement for setup operations on all machines at the same time. In addition, the setup time for the first job is not dependent on the sequence, and it can be dependent on the job. This time is smaller than the other setup times and can be ignored. Therefore, the setup times for the first jobs are neglected in many studies in the literature (Hamzadayı and Yıldız, 2017). Moreover, there is no precedence among jobs, and preemption is not allowed in the system. An MILP model is proposed for the problem.

2.1. An MILP model for the considered problem

2.1.1. Sets and indices

N: Set of jobs to be processed, N = {1, 2, …, n}
M: Set of machines, M = {1, 2,…, m}
j, h, q: Job indices, where j, h, q ∈ N.
k, o: Position indices, where k, o ∈ N
i: Machine indices, where i ∈ M

2.1.2. Parameters

p_{ij}: *Processing time for job j on machine i*
s_{hj}: *Sequence-dependent setup time to process job j after job h on the same machine*
BM: Very large positive number
w_j: *Weight of job j*
d_j: *Due date of job j*
b_{ij}: $\begin{cases} 1; & \text{if machine } i \text{ is capable of processing job } j \\ 0; & \text{otherwise} \end{cases}$

2.1.3. Decision variables

C_j: Completion time of job j
A_j: Start time of the setup operation for job j
V_j: Wait time of job j for the server to be available
Z_j: Completion time of the setup operation for job j
$X_{ijk} = \begin{cases} 1; & \text{If job } j \text{ is processed on machine } i \text{ in the } k\text{th position} \\ 0; & \text{otherwise} \end{cases}$
$E_{jq} = \begin{cases} 1; & \text{If the completion time of the setup for job } j \\ & \quad \text{is less than the start time of the setup for job } q \\ 0; & \text{If the completion time of the setup for job } q \\ & \quad \text{is less than the start time of the setup for job } j \end{cases}$
T_j: Tardiness of job j

2.1.4. Model

$$Min\ z = \sum_j w_j T_j \tag{1}$$

Subject to

$$C_j + BM\left(1 - X_{ij1}\right) \geq \sum_o X_{ijo} p_{ij} + V_j \quad \forall\ j,\ i \tag{2}$$

$$C_j - BM\left(1 - X_{ij1}\right) \leq \sum_o X_{ijo} p_{ij} + V_j \quad \forall\ j,\ i \tag{3}$$

$$C_j + BM\left(2 - X_{ih(k-1)} - X_{ijk}\right) \geq s_{hj} + \sum_o X_{ijo} p_{ij} + C_h + V_j \quad \forall\ h, j,\ k > 1,\ i \text{ and } h \neq j \tag{4}$$

$$C_j - BM\left(2 - X_{ih(k-1)} - X_{ijk}\right) \leq s_{hj} + \sum_o X_{ijo} p_{ij} + C_h + V_j \quad \forall\ h, j,\ k > 1,\ i \text{ and } h \neq j \tag{5}$$

$$\sum_j X_{ijk} - \sum_h X_{ih(k-1)} \leq 0 \quad \forall\ i \text{ and } k > 1 \tag{6}$$

$$\sum_j X_{ijk} \leq 1 \quad \forall\ k,\ i \tag{7}$$

$$\sum_k \sum_i X_{ijk} = 1 \quad \forall\ j \tag{8}$$

$$A_j \geq C_h + V_j - BM * \left(2 - X_{ijk} - X_{ih(k-1)}\right) \quad \forall\ h, j,\ k > 1,\ i \text{ and } h \neq j \tag{9}$$

$$A_j \leq C_h + V_j + BM * \left(2 - X_{ijk} - X_{ih(k-1)}\right) \quad \forall\ h, j,\ k > 1,\ i \text{ and } h \neq j \tag{10}$$

$$A_j \leq V_j + BM * \left(1 - X_{ijk}\right) \quad \forall\ j,\ i \text{ and } k = 1 \tag{11}$$

$$A_j \geq V_j - BM * \left(1 - X_{ijk}\right) \quad \forall\ j,\ i \text{ and } k = 1 \tag{12}$$

$$Z_j \geq A_j - BM\left(1 - X_{ijk}\right) \quad \forall\ j,\ i \text{ and } k = 1 \tag{13}$$

$$Z_j \leq A_j + BM\left(1 - X_{ijk}\right) \quad \forall\ j,\ i \text{ and } k = 1 \tag{14}$$

$$Z_j \geq A_j + s_{hj} - BM\left(2 - X_{ih(k-1)} - X_{ijk}\right) \quad \forall\ h, j,\ k > 1,\ i \text{ and } h \neq j \tag{15}$$

$$Z_j \leq A_j + s_{hj} + BM\left(2 - X_{ih(k-1)} - X_{ijk}\right) \quad \forall\ h, j,\ k > 1,\ i \text{ and } h \neq j \tag{16}$$

$$Z_q \leq A_j + BME_{jq} \quad \forall j,\ q \text{ and } j < q \tag{17}$$

$$Z_j \leq A_q + BM(1 - E_{jq}) \quad \forall j,\ q \text{ and } j < q \tag{18}$$

$$T_j \geq C_j - d_j \quad \forall j \tag{19}$$

$$X_{ijk} \leq b_{ij} \quad \forall j,\ k,\ i \tag{20}$$

$$C_j,\ A_j, V_j, Z_j, T_j \geq 0 \quad \forall j \tag{21}$$

$$X_{ijk}, E_{jq} \in \{0, 1\} \quad \forall j,\ k, i, q \tag{22}$$

(1) is the objective function. Constraints (2) and (3) calculate the completion times of the first-positioned jobs on each machine. Constraints (4) and (5) calculate the completion times of the other positioned ($k > 1$) jobs. Constraint (6) states that if job j is processed on machine i, it will be immediately followed by at most one other job on the same machine. Constraint (7) ensures that

Table 2
Parameters of the problem.

p_{ij}		b_{ij}		w_j	d_j	s_{hj}										
M1	M2	M1	M2			jobs	1	2	3	4	5	6	7	8	9	10
59	25	1	1	6	103	1	0	10	9	10	19	7	11	17	9	10
33	57	1	1	8	105	2	14	0	7	15	22	9	9	17	23	12
84	66	1	1	1	102	3	25	16	0	9	7	17	8	8	11	6
37	47	1	1	6	117	4	22	5	10	0	11	12	12	12	25	7
39	84	1	1	5	133	5	20	10	10	17	0	17	11	16	24	15
81	94	1	1	4	113	6	12	15	7	15	9	0	6	16	21	21
100	35	1	0	7	96	7	19	19	11	15	13	19	0	10	22	11
32	44	1	1	7	122	8	22	14	8	5	7	21	7	0	10	19
61	19	1	1	3	100	9	20	18	11	12	13	9	14	18	0	15
47	33	1	1	4	99	10	18	24	21	14	19	12	17	19	12	0

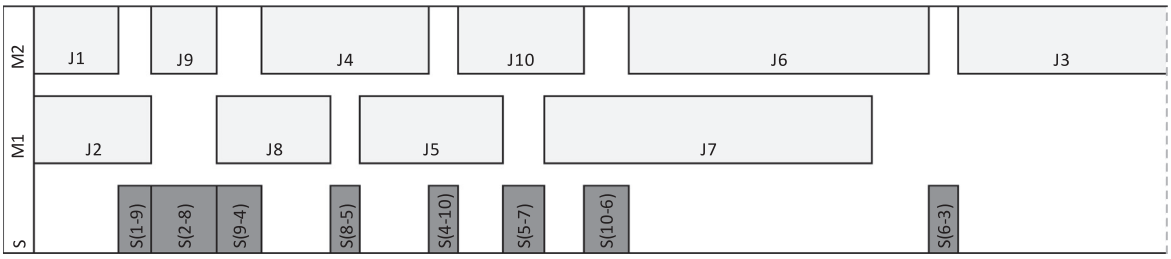

Fig. 1. Schedule of the small-sized problem obtained by solving the proposed MILP model.

only one job can be processed at one position on a machine. Constraint (8) ensures that each job is assigned to one position on a machine. Constraints (9)–(12) calculate the start times of the setup operations. Constraints (13)–(16) calculate the completion times of the setup operations. Constraints (17),(18) ensure that the server is processing only one setup operation at any time. In other words, either the setup of job j starts after the setup of job q is completed or the setup of job q starts after the setup of job j is finished. Constraint (19) calculates the job tardiness. Constraint (20) is a set of machine eligibility restrictions. Constraints (21),(22) are sign constraints.

The handling of the problem is illustrated by a small problem, such as 10 jobs on 2 machines. The parameters of the problem are given in Table 2. The problem is solved with the proposed MILP model. The objective function value is 2029, and the obtained schedule of the problem is given in Fig. 1.

3. Proposed dispatching rule, SA and TS algorithms

3.1. Dispatching rule for the $R_m|ST_{sd}, S_1, M_j|\Sigma W_J T_J$ *(Initial solution-finding mechanism)*

A modified ATCS dispatching rule is proposed to find an initial solution for the SA and TS algorithms. In scheduling problems, when a machine is idle, dispatching rules are used to determine which job is assigned to the relevant position on the machine. These rules are used to prioritize jobs, and jobs with the highest priority are assigned to the appropriate machine for processing. These algorithms are greedy polynomial time algorithms, and their application is simple. Dispatching rules can also guarantee that the optimum schedule is achieved in some cases, but the associated applications are limited due to the loss of applicability when different constraints are added to the scheduling problem. In these cases, different dispatching rules can be combined (Lee et.al., 1997).

The ATCS dispatching rule was proposed by Lee et al. (1997) to minimize the total weighted tardiness when jobs are subject to sequence-dependent setup times. The priority index is calculated by the following equation:

$$I_j = \frac{w_j}{p_j} exp\left[\frac{-max\left(d_j - p_j - t,\ 0\right)}{k_1 \bar{p}}\right] exp\left[\frac{-s_{hj}}{k_2 \bar{s}}\right] \tag{23}$$

where t denotes the idle state time of the machine, h is the index of the job that was previously completed on the machine, the values of the parameters k_1 and k_2 depend on the problem instance and are scale related, $\bar{s}$ denotes the average setup time and $\bar{p}$ denotes the average processing time.

In the literature, ATC and ATCS dispatching rules have been modified to solve scheduling problems with different constraints. Mönch et.al. (2005) proposed a modified ATC dispatching rule for the problem of scheduling parallel batch processing machines with unequal ready times. Pfund et al. (2008) proposed a modified ATCS dispatching rule for the problem of $P_m|r_j, s_{jk}|\Sigma w_j T_j$. Lin et al. (2013) used the ATC dispatching rules in a phase of their heuristic algorithm and studied the problem of $R_m||C_{max}$, $\Sigma w_j C_j$, $\Sigma w_j T_j$. Kuei and Wei (2013) proposed the apparent tardiness cost with release date (ATCR) dispatching rule for the problem of $R_m|r_j|\Sigma w_j T_j$.

In contrast to the ATCS rule in the literature, this study considers unrelated parallel machines, a common server and machine eligibility restrictions. Therefore, the ATCS rule is modified based on these constraints.

The priority index of jobs is calculated by the following equation.

$$I_{i^*j} = \frac{w_j}{p_{i^*j}} \exp\left[\frac{-\max\left(d_j - p_{i^*j} - \max\left(t_{i*}, CST\right), 0\right)}{k_1 \bar{p}_{i^*}}\right] \exp\left[\frac{-s_{hj}}{k_2 \bar{s}}\right] \tag{24}$$

According to Eq. (24), the common server can process only one setup at any given time. Moreover, i^* denotes the selected machine, h denotes the index of the job completed immediately before job j at machine i^*, index i is for machines, and j is the index for jobs. t_i denotes the time at which machine i waits for processing. The common server time (CST) is the time at which the common server waits for the setup operation. $\bar{p}_{i^*}$ is the average processing time of

```
Procedure modified ATCS algorithm
Input: p_ij, b_ij, s_hj, d_j, w_j, n and m
Output: Initial solution (S_0), Objective function value of S_0 (E_0)
Initialize the set of unscheduled jobs (U=1,…,n)
Initialize the set of unscheduled jobs that can be processed on machine i (B_i⊂U)
t_i←0; CST←0; h_i←0; exit1←0; T_j←0;
While exit1==0
      exit2←0;
      While exit2==0
            For i=1 to m
            Select the machine i* with the minimum ti value;
            End
            If B_i*=={}
                  t_i*←BM;
            Else
                  exit2 ←1;
            End
      End
      For j=1 to n /identifying, the job j* will be assigned to machine i*/
            Select the job j* with highest I_i*j values from the jobs in B_i* ;
      End
      For i=1 to m /re-select machine i**/
            Re-select the machine i** with the minimum C_i**j* value;
            (C_i**_j*=min{max(t_i,CST)+p_ij*+ s_hj*})
      End
      /Assign job j* to the next available position on machine i**/
      x←max(CST, t_i**); CST←x+s_hj* ; t_i**←x+s_hj*+ p_i**j* ;
      U←U\{j*};B_i*←B_i*\{j*}; h_i**←j*; T_j*←max(t_i**- d_j, 0);∑w_jT_j= ∑w_jT_j+w_j*T_j*;
      If U=={}
            exit1←1;
      End
End
```

Fig. 2. Steps of the modified ATCS rule.

Table 3
Application of the modified ATCS rule.

		i^*	B_i	$maxI_{i^*j}$	j^*	C_{j^*}	i^{**}	CST	$t_{i^{**}}$	$h_{i^{**}}$	$w_{j^*}T_{j^*}$
Initial	Machine1	–	U	–	–	–	–	0	0	0	–
	Machine 2	–	U\{7}	–	–	–	–	0	0	0	–
1	Machine 1	√	U	0.185	2	33	√	0	33	2	0
	Machine 2		U\{7}			57			0	0	–
2	Machine 1		U\{2}			106		0	33	2	–
	Machine 2	√	U\{7,2}	0.173	1	25	√		25	1	0
3	Machine 1		U\{2,1}			117		34	33	2	–
	Machine 2	√	U\{7,2,1}	0.036	9	53	√		53	9	0
4	Machine 1	√	U\{2,1,9}	0.020	7	143	√	43	143	7	329
	Machine 2		U\{7,2,1,9}			–			53	9	–
5	Machine 1		U\{2,1,9,7}			195		65	143	7	–
	Machine 2	√	U\{7,2,1,9,}	0.022	4	112	√		112	4	0
6	Machine 1		U\{2,1,9,7,4}			201		119	143	7	–
	Machine 2	√	U\{7,2,1,9,4}	0.045	10	152	√		152	10	212
7	Machine 1	√	U\{2,1,9,7,4,10}	0.054	8	185	√	153	185	8	441
	Machine 2		U\{7,2,1,9,4,10}			215			152	10	–
8	Machine 1		U\{2,1,9,7,4,10,8}			287		165	185	8	–
	Machine 2	√	U\{7,2,1,9,4,10,8}	0.008	6	259	√		259	6	584
9	Machine 1	√	U\{2,1,9,7,4,10,8,6}	0.048	5	231	√	192	231	5	490
	Machine 2		U\{7,2,1,9,4,10,8,6}			352			259	6	–
10	Machine 1	√	U\{2,1,9,7,4,10,8,6,5}	0.003	3	325	√	241	325	3	223
	Machine 2		U\{7,2,1,9,4,10,8,6,5}			332			259	6	
			U = {}								Total:2279

unscheduled jobs on machine i^*. $\bar{s}$ denotes the average setup time. The values of the parameters k_1 and k_2 depend on the problem instance and are scale related. Equations for properly selecting k_1 and k_2 values are given in Section 4 Eqs. (28)–(30). If the I_{i^*j} or t_{i^*} values are equal, the algorithms prefer the job (machine) with the smaller index value. The proposed modified ATCS algorithm is given in Fig. 2.

An Illustrative Example of the ATCS Rule

The parameters of the example are given in Table 2. The scaling parameters k_1 and k_2 are set to 4.7 and 0.48, respectively, for the example based on the method of Lee et al. (1997) described in Section 4 Eqs. (28)–(30). The application of the proposed algorithm is given in Table 3. According to Table 3, column i^* shows the first available machine with the minimum t_i value. B_i is the set of unscheduled jobs that can be processed on machine i. I_{i^*j} values are calculated, and the maximum value is given in column max I_{i^*j}. The job with the max I_{i^*j} value is shown in column j^*. The completion time of job j^* for each machine is given in column C_{j^*}. As a result, job j^* is assigned to machine i^{**} based on the minimum completion time, and the CST, $t_{i^{**}}$, $h_{i^{**}}$ and w_jt_j values are updated in the related columns. For example, jobs 2, 7, 8, 5 and 3 are assigned to machine 1, and jobs 1, 9, 4, 10 and 6 are assigned to machine 2. According to this schedule, the total weighted tardiness is 2279. The optimal value for the problem is 2029, and the Gantt chart of the solution is given in Fig. 1.

3.2. Tabu Search

The TS algorithm is a successful algorithm used in many optimization problems (Güngör and Ünler, 2008). It was first suggested by Glover (1986). However, the TS algorithm can cycle. To prevent cycling, TS utilizes diversification/intensification strategies. The TS algorithm uses a memory structure to escape local optima (Sarıçiçek and Çelik, 2011). The basic TS algorithm has a short-term memory structure, and a long-term memory structure can increase the success of the algorithm (Özpeynirci et.al., 2016). In

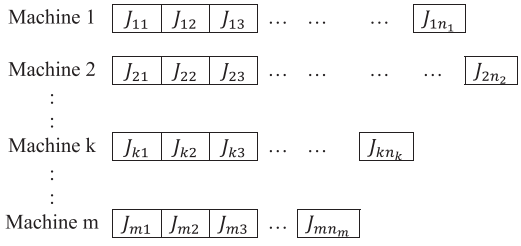

Fig. 3. Representation of the solution.

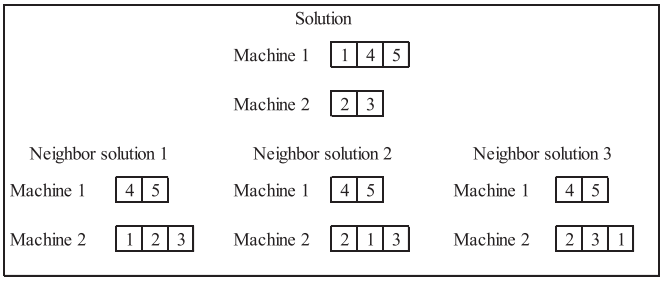

Fig. 4. An example of a candidate list strategy.

the unrelated machine scheduling literature, Glass et al. (1994) and Sels et al. (2015) proposed TS-based heuristic algorithms for the scheduling of unrelated parallel machines. They considered the problem of $R_m||C_{max}$. Bozorgirad and Logendran (2012) considered the problem of scheduling unrelated parallel machines with sequence- and batch-dependent setups and release dates. They proposed a TS-based heuristic algorithm. Kim and Shin (2003) proposed a TS algorithm for the problem of unrelated parallel machine scheduling with sequence-dependent setups and release dates.

The proposed TS algorithm for the problem of $R_m|ST_{sd}$, S_1, $M_j|\Sigma w_jT_j$ is given in this section.

```
Procedure decoding algorithm
Input: schedule, p_ij, s_hj, d_j and w_j
Output: Objective function value of the schedule
Initialize the α_ij values (If α_ij=0, the tardiness of job j on machine i is
calculated; if α_ij=1, the tardiness of job j is not calculated)
t_i←0; CST←0; h_i←0; exit1←0;
While exit1==0
      exit2←0;
      while exit2==0
            For i=1 to m
                  Select the machine i* with the minimum t_i value;
            End
            If ∑_j α_i*_j==0
                  t_i←BM;
            Else
                  exit2←1;
            End
      End
      /Schedule the first job j* with the value α_i*j=1/
      T_j*←max(t_i*-d_j*, 0);
      x←max(CST, t_i*); CST←x+s_hj*; t_i*←x+s_hj*+p_i*j*; h_i*←j*; α_i*j* ←0;
      If ∑_i ∑_j α_ij==0
            exit1←1;
      End
End
```

Fig. 5. Steps of the decoding algorithm.

Table 4
Application of the decoding algorithm.

		i^*	j^*	CST	t_{i^*}	h_{i^*}	$\sum_i \sum_j \alpha_{ij}$.	w_jT_j
Initial	Machine 1	–	–	0	0	0	4	
	Machine 2	–	–		0	0	6	
1	Machine 1	√	2	0	33	2	3	–
	Machine 2				0	0	6	
2	Machine 1			0	33	2	3	
	Machine 2	√	1		25	1	5	–
3	Machine 1			34	33	2	3	
	Machine 2	√	9		53	9	4	–
4	Machine 1	√	8	51	83	8	2	–
	Machine 2				53	9	4	
5	Machine 1			65	83	8	2	
	Machine 2	√	4		112	4	3	–
6	Machine 1	√	5	90	129	5	1	–
	Machine 2				112	4	3	
7	Machine 1			119	129	5	1	
	Machine 2	√	10		152	10	2	212
8	Machine 1	√	7	140	240	7	0	1008
	Machine 2				152	10	2	
9	Machine 1			164	BM	7	0	
	Machine 2	√	6		258	6	1	580
10	Machine 1			265	BM	7	0	
	Machine 2	√	3		331	3	0	229
	$\sum_i \sum_j \alpha_{ij}$=0						$\Sigma w_jT_j = 2029$	

```
Procedure TS algorithm with long-term memory
Input: p_ij, b_ij, s_hj, d_j, w_j, v, g, MTLS
Output: Near optimal solution
E_best←BM; iterlong←0; g[i][j]←0;  n[h][j]←0;
While iterlong<g
      Obtain S_0 and E_0 using the ATCS rule;
      iterlong←iterlong+1; TLL←1; itershort←1;
      s_hj←s_hj-n[h][j]; p_ij←p_ij-g[i][j]; Reset g[i][j] and n[h][j] matrices;
      S_c←S0; E_c←E_0;
            If  E_0<E_best
                  S_best←S_0 and E_best←E_0;
            End
      While itershort<v
            Generate neighbor solutions and sort ascending order according
            to obj. func.value (S_n^t); t←1; check←0;
            While check==0
                  If the movement of S_n^t not tabu or E_n^t<E_best
                        S_c←S_n^t; E_c←E_n^t;
                        Insert the movement of S_c at the bottom of tabu list;
                        TLL←TLL+1; Update n[h][j] and g[i][j]; check←1;
                  Else
                        t←t+1;
                  End
            End
            If TLL==MTLS+1
                  Delete the first element in the tabu list;
                  TLL←TLL-1;
            End
            If E_c<E_best
                  S_best←S_c; E_best←E_c;
            End
      itershort←itershort+1;
      End
      s_hj←s_hj+n[h][j]; p_ij←p_ij+g[i][j];
End
```

Fig. 6. Steps of the proposed TS.

```
Procedure SA algorithm
Input: p_ij, b_ij, s_hj, d_j, w_j, T_0, T_final, q, IT, S_0, E_0
Output: Near optimal solution
E_best←E_0; E_c←E_0; S_best←S_0; S_c←S_0; T_c←T_0;
While T_c<T_final
      il←0;
      While il<IT
            Generate the neighboring solutions using all six methods and
            determine the neighboring solution S_n with the best objective
            function value E_n;
            il←il+1;
            Δ←E_n-E_c;
            If (Δ<0) or (Δ≥0 and random(0-1)<exp(-Δ/T_c))
                  E_c←E_n and S_c←S_n;
            End
            If  E_c<E_best
                  S_best←S_0 and E_best←E_0;
            End
      End
      T_c ← T_c*q;
End
```

Fig. 7. Steps of the proposed SA.

Table 5
Distribution/value of the parameters.

Parameters	Distribution/Value	Number of alternatives
m	*2, 3, 5, 7 and 10*	*5*
n	*5*m and 10*m*	*2*
p_{ij}	*U(10,100)*	*1*
w_j	*U(1,10)*	*1*
s_{hj}	*U(5, 25), U(5, 50) and U(25, 50)*	*3*
b_{ij}	*The machine is eligible with a probability of 1* *The machine is eligible with a probability of 0.7*	*2*
τ, R	$\tau = 0.5$ *and* 0.8, $R = 0.2$	*2*

3.2.1. Solution Representation

In the solution representation, the jobs assigned to each machine are shown as a string. The representation of the solution is given in Fig. 3. J_{ik} denotes the index of the job assigned to machine i and with sequence k. n_i denotes the number of jobs allocated to machine i.

An example is given below for 7 jobs and 2 machines.

Machine 1: 7-1-5

Machine 2: 4-6-3-2

In this example, jobs 7, 1, and 5 are assigned to machine 1, and jobs 4, 6, 3, and 2 are assigned to machine 2. With the relevant representation, each job is assigned to one machine based on a determined sequence.

3.2.2. Neighborhood generation

Two methods are used when a neighbor solution is derived. In the insertion method, a randomly selected job precedes the randomly selected job, and the neighbor solution is created (Bilge et al., 2004). In the swap method, two randomly selected jobs replace each other.

Because the server, sequence-dependent setup times, unrelated parallel machines and machine eligibility restrictions are taken into account, the calculation of the total weighted tardiness for a given move is a tedious task. A candidate list strategy is used to restrict the number of neighbor solutions. In this study, the candidate list strategy is used for only insertion moves, and all swap moves are generated. The candidate list strategy is explained below.

First, the contribution of each machine i to the total weighted tardiness is calcualted as $\sum_{j \in I_i} w_j T_j$, where I_i is the set of jobs scheduled on machine i.

Next, the machine with the highest contribution to the total weighted tardiness is selected and denoted as machine f^T. Subsequently, the machine with the lowest contribution to the total weighted tardiness is selected and denoted as f^E. Every job on machine f^T is consdiered for insertion on machine f^E.

An example of a candidate list strategy is given in Fig. 4 for 5 jobs and 2 machines. In this case, we assume that $f^T$is machine 1 and f^E is machine 2. Every job (job 1, job 4 and job5) on machine 1 is considered for insertion on machine 2. We selected job 1 for

Table 6
Parameters of the test problems.

No	m	n	τ	s_{ij}	b	No	m	n	τ	s_{ij}	b	No	m	n	τ	s_{ij}	b	No	m	n	τ	s_{ij}	b
1				U(5,25)	1	25				U(5,25)	1	49				U(5,25)	1	73				U(5,25)	1	97				U(5,25)	1
2					0,7	26					0,7	50					0,7	74					0,7	98					0,7
3			0,5	U(5,50)	1	27			0,5	U(5,50)	1	51			0,5	U(5,50)	1	75			0,5	U(5,50)	1	99			0,5	U(5,50)	1
4					0,7	28					0,7	52					0,7	76					0,7	100					0,7
5				U(25,50)	1	29				U(25,50)	1	53				U(25,50)	1	77				U(25,50)	1	101				U(25,50)	1
6		10			0,7	30		15			0,7	54		25			0,7	78		35			0,7	102		50			0,7
7				U(5,25)	1	31				U(5,25)	1	55				U(5,25)	1	79				U(5,25)	1	103				U(5,25)	1
8					0,7	32					0,7	56					0,7	80					0,7	104					0,7
9			0,8	U(5,50)	1	33			0,8	U(5,50)	1	57			0,8	U(5,50)	1	81			0,8	U(5,50)	1	105			0,8	U(5,50)	1
10					0,7	34					0,7	58					0,7	82					0,7	106					0,7
11				U(25,50)	1	35				U(25,50)	1	59				U(25,50)	1	83				U(25,50)	1	107				U(25,50)	1
12	2				0,7	36	3				0,7	60	5				0,7	84	7				0,7	108	10				0,7
13				U(5,25)	1	37				U(5,25)	1	61				U(5,25)	1	85				U(5,25)	1	109				U(5,25)	1
14					0,7	38					0,7	62					0,7	86					0,7	110					0,7
15			0,5	U(5,50)	1	39			0,5	U(5,50)	1	63			0,5	U(5,50)	1	87			0,5	U(5,50)	1	111			0,5	U(5,50)	1
16					0,7	40					0,7	64					0,7	88					0,7	112					0,7
17				U(25,50)	1	41				U(25,50)	1	65				U(25,50)	1	89				U(25,50)	1	113				U(25,50)	1
18		20			0,7	42		30			0,7	66		50			0,7	90		70			0,7	114		100			0,7
19				U(5,25)	1	43				U(5,25)	1	67				U(5,25)	1	91				U(5,25)	1	115				U(5,25)	1
20					0,7	44					0,7	68					0,7	92					0,7	116					0,7
21			0,8	U(5,50)	1	45			0,8	U(5,50)	1	69			0,8	U(5,50)	1	93			0,8	U(5,50)	1	117			0,8	U(5,50)	1
22					0,7	46					0,7	70					0,7	94					0,7	118					0,7
23				U(25,50)	1	47				U(25,50)	1	71				U(25,50)	1	95				U(25,50)	1	119				U(25,50)	1
24					0,7	48					0,7	72					0,7	96					0,7	120					0,7

Table 7
SA parameter levels.

Factors	Levels
Initial temp. (T_0)	100, 150, 200, 250 and 300
Number of neighbor solutions (IT)	100, 150, 200, 250 and 300
Cooling rate (q)	0.99,0.95, 0.90, 0.85 and 0.80

the example. Additionally, jobs 4 and 5 must be inserted. The total number of neighbor solutions must be 9.

3.2.3. Decoding algorithm

When a current solution is given, the following algorithm is used to calculate the objective function value. Using this algorithm, the common server can process one setup at a given time. The algorithm that is used to calculate the objective function value of the solutions is given in Fig. 5.

An illustrative example of the decoding algorithm

In this example, 10 jobs are scheduled on two machines. The parameters of the problem are given in Table 2.

The corresponding schedule is jobs 2, 8, 5 and 7 on machine 1 and jobs 1, 9, 4, 10, 6 and 3 on machine 2 (see Fig. 1). The application of the proposed algorithm is given in Table 4. In Table 4, column i^* shows the machine with the minimum t_i value. The first unscheduled job on machine i^* is shown in column j^*. Job j^* is scheduled on machine i^* and the corresponding *CST*, t_{i^*}, h_{i^*} and w_jT_j values are updated. The number of unscheduled jobs for each machine is given in column $\sum_i \sum_j \alpha_{ij}$. According to the algorithm, the objective function value of the example is 2029.

3.2.4. Tabu classification

If the neighboring solution is the result of an insertion movement, the inserted job is added to the tabu list. Movement of the related job is prohibited if it is on the tabu list. If the neighboring solution is obtained as the result of a swap movement, the swap movement of these jobs is prohibited if the jobs are on the tabu list.

The tabu list length is assumed to be constant. When the tabu list is full, the movement that is on the tabu list for the longest time is deleted. The objective criterion is to obtain a better solution than the previous best solution. In this situation, tabu is not considered, and the obtained solution is accepted.

3.2.5. Long-term strategy

A long-term strategy is used for diversification. In long-term memory, a search of unvisited regions occurs. In this phase, the processing times and setup times of the operations are updated considering frequency-based memory, and the algorithm starts with a different initial solution using the modified ATCS rule. Therefore, the search space is explored. The frequency-based memory records the number of times that operation j is assigned to machine i and that operation j is processed after job h using array $g[i][j]$ for the processing times and array $n[h][j]$ for the sequence-dependent setup times. When operation j^* is assigned to machine i^*, $g[i^*][j^*]$ is increased by one. When operation j^* is processed after job h^*, we increase $n[h^*][j^*]$ by one. When the short-term memory is terminated, the processing time of operation j on machine i and the sequence-dependent setup time of operation j after operation h are updated by adding the frequency of the machine-operation pair to the processing time, i.e., $p_{ij} = p_{ij} + g[i][j]$, and adding the frequency of the setup time of job j after job h on the same machine, i.e., $s_{hj} = s_{hj} + n[h][j]$. Then, a new initial solution is found with the updated p_{ij} and s_{hj} values, and the short-term memory process is restarted. A similar strategy was used by Özpeynirci et al. (2016).

Table 8
TS parameter levels.

Factors	Levels
The iteration number for the short-term memory (v)	50, 100,150, 200 and 250
The iteration number for the long-term memory (g)	10, 20, 30, 40 and 50
Tabu list length (TLL)	5, 10,15, 20 and 25

Table 9
L25 orthogonal array of the Taguchi experimental design.

Exp. no.	Initial temperature	Number of neighbor solutions	Cooling rate	Average calculated obj. func. values
1	100	100	0.99	75,462
2	100	150	0.95	78,322
3	100	200	0.9	78,927
4	100	250	0.85	80,162
5	100	300	0.8	79,955
6	150	100	0.95	78,473
7	150	150	0.9	79,254
8	150	200	0.85	80,098
9	150	250	0.8	80,224
10	150	300	0.99	75,710
11	200	100	0.9	80,375
12	200	150	0.85	78,762
13	200	200	0.8	79,292
14	200	250	0.99	75,534
15	200	300	0.95	76,566
16	250	100	0.85	80,375
17	250	150	0.8	81,472
18	250	200	0.99	75,050
19	250	250	0.95	75,978
20	250	300	0.9	76,368
21	300	100	0.8	81,676
22	300	150	0.99	75,879
23	300	200	0.95	75,399
24	300	250	0.9	77,049
25	300	300	0.85	78,183

Table 10
Determined SA and TS Parameters.

Test problem (Machine number × job number)	SA			TS		
	T_0	IT	q	v	g	TLL
2 × 10	150	150	0.99	50	10	5
2 × 20	150	150	0.99	100	10	5
3 × 15	300	300	0.99	100	10	5
3 × 30	300	300	0.99	150	10	10
5 × 25	300	300	0.99	200	20	10
5 × 50	300	300	0.99	150	20	5
7 × 35	300	300	0.99	150	20	10
7 × 70	300	300	0.99	200	30	10
10 × 50	300	300	0.99	200	40	15
10 × 100	300	300	0.99	200	20	20

Before describing the TS algorithm, the abbreviations used in the algorithm are given as follows.

MTLS: Maximum tabu list size
v: Maximum iteration number for the short-term memory
g: Maximum iteration number for the long-term memory
TLL: Tabu list length
iterlong: Iteration counter for the long-term memory
itershort: Iteration counter for the short-term memory
$E_{c(n)(0)}$*:* Objective function value of the current (neighbor) (initial) solution
$S_{0(c)}$*:* Initial (current) solution
$S_n{}^t$*:* tth neighbor solution
The steps of the proposed TS are given in Fig. 6.

3.3. Simulated annealing algorithm

The SA algorithm was proposed by Kirkpatrick et al. (1983). To escape local optima, the SA algorithm accepts solutions that do not improve the objective function value based on a certain probability (Kim et al. 2002). The performance of the SA is influenced by certain factors, such as the initial temperature (T_0), final temperature (T_{final}), cooling rate (q) and number of iterations at each temperature (IT). In previous studies of unrelated parallel machine scheduling, Kim et al. (2002) proposed an SA algorithm for the problem of $R_m|ST_{sd,b}|\Sigma T_j$. They considered sequence- and batch-dependent setup times. Chang and Chen (2011) and Lee et.al. (2013), Chen (2009) considered the problem of unrelated parallel machine scheduling with sequence- and machine-dependent setups and proposed SAs. The proposed SA for the problem of $R_m|ST_{sd}$, S_1, $M_j|\Sigma w_j T_j$ is given in this section.

3.3.1. Neighbor generation

The solution representation is the same as that for the TS algorithm given in Section 4.2.1. Six methods are used to create neighborhood structures.

Table 11
Test problems with 2 machines and 10 or 20 jobs.

No	MILP			ATCS		SA I		TS I		SA II		TS II	
	Solution		CPU	Solution	CPU	Solution	CPU	Solution	CPU	Solution	CPU	Solution	CPU
1	922	*	18,522	1162	0.001	**922**	1.15	**922**	0.02	**922**	1.13	**922**	0.02
	3087	*	100,800	3548	0.001	**3087**	1.15	**3087**	0.02	**3087**	1.13	**3087**	0.02
2	1332	*	17,340	1834	0.001	**1332**	1.15	**1332**	0.02	1341	1.13	**1332**	0.02
	1815	*	14,256	2443	0.001	**1815**	1.15	**1815**	0.02	**1815**	1.13	1902	0.02
3	1007	*	12,780	1332	0.001	**1007**	1.15	**1007**	0.03	**1007**	1.13	1409	0.02
	2021	*	18,136	2604	0.001	**2021**	1.15	**2021**	0.03	**2021**	1.13	2123	0.02
4	1616	*	995.8	2177	0.001	**1616**	1.15	**1616**	0.03	1656	1.13	1735	0.02
	3337	*	3044	5747	0.001	**3337**	1.15	**3337**	0.03	3420	1.13	3420	0.02
5	1244	*	11,680	1739	0.001	**1244**	1.15	**1244**	0.03	**1244**	1.13	1289	0.02
	1658	*	35,192	3076	0.001	**1658**	1.15	**1658**	0.03	1741	1.13	1792	0.02
6	3647	*	10,252	4463	0.001	**3647**	1.15	**3647**	0.03	**3647**	1.13	4122	0.02
	789	*	1149	1427	0.001	**789**	1.18	**789**	0.03	**789**	1.13	**789**	0.02
7	3437	*	41,160	3675	0.001	**3437**	1.18	**3437**	0.29	**3437**	1.15	**3437**	0.16
	4361	*	38,255	5391	0.001	**4361**	1.19	**4361**	0.32	**4361**	1.15	4628	0.15
8	4663	*	1031	4948	0.001	**4663**	1.18	**4663**	0.13	**4663**	1.15	4760	0.07
	3989	*	10,993	4455	0.001	**3989**	1.19	**3989**	0.15	**3989**	1.15	4152	0.05
9	3460	*	33,780	4266	0.001	**3460**	1.19	**3460**	0.27	**3460**	1.15	3744	0.16
	4143	*	60,498	5129	0.001	**4143**	1.2	**4143**	0.32	**4143**	1.15	4412	0.12
10	4722	*	7951	6030	0.001	**4722**	1.18	**4722**	0.2	4827	1.15	4777	0.07
	4189	*	13,554	5990	0.001	**4189**	1.18	**4189**	0.2	**4189**	1.15	**4189**	0.16
11	4865	*	46,022	7395	0.001	**4865**	1.18	**4865**	0.27	**4865**	1.15	5165	0.16
	5463	*	56,045	7317	0.001	**5463**	1.18	**5463**	0.32	**5463**	1.15	**5463**	0.16
12	6049	*	5968	7246	0.001	**6049**	1.18	**6049**	0.19	**6049**	1.15	6206	0.08
	4967	*	24,622	6332	0.001	**4967**	1.17	**4967**	0.21	**4967**	1.15	5321	0.07
13	10,236		18,000	5380	0.001	3996	3.11	**3745**	3.33	4069	3.02	4476	2.27
	13,307		18,000	8368	0.001	**5661**	3.16	**5661**	2.29	**5661**	3	**5661**	2.56
14	3370		18,000	4207	0.001	2161	3.1	**1991**	2.27	2173	3	2618	1.21
	7481		18,000	6651	0.001	**3682**	3.15	3699	3.33	3852	3	3954	1.28
15	12,803		18,000	7930	0.001	4461	3.12	**4361**	3.33	4801	3.01	5500	2.25
	14,298		18,000	7386	0.001	4751	3.12	**4490**	3.33	4652	3.01	4852	2.25
16	11,703		18,000	11,993	0.001	5723	3.12	**5195**	2.25	5723	3.01	5959	1.24
	14,848		18,000	6212	0.001	3212	3.16	**2825**	2.85	3212	3.01	3212	1.52
17	18,880		18,000	12,929	0.001	8681	3.11	**8037**	3.33	9069	3.01	9084	2.22
	13,584		18,000	8982	0.001	6648	3.11	**6383**	2.85	6852	3.01	6852	2.85
18	10,790		18,000	8352	0.001	**5391**	3.1	5407	2.27	5722	3.01	6107	1.21
	15,041		18,000	8265	0.001	6156	3.11	**5901**	2.36	6100	3.01	6420	1.41
19	19,357		18,000	12,166	0.001	11,428	3.37	**11,309**	3.33	11,420	3.21	11,666	2.27
	19,119		18,000	15,203	0.001	**10,463**	3.2	**10,463**	3.52	**10,463**	3.21	10,995	2.52
20	24,689		18,000	15,611	0.001	13,876	3.36	**13,523**	2.26	14,414	3.21	13,821	1.11
	17,541		18,000	14,252	0.001	12,123	3.36	**10,922**	2.12	14,521	3.21	15,620	1.41
21	21,431		18,000	11,263	0.001	10,510	3.37	**10,106**	3.33	10,582	3.21	10,393	2.25
	19,576		18,000	14,863	0.001	**10,376**	3.37	**10,376**	3.33	**10,376**	3.21	12,526	2.52
22	22,635		18,000	20,579	0.001	19,888	3.37	**19,059**	2.25	19,740	3.21	19,905	1.41
	22,618		18,000	18,363	0.001	**12,571**	3.37	**12,571**	2.2	13,205	3.21	13,205	1.32
23	26,351		18,000	18,869	0.001	15,509	3.37	**14,543**	3.32	16,414	3.21	14,999	2.28
	29,222		18,000	25,282	0.001	19,313	3.37	**18,006**	2.25	19,313	3.21	21,365	2.16
24	–		18,000	20,365	0.001	18,103	3.37	**16,786**	2.25	17,589	3.21	16,975	1.19
	22,458		18,000	20,734	0.001	18,053	3.37	**17,066**	2.25	18,852	3.21	18,005	1.19

* optimal solution

Method 1: The machine with the largest total weighted tardiness is identified. A job randomly selected from this machine is assigned to the machine with the smallest total weighted tardiness, and the sequence of the job is randomly determined. If the job specified by the machine with the largest total weighted tardiness is not eligible on the machine with the smallest total weighted tardiness, then a new job from the machine with the largest total weighted tardiness is randomly selected (insertion).

Method 2: The machine with the largest total weighted tardiness is identified. The job with the largest total weighted tardiness from this machine is assigned to the machine with the smallest total weighted tardiness. The sequence of the job is randomly determined. If the machine with the smallest total weighted tardiness is not capable of processing the job, then the job is assigned to the machine with the second smallest total weighted tardiness (insertion).

Method 3: A randomly selected job is assigned to a randomly selected sequence on a randomly selected machine. Machine eligibility restrictions are taken into account when selecting the machine to which the job will be assigned (insertion).

Method 4: Two randomly selected jobs are swapped. The jobs can be on different machines or on the same machine. If jobs on different machines are swapped, then machine eligibility restrictions are taken into account (swap).

Method 5: A randomly selected job from the machine with the largest total weighted tardiness and a randomly selected job from the machine with the smallest total weighted tardiness are swapped. If the related job cannot be processed on the machine with the largest or smallest total weighted tardiness, then a new job is randomly selected for swapping (swap).

Method 6: Two randomly selected jobs are swapped on the machine with the largest total weighted tardiness (swap).

Table 12
Test problems with 3 machines and 15 or 30 jobs.

No	MILPM		ATCS		SA I		TS I		SA II		TS II	
	Solution	CPU	Solution	CPU	Solution	CPU	Solution	CPU	Solution	CPU	Solution	CPU
25	4757	18,000	3459	0.003	1908	8.825	**1889**	1.14	1955	7.71	**1889**	1.11
	2585	18,000	3702	0.003	2273	8.8	**2062**	1.48	2273	7.71	2415	1.11
26	2649	18,000	2080	0.003	1350	7.79	**1305**	0.86	1326	7.71	1388	0.07
	2833	18,000	2833	0.003	**2466**	7.71	**2466**	1.12	2491	7.71	2652	0.05
27	9044	18,000	8876	0.003	3746	7.77	**3669**	1.13	3746	7.7	3964	1.12
	2864	18,000	2722	0.003	2110	7.77	**1938**	0.98	2315	7.52	2563	1.51
28	6034	18,000	5439	0.003	3359	7.77	**2619**	0.95	3452	7.52	**2619**	0.07
	2524	18,000	3786	0.003	2269	7.74	**2193**	0.85	2269	7.45	2352	0.07
29	8118	18,000	6256	0.003	4961	7.79	**3874**	1.13	4990	7.52	4021	1.15
	5859	18,000	5124	0.003	4236	7.79	**3701**	1.12	4428	7.12	**3701**	1.19
30	6977	18,000	7206	0.003	5397	7.79	5290	0.93	5563	7.52	**5226**	0.07
	4815	18,000	5796	0.003	4092	7.72	**3826**	0.85	4150	7.52	4092	0.05
31	7654	18,000	5893	0.003	5385	7.77	**5149**	1.13	5452	7.45	5345	1.15
	7626	18,000	5527	0.003	4455	7.71	**4275**	1.58	4352	7.42	4585	1.19
32	6814	18,000	6477	0.003	6139	7.77	**6025**	0.94	6172	7.42	6139	0.08
	5521	18,000	6831	0.003	4544	7.7	**4334**	0.85	**4334**	7.41	4359	0.07
33	9239	18,000	8188	0.003	6330	7.77	**6095**	1.13	6215	7.41	6251	1.17
	7410	18,000	6682	0.003	**5589**	7.77	5615	1.13	5615	7.41	5689	1.15
34	10,342	18,000	9390	0.003	5452	7.78	**5297**	1.11	5895	7.4	5955	0.08
	6715	18,000	7289	0.003	6175	7.45	**5902**	1.18	6175	7.52	6320	0.07
35	11,536	18,000	10,013	0.003	6997	8.8	**6793**	1.13	**6793**	7.4	7507	1.17
	7780	18,000	7913	0.003	7262	8.84	**6874**	1.17	7302	7.4	6910	1.12
36	16,038	18,000	14,325	0.003	8575	8.83	**8556**	0.92	8686	7.4	**8556**	0.08
	10,701	18,000	10,049	0.003	9369	8.83	9033	0.85	**8963**	7.4	**9396**	0.05
37	–	18,000	8511	0.003	5191	18.18	**4775**	24.2	5191	16.12	5806	12.12
	–	18,000	12,432	0.003	**6799**	18.18	6829	20.2	6985	16.12	6892	12.52
38	–	18,000	8855	0.003	7186	18.18	**6635**	17.2	7879	16.12	8203	9.93
	–	18,000	9577	0.003	6571	18.17	**5675**	18.5	6505	16.12	6052	9.52
39	–	18,000	12,432	0.003	**8580**	18.18	9240	24.2	10,361	16.12	10,803	12.11
	–	18,000	11,953	0.003	8041	18.17	**7050**	15.5	8252	16.12	7452	12.41
40	–	18,000	20,790	0.003	13,999	18.18	**11,843**	11.1	14,219	15.12	13,850	9.55
	–	18,000	19,552	0.003	12,522	18.15	**11,885**	12.2	14,520	15.12	14,520	8.52
41	–	18,000	19,686	0.003	17,171	18.18	**14,960**	23.2	15,696	15.15	18,029	12.16
	–	18,000	26,943	0.003	18,165	18.18	**15,453**	12.5	17,852	15.12	20,520	12.41
42	–	18,000	21,472	0.003	**13,287**	18.18	13,706	17.2	13,907	15.15	15,810	9.91
	–	18,000	25,183	0.003	20,598	18.18	**16,541**	18.5	22,541	15.1	23,584	9.94
43	–	18,000	19,711	0.003	15,873	22.21	**15,663**	24.2	16,185	21.2	17,859	12.11
	–	18,000	21,831	0.003	19,300	22.15	**18,223**	28.5	20,584	21.21	22,541	11.12
44	–	18,000	20,294	0.003	17,965	18.18	17,380	15.2	**17,035**	21.21	17,459	9.88
	–	18,000	21,057	0.003	16,874	20.15	**15,900**	25.1	17,873	21.41	16,879	9.45
45	–	18,000	20,513	0.003	**17,934**	18.18	**17,934**	24.2	19,627	20.12	18,303	12.15
	–	18,000	25,664	0.003	24,542	18.1	23,724	25.1	**23,452**	20.12	26,852	12.15
46	–	18,000	21,151	0.003	**19,334**	18.18	**19,334**	16.2	19,629	22.21	20,663	9.41
	–	18,000	30,545	0.003	24,710	18.12	**23,595**	20.5	25,852	21.12	25,965	9.41
47	–	18,000	47,271	0.003	**37,705**	18.18	37,993	24.1	39,918	22.21	44,145	12.01
	–	18,000	45,241	0.003	**37,391**	18.18	38,523	24.1	38,985	22.52	40,528	12.14
48	–	18,000	56,801	0.003	42,431	18.18	**42,389**	11.1	42,429	22.21	42,895	9.41
	–	18,000	40,560	0.003	32,557	18.18	**30,143**	15.1	34,852	21.12	34,985	9.41

All of the methods used invole basic swap and insertion. To obtain better neighboring solutions more quickly, more intelligent approaches can be considered. The algorithm in this study generates solutions according to the six methods and selects the best solution as a neighbor solution.

3.3.2. Steps of the SA

The steps of the proposed SA are given in Fig. 7. In Fig. 7, $T_{c(0)}$ is the current (initial) temperature, and *il* is the iteration counter.

The decoding algorithm is used to calculate the objective function value of the solutions as in Section 4.2.3.

4. Computational experiments

4.1. Test Problems

The number of machines (m) is 2, 3, 5, 7 and 10. The number of jobs depends on the number of machines and is taken as 5^*m and 10^*m. The processing times are derived from a uniform distribution between 10 and 100. Setup times are considered at three levels: U(5, 25), U(5, 50) and U(25, 50) (Huang et al., 2010). Due dates are generated according to Lee et al. (1997). The due date tightness factor (τ) and due date range factor (R) have been used by researchers to determine the due date of a job. τ is defined as $\tau = 1 - \bar{d}/C_{max}$, where $\bar{d}$ is the average due date and C_{max} is the estimated makespan. A large value of τ indicates tight due dates, and small value of τ indicates loose due dates. R is defined as $R = (d_{max} - d_{min})/C_{max}$, where d_{max} and d_{min} are the maximum and minimum due dates, respectively. R provides the measure of variability of the due dates. The higher the value of R is, the larger the range of due dates generated. Because dates can be generated from a composite uniform distribution based on R and τ, with the probability τ of the due date uniformly distributed in the interval $[\bar{d} - R\bar{d}, \bar{d}]$ and the probability $(1-\tau)$ in the interval $[\bar{d}, \bar{d} + (Cmax - \bar{d})R]$. τ, $\bar{d}$, and C_{max} are related by $\tau = 1 - \bar{d}/C_{max}$. The τ value is taken as 0.5 and 0.8, and the value of R is 0.2 (Bozorgirad and Logendran, 2012).

Table 13
Test problems with 5 machines and 25 or 50 jobs.

No	MILPM	ATCS		SA I		TS I		SA II		TS II	
	Solution	Solution	CPU	Solution	CPU	Solution	CPU	Solution	CPU	Solution	CPU
49	12,565	6866	0.03	3979	19.19	**3371**	22.21	4190	18.21	5130	18.17
	15,214	6257	0.03	5529	19.19	**4303**	24.25	5685	18.21	5874	18.18
50	15,482	6368	0.03	4091	19.19	**3523**	13.13	4095	18.18	3774	9.91
	13,379	6302	0.03	4734	19.19	**4320**	12.14	4885	18.14	5026	9.81
51	22,621	8165	0.03	5484	19.19	**5253**	24.2	7026	18.23	6566	18.12
	20,833	7956	0.03	5086	19.19	4886	20.21	**4852**	18.2	6052	18.52
52	20,771	11,029	0.03	5962	19.19	**5893**	13.11	6486	18.41	7275	9.12
	17,812	6519	0.03	5252	19.19	**4870**	12.12	5520	18.36	6024	8.15
53	–	13,813	0.03	11,250	19.19	**9832**	24.24	11,416	18.18	10,348	17.45
	28,055	17,295	0.03	13,111	19.19	**12,167**	20.1	13,521	18.18	13,582	16.45
54	–	13,532	0.03	9794	19.19	**9064**	16.16	9732	18.42	10,494	8.12
	22,423	17,567	0.03	14,080	19.19	**12,546**	12.12	14,080	18.18	16,528	7.45
55	20,495	14,844	0.03	10,469	18.18	**9250**	25.24	11,469	17.45	10,786	19.21
	28,441	9999	0.03	9077	19.18	**8547**	21.1	9252	17.52	9152	20.41
56	21,393	11,074	0.03	9246	18.18	**8955**	15.14	9280	17.42	9989	11.12
	12,190	10,449	0.03	6623	17.18	**6560**	12.14	6698	16.52	6785	11.45
57	–	12,473	0.03	9638	18.16	**8694**	25.25	11,206	17.21	10,803	18.12
	–	10,775	0.03	9605	18.16	**8540**	22.1	9952	17.26	9605	20.12
58	–	10,897	0.03	8551	18.16	**8183**	12.12	10,304	17.17	9475	10.12
	–	13,335	0.03	11,654	18.18	**10,665**	11.15	13,258	17.17	13,250	12.15
59	–	26,241	0.03	23,832	18.16	**22,872**	25.24	23,317	17.17	24,587	18.18
	–	26,310	0.03	21,399	18.18	**21,006**	21.2	21,987	17.41	23,528	20.2
60	–	28,209	0.03	22,221	18.16	**20,806**	13.13	22,881	17.17	22,125	11.1
	–	27,432	0.03	24,149	20.21	**23,373**	10.15	26,528	17.17	26,985	12.45
61	–	30,513	0.04	20,280	53.51	**18,902**	171.11	20,280	50.14	23,150	155.15
	–	20,120	0.04	15,306	55.26	**12,861**	164.12	15,985	49.85	16,852	157.45
62	–	19,866	0.04	17,369	53.53	**17,321**	93.93	19,823	50.54	21,837	82.82
	–	15,859	0.04	11,852	53.26	**10,520**	95.52	11,952	50.52	12,520	84.12
63	–	28,185	0.04	26,932	53.45	**19,632**	170.18	28,029	52.52	25,385	154.12
	–	28,182	0.04	**24,205**	50.2	26,478	156.52	26,852	52.52	27,852	156.2
64	–	40,026	0.04	24,534	52.12	**19,351**	88.84	23,624	53.52	26,724	80.81
	–	35,210	0.04	20,568	51.12	**17,458**	88.52	21,587	52.52	21,985	75.41
65	–	94,689	0.04	75,266	50.1	**73,722**	175.15	78,918	50.12	84,593	153.12
	–	88,730	0.04	66,354	50.12	**64,179**	188.52	66,052	49.52	65,852	155.42
66	–	99,689	0.04	78,515	53.52	**74,477**	99.52	80,540	50.1	85,257	80.81
	–	105,899	0.04	76,521	52.41	**75,002**	101.12	77,852	50.1	76,985	78.45
67	–	39,083	0.04	29,640	47.45	30,205	161.21	**28,425**	46.41	33,384	153.15
	–	27,654	0.04	26,369	45.45	**24,848**	152.52	26,054	48.41	28,985	156.41
68	–	42,715	0.04	**33,082**	47.47	34,520	90.95	35,104	46.12	33,690	84.45
	–	49,852	0.04	30,521	50.21	**28,521**	80.54	29,854	45.12	32,520	80.12
69	–	58,708	0.04	45,136	48.11	**43,520**	150.48	47,595	46.1	59,891	151.1
	–	46,827	0.04	38,699	48.12	**35,753**	120.52	41,528	44.16	40,521	156.52
70	–	55,278	0.04	44,568	47.45	**43,320**	88.14	48,392	45.45	59,834	82.12
	–	58,742	0.04	42,587	45.12	**40,258**	90.58	42,587	44.2	41,574	83.15
71	–	119,194	0.04	103,324	42.41	**100,809**	170.15	106,142	42.41	104,485	151.1
	–	119,349	0.04	**101,324**	41.18	101,382	162.58	115,285	42.41	125,241	152.41
72	–	132,049	0.04	107,939	41.12	**106,822**	89.42	108,798	40.12	113,470	84.12
	–	158,540	0.04	109,851	41.12	**105,140**	90.52	115,289	40.11	118,952	84.52

The value of C_{max} is necessary to generate *dj* values. The lower bound of C_{max} is determined by the following method.

The processing time of a job is equal to the minimum processing time. Because unrelated parallel machines are taken into account, job processing times vary according to the machines. For the lower bound, the job processing time is calculated by Eq. (25). According to this equation, the job processing time is equal to the minimum processing time of the job among all machines capable of processing that job (M': the machines that are capable of processing the job).

$$P_j{}^1 = \min_{i \in M'} P_{ij} \quad \forall j \tag{25}$$

Jobs have sequence-dependent setup times, and the setup time of job j (S_j) is calculated by Eq. (26) (Huang et al., 2010).

$$S_j = \min_{h \in J} s_{hj} \quad \forall j \tag{26}$$

S_j values are sorted in ascending order, and because setup is not required for the first jobs at each machine, the sum of the first $|J-M|$ setup times is deemed S^1 (J denotes the number of jobs, and M denotes the number of machines). The first two minimum p_{ij} values are p^1 and p^2.

The lower bound of the machine is $((\sum_j P_j{}^1)+S^1)/M$ (Huang et al., 2010), and the lower bound of the common server is $(S^1+p^1+p^2)$ because there is at least one job before the first setup and one job after the last setup. The lower bound of C_{max} is calculated by Eq. (27).

$$C_{max} = max\left\{\left(\left(\sum_j P_j^1\right)+S^1\right)/M, \left(S^1+p^1+p^2\right)\right\} \tag{27}$$

The following Eqs. (28)–(30) can be used to select the proper values of k_1 and k_2 for use in the modified ATCS rule (Lee et al., 1997):

$$k_1 = 4.5 + R \qquad for R \le 0.5 \tag{28}$$

$$k_1 = 6 - 2R \qquad for R \ge 0.5 \tag{29}$$

$$k_2 = \tau/2\sqrt{\eta} \; and \; \eta = \bar{s}/\bar{p} \tag{30}$$

Table 14
Test problems with 7 machines and 35 or 70 jobs.

No	ATCS		SA I		TS I		SA II		TS II	
	Solution	CPU	Solution	CPU	Solution	CPU	Solution	CPU	Solution	CPU
73	15,738	0.07	8388	31.32	**8236**	43.43	9628	35.23	11,095	41.41
	10,576	0.07	8377	29.31	**7141**	42.24	8574	35.12	10,520	40.41
74	16,274	0.07	8984	32.16	**8352**	21.2	9984	34.21	11,359	25.21
	9959	0.07	7634	30.3	**7406**	20.25	8452	33.12	9852	24.12
75	16,362	0.07	11,604	31.31	**10,188**	41.4	11,809	32.12	14,488	40.23
	13,627	0.07	10,973	31.2	**8771**	38.52	10,852	30.1	11,854	40.32
76	25,732	0.07	13,987	31.31	**12,628**	21.21	14,615	35.11	13,796	22.12
	18,305	0.07	12,063	29.2	**11,027**	20.12	11,854	38.41	12,841	21.51
77	32,188	0.07	25,824	31.31	**25,167**	41.41	28,191	36.11	28,264	42.12
	43,346	0.07	35,868	31.31	**33,752**	40.12	35,998	37.41	38,521	40.47
78	46,684	0.07	35,075	31.31	**34,265**	20.22	38,073	35.87	42,800	43,121
	37,532	0.07	31,619	31.31	**29,303**	18.21	36,521	36.41	37,584	20.84
79	18,552	0.07	16,131	26.25	15,044	42.41	16,141	34.34	**14,735**	41.41
	16,656	0.07	13,575	20.12	**12,998**	40.52	13,258	33.42	15,210	42.57
80	24,109	0.07	20,308	20.21	**16,329**	20.2	20,748	34.34	21,441	22.21
	16,097	0.07	13,622	43,151	**12,985**	18.45	13,959	33.2	15,854	20.14
81	30,033	0.07	24,745	25.21	22,785	41.41	28,432	34.34	**22,095**	44.32
	22,941	0.07	19,628	20.21	**18,882**	42.12	19,523	33.87	23,568	41.52
82	29,737	0.07	23,494	21.2	**19,992**	20.22	23,684	34.34	25,202	22.22
	22,102	0.07	17,703	20.16	**17,417**	15.41	20,457	35.36	19,852	20.48
83	61,463	0.07	50,648	22.21	47,420	43.41	51,347	34.34	**42,126**	40.21
	38,537	0.07	35,859	22.22	**34,852**	41.2	35,558	33.41	36,521	40.38
84	64,553	0.07	49,040	26.26	**39,446**	20.2	49,985	34.34	46,752	21.12
	45,766	0.07	38,938	26.2	**37,769**	18.52	40,512	37.49	42,584	19.54
85	36,923	0.09	29,449	95.45	**28,899**	557.12	32,445	99.85	30,232	550.32
	39,190	0.09	34,008	90.1	**30,212**	500.12	35,854	101.46	36,852	554.84
86	49,671	0.09	30,462	96.46	**29,983**	300.3	33,120	97.25	39,478	300.21
	55,854	0.09	35,854	95.2	**34,585**	286.25	38,521	99.45	39,024	300.52
87	52,105	0.09	37,155	98.98	**35,346**	547.2	36,521	95.52	39,852	498.23
	56,225	0.09	41,186	99.85	**41,073**	544.2	41,084	98.42	42,541	501.52
88	54,431	0.09	41,365	98.45	**39,999**	300.41	42,532	90.54	42,521	298.52
	45,852	0.09	37,411	98.12	**35,745**	285.58	40,251	89.62	40,789	300.56
89	200,093	0.09	165,916	98.95	**160,011**	547.11	169,990	91.45	192,211	544.21
	197,458	0.09	172,137	98.55	**161,250**	541.41	170,521	92.35	165,211	545.2
90	238,904	0.09	183,874	99.98	**173,500**	300.3	189,921	92.1	205,233	298.21
	301,478	0.09	221,054	88.85	**198,540**	288.52	229,951	91.52	244,152	300.21
91	70,667	0.09	52,295	98.78	**50,123**	555.41	52,992	97.75	60,002	540.42
	61,495	0.09	53,992	90.12	**53,571**	551.26	56,854	98.23	57,850	545.29
92	71,680	0.09	52,401	101.4	**50,411**	300.13	53,620	95.44	55,122	310.24
	69,487	0.09	**51,411**	102.1	52,415	288.54	53,214	99.26	53,987	315.54
93	90,207	0.09	**69,014**	103.1	69,122	554.12	71,520	91.52	72,321	488.24
	85,538	0.09	67,509	100.52	66,002	552.15	**65,785**	92.68	68,597	500.51
94	112,700	0.09	80,051	97.12	**78,852**	299.85	82,520	88.23	82,123	258.55
	98,474	0.09	85,487	99.54	**84,251**	300.21	86,985	87.54	88,789	256.54
95	261,304	0.09	221,852	103.1	**220,052**	500.45	220,985	90.15	254,420	514.24
	219,705	0.09	191,860	101.52	**184,152**	521.2	198,540	91.24	205,210	519.41
96	253,805	0.09	203,538	101.1	**185,222**	300.45	191,299	94.1	205,632	288.27
	278,845	0.09	205,487	100.84	**185,240**	289.41	214,521	94.12	219,984	300.54

where k_1 is set to 4.7. For the test problems with sequence-dependent setup times of U(5, 25), U(5, 50) and U(25, 50), k_2 is set to 0.48, 0.35 and 0.30, respectively.

Our experiments imply that based on a probability of 0.7, the machine is eligible to process the job; with a probability of 0.3, it cannot (Alagöz and Azizoğlu, 2003). With a probability of 1, the machine is also eligible to process the job. Thus, machine eligibility restrictions are ignored for these test problems. The weights of the jobs that are generated are suitable for U(1,10). Table 5 shows how the parameters are generated. We generated two different test problems for each combination. The total number of test problems was 240. The parameters of the test problems are given in Table 6.

The proposed MILP model was coded in the GAMS 24.1.3 program, and the Cplex solver was used. The SA, TS and modified ATCS rule were coded in the Microsoft Visual Studio 2015 C# programming language and implemented on a an Intel (R) Core (TM) i7- 5500 U CPU at 2.40 GHz with 12GB of RAM memory and the Windows 10 operating system. The SA and TS parameters are determined using the Taguchi optimization technique for all problem sizes. The levels of the parameters for SA are given in Table 7, and the TS parameter levels are given in Table 8.

Ten problems are used for the preliminary trials. The parameters of the problems are 2, 3, 5, 7 and 10 machines with 5^*m and 10^*m jobs. The setup times of the problems are based on U(25, 50), the value of b_{ij} is equal to 1 with probability 1, and the τ value is 0.5. The experiments were designed with three factors at five levels considering the orthogonal array layout of L25. All statistical experimental results were analyzed using Minitab 16 for Windows (Minitab Inc.). The corresponding Taguchi design table for the SA and the average response values for the example problem (5 machines and 50 jobs; problem no. 65) are displayed in Table 9.

Taguchi used a measure signal-to-noise ratio (S/N) to determine the characteristics of engineering problems. To optimize the SA parameters, " the smaller, the better" performance criterion was used in the Taguchi method. The S/N ratio was calculated using Eq. (31):

$$\frac{S}{N} = -10 * log\left(\frac{1}{n}\sum_{i=1}^{n} Y_i^2\right) \tag{31}$$

Table 15
Test problems with 10 machines and 50 or 100 jobs.

No	ATCS		SA I		TS I		SA II		TS II	
	Solution	CPU	Solution	CPU	Solution	CPU	Solution	CPU	Solution	CPU
97	22,346	0.09	17,306	73.73	**17,199**	360.18	18,199	72.71	19,739	419.18
	20,783	0.09	16,591	71.23	**16,108**	354.24	18,521	70.25	21,523	425.35
98	28,895	0.09	**14,934**	72.42	16,201	190.22	18,452	72.71	20,643	228.13
	26,025	0.09	18,523	73.45	**16,852**	188.24	18,958	73.2	20,523	232.38
99	29,506	0.09	24,145	73.45	**21,204**	360.15	25,760	72.71	25,666	420.1
	27,701	0.09	**21,633**	77.2	22,143	355.42	23,526	70.27	23,998	423.2
100	36,728	0.09	20,686	72.74	**19,290**	197.32	23,054	70.79	29,842	228.11
	42,580	0.09	30,237	70.56	**27,853**	200.42	29,634	68.12	31,203	232.54
101	85,314	0.09	68,398	77.45	**67,433**	370.12	69,096	70.41	72,262	421.12
	83,227	0.09	70,795	75.13	**68,097**	368.52	70,995	68.18	71,256	430.28
102	76,398	0.09	57,842	72.12	**56,266**	197.12	59,958	70.11	60,212	226.12
	79,991	0.09	60,012	77.43	**58,745**	199.42	60,856	65.48	62,568	229.54
103	35,244	0.09	23,893	71.72	**22,204**	377.12	24,683	69.68	28,251	420.18
	29,690	0.09	27,164	70.34	**24,167**	380.32	26,568	71.41	29,563	423.62
104	38,650	0.09	30,731	72.12	**28,145**	188.12	31,124	69.45	34,488	218.11
	44,582	0.09	35,698	74.52	**35,023**	190.16	37,546	71.52	37,523	221.37
105	45,482	0.09	32,115	74.1	**30,205**	377.11	35,051	68.58	34,350	420.17
	37,678	0.09	**26,630**	71.26	27,478	374.54	28,523	70.52	29,369	422.14
106	52,104	0.09	**33,742**	74.1	34,487	180.22	37,361	68.45	46,584	227.15
	60,236	0.09	48,523	75.36	**47,023**	182.12	48,325	71.54	51,851	231.26
107	104,339	0.09	88,124	74.1	**86,376**	377.11	90,902	66.12	90,675	421.12
	116,410	0.09	97,878	77.26	**94,940**	379.15	98,569	68.41	100,523	425.3
108	104,431	0.09	84,929	74.1	**83,765**	180.63	84,680	65.45	87,984	227.18
	79,985	0.09	**60,056**	71.65	60,523	182.24	62,512	66.45	64,513	229.38
109	104,172	0.13	85,089	245.24	**82,255**	1324.1	92,665	241.12	97,640	1439.14
	74,034	0.13	61,488	241.68	**60,058**	1300.41	67,852	242.56	69,998	1386.35
110	111,195	0.13	**72,510**	249.24	73,614	655.12	79,621	245.12	95,552	795.45
	152,365	0.13	108,541	251.3	**98,258**	652.43	116,545	240.85	135,485	785.24
111	122,008	0.13	100,410	240.12	**99,511**	1260.45	101,699	241.45	114,336	1412.11
	123,700	0.13	89,148	245.57	**88,741**	1285.12	91,520	235.42	91,985	1419.25
112	156,301	0.13	102,245	240.15	**101,086**	660.1	115,019	240	121,246	791.41
	109,908	0.13	70,845	240.52	**69,541**	665.42	72,587	242.35	74,125	784.38
113	435,874	0.13	375,190	240.2	**374,384**	1265.23	381,648	241.13	383,316	1422.12
	449,571	0.13	384,383	245.63	**374,512**	1198.2	389,988	245.32	398,541	1432.54
114	369,165	0.13	317,855	240.12	**316,705**	654.22	331,260	240.1	335,442	771.15
	312,410	0.13	269,954	238.59	**260,021**	650.42	285,221	243.23	302,154	784.35
115	128,771	0.13	92,802	245.2	**91,221**	1380	96,465	240.13	108,433	1488.12
	123,671	0.13	107,215	241.87	**98,541**	1485.41	115,284	241.32	119,988	1409.35
116	141,525	0.13	107,947	252.12	**98,584**	780.12	110,519	241.12	122,210	774.41
	120,983	0.13	109,548	255.68	**100,052**	778.52	115,296	238.16	135,294	762.37
117	173,090	0.13	128,908	255.41	**124,136**	1380	130,163	241.18	145,220	1414.2
	166,326	0.13	130,970	253.42	**125,269**	1374.15	139,845	243.43	145,232	1485.31
118	168,401	0.13	111,768	252.12	**108,732**	758.75	124,289	240.13	130,399	773.16
	187,450	0.13	148,532	250.23	**132,052**	768.41	168,452	243.16	159,652	701.38
119	461,302	0.13	395,218	255.11	**383,215**	1337.13	418,526	247.13	414,152	1444.12
	493,362	0.13	440,811	253.63	**431,200**	1389.1	449,745	243.53	459,872	1532.38
120	528,952	0.13	**456,525**	250.11	457,804	743.1	485,852	248.13	481,046	771.1
	502,698	0.13	429,685	245.35	**415,896**	795.15	445,269	245.38	449,985	732.28

where n is the number of observations in each experiment and Y_i is the objective function value of the SA.

The optimum parameters are determined based on the highest S/N values. According to the S/N ratios, the levels of the factors are determined for all problem sizes and given in Table 10. According to Table 10, the parameters of the SA (T_0, IT and q) are 150, 150 and 0.99, and the parameters of the TS (v, g, TLL) are 50, 10 and 5 for a problem with 2 machines and 10 jobs.

4.2. Comparison of the proposed methods

Test problems were solved with the proposed MILP model. The MILP model was run until the optimal solution was obtained for the test problem with 2 machines and 10 jobs. Because it took a very long time for the mathematical model to reach optimal solutions for the other test problems, setting a time limit for solutions was necessary. The time limit was set to 18,000 s for all problems except the test problem with 2 machines and 10 jobs. Because of the NP-hard nature of the problem, the optimal solution was obtained for problems with only 10 jobs and 2 machines. Feasible solutions were obtained only for the problems with 2 machines and 20 jobs, 3 machines and 15 jobs and 5 machines and 25 jobs. No solution was obtained for the problems with 7 and 10 machines. The TS algorithm and the SA algorithm were proposed to solve large problems. In the proposed TS I algorithm, the initial solution is obtained by the modified ATCS rule, and a long-term memory structure is used. The parameters were determined by the Taguchi experimental design method for TA I. In TA II, the initial solution was randomly generated, and long-term memory was not considered. The parameters of the TS II algorithm are iteration numbers of short term memory and tabu list length. For TS II, the iteration number of the short-term memory is equal to $v \times g$ (v: iteration number of the short-term memory of TS I; g: iteration number of the long-term memory of TS I). Thus, both TS algorithms are run with an equal number of iterations and with the same tabu list length. In the proposed SA I algorithm, the initial solution is obtained by the modified ATCS rule. For the SA II algorithm, the initial solution was randomly generated. The parameters of SA I were

Table 16
Comparison of the heuristic algorithms.

n	τ	% Error SA I	% Error TS I	% Error SA II	% Error TS II
10	0.5	**0**	**0**	0.88	7.02
	0.8	**0**	**0**	0.18	3.98
15	0.5	9.17	**0.10**	11.37	7.44
	0.8	3.36	**0.10**	3.01	4.43
20	0.5	5.30	**0.06**	7.47	13.61
	0.8	4.21	**0**	7.43	9.57
25	0.5	11.07	**0.05**	15.81	22.78
	0.8	6.40	**0**	13.64	13.05
30	0.5	10.60	**0.94**	15.52	19.90
	0.8	3.02	**0.58**	6.25	10.21
35	0.5	9.00	**0**	15.65	27.81
	0.8	10.73	**1.48**	14.33	14.27
50	0.5	7.81	**0.84**	12.95	19.74
	0.8	3.87	**0.69**	7.45	15.15
70	0.5	5.08	**0**	9.36	14.30
	0.8	3.37	**0.20**	5.04	9.79
100	0.5	2.30	**0.12**	8.15	15.07
	0.8	5.15	**0.02**	11.28	16.82
Average		5.58	0.28	9.20	13.60

determined by the Taguchi experimental design method. The parameters of the SA II algorithm were the same as those of the SA I algorithm.

The results of the MILP model, ATCS dispatching rule and heuristic algorithms are given in Tables 11–15. Because the SA is a stochastic algorithm, SA I and SA II were run five times for each problem. The results of TS II depend on the initial solution of the algorithm, and TS II was run five times for each problem. The solutions of the SAs and TS II were those with the best values obtained in 5 experiments. The CPU time of the algorithms was the average run time of the 5 experiments. Because the proposed TS I algorithm is deterministic, TS I was run one time for each problem. According to Tables 11–15, TS I obtained better results than the other methods for nearly all test problems.

The heuristic error was calculated as follows for the test problems for which the optimal solution was known.

% Error = (Obj.Fnc.Value of Heuristic Solution-Obj.Fnc.Value of Optimal Solution)/(Obj.Fnc.Value of Optimal Solution) × 100

The results of the heuristic methods were compared with the results of the optimal solutions obtained by the MILP model, and the errors of these heuristic methods were 0 for TS I and SAI, 0.53 for SA II and 5.5 for TS II. According to these values, for test problems with 2 machines and 10 jobs, TS I and SA I obtained the optimal solutions.

Because the optimal solutions of problems in the range of [13–120] were not obtained, the results of the heuristic algorithms were compared with the best heuristic solution to assess their performance. The best heuristic solutions are shown in bold font in Tables 11–15. In this comparison, the error, formulated below, was used as a performance measure.

% Error = (Obj.Fnc.Value of Heuristic Solution-Obj.Fnc.Value of Best Heuristic Solution)/(Obj.Fnc.Value of Best Heuristic Solution) × 100

Table 16 presents the average percent error of the heuristic algorithms for various problems according to the τ values and numbers of machines and jobs. TS I displayed the best performance, and TS II exhibited the worst performance. According to these results, long-term memory usage by the TS algorithm greatly affected the performance of the algorithm. The TS algorithm will cycle if there is no long-term memory. Moreover, SA I displayed better performance than SA II. In conclusion, the initial solution-finding mechanism, the modified ATCS rule, has a positive effect on the performances of both the SA and TS algorithms.

5. Conclusions

In this article, the unrelated parallel machine scheduling problem with a common server was considered. In addition, sequence-dependent setup times and machine eligibility restrictions were examined. The objective function involved total weighted tardiness minimization. An MILP model was proposed for the optimal solutions. Due to the NP hardness of the problem, heuristic methods, including a TS and an SA, were proposed to obtain the solutions of the problem. According to the experimental comparisons, the TS algorithm with long-term memory and an initial solution-finding mechanism based on the modified ATCS rule yielded better results than the other heuristic methods. This study is the first to modify the ATCS dispatching rule for scheduling problems with a common server. In addition, this study is the first to use the TS algorithm for scheduling problems with a server and to consider the total weighted tardiness and unrelated parallel machines in scheduling with a common server. Future studies will focus on sequence- and machine-dependent setup times, different heuristic methods and the lower bounds of the problem.

Funding

This work was supported by Eskişehir Osmangazi University Scientific Research Projects [201715D10].

References

Abdekhodaee, A.H., Wirth, A., 2002. Scheduling parallel machines with a single server: Some solvable cases and heuristics. Comput. Oper. Res. 29 (3), 295–315.
Abdekhodaee, A.H., Wirth, A., Gan, H.S., 2004. Equal processing and equal setup time cases of scheduling parallel machines with a single server. Comput. Oper. Res. 31 (11), 1867–1889.
Abdekhodaee, A.H., Wirth, A., Gan, H.S., 2006. Scheduling two parallel machines with a single server: the general case. Comput. Oper. Res. 33 (4), 994–1009.
Afzalirad, M., Rezaeian, J., 2017. A realistic variant of bi- objective unrelated parallel machine scheduling problem: NSGA II and MOACO approaches. Appl. Soft Comput. 50, 109–123.
Alagöz, O., Azizoğlu, M., 2003. Rescheduling of identical parallel machines under machine eligibility constraints. Eur. J. Oper. Res. 149 (3), 523–532.
Bektur, G., Saraç, T., 2016. Two parallel injection machine scheduling under crane constraint. J. Fac. Eng. Archit. Gazi Univ. 21 (4), 903–911.
Bilge, Ü., Kıraç, F., Kutulan, M., Akgün, P., 2004. A tabu search algorithm for parallel machine total tardiness problem. Comput. Oper. Res. 31 (3), 397–414.
Bozorgirad, M.A., Logendran, R., 2012. Sequence dependent group scheduling problem on unrelated parallel machines. Expert Syst. Appl. 39 (10), 9021–9030.
Brucker, P., Dhaenens- Flipo, C., Knust, S., Kravchenko, S.A., Werner, F., 2002. Complexity results for parallel machine problems with a single server. J. Sched. 5 (6), 429–457.
Chang, P.C., Chen, S.H., 2011. Integrating dominance properties with genetic algorithms for parallel machine scheduling problems with setup times. Appl. Soft Comput. 11, 1263–1274.
Chen, C.L., Chen, C.L., 2009. Hybrid metaheuristics for unrelated parallel machine scheduling with sequence dependent setup times. Int. J. Adv. Manuf. Technol. 43, 161–169.
Chen, J.F., 2009. Scheduling on unrelated parallel machines with sequence and machine dependent setup times and due date constraints. Int. J. Adv. Manuf. Technol. 44, 1204–1212.
Chen, C.L., 2012. Iterated hybrid metaheuristic algorithms for unrelated parallel machines problem with unequal ready times and sequence dependent setup times. Int. J. Adv. Manuf. Technol. 60, 693–705.
Du, J., Leung, J.Y.T., 1990. Minimizing total tardiness on one machine is NP- hard. Math. Oper. Res. 15 (3), 483–495.
Gan, H., Wirth, A., Abdekhodaee, A., 2012. A branch-and-price algorithm for the general case of scheduling parallel machines with a single server. Comput. Oper. Res. 39 (9), 2242–2247.
Glass, C.A., Shafransky, Y.M., Strusevich, V.A., 2000. Scheduling for parallel dedicated machines with a single server. Nav. Res. Logist. 47 (4), 304–328.
Glass, C.A., Potts, C.N., Shade, P., 1994. Unrelated parallel-machine scheduling using local search. Math. Comput. Model. 20, 41–52.
Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. 13, 533–549.

Graham, R., Lawler, L.E.L., Lenstra, J.K.L., Rinnooy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling: a surver. Ann. Discrete Math. 5, 287–326.
Guirchoun, S., Souhkal, A., Martineau, P., 2005. Complexity results for parallel machine scheduling problems with a server in computer systems. In: In Proceedings of the 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA), pp. 232–236.
Güngör, Z., Ünler, A., 2008. K- Harmonic means data clustering with tabu- search method. Appl. Math. Model. 32 (6), 1115–1125.
Hall, N.G., Potts, C.N., Sriskandarajah, C., 2000. Parallel machine scheduling with a common server. Discrete Appl. Math. 102 (3), 223–243.
Hamzadayı, A., Yıldız, G., 2016a. Event driven strategy based complete rescheduling approaches for dynamic m identical parallel machines scheduling problem with a common server. Comput. Ind. Eng. 91, 66–84.
Hamzadayı, A., Yıldız, G., 2016b. Hybrid strategy based complete rescheduling approaches for dynamic m identical parallel machines scheduling problem with a common server. Simul. Model. Pract. Theory 63, 104–132.
Hamzadayı, A., Yildiz, G., 2017. Modeling and solving static m identical parallel machines scheduling problem with a common server and sequence dependent setup times. Comput. Ind. Eng. 106, 287–298.
Hasani, K., Kravchenko, S.A., Werner, F., 2014a. Simulated annealing and genetic algorithms for the two machine scheduling problem with a single server. Int. J. Prod. Res. 52 (13), 3778–3792.
Hasani, K., Kravchenko, S.A., Werner, F., 2014b. Block models for scheduling jobs on two parallel machines with a single server. Comput. Oper. Res. 41, 94–97.
Huang, S., Cai, L., Zhang, X., 2010. Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server. Comput. Opera. Res. 58 (1), 165–174.
Jiang, Y., Dong, J., Ji, M., 2013. Preemptive scheduling on two parallel machines with a single server. Comput. Ind. Eng. 66 (2), 514–518.
Jiang, Y., Zhang, Q., Hu, J., Dong, J., Ji, M., 2015. Single server parallel machine scheduling with loading and unloading times. J. Comb. Optim. 30 (2), 201–213.
Kayvanfar, V., Teymourian, E., 2014. Hybrid intelligent water drops algorithm to unrelated parallel machines scheduling problem: a just-in-time approach. Int. J. Prod. Res. 52 (19), 5857–5879.
Kim, M., Lee, Y.H., 2012. MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. Comput. Oper. Res. 39 (11), 2457–2468.
Kim, C.O., Shin, H.J., 2003. Scheduling jobs on parallel machines: a restricted tabu search approach. Int. J. Adv. Manuf. Technol. 22, 278–287.
Kim, D.W., Kim, K.H., Jang, W., Chen, F.F., 2002. Unrelated parallel machine scheduling with setup times using simulated annealing. Robot. Comput. Integr. Manuf. 18 (3), 223–231.
Kirkpatrick, S., Gelatt, S.C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680.
Koulamas, C.P., 1996. Scheduling two parallel semiautomatic machines to minimize machine interference. Comput. Oper. Res. 23 (10), 945–956.
Kravchenko, S.A., Werner, F., 1997. Parallel machine scheduling problems with a single server. Math. Comput. Modell. 26 (12), 1–11.
Kravchenko, S.A., Werner, F., 1998. Scheduling on Parallel Machines with a Single and Multiple Servers. Otto- van- Guericke- Universitat Mgdeburg, pp. 1–18.
Kravchenko, S.A., Werner, F., 2001. A heuristic algorithm for minimizing mean flow time with unit setups. Inf. Process. lett. 79 (6), 291–296.
Kuei, L.Y., Wei, L.C., 2013. Dispatching rules for unrelated parallel machine scheduling with release dates. Int. J. Adv. Manuf. Technol. 67, 269–279.
Lee, Y.H., Bhaskaran, K., Pinedo, M., 1997. A heuristic to minimize the total weigted tardiness with sequence- dependent setups. IIE Trans. 29 (1), 45–52.
Lee, J.H., Yu, J.M., Lee, D.H., 2013. A tabu search algorithm for unrelated parallel machine scheduling with sequence and machine dependent setups: minimizing total tardiness. Int. J. Adv. Manuf. Technol. 69, 2081–2089.
Lin, Y.K., Fowler, J.W., Pfund, M.E., 2013. Multiple- objective heuristics for scheduling unrelated parallel machines. Eur. J. Oper. Res. 227, 239–253.
Mönch, L., Balasubramanian, H., Fowler, J.W., Pfund, M.E., 2005. Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. Comput. Oper. Res. 32, 2731–2750.
Ou, J., Qi, X., Lee, C.Y., 2010. Parallel machine scheduling with multiple unloading servers. J. Sched. 13 (3), 213–226.
Özpeynirci, S., Gökgür, B., Hnich, B., 2016. Parallel machine scheduling with tool loading. Appl. Math. Model. 40 (9-10), 5660–5671.
Pfund, M., Fowler, J.W., Gadkari, A., Chen, Y., 2008. Scheduling jobs on parallel machines with setup times and ready times. Comput. Ind. Eng. 54, 764–782.
Sarıçiçek, İ., Çelik, C., 2011. Two meta- heuristics for parallel machine scheduling with job splitting to minimize total tardiness. Appl. Math. Model. 35 (8), 4117–4126.
Sels, V., Coelho, J., Dias, A.M., Vanhoucke, M., 2015. Hybrid tabu search and truncated branch and bound for the unrelated parallel machine scheduling problem. Comput. Oper. Res. 53, 107–117.
Su, C., 2011. Online LPT algorithms for parallel machines scheduling with a single server. J. Comb. Optim. 26 (3), 480–488.
Türker, A.K., Sel, Ç., 2011. A hybrit approach on single server parallel machines scheduling problem with sequence dependent setup times. J. Fac. Eng. Archit. Gazi Univ. 26 (4), 731–740.
Vallada, E., Ruben, R., 2011. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. Eur. J. Oper. Res. 211 (3), 612–622.
Werner, F., Kravchenko, S.A., 2010. Scheduling with multiple servers. Autom. Remote Control 71 (10), 2109–2121.
Yang-Kueui, L., Chi- Wei, L., 2013. Dispatching rules for unrelated parallel machine scheduling with release dates. Int. J. Adv. Manuf. Technol. 67 (1), 269–279.
Zhang, L., Wirth, A., 2009. On-line scheduling of two parallel machines with a single server. Comput. Oper. Res. 36 (5), 1529–1553.
Zhang, A., Wang, H., Chen, Y., Chen, G., 2016. Scheduling jobs with equal processing times and a single server on parallel identical machines. Discrete Appl. Math. 213, 196–206.