TOP (2019) 27:94–124
https://doi.org/10.1007/s11750-018-00494-x

ORIGINAL PAPER

MIP models and a matheuristic algorithm for an identical parallel machine scheduling problem under multiple copies of shared resources constraints

Emine Akyol Ozer[1] · Tugba Sarac[2]

Received: 4 July 2017 / Accepted: 27 November 2018 / Published online: 7 December 2018

Abstract
If parallel machines use shared resources during production, jobs on machines must wait until the required resources are available. If the shared resource is single, only one job can use it at a time, but if there are multiple copies of this resource, multiple jobs can be scheduled up to the number of copies at a time. For this reason, it is crucial to consider resource usage when scheduling this type of machine. In recent years, various studies have been carried out to address identical parallel machine scheduling problems. However, although shared resources in parallel machines are an important aspect of this problem, resources are rarely considered in these studies and, in fact, have not been studied for this particular aspect. In this study, an identical parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and multiple copies of shared resources (IPMS-SMS) is considered. Mixed-integer programming (MIP) models and a model-based genetic algorithm (matheuristic) are proposed, and the objective function of the problem seeks to minimize the total weighted completion time. Randomly generated instances are solved using the proposed models and the matheuristic. Optimal schedules are obtained for almost all small problems using a mixed-integer programming model. However, better solutions are obtained for medium and large instances using the proposed matheuristic.

Keywords Matheuristic · Genetic algorithm · Identical parallel machine scheduling problem · Sequence-dependent setup times · Machine eligibility restrictions · Multiple copies of shared resources

✉ Emine Akyol Ozer
emineakyol@anadolu.edu.tr

Tugba Sarac
tsarac@ogu.edu.tr

1 Department of Industrial Engineering, Eskisehir Technical University, Iki Eylul Campus, 26555 Eskisehir, Turkey

2 Department of Industrial Engineering, Osmangazi University, Meselik, 26480 Eskisehir, Turkey

1 Introduction

The parallel machine scheduling problem is a generalization of the single machine scheduling problem. Many production environments consist of several stages or workcenters, each with a number of machines working in parallel (Pinedo 2009). The parallel machine scheduling problem is significant because it is a common phenomenon in real life and a subproblem of multistage complex problems.

Parallel machine scheduling is one of the most studied problems in the scheduling literature. This problem is usually divided into three main classes based on the characteristics of processing time in the machines (Pinedo 2011): identical parallel machines, uniform parallel machines and unrelated parallel machines. Our problem is an identical parallel machine scheduling problem. Identical parallel machine scheduling problems can be studied along with different constraints. In this study, we consider an identical parallel machine scheduling problem with constraints of sequence-dependent setup time, machine eligibility and multiple copies of shared resources. Studies on parallel machine scheduling with each of these constraints are briefly discussed below in a separate paragraph.

The setup time is a period required to prepare a machine for processing. There are many studies on identical parallel machine scheduling problems with *sequence-dependent setup times*. In these studies, the objective of the problem is generally to minimize the makespan (Arnaout 2010; Montoya-Torres et al. 2009; Montoya-Torres et al. 2010; Kim and Kim 2011). Chung et al. (2009) analyzed a scheduling problem with job-cluster processing time, machine capacity and daily production target volumes on a polyimide printing process operation. They proposed a mixed-integer linear programming (MIP) model. The problem is also transformed into a multiple tour maximum collection problem. Driessel and Moench (2009) discussed an identical parallel machine scheduling problem in semiconductor manufacturing and considered ready times of the jobs, precedence constraints and sequence-dependent setup times. They suggested a variable neighborhood search scheme for the problem. The proposed approach was compared to heuristics based on the apparent tardiness cost of setups and the ready time dispatching rule. Li et al. (2010) considered an identical parallel machine scheduling problem with release date and due date. The objective function of the problem was to minimize the makespan and total tardiness. They developed a 0-1 mixed-integer program and a genetic algorithm (GA) with a fuzzy logic controller. Driessel and Moench (2011) obtained an initial solution for a variable neighborhood search. The entire scheme, with the initial solution obtained by the apparent tardiness cost with setups and ready times, was also fast. Lin et al. (2011) studied an identical parallel machine scheduling problem with sequence-dependent setup times and job release dates. The objective was to minimize the maximum lateness. They developed an improved iterated greedy heuristic and compared it with the basic iterated greedy heuristic. Turker and Sel (2011) analyzed a scheduling problem on two identical parallel machines with sequence-dependent setup times and setup operations. The objective was to minimize the makespan. They proposed an algorithm combining a genetic algorithm and a tabu search. Gokhale and Mathirajan (2012) addressed a scheduling problem with release times and machine eligibility restrictions for minimizing the total weighted flow time. They proposed a mathematical model and a number of heuristic algorithms to solve the problem. Park et al. (2012) took into account a parallel machine scheduling problem with job splitting and sequence-dependent major/minor setup times. They proposed heuristic algorithms for minimizing total tardiness. The performance of the proposed heuristics was compared with that of three previous algorithms in the literature. Li et al. (2012) developed two meta-heuristics (FLC-NSGA-II and FLC-SPEA-II) based on the traditional NSGA-II and SPEA-II for a parallel machine scheduling problem. They considered a multiobjective optimization problem to minimize the makespan and the total tardiness. Their experimental results showed the advantage of the proposed FLC-NSGA-II algorithm compared to NSGA-II and FLC-SPEA-II. Joo and Kim (2012) proposed two meta-heuristic algorithms for a parallel machine scheduling problem with ready times, due times and sequence-dependent setup times. The optimal solution was investigated with a mathematical model, and the performances of the developed meta-heuristic algorithms were evaluated with optimal solutions. Gedik et al. (2016) examined a parallel machine scheduling problem with setup times and time windows. Each job had a profit and cost. They proposed a constraint programming model and logic-based Benders algorithm to maximize total profit. The models were tested on a real-life case study by the U.S. Army Corps of Engineers.

Some machines are incapable of processing a particular job. This is referred to as machine eligibility restriction in the scheduling literature. There are also studies on identical parallel machine scheduling problems with *machine eligibility restrictions*. For example, Alagoz and Azizoglu (2003) analyzed a rescheduling problem in parallel machine environments under machine eligibility constraints. Their objective was to minimize total flow time, and they presented an optimizing algorithm. Eliiyi and Azizoglu (2009) considered an identical parallel machine scheduling problem with machine-dependent job weights. The objective was to maximize the total weight of the processed jobs. They presented a branch and bound algorithm and obtained successful results for large-sized problems. Additionally, Su et al. (2011) addressed an identical parallel machine scheduling problem with job deadlines and machine eligibility constraints to minimize the total job completion time. A heuristic and a branch and bound algorithm was developed to solve the problem. The proposed heuristic generated a good-quality schedule in a reasonable time. Finally, Lee et al. (2013) provided a brief overview of online scheduling in parallel machine environments with machine eligibility constraints with an objective function to minimize makespan. They surveyed different parallel machines and various types of machine eligibility restrictions.

In real-life problems, resources are used to produce products. Generally, jobs share these resources. To obtain an efficient schedule, this case must be considered. Although the use of *resources* is important for parallel machine scheduling problems, few studies have considered resources. For example, Li et al. (2011) recently proposed a simulated annealing algorithm to minimize makespan for an identical parallel machine scheduling problem. The processing times were controllable by consumed resources, and the total resource consumption was limited. Edis et al. (2012) addressed a real-life problem that required the simultaneous scheduling of jobs and operators over parallel machines. The operators were responsible for monitoring the machines, unloading the parts and trimming extra material. They proposed integer and constraint programming models for minimizing the completion time of the last job. Additionally, Ruiz and Andrés-Romano (2011) considered an unrelated parallel machine scheduling problem and job sequence-dependent setup times. The duration of the setup times depended on assignable resources. They developed a mixed-integer programming model to minimize total completion time and the total amount of resources assigned. Edis and Ozkarahan (2012) investigated a resource-constrained identical parallel machine scheduling problem with machine eligibility restrictions. They developed three optimization models: an integer programming model (IP), a constraint programming model (CP) and a combined IP/CP model. Edis and Ozkarahan (2011) presented mathematical models of static and dynamic parallel machine flexible resource scheduling (PMFRS) and unspecified PMFRS problems. In this problem, additional flexible resources could be freely allocated to any jobs and/or any machines. They proposed an integer programming-based constraint programming approach for large-sized dynamic PMFRS and UPMFRS problems. Fanjul-Peyro et al. (2017) focused on unrelated parallel machine scheduling problems with additional resource constraints in which the number of resources depends on machine and job. First, the authors proposed two linear programming models. Because of the NP-hard nature of the problem, they combined matheuristic strategies with each of the two mathematical models and demonstrated the success of the algorithms using computational tests. Afzalirad et al. (2016) addressed an unrelated parallel machine scheduling problem with resource constraints, sequence-dependent setup times, different release dates, machine eligibility and precedence constraints. The considered problem is the implementation of a real-life problem that is a block erection scheduling problem in a shipyard. The authors present a model and two new meta-heuristics. The results are discussed.

Using the three-field classification, IPMS–SMS is denoted in the scheduling literature as $P_m|s_{jk}$, M_j, shared resource$|\sum w_j C_j$, where j and k indicate jobs, P_m and $\sum w_j C_j$ denote identical parallel machines and the total weighted completion times, respectively. No relevant previous study has considered these three constraints: sequence-dependent setup times (s_{jk}), machine eligibility restrictions (M_j) and multiple copies of shared resources. Similar to that of Edis and Ozkarahan (2011), our motivation is the scheduling of injection molding machines in factory manufacturing plastic parts. Our problems are similar to those in their study, but there are three important differences: (1) they did not consider sequence-dependent setup times, but we did; (2) to cope with resource constraint, they assume that all the jobs used the same mold, assigned to the same machine consecutively, but as this assumption may hamper a search for better solutions, there is no such assumption in our study; and (3) a time index was used in a description of their decision variable. Therefore, when the scheduling horizon increases, the number of decision variables gradually increases, even if the number of jobs is fixed. On the other hand, the number of decision variables depends on the number of jobs in our study.

In this study, MIP models are proposed for IPMS-SMS. When the number of machines is two and the objective is to minimize the total weighted completion times, the parallel machine scheduling problem is NP-hard (Pinedo 2011). Although it is possible to find optimal solutions using mathematical models, this approach may require an enormous amount of computation time with increasing problem size. A large number of articles presented mathematical models of an investigated parallel machine scheduling problem. However, most of these articles used heuristic or meta-heuristic approaches to tackle the problems. In recent years, the genetic algorithm has been used as a significant meta-heuristic approach (Liu and Wu 2003; Chaudhry and Drake 2009; Chan et al. 2011; Keskinturk et al. 2012). However, if we want to use the GA in the case with a shared resource use, it is not sufficient to merely determine the job sequence for computing the value of the objective function (the fitness value). In addition, if jobs using a shared resource are scheduled at the same time, determining which resource will primarily use the shared resource is also necessary. In other words, the stage of determining the value of the objective function is also a matter of decision. Therefore, if we want to use a GA to solve this problem, a solution algorithm should be developed for computing the objective function. The proposed matheuristic is a GA using a mathematical model to determine the value of the objective function. Since the optimal value of the objective function can be found, better solutions can be obtained for a classical GA with fewer population and generation sizes.

The remainder of this paper is organized as follows: In Sect. 2, the problem definition and proposed mathematical models are given. The proposed matheuristic is introduced in Sect. 3. Test problems and computational results are represented in Sect. 4. Finally, Sect. 5 summarizes conclusions and recommendations for future research.

2 Problem definition and mathematical models of the problem

The scheduling of jobs is a major and difficult problem for most plastic parts manufacturers. Our motivation is scheduling plastic injection molding machines in the Boyplast plastic factory. This factory produces several plastic parts for electrical home appliance manufacturers.

There are three significant characteristics that should be considered in the injection machine scheduling problem: (1) production of a plastic part using an injection molding machine is a single-stage production. Each job can be processed with the same processing time on any one of the machines that belongs to the given subset. Thus, injection molding machines are identical machines that operate in parallel. To produce a product in an injection molding machine, the relative mold has to be fixed to it. All molds may not be fixed to all of the machines due to technical conditions. For instance, for a mold to be fixed to the injection molding machine, its width and length must be shorter than the column space, and the mold depth must be suitable for the closing space of the machine. Therefore, machine eligibility restrictions should be considered. (2) Products with the same form in different colors are considered different products. Therefore, it is possible to have different products using the same mold. Molds are shared resources of jobs. Jobs using the same mold cannot be scheduled simultaneously. However, if there are multiple copies of the mold, jobs can be simultaneously scheduled up to the number of copies. (3) Switching from the production of a certain product to another requires setup times. Setup times cover the changing of molds or the preparation of raw materials and paint mixtures, if necessary. Therefore, setup times are short for similar jobs. If similar jobs are not assigned to the same machines, the total setup time needed for production may increase significantly. For example, if the previous job used a light color, the setup time is short, but if the previous job has a dark color, the setup time can be much longer. Therefore, setup times are dependent on the job sequence.

After all, an identical parallel machine scheduling problem have three significant characteristics of injection machines: sequence-dependent setup times, machine eligibility restrictions and multiple copies of shared resources are considered. Two MIP models are proposed for the considered problem. The objective of the models is to minimize the total weighted completion time.

The presented MIP models are based on the following assumptions:

(1) All parts are available for processing at time zero.
(2) A machine can process one job at a time at most.
(3) Machines are available for the scheduling horizon.
(4) Preemption of the jobs is not allowed.
(5) All processing and setup times of the jobs are deterministic.
(6) Each job needs a resource to produce. Some jobs use the same resources, but a resource can be used by one job at a time at most.
(7) There are different types of resources, and each type of resource may have copies.

The IPMS-SMS problem takes the following sets:
$N = \{1, 2, \ldots, n\}$ jobs set and
n refers number of jobs to be processed;
$L = \{1, 2, \ldots, m\}$ machines set and
m denotes number of machines;
$R = \{1, 2, \ldots, g\}$ resources set and
g refers number of resource types which required processing of jobs at machines.
We will also use the following indices, parameters and decision variables.
Indices:
i, j and $q \in N$ are indices to denote a job.
$l \in L$ is the index to denote a machine.
$k \in N$ is the index to denote the position of a job in the sequence of its machine.
$r \in R$ is the index to denote a resource.
Parameters:
n: number of jobs
m: number of machines
g: number of resource types
p_j: processing time of job j
M: a large positive number
h_j: setup time of job j is first in the sequence
s_{ij}: sequence-dependent setup time between job i and job j
f_r: number of available resources of type r
w_j: weight of job j

$$\text{res}_{jr}: \begin{cases} 1, & \text{if resource } r \text{required by job } j \\ 0, & \text{otherwise} \end{cases}$$

$$b_{jl}: \begin{cases} 1, & \text{if job } j \text{can be processed on machine } l \\ 0, & \text{otherwise} \end{cases}$$

Decision variables:
C_j: completion time of job j
a_j: starting time of job j

$$x_{jkl}: \begin{cases} 1, & \text{if job } j \text{is scheduled in } k\text{th order in machine} l \\ 0, & \text{otherwise} \end{cases}$$

$$\lambda_{jl}: \begin{cases} 1, & \text{if at least one other job that uses the same resource is scheduled in machine} l \\ & \text{while job } j \text{is processing,} \\ 0, & \text{otherwise} \end{cases}$$

$e1_{jq}$, $e2_{jq}$, $e3_{jq}$, $e4_{jq}$: 0-1 integer decision variables for ensuring particular constraints
Objective function:

$$\min z = \sum_{j=1}^{n} w_j C_j \tag{1}$$

Objective (1) represents the minimization of the total weighted completion time. It is equal to the sum of the values that are obtained by multiplying completion time and weight for each job.

The constraints of the presented MIP model are explained in detail in the following parts:

$$C_j + M * \left(1 - x_{jkl}\right) \geq p_j + h_j \quad \forall j, k, l \quad k = 1 \tag{2}$$

$$C_j + M * \left(2 - x_{jkl} - x_{ik-1l}\right) \geq C_i + s_{ij} + p_j \quad \forall i, j, k, l \quad i \neq j, k > 1 \tag{3}$$

Constraint (2) indicates that the completion time of the first job assigned to any machine should be equal to the sum of the processing time, setup time for the first job and waiting time of job j. Constraint (3) calculates the completion time of jobs apart from the first sequence.

$$\sum_{j} x_{jkl} \leq 1 \quad \forall k, l \tag{4}$$

$$\sum_{k} \sum_{l} x_{jkl} = 1 \quad \forall j \tag{5}$$

Constraints (4) and (5), respectively, ensure that at most one job can be assigned to any sequence in a machine, and each job must be assigned to any sequence in a machine.

$$b_{jl} \geq x_{jkl} \quad \forall j, k, l \tag{6}$$

Constraint (6) was formed to ensure that jobs are only assigned to a machine that is available considering technical conditions.

$$\sum_{j} x_{jkl} - \sum_{i} x_{ik-1l} \leq 0 \quad \forall k, l, \quad k > 1 \tag{7}$$

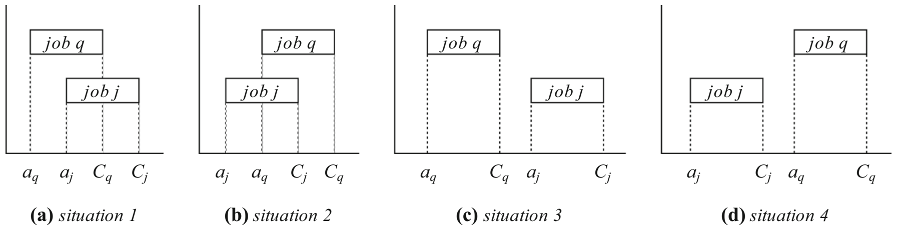


Fig. 1 Four situations for job j and job q that use shared resources

Constraint (7) ensures that the jobs are scheduled without skipping any order.

$$a_j \le C_j - p_j - s_{ij} + M * \left(2 - x_{jkl} - x_{ik-1l}\right) \quad \forall i, j, k, l \quad i \ne j, k > 1 \tag{8}$$

$$a_j \le C_j - p_j - h_j + M * \left(1 - x_{jkl}\right) \quad \forall j, k, l \quad k = 1 \tag{9}$$

Constraints (8) and (9) determine the starting times of jobs.

There are at most four situations for job j and q that use shared resources in any schedule. These situations are shown in Fig. 1.

(a) If job q starts earlier than job j and these jobs overlap, $a_q \le a_j$ and $a_j \le C_q$. Constraints (10) and (11) are for situation 1. If situation 1 occurs, the binary decision variable $e1_{jq}$ is equal to one.

$$a_q \le a_j + M * (1 - e1_{jq}) \quad \forall j, q, r \quad j < q, \ \text{res}_{jr} = 1 \text{ and } \text{res}_{qr} = 1, q \le n \tag{10}$$

$$a_j \le C_q + M * (1 - e1_{jq}) \quad \forall j, q, r \quad j < q, \ \text{res}_{jr} = 1 \text{ and } \text{res}_{qr} = 1, q \le n \tag{11}$$

(b) If job j starts earlier than job q and these jobs overlap, $a_j \le a_q$ and $a_q \le C_j$. Constraints (12) and (13) are for situation 2. If situation 2 occurs, the binary decision variable $e2_{jq}$ is equal to one.

$$a_j \le a_q + M * (1 - e2_{jq}) \quad \forall j, q, rj < q, res_{jr} = 1 \text{ and } res_{qr} = 1, q \le n \tag{12}$$

$$a_q \le C_j + M * \left(1 - e2_{jq}\right) \quad \forall j, q, r \quad j < q, \ \text{res}_{jr} = 1 \text{ and } \text{res}_{qr} = 1, q \le n \tag{13}$$

(c) If job q is completed before job j starts, $C_q \le a_j$. Therefore, these jobs do not overlap. Constraint (14) is for situation 3. If this situation occurs, the binary decision variable $e3_{jq}$ is equal to one.

$$C_q \le a_j + M * (1 - e3_{jq}) \quad \forall j, q, r \quad j < q, \ \text{res}_{jr} = 1 \text{ and } \text{res}_{qr} = 1, q \le n \tag{14}$$

(d) If job j is completed before job q starts,$C_j \le a_q$. Therefore, these jobs do not overlap. Constraint (15) is for situation 4. If this situation occurs, the binary decision variable $e4_{jq}$ is equal to one.

$$C_j \le a_q + M * (1 - e4_{jq}) \quad \forall j, q, r \quad j < q, \ \text{res}_{jr} = 1 \text{ and } \text{res}_{qr} = 1, q \le n \tag{15}$$

Constraint (16) ensures that only one situation (1, 2, 3 or 4) can occur for each job couple sharing the same resource.

$$e1_{jp} + e2_{jq} + e3_{jq} + e4_{jq} = 1 \quad \forall j, q, r \quad j < q, \ \text{res}_{jr} = 1 \text{ and } \text{res}_{qr} = 1, q \leq n \tag{16}$$

If some resources have copies, it is possible to schedule two jobs requiring the same resource in different machines at the same time. For example, if one resource has one copy, the jobs share this resource and can overlap once. Constraint (17) allows job couples using the same resource to be used at most (f_r) times in different machines. The number of overlapping jobs using the same resource is limited to the number of available related resources.

$$\sum_{l=1} \lambda_{jl} \leq f_r \quad \forall j, r \ \ \text{res}_{jr} = 1 \tag{17}$$

To determine overlapping jobs, the model should count how many times job j can overlap with job q that uses resource r in different machines and is provided by Constraints (18)–(21) with the help of decision variable λ_{jl}.

$$1 + \lambda_{jl} \geq e1_{jq} + \sum_{k=1} x_{jkl} \quad \forall j, q, l, \ \ j \neq q \tag{18}$$

$$1 + \lambda_{jl} \geq e1_{jq} + \sum_{k=1} x_{qkl} \quad \forall j, q, l \quad j \neq q \tag{19}$$

$$1 + \lambda_{jl} \geq e2_{jq} + \sum_{k=1} x_{jkl} \quad \forall j, q, l \quad j \neq q \tag{20}$$

$$1 + \lambda_{jl} \geq e2_{jq} + \sum_{k=1} x_{qkl} \quad \forall j, q, l \quad j \neq q \tag{21}$$

Finally, constraints (22)–(26) are the sign constraints of the decision variables:

$$x_{jkl} \in \{0, 1\} \quad \forall j, k, l \tag{22}$$

$$\lambda_{jl} \in \{0, 1\} \quad \forall j, l \tag{23}$$

$$e1_{jq}, e2_{jq}, e3_{jq}, e4_{jq} \in \{0, 1\} \quad \forall j, q \in N \tag{24}$$

$$C_j \geq 0 \quad \forall j \in N \tag{25}$$

$$a_j \geq 0 \quad \forall j \in N \tag{26}$$

We called the mathematical model M1. Alternatively, another mixed-integer programming model proposed by Avalos-Rosales et al. (2015) is modified based on overlapping constraints, referred to as M2. The job set N plus a dummy job 0 is denoted by N_0 in M2. The processing times and setup times associated with the dummy job are equal to zero. M2 has a new variable y_{ijl}, instead of x_{jkl}, in addition to the above defined variables that can be denoted as follows:

$$y_{ijl} = \begin{cases} 1, & \text{if job } i \text{ is scheduled before job } j \text{ on machine } l, \\ 0, & \text{otherwise.} \end{cases}$$

The constraints of M2 with the objective function (1) can be stated as follows:

$$\sum_{l \in L} \sum_{i \in N_0, i \neq j} y_{ijl} = 1 \quad \forall j \in N \tag{27}$$

$$\sum_{l \in L} \sum_{j \in N_0, j \neq i} y_{ijl} = 1 \quad \forall i \in N \tag{28}$$

Constraints (27) and (28), respectively, ensure that each job has exactly one predecessor and one successor.

$$\sum_{j \in N_0, i \neq j} y_{ijl} - \sum_{q \in N_0, q \neq i} y_{qil} = 1 \quad \forall i \in N, \quad \forall l \in L \tag{29}$$

Constraint (29) establishes that each job in a machine should have a predecessor and a successor in the same machine.

$$\sum_{j \in N} y_{0jl} \leq 1 \quad \forall l \in L \tag{30}$$

Constraint (30) guarantees that at most one job can be assigned to the first sequence in a machine.

$$C_j - C_i + M * \left(1 - y_{ijl}\right) \geq s_{ij} + p_j + wt_j \quad \forall i \in N_0, \forall j \in N, i \neq j \tag{31}$$

The completion time of jobs calculated by Constraints (31) and (32) are formulated for machine eligibility restrictions.

$$b_{jl} \geq \sum_{i \in N_0} y_{ijl} \quad \forall j \in N_0, \forall l \in L \tag{32}$$

Constraints (33) and (34) provide the calculation of starting times:

$$a_j \geq C_i + wt_j - M * \left(1 - y_{ijl}\right) \quad \forall i, j \in N_0, l \in L \quad i \neq j, j > n \tag{33}$$

$$a_j \leq C_i + wt_j + M * \left(1 - y_{ijl}\right) \quad \forall i, j \in N_0, l \in L \quad i \neq j, j > n \tag{34}$$

The completion time of a dummy job is equal to zero and is explained by Constraint (35).

$$C_0 = 0 \tag{35}$$

Table 1 Sequence-dependent setup times between job i and job j (s_{ij})

i/j	1	2	3	4	5
1	0	90	10	240	240
2	30	0	180	20	180
3	150	200	0	120	30
4	150	20	120	0	120
5	150	200	10	120	0

Table 2 The type of resource required by the jobs (res_{jr})

j/r	1	2
1	1	0
2	1	0
3	0	1
4	0	1
5	1	0

Table 3 The amount of resource r (f_r)

r	f_r
1	1
2	2

The equation numbers of overlapping constraints for M2 are (10)–(21). Constraints (23)–(26) and (36) are for sign constraints.

$$y_{ijl} \in \{0, 1\} \quad \forall i, j, l \tag{36}$$

M1 and M2 were tested using a small example:

Example There are five jobs, two machines and two resources. The processing times and setup times of job j are first in the following sequence: $p_j = \{20, 30, 10, 20, 15\}$, $h_j = \{10, 20, 15, 10, 5\}$, where $j = 1, \ldots, 5$. Sequence-dependent setup times between the two jobs are shown in Table 1.

Machine eligibility restrictions are given as follows: job 1 and job 3 can only be processed on machine 1. Job 2, job 4 and job 5 can only be processed on machine 2. res_{jr} denotes the type of resource required by job j, and this parameter is given in Table 2, and the amount of resource r (f_r) is also given in Table 3. The weights of job j (w_j) are given as follows: $w_j = \{9, 2, 6, 5, 7\}$

The example was solved using M1 and M2 with the GAMS/Cplex solver. An optimal solution was found in 28 and 12 s with M1 and M2, respectively. The objective value of the optimum schedule was 2230. The Gantt Scheme of the obtained results is shown in Fig. 2.

In Fig. 2, the jobs using the same resources are shown in the same color. As seen in the figure, jobs 1 and 5 could not be produced at the same time because resource 1

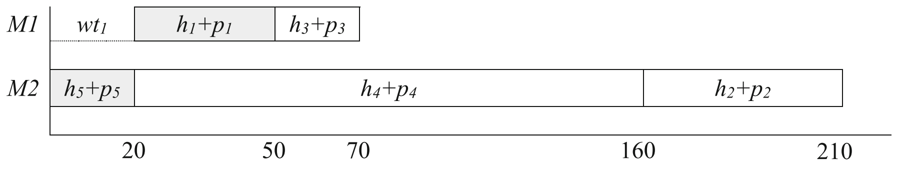


Fig. 2 Gantt scheme of example

had no copies. However, two of jobs 2, 3 and 4 could be scheduled at the same time since resource 2 had two copies.

3 Proposed matheuristics

Matheuristics are model-based (meta)heuristics. These methods combine the advantages of both exact methods and (meta)heuristics. Therefore, researchers have recently shown an increased interest in matheuristics. A number of studies have used matheuristics in scheduling problems. For example, Billaut et al. (2015) handled a scheduling problem in which jobs consumed a perishable resource stored in vials. The problem was modeled as a single machine scheduling problem with additional duration and consumption constraints. They proposed a two-step approach that consisted of a recovering beam search algorithm and a matheuristic algorithm. Guimaraes et al. (2013) considered a single machine capacitated lot sizing and scheduling problem with sequence-dependent setup times and costs. They presented a MIP-based heuristic (matheuristic) model for solving this problem. Finally, Singha et al. (2012) focused sensor coverage scheduling in wireless sensor networks subject to Q-coverage constraints. The objective of the problem was to maximize the network lifetime. A matheuristic algorithm that included a GA and a linear programming model was developed.

The IPMS–SMS instances, especially if they are large-scale problems, cannot be solved easily due to their NP-hard nature. An efficient heuristic search would be useful to address such a problem. In this study, we developed a special version of a model-based GA to solve the IPMS–SMS. The proposed matheuristic algorithms have a different structure from a classic GA. In a classic GA, solutions are coded as chromosomes, and this representation contains all the values of the decision variables of the considered problem, so fitness values of the chromosomes can be calculated using these values. However, solutions may be very far from the optimum solution. In the proposed GA, a chromosome contains only the values of the decision variable x_{jkl}. After determining the value of the decision variables x_{jkl} using the GA, determining the values of the remaining decision variables is still a decision problem. In this study, a subproblem is solved by GAMS/Cplex to calculate the fitness value. The subproblem (M3) is very similar to M1, but a new decision variable (wt_j) is defined for calculating the waiting time of job j in M3. In addition, x_{jkl} are parameters and not decision variables, and M1 is tightened using the x_{jkl} values. Therefore, we can find optimal solutions of the fitness value when subproblem M3 is solved. Since the fitness value is directly affected, better solutions can be obtained than with a classical GA with fewer population and generation sizes. The subproblem (M3) is given below:

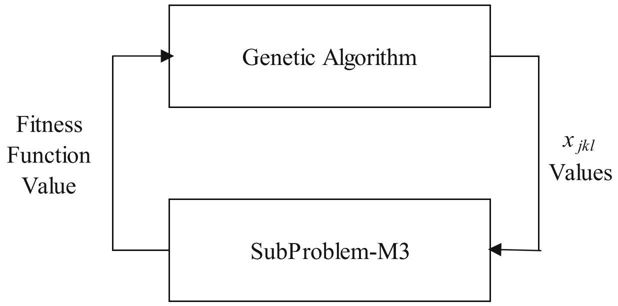


Fig. 3 The relationship between the GA and subproblem-M3

$$C_j = h_j + p_j + wt_j \quad \forall j, k, l \;\; k = 1 \quad \text{and} \;\; x_{jkl} = 1 \tag{37}$$

$$C_j = C_i + p_j + s_{ij} + wt_j \quad \forall i,\; j,\; k,\; l\; i \neq j,\;\; k > 1, \quad x_{jkl} = 1 \,\text{and}\;\; x_{jk-1l} = 1 \tag{38}$$

$$a_j = C_i + wt_j \quad \forall i, j, k, l \quad i \neq j, \quad k > 1 \;\; x_{jkl} = 1 \;\text{and}\;\; x_{jk-1l} = 1 \tag{39}$$

$$a_j = wt_j \quad \forall j, k, l \quad k = 1 \;\text{and}\;\; x_{jkl} = 1 \tag{40}$$

(10)–(21)
(23)–(26)

x_{jkl} values contain information about jobs and their sequence in machines by means of chromosome values. To determine the starting time, completion time and waiting time of jobs, M3 is solved. Constraints (37) and (38) calculate the completion time of jobs. Constraints (39) and (40) determine the starting times of jobs.

In the proposed algorithm (MA), the GA and the subproblem (M3) work cooperatively. The x_{jkl} values obtained by the GA transform the parameters of M3, and the objective function value of M3 transforms to the GA as the fitness value of the related chromosome. This relationship between the GA and M3 is shown in Fig. 3.

3.1 Representation

The chromosome representation of the proposed algorithm is a 2xn-bit integer matrix. All the genes in the first row of the chromosome have values between 1 and m, and the genes in the second row of the chromosome have values between 0 and 1. The sequence of the genes shows the job index for both rows. The first row of the matrix represents the machines to which the jobs are assigned. The second row of the matrix represents the job sequences. For example, a sample solution for the 6-job, 3-machine problem is given as the following bit matrix:

3	1	2	1	2	3
0.2	0.5	0.6	0.3	0.1	0.5

This means that jobs 2 and 4 are selected for machine one, jobs 3 and 5 are selected for machine two and jobs 1 and 6 are selected for machine three. The sequence of the jobs is identified according to the second row of the chromosome. Determining the initial job sequences, organizing the job sequences and assigning the $x_{(jkl)}$ value procedures are given in Sect. 3.2.

GA starts with the encoding of a chromosome and it is followed by generating an initial population. In the proposed algorithm, the initial population is constructed in three ways: (1) M1 assignment model (M1-A), (2) M2 assignment model (M2-A), (3) Random assignment.

(1) M1 assignment model (M1-A) is a reduced version of M1 that considers Constraints (2)–(7) and sign constraints (22) and (25) to minimize total weighted completion time. M1-A performs in 1500 s in GAMS/Cplex, and the first chromosome of the initial population is obtained.
(2) The M2 assignment model (M2-A) is a reduced version of M2 that considers Constraints (27)–(32) and (35) and sign constraints (25) and (36) to minimize the total weighted completion time. M2-A also performs in 1500 s in GAMS/Cplex, and the second chromosome of the initial population is obtained.
(3) The rest of the initial population is determined by a purely random assignment of jobs considering machine eligibility restrictions. The other chromosomes are constructed by generating random numbers in the range [1, m] for each of the genes in the first row and in the range [0, 1] for each of the genes in the second row of chromosomes. Machine eligibility restrictions are considered when generating the first row. Therefore, the chromosomes of the initial population represent feasible solutions. The pseudo-code of the initial population generation procedure for random assignment is given below:

```
Begin Initial Population Algorithm for Random Assignment
    For every chromosome, $kr=3$ to $nf$ do
        For every job $j=1$ to $n$ do
            Ok=0;
            Generate a random number $rnumber$ in the range [1, $m$]
            If $b(j, rnumber)=0$ then
                Repeat $rnumber$=mod ($rnumber$, $m$) +1;
                    If $b(j, rnumber)=1$, then
                        Ok=1;
                    End if
                Until Ok=1;
            Else gen($kr$,$j$)=$rnumber$;
            End if
            Generate a random number $rseq(kr,j)$ in the range (0, 1)
        End For
    End For
End Initial Population Algorithm for Random Assignment
```

where *kr:* chromosome, *j:* job, *l:* machine, *rnumber*: a random number, *gen*(*kr*,*j*): *j*th gene value of chromosome *kr*, *rseq*(*kr*,*j*): a random number for sequence of job *j* in chromosome *kr*

3.2 Fitness function

The fitness value of the chromosome is equal to the total weighted completion time. The genetic algorithm determines the value of the decision variable "x_{jkl}", which gives information about a particular job's assignment, its machines and its order. After determining the value of the decision variables x_{jkl} using the GA, determining the values of the remaining decision variables is still difficult and remains a decision problem. In this study, we solve a subproblem (M3) to calculate the fitness value. Based on the GA's assignment, the M3 model determines the starting time, waiting time and completion time of jobs under all constraints. The pseudo-code used for calculating the fitness value consists of three parts. First, one determines the jobs that are assigned to the same machines. After that, jobs are sorted in increasing order of random numbers in the second part. The last part is for assignment for the value of parameter x and calculating the fitness value. The entire pseudo-code is given below:

* Determining the initial job sequences

```
Part 1
For every chromosome, kr=1 to nf do
  For every machine l=1 to m do
    Num1=1;
      For every job j=1 to n do
        If gen(kr, j)=l   then
          Seq(kr,j)=num1;
          Num1=num1+1;
        End If
      End For
  End For
End For
```

* Organizing the job sequences

```
Part 2
For every chromosome, kr=1 to nf do
  For every machine l=1 to m do
    For every job j=1 to n in condition that j < n-1 do
      For every job i=1 to n in condition that j < i  do
        If gen(kr, j)=l  and gen(kr, i)=l , then
          If rseq(kr, j)>rseq(kr,i) then
              gec=seq(kr,i);
              seq(kr,i)= seq(kr,j);
              seq(kr,j)=gec;
          End If
        End If
      End For
    End For
  End For
End For
```

* Assigning the $x(j,k,l)$ value and calculating the fitness value

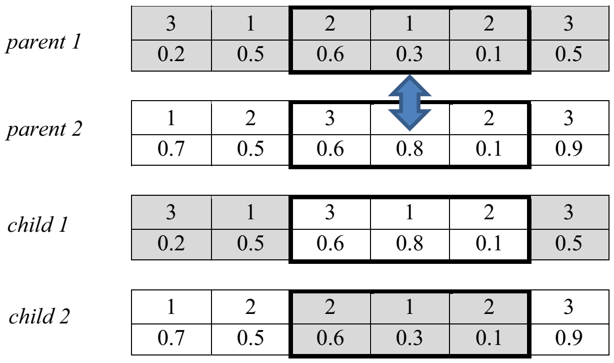
```
Part 3
For every chromosome, $kr=1$ to $nf$ do
  $x(j,k,l)=0;$
  For every job $j=1$ to $n$ do
    For every sequence $k=1$ to $n$ do
      For every machine $l=1$ to $m$ do
        If $gen(kr,j)=l$ and $seq(kr,j)=k$ then
          $x(j,k,l)=1;$
        End If
      End For
    End For
  End For
  Solve submodel (M3)
  $Fit(kr)=z;$
End For
```

where *seq*(*kr*,*j*): sequence of job *j* in chromosome *kr*, *Fit*(*kr*): fitness function value of chromosome *kr*,

3.3 Genetic operators

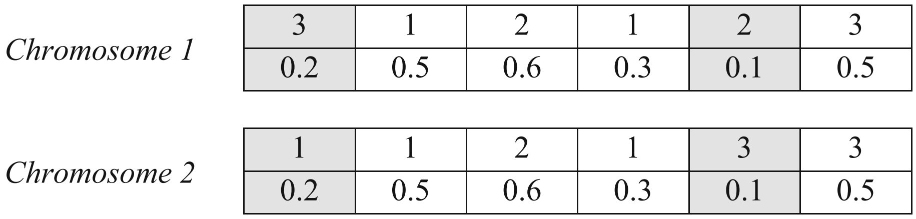
A classic GA is composed of three operators: reproduction, crossover and mutation. The reproduction operator allows individual chromosomes to be copied for possible inclusion in the next generation. The chance that a chromosome will be copied is based on the chromosome's fitness value, calculated from a fitness function. We used a 2-tournament selection method as a reproduction operator. In 2-tournament selection, two chromosomes are randomly selected. The fitness values of the chromosomes are compared. The chromosome with a better fitness value is selected for the next generation. This continues until the number of selected chromosomes is equal to the population size. Crossover enables the algorithm to extract the best genes from different individuals and recombine them into potentially superior children. The proposed GA has a two-point crossover operator. A sample crossover is given in Fig. 4. In the example, a chromosome of Parent 1 and a chromosome of Parent 2 are aligned. Crossing points are selected randomly. It shows the borders of the genes that are given in black boxes. Child 1 gets the genes of Parent 1 except for the crossing point. The remaining genes of Child 1 are copied from the chromosome of Parent 2. The genes in the crossing point of Parent 1 are copied to Child 2. Other genes in Child 2 are the same as in Parent 2. As shown in Fig. 4, only genes in the crossing point of each chromosome are changed, so eligibility constraints are still satisfied.

Reproduction and crossover alone can obviously generate a staggering number of differing chromosomes. However, depending on the initial population chosen, there may not be a sufficient variety of chromosomes to ensure the GA searches the entire problem space, or the GA may find itself converging on chromosomes that are not quite close enough to the optimum it seeks due to a bad initial population. Introducing a mutation operator into the GA may prevent some of these problems. In the proposed algorithm, a random number is generated for all the genes. If the random number is smaller than the mutation rate, the value of the gene is generated according to the initial



parent 1	3	1	2	1	2	3
	0.2	0.5	0.6	0.3	0.1	0.5
parent 2	1	2	3		2	3
	0.7	0.5	0.6	0.8	0.1	0.9
child 1	3	1	3	1	2	3
	0.2	0.5	0.6	0.8	0.1	0.5
child 2	1	2	2	1	2	3
	0.7	0.5	0.6	0.3	0.1	0.9

Fig. 4 Crossover



Chromosome 1	3	1	2	1	2	3
	0.2	0.5	0.6	0.3	0.1	0.5
Chromosome 2	1	1	2	1	3	3
	0.2	0.5	0.6	0.3	0.1	0.5

Fig. 5 Mutation

population generation procedure so that machine eligibility restrictions are considered. The value of the gene is protected if the random number is greater than the mutation rate. A sample mutation is given in Fig. 5. In this example, the mutation rate is equal to 0.25. Random numbers of the first and fourth genes are smaller than the mutation rate. Therefore, these genes are regenerated according to the initial population procedure, and Chromosome 2 is obtained.

Since elitism selection improves the efficiency of a GA considerably, as it prevents a loss of the best results, it is used in the developed GA. In this study, two types of termination conditions are used together. The first condition checks whether the algorithm has run a fixed number (n_f) of generations. The other stops the algorithm if the total solution time reaches 'n_t'.

4 Computational results

To test the performance of the proposed MIP models, randomly generated instances were used. Since the identical parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and multiple copies of shared resources has been defined firstly in this paper, we could not find test instances from the scheduling literature. Therefore, instances were generated randomly. The proposed MIP formulations and the matheuristic algorithm were coded in GAMS 24.0.2 and were run using the Cplex solver embedded in GAMS on a 2.8-GHz Intel Core i7 with 4 GB RAM. The characteristics of the test problems and the obtained results of the GAMS/Cplex solver and the matheuristic algorithm are given in the following subsections.

4.1 Test problems

In the identical parallel machine scheduling literature, generally the number of jobs (n) is approximately eight for small-sized test problems (Gacias et al. 2010; Chen and Chen 2009), approximately forty for medium-sized test problems (Edis and Oguz 2012; Gacias et al. 2010; Nessah et al. 2007) and one hundred for large-sized test problems (Montoya-Torres et al. 2009; Gacias et al. 2010; Tahar et al. 2006). In this study, we generated instances with $n = 8$, $n = 40$ and $n = 100$. Other parameters by generating instances and their levels are given separately for problems $n = 8$, $n = 40$ and $n = 100$ in Table 4.

As shown in Table 4, the number of machines (m) is two or three for instances with $n = 8$, two or six for instances with $n = 40$ and two and eight for instances with $n = 100$. In the literature, similar m values are used (Montoya-Torres et al. 2009; Lin et al. 2011; Tahar et al. 2006). The number of resource types (g) is two or five for instances with $n = 8$, four or twenty for instances with $n = 40$ and eight or fifty for instances with $n = 100$. The processing times (p_j) and the setup times for first jobs (h_j) are drawn from a uniform distribution on [1,100] rounded to the nearest integer. Sequence-dependent setup times (s_{ij}) are drawn from a uniform distribution on [1,10], if job i and job j use the same resource, and on [1,100] if they use different resources. If job j can be processed on machine l, b_{jl} is equal to 1. rb is the rate of b_{jl} parameters other than zero, and we fixed this rate at 0, 25 and 1. If the type of resource r required by job j, res_{jr} is equal to 1, g_{res} is an indicator used to show generation type. If it is (1), res_{jr} is generated randomly. If it is (2), res_{jr} is generated as one resource dense. The amount of resource r (f_r) is generated with a uniform distribution in the range [1,2]. By using these levels of parameters, 16 problem types occur, and we generated three instances for each type using Excel VBA.

All instances of IPMS with SMS are available on the web site (http://endustri.ogu.edu.tr/Personel/Akademik_personel/Tugba_Sarac_Test_Instances/IPMS_with_SMS_instances.rar)http://endustri.ogu.edu.tr/Personel/Akademik_personel/Tugba_Sarac_Test_Instances/TOP_AKYOLOZER_SARAC_Instances.rar).

4.2 Test results of the GAMS/Cplex

Each of the generated test problems with $n = 8$, $n = 40$ and $n = 100$ was solved by means of the proposed models (M1 and M2). The running time of the Cplex solver of GAMS was limited to 7200 s for all the tests. To evaluate the relative performance of the model and matheuristic algorithm, we utilized two types of performance measures, namely relative percentage deviation (RPD1, RPD2) and GAMS relative gap (GAP).

- RPD1 is equal to the difference between solutions of M1 and M2. RPD2 is defined as the difference between the best solution of the models and the solution of the matheuristic algorithm for each instance. RPD1 and RPD2 are calculated by Constraints (41) and (42), as follows:

Table 4 Problem parameters with their levels

Parameters	Levels for problems with $n = 8$		Levels for problems with $n = 40$		Levels for problems with $n = 100$	
	−	+	−	+	−	+
M	2	3	2	6	2	8
G	2	5	4	20	8	50
Rb	Rate of b_{jl} parameters other than zero is 0.25	Rate of b_{jl} parameters other than zero is 1	Rate of b_{jl} parameters other than zero is 0.25	Rate of b_{jl} parameters other than zero is 1	Rate of b_{jl} parameters other than zero is 0.25	Rate of b_{jl} parameters other than zero is 1
G_{res}	res_{jr} generation is random (1)	res_{jr} generation is one resource dense (2)	res_{jr} generation is random (1)	res_{jr} generation is one resource dense (2)	res_{jr} generation is random (1)	res_{jr} generation is one resource dense (2)
p_j	[1,100]		[1,100]		[1,100]	
h_j	[1,100]		[1,100]		[1,100]	
s_{ij}	[1,10], if res_{ir} =1 and res_{ir} =1; [1,100], otherwise		[1,10], if res_{ir} =1 and res_{ir} =1; [1,100], otherwise		[1,10], if res_{ir} =1 and res_{ir} =1; [1,100], otherwise	
fr	[1,2]		[1,2]		[1,2]	

$$\text{RPD1}(\%) = \frac{Z_{\text{M1}} - Z_{\text{M2}}}{Z_{\text{M2}}} \tag{41}$$

$$\text{RPD2}(\%) = \frac{Z_{\text{MA}-\text{Min}} - \text{Min}(Z_{\text{M1}}, Z_{\text{M2}})}{Z_{\text{MA}-\text{Min}}}, \tag{42}$$

where $Z_{\text{MA-Min}}$ denotes the minimum objection function value of the matheuristic algorithm, and Z_{M1} and Z_{M2} denote the objection function values for M1 and M2, respectively.

- GAP is obtained by GAMS/Cplex after solving mathematical models M1 and M2.

The results of small-sized ($n = 8$) problems are presented in Table 5. In this table, the first four columns summarize the properties of instances such as instance number (no), number of resources (g), rate of b_{jl} parameters other than zero (rb) and generation type of res_{jr} (g_{res}). The fifth and sixth columns show the obtained objective value and solution time in seconds using the GAMS/Cplex solver with M1. The last two columns represent the results of the M2 model for small size instances.

As we can see from Table 5, M1 and M2 are capable of solving to optimality almost all small instances in reasonable time. For only one instance (9-2), the running time limit is exceeded in M2 before reporting the obtained best integer solution is optimal. It can be clearly observed that M2 is remarkably more efficient than M1 with regard to running time. Differences between the running times of M1 and M2 are distinctly decreased with increases in the number of machines.

Test results for medium size instances ($n = 40$) obtained using M1 and M2 are summarized in Table 6. M1 could obtain feasible solutions for 80% of the medium-size instances in 7200 s, and the GAP value of all these solutions is 0.99. However, M2 could solve feasible solutions for only 30 percent of the instances, and the average GAP value is 0.72. This implies that M1 indicated a significantly poor performance compared to M2, although M1 has more feasible solutions.

Finally, feasible solutions could not be found in 7200 s using M1 and M2 for all large instances except for one instance (8-1) with M1.

4.3 Test results of the proposed matheuristic algorithm

First, a small-size ($n = 8$) instance (*no*: 6-2) was solved using the matheuristic algorithm with the following parameters: population size is 30, crossover rate is 0.7, mutation rate is 0.2 and the values of the termination parameters are $n_f = 500$ and $n_t = 7200$. The obtained convergence graph of the matheuristic algorithm is shown in Fig. 6. As shown in Fig. 6, the matheuristic algorithm can reach the optimal objective value (6785) of the instance.

The parameter values of any meta-heuristic algorithm directly affect their performance. A full factorial experimental design was used to determine the proper parameter values of the matheuristic algorithm for each problem size. Factors and their levels are given in Table 7.

Population size, crossover rate and mutation rate are critical factors according to analysis of variance for small instances. Related main effect plots are given in Fig. 7.

Table 5 GAMS/Cplex results for small instances with n=8

$m = 2$				M1		M2		$m = 3$				M1		M2	
No	g	rb	g_{res}	Z_{M1}	t_1 (s.)	Z_{M2}	t_2 (s.)	No	g	rb	g_{res}	Z_{M1}	t_1 (s.)	Z_{M2}	t_2 (s.)
1-1	2	0.25	1	8647	144	8647	16	9-1	2	0.25	1	10,142	1174	10,142	3
1-2	2	0.25	1	5824	205	5824	13	9-2	2	0.25	1	12,222	3555	**12,222**[a]	7200
1-3	2	0.25	1	8289	113	8289	11	9-3	2	0.25	1	3038	474	3038	9
2-1	2	0.25	2	6601	126	6601	6	10-1	2	0.25	2	5261	1132	5261	341
2-2	2	0.25	2	6111	146	6111	21	10-2	2	0.25	2	4803	523	4803	15
2-3	2	0.25	2	7553	300	7553	59	10-3	2	0.25	2	8516	752	8516	23
3-1	2	1	1	8487	143	8487	3	11-1	2	1	1	7678	901	7678	49
3-2	2	1	1	3568	209	3568	3	11-2	2	1	1	6830	886	6830	113
3-3	2	1	1	8843	228	8843	11	11-3	2	1	1	3749	602	3749	3
4-1	2	1	2	5811	315	5811	83	12-1	2	1	2	5116	393	5116	5
4-2	2	1	2	6893	124	6893	5	12-2	2	1	2	7110	1949	7110	939
4-3	2	1	2	8152	277	8152	39	12-3	2	1	2	6104	1151	6104	928
5-1	5	0.25	1	6256	128	6256	5	13-1	5	0.25	1	4047	398	4047	8
5-2	5	0.25	1	8899	148	8899	9	13-2	5	0.25	1	4416	458	4416	5
5-3	5	0.25	1	4266	84	4266	3	13-3	5	0.25	1	5676	581	5676	9
6-1	5	0.25	2	7922	127	7922	12	14-1	5	0.25	2	6733	453	6733	58
6-2	5	0.25	2	6785	179	6785	25	14-2	5	0.25	2	5126	479	5126	19
6-3	5	0.25	2	6148	183	6148	17	14-3	5	0.25	2	5057	498	5057	8

Table 5 continued

$m = 2$				M1		M2		$m = 3$				M1		M2	
No	g	rb	g_{res}	Z_{M1}	t_1 (s.)	Z_{M2}	t_2 (s.)	No	g	rb	g_{res}	Z_{M1}	t_1 (s.)	Z_{M2}	t_2 (s.)
7-1	5	1	1	4066	131	4066	11	15-1	5	1	1	3204	506	3204	12
7-2	5	1	1	5358	160	5358	5	15-2	5	1	1	9179	821	9179	23
7-3	5	1	1	7241	200	7241	15	15-3	5	1	1	3659	395	3659	5
8-1	5	1	2	6758	163	6758	21	16-1	5	1	2	3929	418	3929	1
8-2	5	1	2	11184	243	11184	29	16-2	5	1	2	7016	485	7016	4
8-3	5	1	2	9097	199	9097	21	16-3	5	1	2	6511	634	6511	36
				t_{mean}	178	t_{mean}	17.5					t_{mean}	817	t_{mean}	409

[a]Running time limit is exceeded

Table 6 GAMS/Cplex results for medium instances with $n = 40$

$m = 2$				M1	M2		RPD1 (%)	$m = 6$				M1	M2		RPD1 (%)
No	g	rb	g_{res}	Z_{M1}	Z_{M2}	GAP (%)		No	g	rb	g_{res}	Z_{M1}	Z_{M2}	GAP (%)	
1-1	2	0.25	1	556,714	–	–	–	9-1	2	0.25	1	–	107,997	0.72	–
1-2	2	0.25	1	–	–	–	–	9-2	2	0.25	1	85,830	–	–	–
1-3	2	0.25	1	220,761	–	–	–	9-3	2	0.25	1	67,983	–	–	–
2-1	2	0.25	2	246,713	–	–	–	10-1	2	0.25	2	177,749	–	–	–
2-2	2	0.25	2	–	–	–	–	10-2	2	0.25	2	140,977	–	–	–
2-3	2	0.25	2	312,326	–	–	–	10-3	2	0.25	2	171,875	–	–	–
3-1	2	1	1	–	–	–	–	11-1	2	1	1	155,419	–	–	–
3-2	2	1	1	–	–	–	–	11-2	2	1	1	149,608	–	–	–
3-3	2	1	1	–	–	–	–	11-3	2	1	1	166,369	–	–	–
4-1	2	1	2	–	–	–	–	12-1	2	1	2	220,442	–	–	–
4-2	2	1	2	–	–	–	–	12-2	2	1	2	275,886	–	–	–
4-3	2	1	2	–	–	–	–	12-3	2	1	2	190,490	–	–	–
5-1	5	0.25	1	118,568	106,043	0.98	0.12	13-1	5	0.25	1	92,964	52,980	0.59	0.75
5-2	5	0.25	1	150,839	109,234	0.98	0.38	13-2	5	0.25	1	53,425	45,562	0.58	0.17
5-3	5	0.25	1	129,602	126,709	0.99	0.02	13-3	5	0.25	1	52,900	38,610	0.50	0.37
6-1	5	0.25	2	153,010	140,480	0.81	0.09	14-1	5	0.25	2	71,286	49,070	0.52	0.45
6-2	5	0.25	2	–	109,687	0.80	–	14-2	5	0.25	2	53,370	46,807	0.59	0.14
6-3	5	0.25	2	118,452	110,970	0.82	0.07	14-3	5	0.25	2	92,059	57,110	0.59	0.61

Table 6 continued

$m = 2$				M1	M2		RPD1 (%)	$m = 6$				M1	M2		RPD1 (%)
No	g	rb	g_{res}	Z_{M1}	Z_{M2}	GAP (%)		No	g	rb	g_{res}	Z_{M1}	Z_{M2}	GAP (%)	
7-1	5	1	1	143,685	–	–	–	15-1	5	1	1	74,829	–	–	–
7-2	5	1	1	141,144	–	–	–	15-2	5	1	1	89,769	–	–	–
7-3	5	1	1	171,737	–	–	–	15-3	5	1	1	81,367	–	–	–
8-1	5	1	2	235,176	–	–	–	16-1	5	1	2	128,970	–	–	–
8-2	5	1	2	135,408	–	–	–	16-2	5	1	2	93,019	–	–	–
8-3	5	1	2	301,095	–	–	–	16-3	5	1	2	118,527	–	–	–

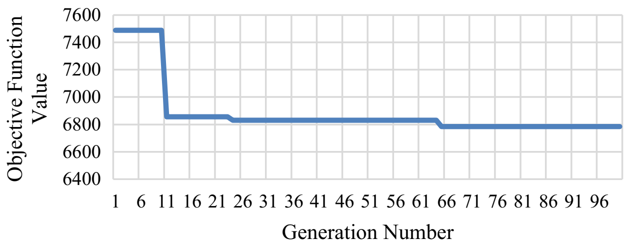


Fig. 6 Convergence graph of the matheuristic algorithm

Table 7 Factors and their levels

Factors	Factor levels		
	1	2	3
pop.size	10	20	30
cross.rate	0.5	0.6	0.7
mut.rate	0.1	0.2	0.3

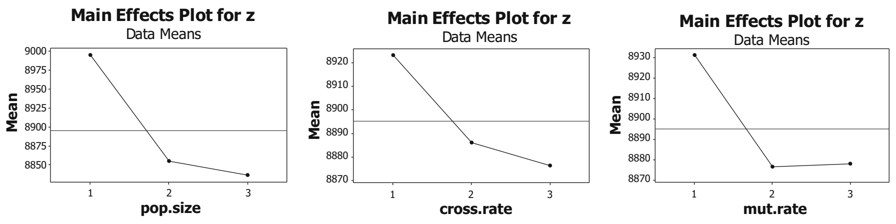


Fig. 7 Main effect plots for small problems

The parameters of the small instances were as follows: the population size was 30, the crossover rate was 0.7 and the mutation rate was 0.2.

Interaction between the population size and mutation rate is critical according to analysis of variance for medium instances, and the mutation rate is a critical factor according to analysis of variance for large instances. Related interaction and main effect plots are given in Figs. 8 and 9, respectively. The parameters of the medium and large instances were as follows: the population size was 10, the crossover rate was 0.6, and the mutation rate was 0.1.

The values of the termination parameters were $n_f = 100$, the solution time limit (n_t) was 7200 s for all instances, and they were carried out in three runs.

The matheuristic algorithm results of the small instances for different values of m (2, 3) are presented in Table 8. In this table, the first column shows the instance number (no). The second column shows the average solution time in seconds of three runs for each problem type. The last column describes the RPD2 value to demonstrate the success of the matheuristic algorithm.

In Table 5, the GAMS/Cplex solver provides the optimal solutions of all small instances. The matheuristic algorithm obtained the optimal solution for approximately 60 percent of the small instances. As shown in Table 8, RPD2 values completely depend

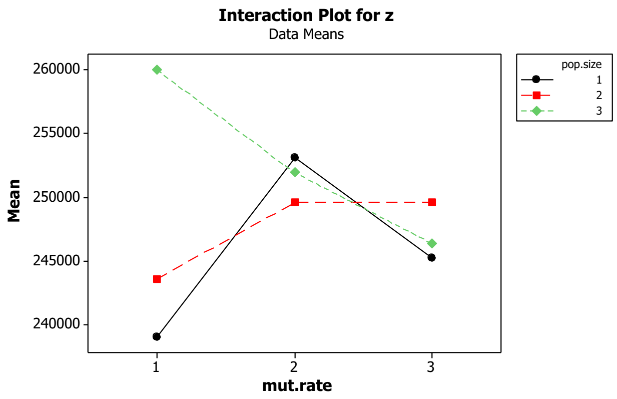
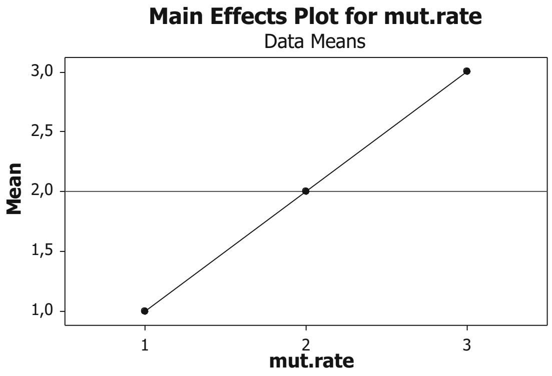


Fig. 8 Interaction plot for medium problems

Fig. 9 Main effect plots for large problems



on the number of machines and the number of resources. With the large number of machines ($m = 3$) and small number of resources ($g = 2$) for small instances (which represent problems between 9-1 and 12-3), RPD2 has peak percentage values (approximately 0.13). As expected, the RPD2 values of small instances with $g = 5$ (which represent problems between (5-1 and 8-3) and (13-1 and 16-3)) are extremely small and approximately zero.

In Table 9, the results of the matheuristic algorithm for medium instances are presented. This table is structured in a similar way as Table 8. As Table 9 shows, RPD2 values of eight medium instances with $m = 2$ (1-2, 2-2, (3-1)-(4-3)) could not be calculated since a feasible solution could not be provided by M1 or M2. RPD2 has a positive value for four instances (%10) since the mathematical models exhibit good performance. However, the proposed matheuristic algorithm has better results with significantly high RPD2 values for the remaining instances. The average RPD2 value is 0.7, and the widest RPD2 value that is larger than 1 is recorded for instance (1-1).

In Table 10, the results of the matheuristic algorithm for large instances are given. In this table, the first column shows the number of machines (m). The second column gives the number of large instances for each machine number. The following two columns

Table 8 Matheuristic algorithm results for small instances with n = 8

$m = 2$			$m = 3$		
MA			MA		
No	t_{Mean}	RPD2 (%)	No	t_{Mean}	RPD2 (%)
1-1	240	0.00	9-1	908	0.00
1-2	157	0.01	9-2	820	0.05
1-3	208	0.00	9-3	387	0.04
2-1	206	0.00	10-1	399	0.13
2-2	180	0.06	10-2	571	0.02
2-3	219	0.04	10-3	741	0.03
3-1	164	0.00	11-1	788	0.05
3-2	142	0.00	11-2	505	0.04
3-3	246	0.00	11-3	579	0.00
4-1	227	0.03	12-1	540	0.05
4-2	143	0.00	12-2	579	0.13
4-3	212	0.05	12-3	388	0.01
5-1	149	0.00	13-1	431	0.00
5-2	184	0.01	13-2	629	0.00
5-3	174	0.00	13-3	476	0.00
6-1	220	0.02	14-1	868	0.00
6-2	189	0.00	14-2	838	0.00
6-3	185	0.00	14-3	648	0.00
7-1	132	0.00	15-1	496	0.00
7-2	179	0.00	15-2	783	0.01
7-3	272	0.01	15-3	611	0.00
8-1	157	0.00	16-1	389	0.00
8-2	320	0.00	16-2	769	0.00
8-3	240	0.00	16-3	455	0.01

show the number of instances that are solved in 7200 s by mathematical models and matheuristic algorithms, respectively. The last two columns describe the averages of the minimum objective values of the three runs and solution times obtained by the proposed matheuristic algorithm. The GAMS/Cplex solver could obtain a feasible solution in 7200 s only for instance (8-1). The RPD2 value of regarding instance is 0.32. The proposed matheuristic algorithm provides feasible solutions for all large instances in a reasonable time. However, instances with $m = 8$ reach running time limits before the number of generations could not be complete, and the average of solution times is equal to the time limit (7200 s).

Table 9 Matheuristic Algorithm results for medium instances with n=40

$m = 2$			$m = 6$		
MA			MA		
No	t_{Mean}	RPD2 (%)	No	t_{Mean}	RPD2 (%)
1-1	3452	− 2.38	9-1	5910	− 0.10
1-2	3891	–	9-2	4116	− 0.44
1-3	3393	− 0.89	9-3	3433	− 0.26
2-1	3210	− 0.74	10-1	7200	− 0.77
2-2	3477	–	10-2	7200	− 0.10
2-3	3223	− 2.06	10-3	7200	− 0.25
3-1	3563	–	11-1	2744	− 0.65
3-2	3754	–	11-2	3914	− 0.51
3-3	3820	–	11-3	3107	− 0.41
4-1	3856	–	12-1	4003	− 2.53
4-2	3559	–	12-2	4113	− 1.98
4-3	6003	–	12-3	4001	− 2.02
5-1	3110	− 0.13	13-1	3168	− 0.02
5-2	3115	0.07	13-2	3162	0.05
5-3	3108	− 0.10	13-3	3165	0.14
6-1	3107	− 0.08	14-1	3176	0.06
6-2	3184	− 0.10	14-2	6309	− 0.14
6-3	3103	− 0.09	14-3	3337	− 0.02
7-1	3105	− 0.32	15-1	3015	− 0.45
7-2	3112	− 0.26	15-2	3124	− 0.55
7-3	3105	− 0.25	15-3	3148	− 0.44
8-1	3152	− 1.25	16-1	3656	− 0.76
8-2	3105	− 0.60	16-2	3696	− 1.06
8-3	3094	− 1.67	16-3	6038	− 1.26

Table 10 Matheuristic algorithm results for large instances with n = 100

m	Count	M1/M2	MA	$\overline{Z_{\text{MA}}}$	$\overline{t_{\text{MA}}}$
$n = 100$					
2	24	1	24	914713	3623
8	24	–	24	917083	7200

5 Conclusion

In this study, an identical parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and multiple copies of shared resources was analyzed. To the best of our knowledge, this problem has not been studied before. A MIP model is proposed. However, to compare the performance of mathematical model M1, a modification of the makespan model that is presented in Avalos-Rosales et al. (2015) is presented and is called M2. To overcome large instances, a matheuristic algorithm is developed. Initial solutions of the proposed algorithm are generated by reduced versions of M1 and M2. We executed computational experiments based on randomly generated instances with small, medium and large sizes. Optimal schedules were obtained for almost all small problems within a reasonable time for M1 and M2. However, M2 exhibits significantly improved performance in the manner of solution time for small problems. In medium instances, although M2 gives a rather high-quality solution, M1 offers more feasible solutions. Furthermore, the RPD2 values are quite wide, which indicates that MA shows clearly good performance. Finally, feasible solutions are obtained for all large instances in a reasonable time with the proposed matheuristic algorithm.

For future research, this problem can be considered multiobjective. For example, total tardiness is a significant objective for this kind of problem. Furthermore, the performance of other meta-heuristic algorithms, such as tabu search and simulated annealing, could be investigated to solve this problem.

Acknowledgements This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Afzalirad, Rezaeian, & (2016) Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions. Comput Ind Eng 98:40–52

Alagoz O, Azizoglu M (2003) Rescheduling of identical parallel machines under machine eligibility constraints. Eur J Oper Res 149(3):523–532

Arnaout JP (2010) Heuristics for the maximization of operating rooms utilization using simulation. Simulation 86(8–9):573–583

Avalos-Rosales O, Angel-Bello F, Alvarez A (2015) Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. Int J Adv Manuf Technol 76:1705–1718

Billaut JC, Croce Federico D, Grosso A (2015) A single machine scheduling problem with two-dimensional vector packing constraints. Eur J Oper Res 243:399–411

Chan TS, Choy KL, Bibhushan (2011) A genetic algorithm-based scheduler for multiproduct parallel machine sheet metal job shop. Expert Syst Appl 38(7):8703–8715

Chaudhry IA, Drake PR (2009) Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using genetic algorithms. Int J Adv Manuf Technol 42(5–6):581–594

Chen CL, Chen CL (2009) Hybrid meta-heuristics for unrelated parallel machine scheduling with sequence-dependent setup times. Int J Adv Manuf Technol 43(1–2):161–169

Chung SH, Tai YT, Pearn WL (2009) An effective scheduling approach for maximizing polyimide printing weighted throughput in cell assembly factories. IEEE Trans Electron Packag Manuf 32(3):185–197

Driessel R, Moench L (2009) Scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints and ready times using variable neighborhood search. In: International Conference on Computers and Industrial Engineering, Troyes, France, July 06-09

Driessel R, Moench L (2011) Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times. Comput Ind Eng 61(2):336–345

Edis EB, Oguz C (2012) Parallel machine scheduling with flexible resources. Comput Ind Eng 63(2):433–447

Edis EB, Ozkarahan I (2011) A combined integer/constraint programming approach to a resource constrained parallel machine scheduling problem with machine eligibility restrictions. Eng Optim 43(2):135–157

Edis EB, Ozkarahan I (2012) Solution approaches for a real-life resource-constrained parallel machine scheduling problem. Int J Adv Manuf Technol 58(9–12):1141–1153

Edis EB, Oguz C, Ozkarahan I (2012) Solution approaches for simultaneous scheduling of jobs and operators on parallel machines. J Fac Eng Arch Gazi Univ 27(3):527–535

Eliiyi D, Azizoglu M (2009) A fixed job scheduling problem with machine-dependent job weights. Int J Prod Res 47(9):2231–2256

Fanjul-Peyro L, Perea F, Ruiz R (2017) Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. Eur J Oper Res 260(2):482–493

Gacias B, Artigues C, Lopez P (2010) Parallel machine scheduling with precedence constraints and setup times. Comput Oper Res 37(12):2141–2151

Gedik R, Rainwater C, Nachtmann H, Pohl E (2016) Analysis of a parallel machine scheduling problem with sequence dependent setup times and job availability intervals. Eur J Oper Res 251(2):640–650

Gokhale R, Mathirajan M (2012) Scheduling identical parallel machines with machine eligibility restrictions to minimize total weighted flow time in automobile gear manufacturing. Int J Adv Manuf Technol 60(9–12):1099–1110

Guimaraes L, Klabjan D, Almada-Lobo B (2013) Pricing, relaxing and fixing under lot sizing and scheduling. Eur J Oper Res 230:75–81

Joo CM, Kim BY (2012) Parallel machine scheduling problem with ready times, due times and sequence-dependent setup times using meta-heuristic algorithms. Eng Optim 44(9):1021–1034

Keskinturk T, Yildirim MB, Barut M (2012) An ant colony optimization algorithm for load balancing in parallel machines with sequence-dependent setup times. Comput Oper Res 39(6):1225–1235

Kim BK, Kim YD (2011) Heuristic algorithms for assigning and scheduling flight missions in a military aviation unit. Comput Ind Eng 61(4):1309–1317

Lee K, Leung JYT, Pinedo ML (2013) Makespan minimization in online scheduling with machine eligibility. Ann Oper Res 204(1):189–222

Li X, Yalaoui F, Amodeo L (2010) A multi objective meta-heuristic with a fuzzy logic controller for solving a scheduling problem. In: Computational intelligence: foundations and applications: proceedings of the 9th international FLINS conference, Emei, CHINA, August 02–04

Li K, Shia Y, Yanga S, Cheng B (2011) Parallel machine scheduling problem to minimize the makespan with resource dependent processing times. Appl Soft Comput 11(8):5551–5557

Li X, Chehade H, Yalaoui F, Amodeo L (2012) Fuzzy logic controller based multi-objective meta-heuristics to solve a parallel machines scheduling problem. J Mult Val Logic Soft Comput 18(5–6):617–636

Lin SW, Lee ZJ, Ying KC, Lu CC (2011) Minimization of maximum lateness on parallel machines with sequence-dependent setup times and job release dates. Comput Oper Res 38(5):809–815

Liu M, Wu C (2003) Scheduling algorithm based on evolutionary computing in identical parallel machine production line. Robot Comput Integr Manuf 19(2003):401–407

Montoya-Torres JR, Soto-Ferrari M, Gonzalez-Solano F, Alfonso-Lizarazo EH (2009) Machine scheduling with sequence-dependent setup times using a randomized search heuristic. In: International conference on computers and industrial engineering troyes, France, July 06–09

Montoya-Torres JR, Soto-Ferrari M, Gonzalez-Solano F (2010) Production scheduling with sequence-dependent setups and job release times. Dyn Colomb 77(163):260–269

Nessah R, Chengbin C, Yalaoui F (2007) An exact method for $\mathrm{P_m/sds/\Sigma_{i=1}^{n} c_i}$ problem. Comput Oper Res 34(9):2840–2848

Park T, Lee T, Kim CO (2012) Due-date scheduling on parallel machines with job splitting and sequence-dependent major/minor setup times. Int J Adv Manuf Technol 59(1–4):325–333

Pinedo ML (2009) Planning and scheduling in manufacturing and services. Springer, New York

Pinedo ML (2011) Scheduling theory, algorithms, and systems. Springer, New York

Ruiz R, Andrés-Romano C (2011) Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times. Int J Adv Manuf Technol 57(5–8):777–794

Singha A, Rossi A, Sevaux M (2012) Matheuristic approaches for Q-coverage problem versions in wireless sensor networks. Eng Optim 45(5):609–626

Su LH, Chang WY, Chou FD (2011) Minimizing maximum lateness on identical parallel machines with flexible resources and machine eligibility constraints. Int J Adv Manuf Technol 56(9–12):1195–1204

Tahar DN, Yalaoui F, Chu C, Amodeo L (2006) A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times *International*. J Prod Econ 99:1–2

Turker AK, Sel C (2011) A hybrid approach on single server parallel machines scheduling problem with sequence-dependent setup times. J Fac. Eng Arch Gazi Univ 26(4):731–740