Computers and Operations Research 104 (2019) 159–173

Contents lists available at ScienceDirect

Computers and Operations Research

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

Exact methods for order acceptance and scheduling on unrelated parallel machines

Shijin Wang*, Benyan Ye

School of Economics and Management, Tongji University, Shanghai 200092, China

ARTICLE INFO

Article history:
Received 7 July 2018
Revised 29 November 2018
Accepted 18 December 2018
Available online 21 December 2018

Keywords:
Order acceptance and scheduling
Unrelated parallel machines
Mixed-integer programming
Branch-and-bound

ABSTRACT

This paper studies an order acceptance and scheduling (OAS) problem on unrelated parallel machines to maximize the total net revenue of accepted orders, which is the difference between sum of revenues and total weighted tardiness. Two mixed-integer programming (MIP) models are formulated, which are further improved with various enhancement techniques. A formulation-based branch-and-bound algorithm is developed in an attempt to handle complicated instances following the principle of "divide and conquer". Extensive computational experiments on various instances are conducted, and the results demonstrate the efficiency of the enhancement techniques for the formulations, as well as the effectiveness and efficiency of the formulation-based branch-and-bound algorithm. The proposed branch-and-bound algorithm can optimally solve instances with up to 50 jobs and different number of machines within the time limit of half an hour.

1. Introduction

To maintain operational agility and flexibility, many companies from different industries, such as engineering tooling, industrial boilers, construction and contracting, adopt the make-to-order (MTO) operational philosophy, thereby laying more focus on customer satisfaction (Mestry et al., 2011). Additionally, there exists an increase in popularity of the MTO philosophy in the service industry, particularly with regard to E-commerce and O2O takeout & catering services offered within restaurants. For example, the ele.me online platform for O2O takeout & catering service covered more than 200 cities in China and had served more than 260 million customers by June 2017 (source from www.ele.me). In this context, how to coordinate operations and sales for effective use of available resource (or limited capacity) is a big challenge for improving customer satisfaction meanwhile obtaining high profit margins.

The order acceptance and scheduling (OAS) problem arises in different MTO production and/or service systems, wherein limited production and/or service capacity and order-delivery requirements necessitate the use of selective order acceptance to satisfy distinct requirements of customers whilst also maximizing total revenue (profit) (Cesaret et al., 2012; Rom and Slotnick, 2009; Silva et al., 2018; Slotnick and Morton, 2007; Wang et al., 2015).

The OAS problem requires one to simultaneously determine which orders should be accepted for processing as well as their corresponding schedule. The complexity of the problem due to their combinatorial nature and intertwined decisions makes optimization extremely difficult, as the problem typically is NP-hard (Ghosh, 1997). Such problems, however, invariably capture the rich and realistic classes of MTO processing, thereby making them easy to explain and tempting to be attempted and solved optimally.

In this paper, a deterministic OAS problem on unrelated parallel machines is addressed, since the prevalence of actual manufacturing environments and service industries are typically equipped with unrelated parallel machines. Typical applications of unrelated parallel machine scheduling include—but are not limited to—semiconductor manufacturing (Şen and Bülbül, 2015; Detienne et al., 2011; Shim and Kim, 2007), multiprocessor computer (Fanjul-Peyro and Ruiz, 2010), operating rooms in hospitals (Fanjul-Peyro and Ruiz, 2012), and car factories and food processing plant (Fanjul-Peyro et al., 2017). For the problem under study, there exists a pool of potential orders (jobs) with known processing times, due dates, revenues, and penalty tardiness weights. The objective here is to maximize the total net revenue, which refers to the difference between the sum of revenues obtained from accepted jobs and total weighted tardiness. This paper investigates the problem from an exact solution viewpoint. The contributions of this research are as follows:

(1) Two MIP formulations are proposed: one is with a dummy job and another is based on linear ordering variables. Formulation

* Corresponding author.
E-mail address: shijinwang@tongji.edu.cn (S. Wang).

https://doi.org/10.1016/j.cor.2018.12.016
0305-0548/

tightening and valid inequalities are proposed to improve the efficiency of the two MIP formulations.
(2) A formulation-based branch-and-bound algorithm is developed based on the idea of "divide and conquer", in which the branch is to determine how many jobs should be accepted, followed by the subproblem of unrelated parallel machine scheduling.
(3) The computational results on various instances show the efficiency of the formulations with enhancement techniques, and demonstrate the efficacy of the proposed branch-and-bound algorithm.

The rest of this paper is organized as follows. Section 2 presents literature review relevant to the problem. In Section 3, two MIP models are formulated. In Section 4, some enhancement techniques for the MIP models are presented. The proposed formulation-based branch-and-bound algorithm is described in Section 5. In Section 6, extensive computational experiments are conducted to evaluate the performance of the developed models and the branch-and-bound algorithm. Lastly, Section 7 concludes the paper and suggests some future research directions.

2. Literature review

2.1. Research streamlines of OAS problems

The OAS problem and its variants have been extensively investigated for more than two decades (Esmaeilbeigi et al., 2016). Interested readers can refer to the survey by Slotnick (2011) and references herein for further details. The scheduling problem with rejection costs is an equivalent version of the OAS problem. The reader is referred to the survey by Shabtay et al. (2013) for more details. Equivalently, the prize-collecting scheduling problem, wherein job acceptance is not mandatory, but their acceptance is awarded a prize in objective value, is also highly related to the OAS problem (Cordone et al., 2018; Wang and Tang, 2010).

The OAS problems have been studied with various objective functions, including (1) maximization of the total net revenue as the difference between the sum of revenues and total weighted tardiness (Cesaret et al., 2012; Chaurasia and Singh, 2017; Emami et al., 2017; 2016; Esmaeilbeigi et al., 2016; Geramipour et al., 2017; Lei and Guo, 2015; Lin and Ying, 2013; Nobibon and Leus, 2011; Oğuz et al., 2010; Silva et al., 2018; Slotnick and Morton, 2007; Wang et al., 2015; 2013; 2013b; Wu et al., 2018; Xiao et al., 2015), (2) maximization of the total net revenue as the difference between sum of revenues and total weighted lateness (Ghosh, 1997; Lewis and Slotnick, 2002; Slotnick and Morton, 1996), (3) minimization of the makespan of the accepted jobs plus the total penalties of all rejected jobs (Bartal et al., 2000; Ou and Zhong, 2017; Ou et al., 2016; 2015; Zhong and Ou, 2017), (4) maximization of the total net profit as the difference between the sum of revenues and costs of using resources (Mestry et al., 2011), (5) minimization of the weighted sum of the maximum lead time of accepted orders and total cost of rejecting and delivering orders (Jiang et al., 2017), and (6) minimization of the makespan (Fanjul-Peyro and Ruiz, 2012; Rahman et al., 2015).

The current researches on OAS problems focus more on two streamlines with objective functions (1) and (3) mentioned above. For OAS problems with objective function (1), various formulations, exact methods, and heuristic and meta-heuristic methods have been developed, as shown in the following Section 2.2. For objective function (3), the existing researches emphasize more on the development of polynomial-time approximation methods with the proof of the worst-case bound.

2.2. Related works on OAS problems with total revenue and tardiness

As previously stated, this paper focuses on the OAS problem on unrelated parallel machines with the objective of maximizing the total net revenue with tardiness related penalties, since tardiness penalties cause loss of revenue. Extant researches with this objective have been performed under different machine environments, including single machine, flow shop and parallel machine.

In the single-machine context, Slotnick and Morton (2007) developed a branch-and-bound (B&B) algorithm with linear programming (LP)-relaxation-based bounds, based on which several heuristic methods were developed. Nobibon and Leus (2011) proved that there is no constant-factor approximation algorithm for the problem. Two LP formulations and two B&B algorithms were proposed to obtain exact solutions. Silva et al. (2018) developed three exact approaches and an iterative local search based heuristic for the problem with sequence-dependent setup times (SDST). Three exact approaches are arc-time-indexed formulation, a B&B with Lagrangian relaxation and a branch and price (B&P). For the same problem, Nguyen (2016) proposed a learning and optimizing system. Geramipour et al. (2017) developed a heuristic method and two B&B procedures. There also exist other heuristic and meta-heuristic methods, including genetic algorithms (Rom and Slotnick, 2009), tabu search method (Cesaret et al., 2012), artificial bee colony (Lin and Ying, 2013) and hybrid evolutionary algorithms (Chaurasia and Singh, 2017). For problems with the additional consideration of a deadline, (Oğuz et al., 2010) proposed several problem-feature-based heuristic methods.

Compared to abundant researches in the single machine environments, the OAS problems on multiple machine are still under exploration. The inherent complexity of multiple machines with the consideration of order acceptance further complicates the determination of optimum or near-optimum solutions, which is worth for more exploration. In the following, we mainly review the studies about the OAS problems on multiple machines with tardiness related penalties.

For two machine flow shops, (Wang et al., 2013) proposed two MIP models and B&B algorithms. The proposed B&B algorithms can solve instances with up to 20 jobs to optimality within a time limit of 1 h. Later, they developed a modified artificial bee colony algorithm for the same problem (Wang et al., 2013b). Esmaeilbeigi et al. (2016) presented two new MIP formulations and compared them against models in Wang et al. (2013) in terms of size complexity and number of disjunctive constraints. The computational results show the efficiency of their models and the enhancements, which can solve optimally instances with up to 100 jobs within a limit of half an hour.

For OAS problems in permutation flow shops, Xiao et al. (2012) developed a simulated annealing algorithm. Later, Xiao et al. (2015) proposed a two-phase genetic algorithm for the problem in non-permutation flow shops. Lei and Guo (2015) developed a parallel neighborhood search method for solving a bi-objective OAS problem in flow shops to simultaneously minimize the makespan and maximize the total net revenue.

2.2.1. Related works on OAS problems on parallel machines

There are relatively few papers in the literature that study the OAS problem on parallel machines with the objective of maximization of total net revenue with tardiness related penalties.

For the first time, Wang et al. (2015) studied the OAS problem on two identical parallel machines and developed two heuristic methods and one exact algorithm based on problem properties and the Lagrangian relaxation. The experimental results show that the exact algorithm can solve the problems with up to 15 jobs within a limit of an hour. This study differs from

that of Wang et al. (2015) on two fronts. First, in this study, the OAS problem on general multiple unrelated parallel machines is considered, as opposed to two identical parallel machines in Wang et al. (2015). Secondly, our solution procedures are considerably different, since we focus more on the formulation-based branch-and-bound method.

Emami et al. (2016) laid more focus on robust scheduling with the assumption of uncertain revenue and job-processing times in the problem with SDST. A Lagrangian relaxation algorithm with a cutting plane method was proposed, and it was used to solve problems with up to 40 orders and 6 machines. In Emami et al. (2017), a Benders decomposition method was developed to handle the same problem. Our work differs from Emami et al. (2016) and Emami et al. (2017) in two ways. First, our model is deterministic. Secondly, solution approaches developed in this study are different from theirs.

Wu et al. (2018) developed a water-flow-like algorithm for solving the OAS problem on identical parallel machines with SDST. The proposed algorithm was further improved by using a combination of particle swarm optimization and variable neighborhood search. Our work mainly differs from Wu et al. (2018) in two aspects in that it considers unrelated parallel machines and focuses on exact methods.

As has been demonstrated by previous works, OAS problems in various machine settings have been treated with both exact and heuristic approaches. To the best of the authors' knowledge, the largest OAS problem on unrelated parallel machines reported in the literature contained 40 orders and 6 machines with the time limit of an hour in the robust setting (Emami et al., 2016). Although robust versions of the problem are usually more difficult to solve than their deterministic counterparts, there is no report yet in literature on deterministic OAS problems on unrelated parallel machines. In this paper, we develop a formulation-based branch-and-bound algorithm which can solve instances with up to 50 jobs and 10 machines to optimality with the time limit of half an hour.

2.3. Related works on unrelated parallel machine scheduling problems

The OAS problems on unrelated machines could be reduced to a special case of the classical unrelated parallel machine scheduling problem by introducing a dummy machine to which all rejected jobs could be assigned with zero processing time, zero revenue and zero penalty. In the following, we mainly summarize the most important relevant works. The reader is referred to excellent researches by Chen and Powell (1999); Fanjul-Peyro et al. (2017); Fanjul-Peyro and Ruiz (2010, 2011); Pinedo (2008), and Fanjul-Peyro et al. (2019) for recent trends on deterministic unrelated parallel machine scheduling problems.

For unrelated parallel machine scheduling problem, Van De Vel (1993) developed an effective exact method by using of surrogate relaxation and duality to minimize makespan. Chen and Powell (1999) developed a column generation based exact method for identical, uniform, and unrelated parallel machine scheduling problems to minimize the total weighted completion time and weighted number of tardy jobs, separately. The proposed method can solve problems with up to 100 jobs to optimality within reasonable computation time. Unlu and Mason (2010) compared four different MIP formulations: time-indexed variables ($M1$), network variables ($M2$), assignment- and positional-date variables ($M3$) and linear-ordering variables ($M4$). They showed that $M4$ provides relatively shorter computation times.

For the unrelated parallel machine scheduling problems with SDST, Avalos-Rosales et al. (2015) reported optimal solutions for instances with up to 60 jobs and 5 machines. Based on a model similar to that in Avalos-Rosales et al. (2015); Tran et al. (2016) developed two exact decomposition-based methods, logic-based Benders decomposition and branch-and-check algorithm. The branch-and-check algorithm can solve problems with up to 60 jobs and 5 machines to optimality in less than 30 min. Fanjul-Peyro et al. (2019) reformulated the problem into a heterogeneous multiple traveling salesmen problem and developed several valid inequalities. A mathematical-programming-based algorithm was developed, which can obtain solutions close to optimality (deviation less than 1%) for instances with up to 1000 jobs and 8 machines within 3 h. In addition, nine MIP models were tested by using two commercial IP solvers, CPLEX and Gurobi. For problems with additional consideration of resources, Fanjul-Peyro et al. (2017) presented a novel reformulation technique based on the strip-packing model, and its superiority was demonstrated.

The above literature review demonstrates that for unrelated parallel machines, researchers have laid more focus on (i) novel reformulations or extensions of basic formulations with valid inequalities; and (ii) decomposition-based exact methods. In this study, extensions to two MIP formulations for the OAS problem on unrelated parallel machines are developed, and a formulation-based branch-and-bound algorithm is proposed.

3. Mathematical description of OAS problem

In this section, after the problem description, two MIP formulations are presented.

There is a set of jobs denoted by $\mathcal{N} = \{1, 2, \ldots, n\}$, which are all ready at time zero. There is a set of machines $\mathcal{M} = \{1, 2, \ldots, m\}$, which are all available from time zero. Each job is processed non-preemptively on exactly one of the machines and the processing times of jobs on machines are $p_{ij} \in \mathbb{Z}^+$ (where $\mathbb{Z}$ denotes the set of positive integers), corresponding to the time required to process job j ($j \in \mathcal{N}$) on machine i ($i \in \mathcal{M}$). If a job j is accepted, its revenue is $u_j \in \mathbb{Z}^+$. Then, it will be decided which machine should be assigned to process the job. Each job has a due date, denoted by $d_j \in \mathbb{Z}^+$. Let C_j denote the completion time of job j. We assume that there is a delay penalty, $w_j \in \mathbb{Z}^+$, for each unit of the completion time that exceeds d_j. The total penalty cost for job j is denoted by $w_j T_j$, where $T_j = \max\{0, C_j - d_j\}$. There is no penalty or reward for early delivery. The net revenue for each job $j \in \mathcal{N}$ is defined by $\pi_j = u_j - w_j T_j$. The schedule is to decide which orders should be accepted and if accepted, which machines should be assigned to process those accepted jobs, as well as the sequence of jobs on each machine. The objective is to maximize the total net revenue.

Two MIP formulations are considered in this study. The first one is with a dummy job (as described in Avalos-Rosales et al. (2015) and Tran et al. (2016)), since this may represent the most efficient MIP for unrelated parallel machine scheduling problems with SDST (Fanjul-Peyro et al., 2019). The second one is based on linear ordering variables, since in accordance with experimental evaluations reported in Unlu and Mason (2010), models with linear ordering variables yield much shorter computation time compared to other formulations. To the best of the authors' knowledge, there is no work that compares these two MIP formulations on OAS problems. Hence, we attempted to gain some insight by performing computational experiments with these two formulations.

3.1. The MIP with a dummy job

The model is based on the concept of a dummy job in Avalos-Rosales et al. (2015) and Tran et al. (2016). To formulate the problem, let

- $\mathcal{N}_0$: a set of jobs to be scheduled, including a dummy job (denoted by 0), with processing time of the dummy job on each machine being 0, i.e., $p_{i0} = 0$.
- L: a large enough integer.

Additionally, we define three binary decision variables and two continuous ones in the following.

- x_j: 1 if job j is accepted, 0 otherwise.
- y_{ij}: 1 if job j is assigned on machine i, 0 otherwise.
- z_{ijk}: 1 if job k is processed immediately after job j on machine i, 0 otherwise.
- C_j: completion time of job j.
- T_j: tardiness of job j.

The resultant MIP model is as follows.

$$[\textbf{MIP1}] \quad \max \quad Z = \sum_{j\in\mathcal{N}} (u_j x_j - w_j T_j) \tag{1a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{M}} y_{ij} = x_j, \qquad \forall j \in \mathcal{N}; \tag{1b}$$

$$y_{ij} = \sum_{j\in\mathcal{N}_0, j\neq k} z_{ijk}, \qquad \forall k \in \mathcal{N}_0, i \in \mathcal{M}; \tag{1c}$$

$$y_{ij} = \sum_{k\in\mathcal{N}_0, j\neq k} z_{ijk}, \qquad \forall j \in \mathcal{N}_0, i \in \mathcal{M}; \tag{1d}$$

$$y_{i0} = 1, \qquad \forall i \in \mathcal{M}; \tag{1e}$$

$$C_k + L(1 - z_{ijk}) \geq C_j + p_{ik}, \qquad \forall j \in \mathcal{N}_0, k \in \mathcal{N}, j \neq k, i \in \mathcal{M}; \tag{1f}$$

$$C_0 = 0; \tag{1g}$$

$$T_j \geq C_j - d_j, \qquad \forall j \in \mathcal{N}; \tag{1h}$$

$$C_j, T_j \geq 0, \qquad \forall j \in \mathcal{N}; \tag{1i}$$

$$x_j, y_{ij} \in \{0, 1\}, \qquad \forall j \in \mathcal{N}, i \in \mathcal{M}; \tag{1j}$$

$$z_{ijk} \in \{0, 1\}, \qquad \forall j, k \in \mathcal{N}_0, j \neq k, i \in \mathcal{M}; \tag{1k}$$

The objective function (1a) maximizes the total net revenue, and can be expressed as $\sum_{j\in\mathcal{N}} x_j(u_j - w_j T_j)$. However, since $T_j = 0$ if job j is rejected, (1a) is equivalent (Emami et al., 2017; 2016; Esmaeilbeigi et al., 2016; Wang et al., 2013). Constraints (1b) ensure that only a job is accepted can the job be assigned on a machine. Constraints (1c) and (1d) enforce that if job j is assigned on machine i, there must exactly be one job that precedes and succeeds the processing of job j on machine i. Constraints (1e) assign the dummy job on each machine. The dummy job is used to represent the start and end of a sequence of jobs on a machine. Constraints (1f) compute the completion time of a job by considering the job sequence. Constraint (1g) places the dummy job at time 0. Constraints (1h) capture the tardiness of each job. If a given job is not accepted, the tardiness of that job equals zero in accordance with the objective function. Constraints (1i)-(1k) give the region of decision variables.

3.2. The linear-ordering-based MIP model

The second model is based on the linear ordering variables (Unlu and Mason, 2010). Here, decision variables x_j, y_{ij}, C_j, and T_j are defined as same as those in Section 3.1. Two other decision variables are defined as,

- z_{ijk}: 1 if job k is processed after job j on machine i (not necessary immediately), 0 otherwise.
- p_j: actual processing time of job j.

The formulation is as follows.

$$[\textbf{MIP2}] \quad \max \quad Z = \sum_{j\in\mathcal{N}} (u_j x_j - w_j T_j) \tag{2a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{M}} y_{ij} = x_j, \qquad \forall j \in \mathcal{N}; \tag{2b}$$

$$z_{ijk} + z_{ikj} \geq \frac{(y_{ij} + y_{ik})}{2} - 0.5, \qquad \forall j, k \in \mathcal{N}, j \neq k, i \in \mathcal{M}; \tag{2c}$$

$$z_{ijk} + z_{ikj} \leq \frac{(y_{ij} + y_{ik})}{2}, \qquad \forall j, k \in \mathcal{N}, j \neq k, i \in \mathcal{M}; \tag{2d}$$

$$\sum_{i\in\mathcal{M}} p_{ij} y_{ij} = p_j, \qquad \forall j \in \mathcal{N}; \tag{2e}$$

$$C_k + L(1 - z_{ijk}) \geq C_j + p_k, \qquad \forall j, k \in \mathcal{N}, j \neq k, i \in \mathcal{M}; \tag{2f}$$

$$C_j + L z_{ijk} + L(2 - y_{ij} - y_{ik}) \geq C_k + p_j, \qquad \forall j, k \in \mathcal{N}, j \neq k, i \in \mathcal{M}; \tag{2g}$$

$$C_j \geq p_j, \qquad \forall j \in \mathcal{N}; \tag{2h}$$

$$T_j \geq C_j - d_j, \qquad \forall j \in \mathcal{N}; \tag{2i}$$

$$p_j, C_j, T_j \geq 0, \qquad \forall j \in \mathcal{N}; \tag{2j}$$

$$x_j, y_{ij}, z_{ijk} \in \{0, 1\}, \qquad \forall j, k \in \mathcal{N}, j \neq k, i \in \mathcal{M}; \tag{2k}$$

The objective function (2a) maximizes the total net revenue. Constraints (2b) ensure that only a job is accepted can the job be assigned on a machine. Constraints (2c) and (2d) enforce that only two jobs j and k are assigned to the same machine i can jobs j and k have a sequence on machine i. Constraints (2e) calculate the actual processing time of job j. Constraints (2f)–(2h) compute the completion time of each job by considering the sequence of jobs. Constraints (2i) capture the tardiness of each job. Constraints (2j)–(2k) give the region of decision variables.

4. Enhancement of proposed MIP formulations

The OAS problem on unrelated parallel machines is NP-hard, since it can be reduced to a classical unrelated parallel machine problem with a dummy machine (see Appendix), which is known to be NP-hard in the strong sense (Fanjul-Peyro and Ruiz, 2012; Fleszar and Hindi, 2018). It is, therefore, necessary to develop enhancement techniques to improve the performance of proposed MIP models.

It is easy to know that there is no idle time between any two consecutive accepted jobs on a machine. In addition, the MIP models mentioned above can be enhanced by using some techniques.

4.1. Moderating big-M coefficients

In the MIP models, there is a big-M coefficient L within constraints (1f) and constraints (2f)–(2g), which can be safely set as $L_{\max} = \sum_{j\in\mathcal{N}} \max_{i\in\mathcal{M}} p_{ij}$. In addition, smaller values of L could be calculated in accordance with the proposition in Esmaeilbeigi et al. (2016), which states that in an optimum schedule, $C_j^U = \frac{U_j}{w_j} + d_j$ is an upper bound for the completion time of an accepted job $j \in \mathcal{N}$. Let $L_{ijk} = \lfloor C_j^U \rfloor + p_{ik}$, wherein $j, k \in \mathcal{N}$, $j \neq k$, and $i \in \mathcal{M}$, and replace L in constraints (1f) and obtain new constraints (3) to replace constraints (1f).

$$C_k + L_{ijk}(1 - z_{ijk}) \geq C_j + p_{ik} \quad \forall j \in \mathcal{N}_0, k \in \mathcal{N}, j \neq k, i \in \mathcal{M} \tag{3}$$

It can easily be observed that L_{ijk} is a suitable value for L because if $z_{ijk} = 0$, the inequality $C_k + \lfloor C_j^U \rfloor + p_{ik} \geq C_j + p_{ik}$ holds. Note that the reason for using $\lfloor C_j^U \rfloor$ instead of C_j^U is that all parameters are positive integers.

Similarly, let $L_{jk} = \lfloor C_j^U \rfloor + \max_{i\in\mathcal{M}} p_{ik}$, $L_{kj} = \lfloor C_k^U \rfloor + \max_{i\in\mathcal{M}} p_{ij}$, so as to obtain the following two constraints by replacing L within constraints (2f) and (2g) in the MIP2 formulation.

$$C_k + L_{jk}(1 - z_{ijk}) \geq C_j + p_k, \quad \forall j, k \in \mathcal{N}, j \neq k, i \in \mathcal{M} \tag{4}$$

$$C_j + L_{kj} z_{ijk} + L_{kj}(2 - y_{ij} - y_{ik}) \geq C_k + p_j, \quad \forall j, k \in \mathcal{N}, j \neq k, i \in \mathcal{M} \tag{5}$$

4.2. Valid inequalities

In this section, some valid inequalities are developed, which can be appended to the MIP models or can be substituted in certain constraints in the MIP models.

The following valid bounding inequalities for the completion time and tardiness of each job $j \in \mathcal{N}$ can be directly added into MIP models Esmaeilbeigi et al. (2016).

$$T_j \leq \lfloor C_j^U \rfloor - d_j, \quad \forall j \in \mathcal{N} \tag{6}$$

$$C_j \leq \lfloor C_j^U \rfloor, \quad \forall j \in \mathcal{N} \tag{7}$$

Proposition 1. *For two jobs j and $k \in \mathcal{N}$, if $p_{ij} \leq p_{ik}$ for all machines $i \in \mathcal{M}$, $w_j \leq w_k$, $d_j \geq d_k$, $u_j \geq u_k$, and if at least one of the inequalities is strict, then job j dominates job k.*

Proof. This is inspired from the dominance rule in Cordone et al. (2018). Assuming an optimum schedule S contains job k but does not contain job j. Consider the schedule S' obtained by replacing job k by job j on the same machine, with job j starting in S' at the same time as job k starts in schedule S whilst leaving starting times of all other jobs unchanged. It is easy to realize that schedule S' is feasible, since $w_j \leq w_k$, $d_j \geq d_k$, $u_j \geq u_k$ (i.e., $C_j^U \geq C_k^U$), and $p_{ij} \leq p_{ik}$. The completion time of job j within schedule S' does not exceed that of job k within schedule S, and all other jobs remain unaffected. Since $d_j \geq d_k$ and $w_j \leq w_k$, the total weighted tardiness of schedule S' does not exceed that of schedule S. In addition, since $u_j \geq u_k$, the total revenue of schedule S' is not smaller compared to that of schedule S. Therefore, S' is also an optimum schedule. □

Proposition 1allows us to define a set of jobs that dominate job k, and such a set can be denoted by Δ_k. This implies that if job k is accepted for processing, then any job within the set Δ_k should also be accepted. Consequently, the following valid inequalities can be added into the MIP models.

$$x_k \leq x_j, \quad \forall\ j, k \in \mathcal{N}, j \neq k, j \in \Delta_k \tag{8}$$

Let Δ_k^i denote a set of jobs that dominates each job k on machine i, i.e., $\Delta_k^i = \{j | p_{ij} \leq p_{ik}, w_j \leq w_k, d_j \geq d_k, u_j \geq u_k\}$. The following proposition is proposed.

Proposition 2. *Consider a job k with a dominating subset Δ_k^i. Additionally, consider any job $j \in \Delta_k^i \cup \{k\}$ assigned on machine i. If $\sum_{h\in\Delta_k^i\cup\{k\}:\ C_h^U \leq C_j^U} p_{ih} > C_j^U$ is satisfied for any job $j \in \Delta_k^i \cup \{k\}$, then job k cannot be processed on machine i.*

Proof. If the above inequalities are satisfied, not all jobs assigned on machine i can be included in an optimum solution. Since $j \in \Delta_k^i$ dominates k, job k can be moved to another machine or be rejected to ensure the optimum schedule on machine i. □

Based on Proposition 2, for $j \in \Delta_k^i \cup \{k\}$, the following valid inequalities can be inserted into the MIP models.

$$\sum_{h\in\Delta_k^i\cup\{k\}:\ C_h^U \leq C_j^U} p_{ih} y_{ih} \leq C_j^U + (1 - y_{ij}) L_{max}, \quad \forall i \in \mathcal{M}, k \in \mathcal{N}, j \in \Delta_k^i \cup \{k\} \tag{9}$$

Proposition 3. *There exists an optimum solution, wherein job k can be assigned on machine i, for jobs j assigned on the same machine with C_j^U not exceeding C_k^U, the total processing time for jobs j (including job k) must not exceed C_k^U.*

Proof. Proof can be realized via contradiction. Assume that there exists an optimum solution, wherein job k can be assigned on machine i, for jobs j assigned to the same machine i with C_j^U no exceeding C_k^U, the total processing time for jobs j (including job k) exceeds C_k^U. There are two possible cases concerning job k in this optimum solution.

Case (1): Job k is scheduled after all jobs j. In this case, the completion time of job k must exceed C_k^U, which contradicts the optimality condition in terms of the upper bound of the completion time for an accepted job.

Case (2): Job k is not scheduled after all jobs j, and that its completion time does not exceed C_k^U. In this case, the completion time of at least one job must exceed the upper bound of the completion time, since the total processing time of jobs j, whose C_j^U does not exceed C_k^U, exceeds C_k^U. This also contradicts the optimality condition of the upper bound of the completion time for an accepted job.

This completes the proof. □

Based on this proposition, the following inequalities can be added into the MIP models.

$$\sum_{j\in\mathcal{N}:\ C_j^U \leq C_k^U} p_{ij} y_{ij} \leq C_k^U + (1 - y_{ik}) L_{max}, \quad \forall k \in \mathcal{N},\ i \in \mathcal{M} \tag{10}$$

Proposition 4. *If there exist two accepted jobs j and k assigned on machine i with $d_j \leq p_{ik}$, $p_{ij} \leq p_{ik}$, and $w_j \geq w_k$, then there is an optimum sequence wherein job k cannot be the first job on machine i.*

Proof. This is based on the corollary reported by Rinnooy Kan (1976), which states that "For single machine with the objective of total weighted tardiness minimization, if there are two jobs j and k with $d_j \leq C_k$, $p_j \leq p_k$, and $w_j \geq w_k$, then there is an optimal sequence in which job j appears before job k". If jobs j and k satisfying all these three conditions are accepted and assigned on machine i, job j appears before job k, thereby implying that job k cannot be the first job to be processed on the assigned machine. If job k is the first job on machine i, its completion time $C_k = p_{ik}$, and the proposition follows. □

A set of jobs that dominates job k on each machine can be defined, and such a set can be denoted by Ψ_{ik}. The following valid inequalities can be added into the MIP1 model.

$$z_{i0k} \le 1 - y_{ij}, \quad \forall\ k \in \mathcal{N},\ \ j \in \Psi_{ik},\ \ i \in \mathcal{M} \tag{11}$$

Proposition 5. *(Azizoglu and Kirca, 1999; Liaw et al., 2003) There exists an optimum schedule, wherein the sum of processing times of jobs processed on machine i does not exceed*

$$P_i^{lim} = \frac{1}{m}\left\{\sum_{j\in\mathcal{N}} \max_{i\in\mathcal{M}} p_{ij} + \sum_{h\in\mathcal{M},h\neq i} \max_{j\in\mathcal{N}} p_{hj}\right\}, \quad \forall\ i \in \mathcal{M}$$

Based on this proposition, the following inequalities can be added into the proposed MIP models,

$$\textstyle\sum_{j\in\mathcal{N}} p_{ij}y_{ij} \le P_i^{lim}, \quad \forall\ i \in \mathcal{M} \tag{12}$$

$$C_j - (1 - y_{ij})L_{max} \le P_i^{lim}, \quad \forall\ j \in \mathcal{N},\ i \in \mathcal{M} \tag{13}$$

Moreover, if $C_j^U > P_i^{lim}$ for all $i \in \mathcal{M}$, job j can directly be rejected.

Additionally, the first Emmons' rule says: for $1||\Sigma w_jT_j$, there exists an optimum sequence, wherein job j is processed before job k if $p_j \le p_k,\ w_j \ge w_k,\ d_j \le \max\{d_k, \sum_{h\in B_k} p_h + p_k\}$ (Emmons, 1969; Rinnooy Kan, 1976), where B_k denotes the set of jobs that must be processed before job k. If only the simplified version with $d_j \le d_k$ is considered, the following constraints are proposed and can be added into the MIP2 model.

$$z_{ijk} \ge z_{ikj}, \quad \text{if } d_j \le d_k,\ p_{ij} \le p_{ik} \text{ and } w_j \ge w_k, \forall\ i \in \mathcal{M}, j, k \in \mathcal{N}, j \neq k \tag{14}$$

With these enhancements, the following two MIP models can be obtained.

$$[\textbf{MIP1V}] \quad \max \quad Z = \sum_{j\in\mathcal{N}} (u_jx_j - w_jT_j) \tag{15a}$$

$$\text{s.t.} \quad \textit{constraints}\ (1b) - (1e), (1g) - (1k), \tag{15b}$$

$$(3), (6) - (13), \tag{15c}$$

$$[\textbf{MIP2V}] \quad \max \quad Z = \sum_{j\in\mathcal{N}} (u_jx_j - w_jT_j) \tag{16a}$$

$$\text{s.t.} \quad \textit{constraints}\ (2b) - (2e), (2h) - (2k), \tag{16b}$$

$$(4) - (10), (12) - (14), \tag{16c}$$

Based on the computational results provided in Section 6, the MIP2V model can be observed to be relatively more efficient compared to MIP1V. Consequently, in the formulation-based branch-and-bound algorithm proposed in the next section, the MIP2V model is employed for each node.

5. A formulation-based branch-and-bound algorithm

For the exact optimization, a formulation-based branch-and-bound (B&B) algorithm is designed based on the idea of "divide and conquer". The proposed algorithm branches on the number of accepted jobs n^A, which may assume values between 1 and n. Once the number of accepted jobs, n^A, is determined, the problem is reduced to an unrelated parallel machine scheduling problem with the objective of minimizing the total weighted tardiness, which can be explored by the formulations mentioned above. A lower and an upper bound on its best solution are computed. Subsequently, the MIP2V model with additional cuts related to the lower and upper bounds is employed to solve the problem at each node. The key elements of the proposed algorithm are explained as follows.

5.1. Upper bounding

Since the objective function comprises two parts (total revenue and total weighted tardiness), the problem can naturally be separated into two subproblems. The first deals with maximization of the total revenue while the second minimizes the total weighted tardiness. The difference between the two corresponding optimum values is an upper bound.

For a node with $n^A \le n$ accepted jobs, the upper bound on the total revenue can be achieved by selecting the maximum revenue subset of jobs with respect to u_j with the constraint of C_j^U. This upper-bound scheme is inspired from the maximum prize reported in Cordone et al. (2018). It corresponds to solving the following IP problem.

$$[\textbf{IP1}] \quad \max \quad Z_1 = \sum_{j\in\mathcal{N}} u_jx_j \tag{17a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{N}:\ C_j^U \le C_k^U} p_{ij}y_{ij} \le C_k^U + (1 - y_{ik})L_{max}, \qquad \forall k \in \mathcal{N},\ i \in \mathcal{M}; \tag{17b}$$

$$\sum_{i\in\mathcal{M}} y_{ij} = x_j, \qquad \forall j \in \mathcal{N}; \tag{17c}$$

$$\sum_{j\in\mathcal{N}} x_j = n^A, \tag{17d}$$

$$x_j, y_{ij} \in \{0, 1\}, \qquad \forall j \in \mathcal{N}, i \in \mathcal{M}; \tag{17e}$$

Constraints (17b) require that for each machine, the total processing time of all accepted jobs, for which C_j^U precedes C_k^U, does not exceed C_k^U. Constraints (17c) ensure that only an accepted job can be assigned on a machine. Constraint (17d) ensures that there are n^A accepted jobs.

For a node with n^A accepted jobs, the lower bound of the total weighted tardiness can be achieved by solving an assignment problem. This lower-bound scheme is based on the idea in Azizoglu and Kirca (1999) and Liaw et al. (2003).

Let $\beta_{ij}^t = 1$, if job j is scheduled in the tth position on machine i, 0 otherwise. Let $\hat{T}_{ij}^t = \max(\hat{C}_{ij}^t - d_j, 0)$ denote an estimate of the tardiness of job j scheduled in the tth position on machine i. If $\hat{C}_{ij}^t \le C_{ij}^t$ for all t, i and j (where C_{ij}^t is the actual completion time of job j in the tth position on machine i), the objective value of the following IP2 model is the lower bound of the total weighted tardiness for a node with n^A accepted jobs.

$$[\textbf{IP2}] \quad \min \quad Z_2 = \sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{M}}\sum_{t\in\mathcal{N}} w_j\hat{T}_{ij}^t\beta_{ij}^t \tag{18a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{t\in\mathcal{N}} \beta_{ij}^t = n^A, \tag{18b}$$

$$\sum_{j\in\mathcal{N}} \beta_{ij}^t \le 1, \qquad \forall t \in \mathcal{N}, i \in \mathcal{M}; \tag{18c}$$

$$\beta_{ij}^t \in \{0, 1\}, \qquad \forall j \in \mathcal{N}, t \in \mathcal{N}, i \in \mathcal{M}; \tag{18d}$$

Constraint (18b) enforces that there are n^A accepted jobs. Constraints (18c) guarantee the assignment of at most one job at each position on each machine. The following definition of $\hat{C}_{ij}^t$ satisfies the requirement of $\hat{C}_{ij}^t \le C_{ij}^t$ for any $t \in \mathcal{N}$, $i \in \mathcal{M}$ and $j \in \mathcal{N}$.

$$\hat{C}_{ij}^t = \begin{cases} \sum_{l=1}^{t} p_{i[l]} & \text{if } S_{ij} \le t \\ \sum_{l=1}^{t} p_{i[l]} + p_{ij} & \text{if } S_{ij} > t \end{cases}$$

where $p_{i[l]}$ denotes the processing time of the lth job when all n jobs are processed on machine i in the shortest processing time (SPT) order, and S_{ij} denotes the position of job j in this ordering.

For each node n^A, an upper bound can be obtained by separately solving the IP1 and IP2 models for Z_1 and Z_2, respectively. The upper bound can be expressed as $UB_{n^A} = Z_1 - Z_2$.

For the root node (for which n^A is not fixed), the IP1 model can be solved without constraint (17d) to obtain Z_1 while the IP2 model can be solved by changing constraint (18d)–(19). The upper bound, thus obtained, can be expressed as $UB^A = Z_1 - Z_2$.

$$\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{t\in\mathcal{N}} \beta_{ij}^t \le n \tag{19}$$

5.2. Lower bounding

5.2.1. A lower bound based on a heuristic method

A constructive heuristic method can be designed for each node of n^A within the branch-and-bound tree, as described in Algorithm 1. The basic idea of the method is to sort jobs according to a non-decreasing order of C_j^U (line 2) and subsequently assign each job to the first available machine (line 4). If the completion time exceeds its $\lfloor C_j^U \rfloor$, the job gets rejected (lines 5 and 6); otherwise, the job is scheduled (lines 7–11).

Algorithm 1 The Constructive Heuristic Method.

1: Compute C_j^U of each job j, $j \in \mathcal{N}$. Let $iter = 0$ and $Am_i = 0$ for all $i \in \mathcal{M}$.
2: Sort C_j^U in a non-decreasing order. Let $C_{[j]}^U$ denote C_j^U of jth job in the order, and let $p_{i[j]}$ denote the processing time of jth job assigned on machine i.
3: **while** $iter < n$ **do**
4: Set $j = iter$ and compute $p_{i[j]} + Am_i$ and $i \in \mathcal{M}$, and get machine i' with the $\text{argmin}_{i\in\mathcal{M}}\{p_{i[j]} + Am_i\}$. Let $temp = p_{i'[j]} + Am_{i'}$.
5: **if** $temp > \lfloor C_{[j]}^U \rfloor$ **then**
6: Job $[j]$ is not accepted.
7: **else**
8: Job $[j]$ is assigned on machine i'.
9: $Am_{i'}$ is updated as $p_{i'[j]} + Am_{i'}$.
10: The net revenue of job $[j]$, $\pi_{[j]} = u_{[j]} - w_{[j]} \max\{0, Am_{i'} - d_{[j]}\}$ is computed.
11: **end if**
12: $iter = iter + 1$.
13: **end while**
14: Obtain the total net revenue $\sum_{j\in\mathcal{N}} \pi_{[j]}$ and output the objective value, which is denoted by F_1.
15: Sort jobs by the weighted shortest processing time (WSPT) first order on each machine, and compute and output the objective value, which is denoted by F_2.
16: Obtain the lower bound $LB_1 = \max\{F_1, F_2\}$.

In line 2 of Algorithm 1, if we change the non-decreasing order by $\frac{u_j}{w_j d_j}$ instead of C_j^U, another lower bound LB_2 can be obtained. Let $LB^H = \max\{LB_1, LB_2\}$.

5.2.2. A formulation-based lower bound

The decision variables z_{ijk} in the MIP2V model can be relaxed to any real-valued number between 0 and 1, i.e., $0 \le z_{ijk} \le 1$, thereby resulting in the relaxed version of the MIP2V model, which is denoted by MIP2^{VR}. Once the relaxed MIP2^{VR} model is solved, the corresponding values of $\hat{x}_j$ and $\hat{y}_{ij}$ can be simultaneously obtained. Subsequently, using known values of $\hat{x}_j$ and $\hat{y}_{ij}$ and turning z_{ijk} back into a binary variable, the MIP2V model can be solved to obtain a lower bound, which is denoted by LB^R.

5.3. The complete procedure of branch-and-bound algorithm

The complete procedure of the proposed branch-and-bound algorithm is shown in Algorithm 2.

Algorithm 2 The formulation-based branch-and-bound (B&B) algorithm.

1: **Obtain the lower bound at the root node**. Let LB^* denote the best incumbent lower bound. It can be initialized with the constructive heuristic method (described in Section 5.2.1) and the formulation relaxation (in Section 5.2.2) and $LB^* = \max\{LB^H, LB^R\}$.
2: **Obtain the upper bound at the root node**. Let UB^* denote the best upper bound. It can be obtained by using methods described in Section 5.1. Let $UB^* = UB^A$.
3: **if** $LB^* = UB^*$ **then**
4: Terminate and return LB^*.
5: **else**
6: **Branching**. Each subproblem concerning the branch-and-bound tree can be indicated by setting the variable $n^A = \{1, 2, \cdots, n\}$. For each node of the branch-and-bound tree, it is processed as follows.
7:
- **Obtain the upper bound at a node**. For each node n^A, its upper bound can be obtained by using the method described in Section 5.1 and denoted by UB_{n^A}. If $UB_{n^A} < LB^*$, the node is pruned by optimality; else, UB_{n^A} is updated as $\min\{UB_{n^A}, UB^*\}$ for this node.
- **Obtain the lower bound at a node**. The lower bound for each node n^A can be obtained with methods described in Sections 5.2.1 and 5.2.2, and $LB_{n^A} = \max\{LB_{n^A}^R, LB_{n^A}^H\}$. If $LB_{n^A} > LB^*$, update the lower bound LB_{n^A} as $\max\{LB^*, LB_{n^A}\}$ for node n^A.
- **Directly solve a node**. If $LB_{n^A} = UB_{n^A}$ for node n^A, $LB^* = \max\{LB_{n^A}, LB^*\}$, and the problem related to the node need not be explored further.
- **Solve the subproblem at a node**. If node n^A is not pruned, solve the MIP2V model with two additional constraints: (1) $Z \ge LB_{n^A} + 1$ and (2) $Z \le UB_{n^A} - 1$. If feasible, the obtained solution is denoted by LB_V, and the best incumbent lower bound is updated as $LB^* = \max\{LB_V, LB^*\}$. If not, the node n^A is pruned by bound.

8: **end if**
9: Terminate and return LB^*.

We provide an example to illustrate the computation process involved in the proposed algorithm.

Example 5.1. There are $n = 10$ jobs and $m = 2$ machines. All job-related information, including p_{ij}, u_j, w_j, d_j, and $\lfloor C_j^U \rfloor$ is presented in Table 1.

Table 1
The job-related information for the example problem.

No.	p_{1j}	p_{2j}	u_j	w_j	d_j	$\lfloor C_j^U \rfloor$
1	3	5	4	9	5	5
2	3	5	9	4	14	16
3	1	1	3	3	12	13
4	4	3	10	1	7	17
5	4	5	10	8	5	6
6	4	5	9	7	10	11
7	3	2	3	8	5	5
8	4	5	10	10	6	7
9	3	3	4	7	10	10
10	1	4	2	5	12	12

Using the heuristic method described in Section 5.2.1, one can obtain $LB^H = 43$, and using the MIP2^{VR} formulation in Section 5.2.2, one can obtain $LB^R = 41$. Thus, $LB^* = \max\{43, 41\} = 43$.

Using the upper bounding method described in Section 5.1, one can obtain $UB^* = UB^A = 61$.

- For $n^A = \{1, \cdots, 10\}$, the corresponding value of $UB_{n^A} = \{10, 20, 30, 39, 48, 52, 56, 59, 61, -\}$; since $UB_{n^A} < LB^*$ for $n^A = 1, 2, 3, 4$, these nodes are pruned by optimality. For the node with $n^A = 10$, the IP1 model with bounds $LB^* = 43$ and $UB^* = 61$ is not feasible (UB_{10} is denoted by "-"); therefore, node $n^A = 10$ is pruned by bound.
- Subsequently, nodes $n^A = 5, 6, 7, 8, 9$ are explored.
  (1) For the node $n^A = 5$, $LB_5 = 43$ and $UB_5 = 48$. Based on these, $LB_V = 47$ is obtained by solving the MIP2V model with LB_5 and UB_5. Correspondingly, the best incumbent lower bound is updated as $LB^* = \max\{47, 43\} = 47$.
  (2) For the node $n^A = 6$, $LB_6 = 48$, $UB_6 = 52$ and $LB_V = 50$, then, LB^* is updated as $LB^* = 50$.
  (3) For the node $n^A = 7$, $LB_7 = 50$, $UB_7 = 56$ and $LB_V = 52$, then, LB^* is updated as $LB^* = 52$.
  (4) For the node $n^A = 8$, $LB_8 = 52$, $UB_8 = 59$ and $LB_V = 53$, then, LB^* is updated as $LB^* = 53$.
  (5) For the node $n^A = 9$, $LB_9 = 53$ and $UB_9 = 61$. However, with these additional constraints: (1) $\sum_{j \in \mathcal{N}} x_j = 9$; (2) $Z \geq LB_9 + 1$; and (3) $Z \leq UB_9 - 1$, the MIP2V model is infeasible; consequently, the node is pruned by bound.
- The optimum objective value is obtained, $LB^* = 53$.

Alternatively, if the branching on n^A is not increased by unity at each step, a parameter b can be introduced to conduct branching using $\lambda \leq n^A \leq \min\{\lambda + b - 1, n\}$, where $\lambda = 1, 1 + b, 1 + 2b, \cdots, \min\{1 + kb, n\}$ with $k = \lceil n/b \rceil$. In this case, constraints (17d) and (18b) should be correspondingly revised.

For better illustration, consider Example 1 with $b = 3$. Nodes can be defined as $1 \leq n^A \leq 3$, $4 \leq n^A \leq 6$, $7 \leq n^A \leq 9$, and $n^A = 10$. It is not clear how exactly does the value of b affect the efficiency of solving the MIP2V model in CPLEX. Thus, different b values could be tested to facilitate performance comparison. As described in Section 6, $b = 1$ performs best compared to $b = 5$ and $b = 10$.

6. Computational experiments

To evaluate the performance of the MIP models, the enhancement techniques, and the formulation-based branch-and-bound algorithm, an extensive computational analysis is conducted. Seven methods, i.e., the MIP1 method described in Section 3.1, the MIP2 in Section 3.2, the MIP1V and MIP2V in Section 4 solved with the IP solver CPLEX, the B&B algorithm with $b = 1, 5, 10$ (denoted as "BB1", "BB5", and "BB10", respectively), are used to test problem instances to compare the performance of the models with CPLEX as well as that of enhancement techniques and the proposed B&B algorithm. In the MIP1 and MIP2 models, $L_{\max}$ is used instead of any large integer L. A time limit of 1800 CPU seconds is used for each instance.

The MIP models and the B&B algorithm are coded in the C++ programming language in Xcode 8.3.3, and run on MacOS machine with a 3.2 GHz processor and 8GB memory. CPLEX 12.6 is used as the IP solver with its configuration being set to default. The results are available online at https://pan.baidu.com/s/1N_vWO3iM1gccmW9ayyuQOg#list/path=%2F.

6.1. Data generation

Sixteen instance classes are generated with different values of n and m. n takes from the set {20, 30, 40, 50} and m takes from the set {2, 3, 5, 10}. Each class with fixed values of n and m contains 9 subclasses, and each subclass has 5 randomly generated instances.

For each subclass, random instances are generated as follows such that instances with different characteristics can be studied. Processing times are randomly chosen from a discrete uniform distribution inside the interval [1, 100], i.e., $p_{ij} \sim U[1, 100]$, $i \in \mathcal{M}$, $j \in \mathcal{N}$. The due dates of jobs d_j are generated with a discrete uniform distribution inside the interval $[\bar{P}(1 - TF - R/2), \bar{P}(1 - TF + R/2)]$, where $\bar{P} = \lfloor \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{N}} p_{ij}/m^2 \rfloor$, and parameters TF and R take values inside the set {0.2, 0.6, 1.0}, thereby yielding 9 subclasses. The parameter TF represents the tardiness factor, which controls average due-date values and parameter R controls the relative range of due dates. Since it is possible that $\bar{P}(1 - TF - R/2) < 0$, $\max\{0, \bar{P}(1 - TF - R/2)\}$ is considered as the lower limit of the interval when generating instances randomly. Generating due dates from such kind of interval follows the methods described in Liaw et al. (2013), Esmaeilbeigi et al. (2016)as well as those in the OR library (Cordone et al., 2018). For each fixed value of TF and R, 5 instances are randomly generated. Both the delay penalty w_j and revenue u_j of jobs are generated with a discrete uniform distribution inside the interval [1, 10].

6.2. Computational results

Table 2 summarizes results of the above seven methods for the instances tested with a combination of $TF = 0.2$ and $R = 0.2$. The first two columns of the table represent the number of machines m and jobs n, respectively. The next seven columns give average values of the objective values for five instances (denoted by "Avg. Z") corresponding to each of the methods MIP1, MIP2, MIP1V, MIP2V, BB1, BB5, and BB10, respectively. The table also indicates within parentheses the number of instances that cannot be solved to optimality within the time limit of 1800 CPU seconds. If certain instances could not be solved to optimality within the time limit, the current best objective value is chosen as the result, thereby leading to smaller average objective values for some methods, as in the case of "Avg. Z" values of MIP1 and MIP2 for the case with $n = 50$ and $m = 2$ in Table 2.

Tables 3–10 list average computation times for the seven methods for instances with other different combinations of TF and R. Average objective values are not listed in these tables, and the reader is referred to the above link for these values. In these tables, the first two columns list values of m and n while the next seven columns list average computation times of the five instances (denoted by "Avg. Time (s)") for each method. If an instance could not be solved to optimality within the specified time limit, 1800 s are considered as the computation time for that instance. Boldfaced characters have been used to highlight the best method in terms of the average computation time. Fig. 1 depicts a compari-

Table 2
Summary of results for computation time and objective value with $TF = 0.2$ and $R = 0.2$.

m	n	Avg. Z							Avg. Time (s)						
		MIP1	MIP2	MIP1V	MIP2V	BB1	BB5	BB10	MIP1	MIP2	MIP1V	MIP2V	BB1	BB5	BB10
2	20	106.0	106.0	106.0	106.0	106.0	106.0	106.0	69.4	2.5	0.5	0.5	0.7	0.8	0.9
	30	167.0	167.0	167.0	167.0	167.0	167.0	167.0	539.1	320.6	2.7	3.8	1.4	1.6	1.7
	40	223.8	223.9	224.0	224.0	224.0	224.0	224.0	1017.8	181.3	3.9	32.1	4.9	5.2	5.3
	50	226.1(4*)	226(4*)	227.7	227.7	227.7	227.7	227.7	1763.3	1269.7	33.7	274.7	7.2	7.7	8.0
								Average	847.4	443.5	10.2	77.8	3.6	3.8	4.0
3	20	110.4	110.4	110.4	110.4	110.4	110.4	110.4	8.9	2.2	0.5	0.7	0.6	0.7	0.8
	30	162.8	162.8	162.8	162.8	162.8	162.8	162.8	175.4	7.7	2.0	3.1	1.1	1.2	1.3
	40	224.9	224.9	224.9	224.9	224.9	224.9	224.9	736.7	87.1	7.5	25.1	3.6	4.0	4.3
	50	267.8(1*)	267.4(2*)	268.0	268.0	268.0	268.0	268.0	1282.9	961.6	8.6	308.5	4.5	5.0	5.3
								Average	551.0	264.6	4.6	84.3	2.5	2.7	2.9
5	20	113.9	113.9	113.9	113.9	113.9	113.9	113.9	5.8	2.3	0.6	0.8	0.6	0.7	0.8
	30	170.9	170.9	170.9	170.9	170.9	170.9	170.9	57.8	17.1	2.2	2.6	1.6	1.8	2.0
	40	220.6	220.6	220.6	220.6	220.6	220.6	220.6	197.1	58.2	9.9	20.2	2.6	3.1	3.4
	50	283.0	283.0	283.0	283.0	283.0	283.0	283.0	664.4	386.5	30.5	78.1	5.1	5.8	6.3
								Average	231.3	116.0	10.8	25.4	2.5	2.8	3.1
10	20	110.5	110.5	110.5	110.5	110.5	110.5	110.5	1.1	0.7	0.4	0.7	0.9	1.1	1.2
	30	168.3	168.3	168.3	168.3	168.3	168.3	168.3	42.6	4.7	2.3	4.1	2.6	3.0	3.4
	40	218.2	218.2	218.2	218.2	218.2	218.2	218.2	165.7	19.9	8.5	17.4	6.0	7.1	7.8
	50	276.0	276.0	276.0	276.0	276.0	276.0	276.0	831.5	168.2	55.3	53.5	12.6	14.9	16.3
								Average	260.2	48.4	16.6	18.9	5.5	6.5	7.1
							Overall	Average	472.5	218.1	10.6	51.6	**3.5**	4.0	4.3

*Number of Instances that cannot be solved to optimality within the time limit of 1800 CPU seconds.

Table 3
Summary of results for computation time with $TF = 0.2$, $R = 0.6$.

m	n	Avg. Time (s)						
		MIP1	MIP2	MIP1V	MIP2V	BB1	BB5	BB10
2	20	33.9	0.8	1.2	0.4	0.5	0.6	0.6
	30	264.5	12.1	8.1	2.3	1.1	1.3	1.4
	40	689.1	35.5	207.6	18.7	3.2	3.4	3.5
	50	1184.9	385.1	663.8	52.1	9.1	9.6	10.0
	Average	543.1	108.4	220.2	18.4	3.5	3.7	3.9
3	20	11.7	1.2	0.6	0.5	0.4	0.4	0.5
	30	70.4	5.9	6.0	3.5	1.0	1.1	1.2
	40	327.3	61.4	49.4	31.2	2.8	3.1	3.4
	50	920.9	319.8	188.7	107.4	4.0	4.4	4.7
	Average	332.6	97.1	61.2	35.7	2.1	2.3	2.5
5	20	5.7	1.8	0.7	0.8	0.7	0.9	0.9
	30	61.9	11.4	4.6	3.5	1.9	2.3	2.5
	40	411.4	43.8	36.3	20.5	3.4	4.0	4.5
	50	427.0	172.5	82.3	83.1	6.2	7.3	8.0
	Average	226.5	57.4	31.0	27.0	3.1	3.6	4.0
10	20	1.4	1.0	0.5	0.8	1.1	1.4	1.6
	30	26.7	4.6	2.9	4.5	3.1	3.7	4.2
	40	145.3	18.8	43.2	18.9	6.6	7.8	8.7
	50	719.4	158.5	211.7	65.3	17.0	19.6	21.3
	Average	223.2	45.7	64.6	22.4	7.0	8.1	9.0
Overall	Average	331.3	77.1	94.2	25.8	**3.9**	4.4	4.8

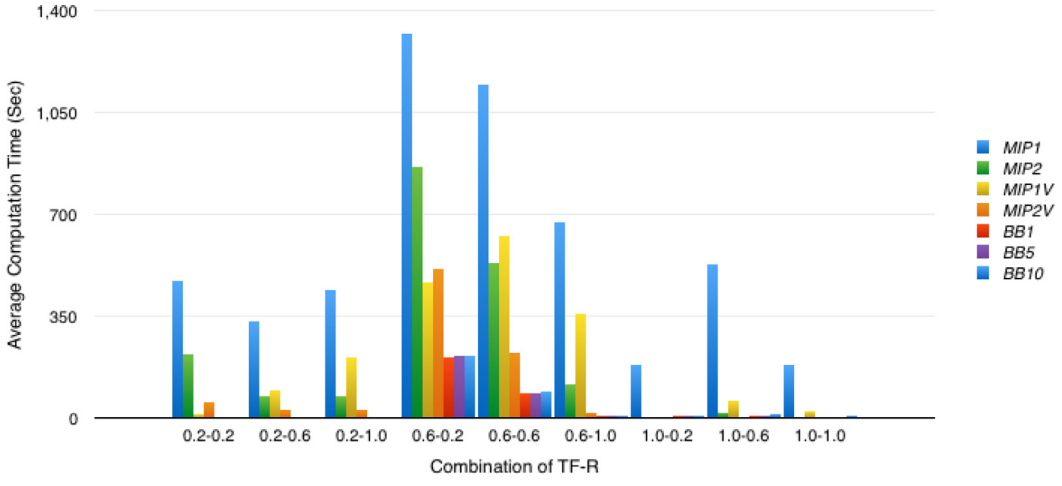

Fig. 1. Comparison between average computation times of methods for combinations of $TF - R$.

Table 4
Summary of results for computation time with $TF = 0.2$ and $R = 1.0$.

m	n	Avg. Time (s)						
		MIP1	MIP2	MIP1V	MIP2V	BB1	BB5	BB10
2	20	17.8	0.7	1.3	0.3	0.4	0.4	0.5
	30	421.0	14.1	75.9	1.7	1.1	1.2	1.3
	40	1048.3	20.2	675.2	22.1	2.7	3.0	3.1
	50	1251.5	540.0	1226.2	51.6	7.2	7.5	7.7
	Average	684.7	143.8	494.7	18.9	2.9	3.0	3.2
3	20	4.4	1.1	0.8	0.4	0.4	0.5	0.5
	30	220.8	5.5	16.9	2.9	1.0	1.1	1.2
	40	926.3	50.9	386.2	35.0	2.1	2.4	2.5
	50	1275.6	166.6	442.0	163.6	5.0	5.7	6.1
	Average	606.8	56.0	211.5	50.5	2.1	2.4	2.6
5	20	2.5	1.3	0.5	0.7	0.5	0.6	0.7
	30	51.2	7.9	10.9	2.0	1.4	1.6	1.7
	40	221.8	36.0	56.7	16.8	2.8	3.2	3.5
	50	789.1	193.9	263.0	82.5	4.8	5.5	6.0
	Average	266.2	59.8	82.8	25.5	2.4	2.7	3.0
10	20	1.3	0.7	0.3	0.7	0.8	1.0	1.1
	30	18.0	4.2	2.1	2.9	2.3	2.7	3.0
	40	126.1	27.5	32.2	10.0	4.7	5.5	6.1
	50	627.9	101.6	175.6	30.1	16.4	18.7	20.4
	Average	193.3	33.5	52.6	10.9	6.1	7.0	7.7
Overall	Average	437.7	73.3	210.4	26.5	**3.4**	3.8	4.1

Table 5
Summary of results for computation time with $TF = 0.6$ and $R = 0.2$.

m	n	Avg. Time (s)						
		MIP1	MIP2	MIP1V	MIP2V	BB1	BB5	BB10
2	20	1800.1	464.7	143.9	1.6	3.2	3.6	3.8
	30	1800.0	1801.0	62.5	22.6	12.5	13.6	15.0
	40	1800.0	1800.0	1800.0	1134.1	37.8	38.9	40.1
	50	1800.0	1800.0	1095.8	1084.7	160.9	165.2	168.8
	Average	1800.0	1466.4	775.6	560.8	53.6	55.3	56.9
3	20	1624.6	12.5	5.8	2.4	7.8	8.9	9.8
	30	1800.1	1800.0	661.8	602.0	420.6	427.8	437.7
	40	1800.0	1800.0	1151.0	1502.2	881.9	896.4	903.3
	50	1800.0	1800.0	1800.0	1800.0	1355.0	1359.4	1365.7
	Average	1756.2	1353.1	904.7	976.7	666.3	673.1	679.1
5	20	470.9	2.7	1.9	0.8	3.1	3.5	3.7
	30	1225.1	45.1	29.5	27.5	27.6	28.1	28.4
	40	1779.3	231.5	107.5	504.5	43.8	44.5	45.0
	50	1800.1	1606.7	348.1	1440.8	353.0	355.2	356.0
	Average	1318.9	471.5	121.8	493.4	106.9	107.8	108.3
10	20	2.4	0.7	0.2	0.2	0.9	1.1	1.3
	30	13.5	5.3	1.6	1.4	2.7	3.3	3.8
	40	159.3	83.7	22.4	7.7	7.6	9.0	10.0
	50	1390.8	562.2	202.2	73.8	25.2	28.9	31.3
	Average	391.5	163.0	56.6	20.8	9.1	10.6	11.6
Overall	Average	1316.6	863.5	464.6	512.9	**209.0**	211.7	214.0

son between the average computation times of the seven methods for problem instances with different combinations of TF and R.

Based on observed results, the key observations could be summarized as below.

(1) Instances with $TF = 0.6$ are relatively harder to be solved compared to others irrespective of the method employed. A possible cause of this is that if TF is small, fewer jobs (sometimes even no job) are expected to be rejected, and the problem is then directly reduced to the unrelated parallel machine scheduling problem. In contrast, if TF is large, more jobs are expected to be rejected owing to tighter due dates, thereby increasing the ease of solution. However, there exists no clear trend in terms of R values relating to the ease or difficulty with which a solution can be obtained.

(2) The performance of formulations is more-or-less related to the ratio of the number of jobs to number of machines (i.e., n/m). In general, the number of jobs n most affects the performance of models, and the value of $TF - R$ affects the performance of models more than that of the n/m ratio. With a fixed combination of n and $TF - R$, it can be seen that when the ratio n/m is small (say, less than 10), the instance is relatively easier to be solved using formulations. This coincides with observations reported by Unlu and Mason (2010) and Chen and Powell (1999). It seems that there exists a certain threshold value for this ratio. However, it is hard to determine the exact value of this threshold.

(3) The use of enhancement techniques improves the performance of the two MIP formulations in terms of average computation time. On average, the MIP1 model is improved by 72.6% in efficiency through use of enhancement techniques, whereas

Table 6
Summary of results for computation time with $TF = 0.6$ and $R = 0.6$.

m	n	Avg. Time (s)						
		MIP1	MIP2	MIP1V	MIP2V	BB1	BB5	BB10
2	20	1800.0	5.1	11.1	1.3	2.5	2.8	2.9
	30	1800.0	761.7	930.8	24.0	7.6	8.1	8.5
	40	1800.0	1800.3	1800.2	348.6	85.3	98.4	100.9
	50	1800.0	1800.0	723.0	1089.8	565.1	565.7	566.2
	Average	1800.0	1091.8	866.3	365.9	165.1	168.8	169.6
3	20	1083.3	2.7	2.6	0.8	1.8	1.9	2.1
	30	1800.1	86.2	1118.1	16.1	12.8	13.2	13.5
	40	1800.0	1324.4	1800.1	614.4	145.7	148.6	148.9
	50	1800.0	1788.7	1791.6	857.4	476.4	477.3	477.9
	Average	1620.9	800.5	1178.1	372.2	159.2	160.3	160.6
5	20	4.7	2.0	0.5	0.4	0.8	0.9	1.0
	30	889.7	14.9	372.5	4.4	8.7	9.4	9.8
	40	1137.9	104.9	82.5	17.8	10.2	10.7	11.1
	50	1800.0	514.2	1221.4	564.5	38.2	39.5	40.4
	Average	958.1	159.0	419.2	146.8	14.5	15.1	15.6
10	20	1.4	0.6	0.2	0.1	1.0	1.2	1.4
	30	8.0	3.6	1.2	1.1	3.1	3.5	3.8
	40	74.5	14.3	8.1	3.9	4.2	5.1	5.6
	50	721.7	256.8	127.7	19.9	8.9	10.6	11.7
	Average	201.4	68.8	34.3	6.3	4.3	5.1	5.6
Overall	Average	1145.1	530.0	624.5	222.8	**85.8**	87.3	87.9

Table 7
Summary of results for computation time with $TF = 0.6$ and $R = 1.0$.

m	n	Avg. Time (s)						
		MIP1	MIP2	MIP1V	MIP2V	BB1	BB5	BB10
2	20	204.1	1.9	4.8	0.6	0.8	0.9	1.0
	30	1087.9	8.1	473.6	1.5	1.3	1.5	1.6
	40	1594.7	112.5	1495.8	23.4	4.1	4.4	4.6
	50	1800.0	559.6	1800.0	71.7	24.4	26.2	27.5
	Average	1171.7	170.5	943.6	24.3	7.7	8.3	8.7
3	20	74.8	1.6	1.2	0.5	1.1	1.2	1.3
	30	414.9	3.3	7.4	1.7	1.4	1.6	1.8
	40	1093.6	38.9	392.1	10.5	6.2	6.6	6.9
	50	1800.0	876.9	989.7	94.7	7.6	9.1	9.8
	Average	845.8	230.2	347.6	26.9	4.1	4.6	5.0
5	20	2.4	1.1	0.3	0.4	1.5	1.8	1.9
	30	392.9	6.0	3.3	1.6	3.4	3.8	4.1
	40	923.6	47.6	405.6	6.9	9.4	11.2	11.9
	50	795.2	140.3	82.4	18.8	11.8	12.6	13.0
	Average	528.5	48.8	122.9	6.9	6.5	7.4	7.7
10	20	3.6	0.8	0.2	0.3	0.8	1.0	1.2
	30	9.3	2.8	0.8	1.1	4.8	5.7	6.3
	40	16.5	9.0	3.6	5.1	6.2	7.0	
	50	406.9	38.8	31.1	10.0	9.3	11.4	12.9
	Average	132.3	14.7	10.3	3.8	5.0	6.1	6.9
Overall	Average	669.6	116.0	356.1	15.5	**5.8**	6.6	7.1

the MIP2 model demonstrated a 73.4% improvement in efficiency, as described in Table 11. In the table, $I_{MIP1}(\%) = (t_{MIP1} - t_{MIP1V})/t_{MIP1} \times 100\%$, and $I_{MIP2}(\%)$ is similarly defined.

(4) Overall, in terms of average computation time, the MIP2 model with CPLEX performs better compared to MIP1 with CPLEX. Similarly, on average, MIP2V performs better compared to MIP1V. However, for several combinations of m and n, the MIP1V model runs faster compared to MIP2V, especially for the instance with $TF = 0.2$ and $R = 0.2$. Such a behavior is highly unexpected on the part of the authors, and a possible reason for this is believed to be that MIP1V may perform better compared to MIP2V with regard to solving unrelated parallel machine scheduling problems with the objective of minimizing the total weighted tardiness, since with $TF = 0.2$ and $R = 0.2$, almost all jobs are expected to be accepted.

(5) The proposed B&B algorithm is capable of solving all instances to optimality within the specified time limit of 1800 CPU seconds. Relatively, BB1 performs best in terms of the average computation time. However, with regard to formulations, namely MIP1, MIP2, MIP1V and MIP2V, not all instances can be solved to optimality within the same time limit, as described in Table 12 for the summarized number of instances that cannot be solved to optimality within the time limit.

(6) The proposed B&B algorithm outperforms other methods for all instances with $TF = 0.2$ and $TF = 0.6$. On average, it saves at least half the computation time compared to MIP formulations in combination with enhancement techniques (for instances with $TF = 0.6$ and $R = 0.2$). However, for the instances with $TF = 1.0$, the MIP2V model performs better compared to the three B&B algorithms in terms of the average computation time. Specifically, on overall average, it takes the MIP2V around 1 CPU seconds to obtain optimum solutions for the instances with $TF = 1.0$, while it takes three B&B algorithms around 10 CPU seconds.

Table 8
Summary of results for computation time with $TF = 1.0$ and $R = 0.2$.

m	n	Avg. Time (s)						
		MIP1	MIP2	MIP1V	MIP2V	BB1	BB5	BB10
2	20	0.2	0.1	0.1	0.1	0.2	0.3	0.3
	30	0.5	0.4	0.1	0.1	0.6	0.7	0.8
	40	3.4	0.8	0.1	0.1	3.5	4.3	4.8
	50	1574.4	6.6	1.2	0.8	9.8	11.4	12.3
	Average	394.6	2.0	0.4	0.3	3.5	4.2	4.6
3	20	0.6	0.3	0.1	0.1	0.4	0.5	0.6
	30	1.6	0.6	0.1	0.1	1.8	2.2	2.5
	40	84.7	2.1	0.2	0.2	6.4	7.6	8.7
	50	552.3	3.5	8.5	0.4	15.2	18.6	20.9
	Average	159.8	1.6	2.2	0.2	6.0	7.2	8.2
5	20	1.8	0.5	0.1	0.1	0.3	0.4	0.5
	30	3.5	1.0	0.1	0.1	3.1	3.7	4.1
	40	10.6	2.0	0.1	0.1	6.7	8.0	8.7
	50	65.5	4.1	0.2	0.2	23.1	26.8	28.2
	Average	20.4	1.9	0.1	0.1	8.3	9.7	10.4
10	20	7.7	0.8	0.5	0.1	0.6	0.7	0.9
	30	10.3	1.3	4.4	0.1	4.8	5.9	6.5
	40	48.7	5.6	0.3	0.3	21.8	26.1	28.2
	50	589.5	7.2	0.1	0.1	23.7	28.7	30.8
	Average	164.1	3.7	1.3	0.2	12.7	15.4	16.6
Overall	Average	184.7	2.3	1.0	**0.2**	7.6	9.1	9.9

Table 9
Summary of results for computation time with $TF = 1.0$ and $R = 0.6$.

m	n	Avg. Time (s)						
		MIP1	MIP2	MIP1V	MIP2V	BB1	BB5	BB10
2	20	0.9	0.3	0.1	0.1	0.3	0.4	0.5
	30	495.4	0.8	0.4	0.2	0.8	0.9	1.0
	40	1800.0	6.0	8.3	0.7	1.9	2.2	2.4
	50	1446.6	218.6	799.6	9.0	10.9	12.6	
	Average	935.7	56.4	202.1	2.5	3.5	4.0	1.3
3	20	0.8	0.5	0.1	0.1	0.6	0.7	0.8
	30	679.1	1.5	0.5	0.4	1.6	1.9	2.1
	40	872.6	3.2	0.7	0.6	6.9	8.2	9.1
	50	748.9	13.2	26.9	2.8	11.9	14.6	16.4
	Average	575.4	4.6	7.1	1.0	5.3	6.4	7.1
5	20	1.7	0.6	0.1	0.1	0.4	0.5	0.5
	30	16.2	2.2	0.3	0.4	2.5	2.9	3.3
	40	729.1	6.2	1.6	0.9	7.3	8.5	9.3
	50	888.2	19.1	104.4	2.7	65.5	72.9	80.0
	Average	408.8	7.0	26.6	1.0	18.9	21.2	23.3
10	20	4.9	0.5	3.2	0.1	1.0	1.3	1.4
	30	15.9	1.5	0.1	0.1	3.2	3.9	4.4
	40	43.5	3.7	0.1	0.2	8.7	10.4	11.3
	50	683.5	7.8	0.3	0.3	18.4	21.7	23.6
	Average	187.0	3.4	0.9	0.2	7.8	9.3	10.2
Overall	Average	526.7	17.9	59.2	**1.2**	8.9	10.2	11.1

7. Conclusions

In this paper we have studied an order acceptance and scheduling problem on unrelated parallel machines, which has not yet been thoroughly explored in available literature. Two different formulations that can be solved with general-purpose IP solvers have been developed. Formulation tightening and valid inequalities have also been proposed to improve the efficiency of the formulations. A formulation-based branch-and-bound algorithm has been developed based on the idea of "divide and conquer", wherein the branching determines how many jobs should be accepted followed by addressing of the unrelated parallel machine scheduling subproblem with the minimization of the total weighted tardiness. Extensive computational experiments have been conducted on various instances to compare the performance of developed formulations, formulations with valid inequalities, and the proposed formulation-based branch-and-bound algorithm. The results demonstrate that the enhanced formulations perform much better compared to basic ones in terms of the average computation time. The proposed branch-and-bound algorithm is observed to be much faster compared to other methods in nearly all instances, and it could efficiently solve all problem instances with values of n and m up to 50 and 10, respectively, to optimality in less than 1800 CPU seconds.

The work could be extended in several research directions. First, more dominant rules, including other Emmon's rules, could be explored and transformed into formulation constraints. Secondly, other practical constraints, such as a renewable-resource constraint, and some mandatorily accepted jobs due to long-term loyal customers, could be considered, thereby making the problem to be more practice-oriented. Additionally, effective heuristic and/or meta-heuristic methods could be developed to tackle large-sized problems. Lastly, since the OAS problem can be reduced to a special unrelated parallel machine scheduling problem with a

Table 10
Summary of results for computation time with $TF = 1.0$ and $R = 1.0$.

m	n	Avg. Time (s)						
		MIP1	MIP2	MIP1V	MIP2V	BB1	BB5	BB10
2	20	0.4	0.1	0.1	0.1	0.2	0.2	0.3
	30	2.9	0.6	0.3	0.2	0.3	0.4	0.5
	40	729.4	2.6	361.0	0.9	3.5	4.0	4.3
	50	784.7	31.3	23.5	2.7	5.7	6.5	7.1
	Average	379.4	8.7	96.2	1.0	2.4	2.8	3.1
3	20	1.5	0.5	0.1	0.1	0.4	0.6	0.7
	30	11.4	1.0	0.5	0.4	1.5	1.8	2.0
	40	122.7	2.0	2.7	0.6	2.5	2.8	3.1
	50	405.4	5.9	2.4	1.1	5.7	6.6	7.2
	Average	135.3	2.4	1.4	0.6	2.5	3.0	3.3
5	20	1.5	0.7	1.1	0.1	0.4	0.5	0.6
	30	6.5	1.4	0.3	0.4	3.2	3.8	4.1
	40	63.9	4.4	2.3	1.2	8.4	10.2	11.1
	50	458.7	6.9	2.7	1.8	12.1	14.2	15.3
	Average	132.7	3.4	1.6	0.9	6.0	7.2	7.8
10	20	4.6	1.0	0.1	0.1	0.6	0.8	1.0
	30	10.8	1.3	0.1	0.1	2.0	2.4	2.7
	40	106.5	5.3	0.4	0.5	13.7	16.7	18.5
	50	247.7	7.0	0.8	1.0	4.2	5.5	6.4
	Average	92.4	3.7	0.4	0.4	5.1	6.4	7.2
Overall	Average	184.9	4.5	24.9	**0.7**	4.0	4.8	5.3

Table 11
Efficiency of enhancement techniques for MIP formulations.

	Avg. Time (s)				Improvement in Time	
$TF - R$	t_{MIP1}	t_{MIP2}	t_{MIP1V}	t_{MIP2V}	$I_{MIP1}(\%)$	$I_{MIP2}(\%)$
0.2-0.2	472.5	218.1	10.6	51.6	97.8	76.3
0.2-0.6	331.3	77.1	94.2	25.8	71.6	66.5
0.2-1.0	437.7	73.3	210.4	26.5	51.9	63.9
0.6-0.2	1316.6	863.5	464.6	512.9	64.7	40.6
0.6-0.6	1145.1	530.0	624.5	222.8	45.5	58.0
0.6-1.0	669.6	116.0	356.1	15.5	46.8	86.6
1.0-0.2	184.7	2.3	1.0	0.2	99.5	91.3
1.0-0.6	526.7	17.9	59.2	1.2	88.8	93.3
1.0-1.0	184.9	4.5	24.9	0.7	86.5	84.4
				Average	72.6	73.4

Table 12
Number of instances that cannot be solved to optimality within the specified time limit.

$TF - R$	$MIP1$	$MIP2$	$MIP1V$	$MIP2V$	$BB1$
0.2-0.2	5	6	0	0	0
0.2-0.6	2	0	1	0	0
0.2-1.0	2	1	1	0	0
0.6-0.2	55	34	18	20	0
0.6-0.6	44	13	20	0	0
0.6-1.0	22	2	13	1	0
1.0-0.2	5	0	0	0	0
1.0-0.6	18	0	1	0	0
1.0-1.0	2	0	0	0	0
Average	17.2	6.2	6.0	2.3	0.0

dummy machine, future efforts could be dedicated towards exploring the use of various decomposition-based methods, including column generation and Benders decomposition.

Acknowledgment

The authors would like to thank the anonymous referees for their constructive comments which contributed to improve the quality of this paper. This work was supported by the National Science Foundation of China (NSFC) with Grant No. 71571135. The work was also supported by the Fundamental Research Funds for the Central Universities.

Appendix

The OAS problem on unrelated parallel machines can be reduced to a special case of the classical unrelated parallel machine scheduling problem by introducing a dummy machine to which all rejected jobs can be assigned with zero processing time, zero revenue and zero penalty. To compare the performance of formulations with and without a dummy machine, the MIP1 and MIP2 models could be reformulated as follows.

Here, a dummy machine M_0 has been introduced, and the MIP1 has been reformulated to the following MIP1-M0 model. To this end, notations additional to those defined in the original MIP1 model have been introduced below.

- $\mathcal{M}_0$: set of machines, including the dummy machine. The dummy machine is indexed by $i = 0$.
- $\mathcal{M}$: set of machines without the dummy machine, i.e., $\mathcal{M} = \{1, \cdots, m\}$.
- $y_{0j} = 1$, if job j is assigned on the dummy machine (i.e., it is rejected), 0 otherwise.

$$[\textbf{MIP1} - \textbf{M0}] \quad \max \quad Z = \sum_{j\in\mathcal{N}}(u_j \sum_{i\in\mathcal{M}} y_{ij} - w_j T_j) \tag{20a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{M}_0} y_{ij} = 1, \quad \forall j \in \mathcal{N}; \tag{20b}$$

$$T_j \le (1 - y_{0j})L_{max} \quad \forall j \in \mathcal{N}; \tag{20c}$$

$$y_{ij} \in \{0, 1\}, \quad \forall j \in \mathcal{N}, i \in \mathcal{M}_0; \tag{20d}$$

$$constraints\ (1c) - (1i), (1k). \tag{20e}$$

The objective function can be described as $\sum_{i\in\mathcal{M}}$, since only jobs assigned on $\mathcal{M}$, i.e., the accepted jobs, are considered. Additionally, with $T_j \ge 0$ in constraints (1i) and (20c), if $y_{0j} = 1$ (i.e., if a job j has been rejected), its tardiness is forced to be 0, i.e., $T_j = 0$. With constraints (20b), $\sum_{i\in\mathcal{M}_0} y_{ij} = 1$, $\forall j \in \mathcal{N}$, any job j can either

Table 13
Computation time of models with and without the dummy machine for $TF = R = 0.2$.

m	n	Avg. Time (s)			
		MIP1	MIP1-M0	MIP2	MIP2-M0
2	20	14.0	13.7	3.0	2.8
	30	442.4	450.9	15.3	731.7
	40	746.2	1287.8	4.7	784.0
	50	694.2	1741.9	391.8	1524.7
	Average	474.2	873.6	103.7	760.8
3	20	9.3	14.8	3.3	4.0
	30	136.5	143.9	20.7	33.8
	40	369.7	920.9	155.4	116.5
	50	432.8	995.3	867.9	804.5
	Average	237.1	518.7	261.8	239.7
5	20	8.2	7.5	2.0	1.8
	30	76.1	58.5	14.5	23.6
	40	688.6	123.3	61.1	153.2
	50	1039.6	1486.6	190.7	554.5
	Average	453.1	419.0	67.1	183.3
10	20	1.5	4.1	1.7	2.1
	30	26.8	47.5	6.8	6.2
	40	181.5	256.6	15.6	28.3
	50	548.1	733.6	105.0	166.4
	Average	189.5	260.5	32.3	50.8
	Overall Average	**338.5**	517.9	**116.2**	308.6

be rejected (i.e., $y_{0j} = 1$) or accepted by assigning it to a machine $i \in \mathcal{M}$.

In the MIP1-M0 model, the decision variable x_j in MIP1 is not used any more. The relationship between the MIP1-M0 model and the MIP1 model can, therefore, be established using the following constraints.

$$x_j + y_{0j} = 1, \quad \forall j \in \mathcal{N} \tag{21}$$

The MIP1 model explicitly considers the decision of acceptance or rejection, whereas the MIP1-M0 model implicitly considers the decision of acceptance. Similarly, with the same objective function in (20a), and constraints (20b)–(20d), (2c)–(2j) and z_{ijk} in (2k), the MIP2 model can be reformulated into the corresponding MIP2-M0.

A2. Comparisons between models with and without the dummy machine

Table 13 summarizes average computation times in terms of CPU seconds for the case with $TF = 0.2$ and $R = 0.2$ with a view to compare models MIP1 and MIP2 against MIP1-M0 and MIP2-M0, respectively. Five instances are randomly generated for the combination of TF and R by following the data-generation scheme described in Section 6.1. The average computation time of the five instances (denoted by "Avg. Time (s)") for each model is listed in the table. If an instance cannot be solved to optimality within the specified time limit, 1800 s are used for the calculation of the average computation time. Model codes, data, and detailed results for Table 13 are available online at https://pan.baidu.com/s/1N_vWO3iM1gccmW9ayyuQOg#list/path=%2F for readers' reference.

As observed, the MIP1-M0 model demonstrates poorer performance compared to MIP1 in terms of the average computation time for almost all cases. Similarly, MIP2-M0 performs worse compared to MIP2 in terms of average computation time. In summary, the preliminary results of the computational experiments in Table 13 show that MIP2 performs better compared to MIP2-M0, MIP1 performs better compared to MIP1-M0, and MIP2-M0 performs better compared to MIP1-M0. Additionally, MIP2 performs better compared to MIP1. In this study, therefore, we only employ the enhanced MIP2 model (i.e., MIP2V) in the proposed branch-and-bound algorithm.

References

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 (9–12), 1705–1718.

Azizoglu, M., Kirca, O., 1999. Scheduling jobs on unrelated parallel machines to minimize regular total cost functions. IIE Trans. 31, 153–159.

Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., Stougie, L., 2000. Multiprocessor scheduling with rejection. SIAM J. Discrete Math. 13 (1), 64–78.

Cesaret, B., Oğuz, C., Salman, F.S., 2012. A tabu search algorithm for order acceptance and scheduling. Comput. Oper. Res. 39, 1197–1205.

Chaurasia, S.N., Singh, A., 2017. Hybrid evolutionary approaches for the single machine order acceptance and scheduling problem. Appl. Soft Comput. 52, 725–747.

Chen, Z.L., Powell, W.B., 1999. Solving para llel machine scheduling problems by column generation. INFORMS J. Comput. 11 (1), 78–94.

Cordone, R., Hosteins, P., Righini, G., 2018. A branch-and-bound algorithm for the prize-collecting single-machine scheduling problem with deadlines and total tardiness minimization. INFORMS J. Comput. 30 (1), 168–180.

Şen, H., Bülbül, K., 2015. A strong preemptive relaxation for weighted tardiness and earliness/tardiness problems on unrelated parallel machines. INFORMS J. Comput. 27 (1), 135–150.

De Vel, V., 1993. Duality-based algorithms for scheduling unrelated parallel machines. INFORMS J. Comput. 5 (2), 192–205.

Detienne, B., Dauzère-Pérès, S., Yugma, C., 2011. Scheduling jobs on parallel machines to minimize a regular step total cost function. J. Sched. 14 (6), 523–538.

Emami, S., Moslehi, G., Sabbagh, M., 2017. A benders decomposition approach for order acceptance and scheduling problem: a robust optimization approach. Comput. Appl. Math. 36 (4), 1471–1515.

Emami, S., Sabbagh, M., Moslehi, G., 2016. A lagrangian relaxation algorithm for order acceptance and scheduling problem: a globalised robust optimisation approach. Int. J. Comput. Integr. Manuf. 29 (5), 535–560.

Emmons, H., 1969. One-machine sequencing to minimize certain functions of job tardiness. Oper. Res. 17, 701–715.

Esmaeilbeigi, R., Charkhgard, P., Charkhgard, H., 2016. Order acceptance and scheduling problems in two-machine flow shops: new mixed integer programming formulations. Eur. J. Oper. Res. 251, 419–431.

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, 482–493.

Fanjul-Peyro, L., Ruiz, R., 2010. Iterated greedy local search methods for unrelated parallel machine scheduling. Eur. J. Oper. Res. 207 (1), 55–69.

Fanjul-Peyro, L., Ruiz, R., 2011. Size-reduction heuristics for the unrelated parallel machines scheduling problem. Comput. Oper. Res. 38, 301–309.

Fanjul-Peyro, L., Ruiz, R., 2012. Scheduling unrelated parallel machines with optional machines and jobs selection. Comput. Oper. Res. 39, 1745–1753.

Fanjul-Peyro, L., Ruiz, R., Perea, F., 2019. Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. Comput. Oper. Res. 101, 173–182.

Fleszar, K., Hindi, K.S., 2018. Algorithms for the unrelated parallel machine scheduling problem with a resource constraint. Eur. J. Oper. Res. 271 (3), 839–848.

Geramipour, S., Moslehi, G., Reisi-Nafchi, M., 2017. Maximizing the profit in customer's order acceptance and scheduling problem with weighted tardiness penalty. J. Oper. Res. Soc. 68 (1), 89–101.

Ghosh, J.B., 1997. Job selection in a heavily loaded shop. Comput. Oper. Res. 24 (2), 141–145.

Jiang, D.K., Tan, J.Y., Li, B., 2017. Order acceptance and scheduling with batch delivery. Comput. Ind. Eng. 107, 100–104.

Lei, D.M., Guo, X.P., 2015. A parallel neighborhood search for order acceptance and scheduling in flow shop environment. Int. J. Prod. Econ. 165, 12–18.

Lewis, H.F., Slotnick, S.A., 2002. Multi-period job selection: planning work loads to maximize profit. Comput. Oper. Res. 29 (8), 1081–1098.

Liaw, C.F., Lin, Y.K., Cheng, C.Y., Chen, M.C., 2003. Scheduling unrelated parallel machines to minimize total weighted tardiness. Comput. Oper. Res. 30, 1777–1789.

Lin, S.W., Ying, K.C., 2013. Increasing the total net revenue for single machine order acceptance and scheduling problems using an artificial bee colony algorithm. J. Oper. Res. Soc. 64 (2), 293–311.

Mestry, S., Damodaran, P., Chen, C.S., 2011. A branch and price solution approach for order acceptance and capacity planning in make-to-order operations. Eur. J. Oper. Res. 211, 480–495.

Nguyen, S., 2016. A learning and optimizing system for order acceptance and scheduling. Int. J. Adv. Manuf. Technol. 86, 2021–2036.

Nobibon, F.T., Leus, R., 2011. Exact algorithm for a generalization of the order acceptance and scheduling problem in a single-machine environment. Comput. Oper. Res. 38, 367–378.

Ou, J.W., Zhong, X.L., 2017. Bicriteria order acceptance and scheduling with consideration of fill rate. Eur. J. Oper. Res. 262, 904–907.

Ou, J.W., Zhong, X.L., Qi, X.T., 2016. Scheduling parallel machines with inclusive processing set restrictions and job rejection. Nav. Res. Logist. 63, 667–681.

Ou, J.W., Zhong, X.L., Wang, G.Q., 2015. An improved heuristic for parallel machine scheduling with rejection. Eur. J. Oper. Res. 241, 653–661.

Oğuz, C., Salman, F.S., Yalçin, Z.B., 2010. Order acceptance and scheduling decisions in make-to-order systems. Int. J. Prod. Econ. 125 (1), 200–211.

Pinedo, M.L., 2008. Scheduling: Theory, algorithms and systems, third edition Springer Science+Business Media, LLC, New York, USA.

Rahman, H.F., Sarker, R., Essam, D., 2015. A real-time order acceptance and scheduling approach for permutation flow shop problems. Eur. J. Oper. Res. 247, 488–503.
Rinnooy Kan, A. H. G., 1976. Machine scheduling problems classification, complexity and computations. Martinus Nijhoff, The Hague.
Rom, W.O., Slotnick, S.A., 2009. Order acceptance using genetic algorithms. Comput. Oper. Res. 36 (6), 1758–1767.
Shabtay, D., Gaspar, N., Kaspi, M., 2013. A survey on offline scheduling with rejection. J. Sched. 16, 3–28.
Shim, S.O., Kim, Y.D., 2007. Minimizing total tardiness in an unrelated parallel-machine scheduling problem. J. Oper. Res. Soc. 58 (3), 346–354.
Silva, Y.L.T.V., Subramanian, A., Pessoa, A.A., 2018. Exact and heuristic algorithms for order acceptance and scheduling with sequence-dependent setup times. Comput. Oper. Res. 90, 142–160.
Slotnick, S.A., 2011. Order acceptance and scheduling: a taxonomy and review. Eur. J. Oper. Res. 212 (1), 1–11.
Slotnick, S.A., Morton, T.E., 1996. Selecting jobs for a heavily loaded shop with lateness penalties. Comput. Oper. Res. 23 (2), 131–140.
Slotnick, S.A., Morton, T.E., 2007. Order acceptance with weighted tardiness. Comput. Oper. Res. 34, 3029–3042.
Tran, T.T., Araujo, A., Beck, J.C., 2016. Decomposition methods for the parallel machine scheduling problem with setups. INFORMS J. Comput. 28 (1), 83–95.
Unlu, Y., Mason, S.J., 2010. Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. Comput. Ind. Eng. 58, 785–800.
Wang, X.L., Huang, G.D., Hu, X.W., Cheng, T.E., 2015. Order acceptance and scheduling on two identical parallel machines. J. Oper. Res. Soc. 66, 1755–1767.
Wang, X.L., Xie, X.Z., Cheng, T.C.E., 2013. A modified artificial bee colony algorithm for order acceptance in two-machine flow shops. Int. J. Prod. Econ. 141, 14–23.
Wang, X.L., Xie, X.Z., Cheng, T.C.E., 2013a. Order acceptance and scheduling in a two-machine flowshop. Int. J. Prod. Econ. 141, 366–376.
Wang, X.P., Tang, L.X., 2010. A hybrid metaheuristic for the prize-collecting single machine scheduling problem with sequence-dependent setup times. Comput. Oper. Res. 37, 1624–1640.
Wu, G.H., Cheng, C.Y., Yang, H.I., Chena, C.T., 2018. An improved water flow-like algorithm for order acceptance and scheduling with identical parallel machines. Appl. Soft Comput. 71, 1072–1084.
Xiao, Y.Y., Yuan, Y.Y., Zhang, R.Q., Konak, A., 2015. Non-permutation flow shop scheduling with order acceptance and weighted tardiness. Appl. Math. Comput. 270, 312–333.
Xiao, Y.Y., Zhang, R.Q., Zhao, Q.H., Kaku, I., 2012. Permutation flow shop scheduling with order acceptance and weighted tardiness. Appl. Math. Comput. 218, 7911–7926.
Zhong, X.L., Ou, J.W., 2017. Improved approximation algorithms for parallel machine scheduling with release dates and job rejection. 4OR-A Q. J. Oper. Res. 15, 387–406.