Contents lists available at ScienceDirect

Computers and Operations Research

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

Exact and matheuristic methods for the parallel machine scheduling and location problem with delivery time and due date

Chuang Zhang [a], Yantong Li [b,*], Junhai Cao [a], Zhen Yang [c], Leandro C. Coelho [d]

[a] Department of Equipment Support and Remanufacturing, Army Academy of Armored Forces, Beijing 100072, China
[b] School of Maritime Economics and Management, Dalian Maritime University, Dalian 116026, China
[c] School of Management, Xi'an Jiaotong University, Xi'an 710049, China
[d] CIRRELT and Canada Research Chair in Integrated Logistics, Université Laval, Canada

ARTICLE INFO

Keywords:
ScheLoc
MILP
Logic-based Benders decomposition
Matheuristic
Delivery time

ABSTRACT

The scheduling and location (ScheLoc) problem is a new and important research topic with a wide range of practical applications. It jointly optimizes machine locations, job assignments, and sequence decisions to yield a globally optimal solution. Current studies on the ScheLoc problem are limited, and most have focused on makespan minimization. This paper investigates a new variant of the parallel-machine ScheLoc problem with job delivery time and due date considerations. This is motivated by practical situations where jobs must be transported back to their original locations upon completion of processing, and a specific due date is associated with each job. The objective is to minimize the total weighted sum of location, transportation, and tardiness penalty costs incurred if a job is delivered after its due date. We propose three models adapted from classic ones for scheduling problems. We then develop an exact logic-based Benders decomposition method, which decomposes the problem into a master problem (MP) for determining machine locations and job assignments, and a subproblem (SP) for determining job sequences on each machine. The MP is augmented by dynamically adding optimality cuts generated by solving the SP. We also develop a matheuristic (MH) and its variant to achieve near-optimal solutions. An approximate model with predetermined job processing sequences on each machine is formulated and solved to determine the machine locations and job assignments. The obtained solutions are further improved by solving a series of single-machine scheduling problems. The variant combines a kernel search framework to make MH more efficient. Computational experiments on 720 randomly generated instances show the effectiveness and efficiency of both the exact and matheuristic methods.

1. Introduction

The scheduling and location (ScheLoc) problem is a new and attractive research field that integrates facility location and machine scheduling problems, traditionally investigated separately. Historically, a facility location problem determines the locations of machines, and then jobs are scheduled on these machines. However, in many real-world applications, machine locations are not fixed. Instead, they must be selected from a set of candidate locations such that jobs can be assigned to and processed on machines at the chosen locations, resulting in a ScheLoc problem, which jointly optimizes the machine location and scheduling decisions to obtain globally optimal solutions. In a generic setting of the discrete ScheLoc problem, a finite set of candidate locations can be selected to deploy machines. A set of jobs stored at different locations must be transported to their assigned locations. Then jobs are processed on machines at their locations. The discrete parallel-machine ScheLoc problem consists of two classic NP-hard problems, i.e., a facility location problem and a parallel machine scheduling problem. Implementing such a complicated combinatorial optimization problem depends on effective and efficient models and algorithms.

Solutions to the discrete parallel-machine ScheLoc problem find various applications in manufacturing and logistics. In container harbor operations, ships (machines) must be assigned to berths (locations), and the loading or unloading sequences of containers (jobs) must be jointly determined (Kalsch and Drezner, 2010; Dkhil et al., 2017). Another typical application is the use of movable crushers in the mining industry. Decision-makers must determine the crushers' locations, assign minerals to them, and determine the processing sequence of minerals on each crusher (Kalsch, 2009). Other potential applications

* Corresponding author.
E-mail addresses: chuangzhang1225@163.com (C. Zhang), yantongli@163.com (Y. Li), zhen.yang@mail.xjtu.edu.cn (Z. Yang), leandro.coelho@fsa.ulaval.ca (L.C. Coelho).

https://doi.org/10.1016/j.cor.2022.105936
Received 10 February 2021; Received in revised form 10 June 2022; Accepted 28 June 2022
Available online 30 June 2022

include supply chain design and military training planning. In a supply chain network, distribution centers are located, while customers are assigned to and served by the centers under certain constraints (Musavi and Bozorgi-Amiri, 2017). In military training planning, locations of training facilities must be determined, and geographically dispersed soldiers must travel to designated facilities to perform training in sequence (Wesolkowski et al., 2014).

It is crucial to meet customers' deadlines and ensure timely delivery of orders in logistics operations. Backorders lead to customer loss, additional costs, and poor service levels (Slotnick and Sobel, 2005; Pundoor and Chen, 2005). Many companies decentralize their factories to ensure timely delivery of orders (Sun et al., 2015; Behnamian and Ghomi, 2016). In this case, they face the problem of properly assigning orders to factories dispersed across a network. In particular, finished orders must be delivered to customers by a specific date; thus, it is important to integrate order delivery times into production and schedule planning (Chen and Pundoor, 2006). Many works have addressed scheduling problems with delivery time considerations (Tian et al., 2007; Lee et al., 2012; Jin et al., 2013; Garmdare et al., 2017; Mönch and Shen, 2021). However, most such studies assume that machine locations are given.

This study investigates the discrete parallel-machine ScheLoc problem with delivery time and due date considerations (DPSL-DD). The objective is to generate an optimal location and scheduling scheme to minimize the weighted sum of location, job transportation, and total tardiness penalty costs. To the best of our knowledge, this work is the first to integrate delivery times and due dates in the ScheLoc problem. We adapt several classic mixed-integer linear programming (MILP) formulations for this new problem, but they perform poorly for solving large instances given the problem complexity. To this end, we propose exact and matheuristic algorithms to solve large instances, which are verified to be efficient and effective. Our main contributions are summarized as follows.

(1) A new variant of the discrete parallel-machine ScheLoc problem is studied by considering delivery times and due dates (DPSL-DD).

(2) Three MILP formulations are proposed for this variant.

(3) A tailored exact logic-based Benders decomposition (LBBD) method is developed, along with several classes of existing and newly developed valid inequalities.

(4) We propose a matheuristic based on predetermined processing sequences and a variant including a kernel search framework to achieve near-optimal solutions with high efficiency.

(5) Extensive numerical experiments on various size instances validate the performance of the proposed formulations, exact method, and matheuristic.

The remainder of this paper is organized as follows. Section 2 reviews related literature. Section 3 describes the DPSL-DD problem and presents three MILP formulations. The developed LBBD and matheuristic methods are presented in Section 4. Extensive computational experiments are conducted and analyzed in Section 5. Conclusions are presented in Section 6.

2. Literature review

We now briefly review the related literature. The single-machine ScheLoc problem has been well-studied under the makespan minimization criterion. Hennes and Hamacher (2002) introduced the ScheLoc problem in which a single machine can be located anywhere on a network and proposed some polynomial-time algorithms to minimize the makespan. Elvikis et al. (2009) studied the single-machine planar ScheLoc problem to minimize the makespan and provided a general formulation using the earliest release date (ERD) rule. They proposed three polynomial-time algorithms, including two global search algorithms and a local search heuristic. Kalsch and Drezner (2010) investigated the single-machine planar ScheLoc problem to minimize the makespan and total completion time, and solved the problem using the big-triangle-small-triangle branch-and-bound approach. Numerical results show that the proposed method can solve instances with up to 1000 jobs in 5–10 min. Recently, Akbarinasaji and Mckendall (2017) considered a planar single-machine ScheLoc problem to minimize the makespan. A mixed-integer nonlinear program was proposed and then linearized to model the problem. The authors proposed a modified version of the heuristic presented in Elvikis et al. (2009) as well as two novel heuristics, which were all tested on randomly generated instances with up to 1000 jobs. Krumke and Le (2020) investigated a robust single-machine ScheLoc problem in which jobs were located at vertices of a tree whose edge length (representing the transportation cost or time) was uncertain. A polynomial algorithm was proposed to obtain a robust min–max regret solution.

The parallel-machine ScheLoc problem has been receiving increasing attention. Rajabzadeh et al. (2016) studied this problem with machines located in discrete and continuous spaces and presented mathematical models to minimize the makespan. Heßler and Deghdak (2017) investigated the discrete parallel machine makespan (DPMM) ScheLoc problem in which a set of machines are to be located on a set of candidate locations to minimize the makespan. They proposed an integer programming formulation and four lower bounds, developed several types of clustering heuristics, and proposed an iterative local search procedure to improve the solution. Ławrynowicz and Jozefczyk (2019) explored integrated and sequential approaches to the DPMM ScheLoc problem. A tabu search-based memetic algorithm was developed for the integrated approach. The sequential approach decomposes the problem into two subproblems, a p-median problem and a parallel-machine scheduling problem with a release date, respectively solved by a gamma heuristic and a particle swarm optimization method. The integrated approach outperformed the sequential method in solution quality and computation time. Wang et al. (2020c) built a network flow-based MILP model for the DPMM ScheLoc problem. They proposed a distance-based heuristic to obtain near-optimal solutions for small-sized instances and a polynomial-time location-density-longest processing time algorithm for large-sized instances. Ławrynowicz and Filcek (2020) dealt with a planar ScheLoc problem in which the number of machines was to be optimized. They developed an evolutionary algorithm (EV) and a simulated annealing (SA) algorithm to obtain near-optimal Pareto solutions. The two methods were competitive, while the SA outperforming the EV in computational efficiency. Wang et al. (2020b) studied the discrete parallel-machine ScheLoc problem to simultaneously minimize the makespan and machine location costs. A proposed bi-objective MILP model was solved by the ϵ-constraint method to obtain a set of Pareto solutions. Fuzzy logic was applied to help decision-makers select a compromised solution based on their preferences. Kramer and Kramer (2021) proposed an arc-flow formulation for the DPMM ScheLoc problem with a pseudo-polynomial number of variables and constraints. A column generation procedure was adopted to solve the linear relaxation of the formulation. Two IP heuristics and an iterated local search metaheuristic were designed to solve the problem. Thus far, all the papers have considered the makespan minimization objective.

Under different objectives, recent papers have studied the stochastic ScheLoc problem in an uncertain environment. Liu and Liu (2019b) considered processing time uncertainty in the DPMM ScheLoc problem to minimize the weighted sum of location cost and expected total completion time. The first stage of a stochastic programming model selects machine locations, and the second stage determines job assignments and sequences. Sample average approximation and a risk-averse measure called conditional value at risk (CVaR) were used to solve the problem. Liu and Liu (2019a) investigated a discrete parallel-machine ScheLoc problem with a partially known distribution of the processing time. The objective is to minimize the total machine operating cost and control the service level, defined by the probability of finishing job processing on time. A distributionally robust chance-constrained formulation was proposed. Bonferroni's inequality was applied, and mixed-integer second-order cone programming was developed to approximate the problem.

A novel feature of the ScheLoc problem is that jobs must be transported to their assigned locations for processing, which incurs a transportation time that reflects the release date of a job. In this paper, we study the DPSL-DD problem, where jobs, each with a specific due date, must be delivered back to their original locations upon completion of processing. To the best of our knowledge, this work is the first to investigate the discrete parallel-machine ScheLoc problem considering delivery time and job due dates. A tardiness penalty is incurred for each job when the due date is exceeded. The urgency of each job is different, so their priority is different. This may cause a situation where two solutions have the same makespan but have different tardiness penalty costs. This setting has wide applications in logistic operations. For example, decision-makers will assign retailers to different distribution centers in a distribution system. Some retailers have urgent orders due to inventory shortages, while others do not. From a perspective of global optimization, the decision-makers will assign urgent retailers to distribution centers that can serve them as early as possible. In such circumstances, tardiness minimization might be more suitable than the makespan as the objective function.

The new variant with total weighted tardiness minimization is more challenging since the single machine scheduling problem with release dates and total weighted tardiness minimization is already strongly NP-hard (Lawler, 1977). In contrast, the single machine scheduling problem with release dates and makespan minimization is polynomially solvable (Baker and Trietsch, 2013). Similarly, studies on parallel machine scheduling problems support our claim since the instances solved for the parallel machine scheduling with makespan minimization are much larger than those for the parallel machine scheduling with total weighted tardiness minimization (Amorim et al., 2017; Yeh et al., 2014).

3. Problem description and formulations

We now formally describe the DPSL-DD problem and introduce several formulations for it.

3.1. Problem description

Consider a set $K = \{1, 2, \ldots, l\}$ of candidate locations, from which we can select at most m locations to deploy machines. Each location k can house at most one machine with a fixed cost c_k. A set $J = \{1, 2, \ldots, n\}$ of jobs must be assigned to and processed on machines without preemption. Each job j has a nonnegative processing time p_j and due date d_j. All jobs are ready at their storage locations from the beginning of the planning horizon. Once job j is assigned to the machine at location k, it must be transported to that location for processing and delivered back to its original location upon completion. The release date r_{jk} and transportation cost e_{jk} are dependent on the distance D_{jk} between the storage location of job j and machine location k, i.e., $r_{jk} = D_{jk}/u_{jk}$, $e_{jk} = D_{jk}f_{jk}$, where u_{jk} and f_{jk} are the transportation speed and cost per unit of distance, respectively.

In all models defined below, let C_j and T_j be the completion time and the tardiness of job j. Since the job must be delivered back to its original location immediately after the processing is completed, the time spent in transit is counted as part of the total consumption. Therefore, if job j is assigned to location k, we define a modified due date $d_j - r_{jk}$, and the job tardiness in this paper is denoted as $T_j = \max\{C_j + r_{jk} - d_j, 0\}$ for the location assigned. We define θ as the tardiness penalty coefficient, q_1, q_2, and q_3 as the weights of the fixed location cost, job transportation cost, and total tardiness penalty cost, respectively. The decisions consist of selecting locations for machines, assigning jobs to machines, and sequencing jobs on their corresponding machines. The objective is to minimize the weighted sum of the fixed location cost, job round-trip transportation cost, and total tardiness penalty cost.

We make the following assumptions: (1) Each job can only be assigned to one location. Order split is not allowed. (2) Each machine can only process one job at one time. (3) Preemption is not allowed in job processing. (4) For each job j, its due date d_j is at least the sum of the transportation time to its nearest location and its processing time, i.e., $d_j \geq 2\min_{k\in K} r_{jk} + p_j$.

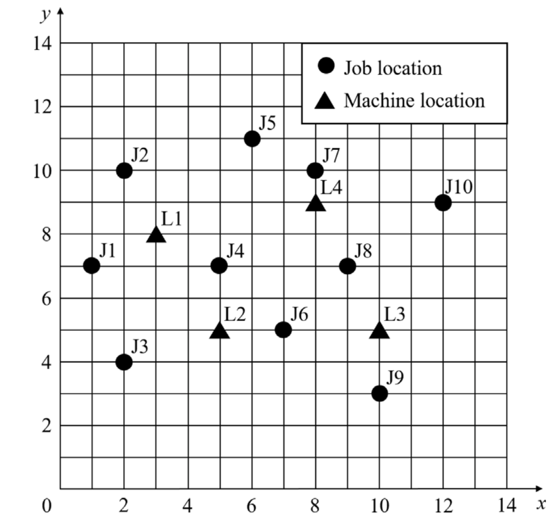


Fig. 1. A simple example of the problem.

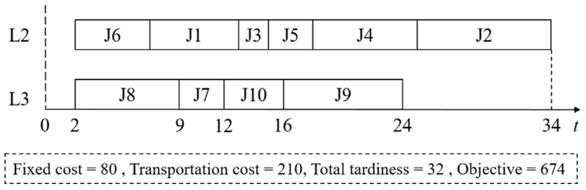


Fig. 2. Optimal solution for the problem in Fig. 1.

3.2. Example

We use a simple example to illustrate the DPSL-DD problem. Consider an instance with 10 jobs, four candidate locations, and two machines. The jobs and candidate locations are dispersed on a network as shown in Fig. 1. The distance D_{jk} between job j and location k is the Euclidean distance, rounded down to the nearest integer. The transportation speed is set to $u_{jk} = 1$, the cost per unit distance is set to $f_{jk} = 3$, the processing time is set to $p_j = \{6, 9, 2, 7, 3, 5, 3, 7, 8, 4\}$, the due date is $d_j = \{22, 28, 20, 21, 27, 12, 14, 9, 16, 22\}$, the fixed location cost is $c_k = \{50, 40, 40, 50\}$, the tardiness penalty coefficient $\theta = 1$, and the weights are $(q_1, q_2, q_3) = (5, 1, 2)$. The problem is to determine which candidate locations should be selected to deploy machines, how to assign jobs to these selected locations, and how to sequence jobs on the machines at those locations. An optimal solution is shown in Fig. 2, where locations $L2$ and $L3$ are selected. Jobs $\{J1, J2, J3, J4, J5, J6\}$ are assigned to location $L2$, and jobs $\{J7, J8, J9, J10\}$ to location $L3$. The Gantt chart indicates the processing sequences of jobs on the machine at the location.

3.3. Formulations

The DPSL-DD problem studied in this paper is new as it generalizes the parallel machine scheduling problem by jointly considering machine location decisions and job deliveries. This section adapts several existing formulations of the parallel machine scheduling problem to our studied problem. The most commonly used formulations include the time-indexed formulation based on time discretization, the linear ordering formulation featuring linear ordering variables, and the sequence-based formulation highlighting machine job sequences. Next, we describe the three adapted formulations in detail, and their performances are compared in Section 5.

3.3.1. *Time-indexed formulation*

The time-indexed (TI) formulation was proposed by Sousa and Wolsey (1992) for a single-machine scheduling problem, which is the most common scheduling problem (Akker et al., 2000; Chan and Simchi-Levi, 1998; Lin Yang-Kuei, 2013; Kramer et al., 2020). The planning horizon is discretized into time periods $P = \{1, 2, \dots, H\}$, where H is an upper bound on the time horizon. We set $H = 2\max_{j\in J, k\in K} r_{jk} + \sum_{j\in J} p_j$. Let binary variable x_{jkt} take value 1 if job j is assigned to a machine at location k and starts processing at the beginning of time period t. Let binary variable w_k be 1 if location k houses a machine. TI is presented as follows:

$$(\text{TI}) \qquad \min \quad q_1 \sum_{k\in K} c_k w_k + 2q_2 \sum_{j\in J}\sum_{k\in K}\sum_{t\in P} e_{jk} x_{jkt} + q_3\theta \sum_{j\in J} T_j \tag{1}$$

s.t.

$$\sum_{k\in K} w_k \le m \tag{2}$$

$$\sum_{k\in K} \sum_{t=r_{jk}+1}^{H-p_j+1} x_{jkt} = 1 \qquad \forall j \in J \tag{3}$$

$$\sum_{j\in J}\sum_{t\in P} x_{jkt} \le n w_k \qquad \forall k \in K \tag{4}$$

$$\sum_{j\in J} \sum_{\tau=\max\{1, t-p_j+1\}}^{t} x_{jk\tau} \le 1 \qquad \forall k \in K, t \in P \tag{5}$$

$$C_j \ge \sum_{k\in K}\sum_{t\in P} t x_{jkt} + p_j - 1 \qquad \forall j \in J \tag{6}$$

$$T_j \ge C_j + \sum_{k\in K}\sum_{t\in P} r_{jk} x_{jkt} - d_j \qquad \forall j \in J \tag{7}$$

$$C_j \ge 0 \qquad \forall j \in J \tag{8}$$

$$T_j \ge 0 \qquad \forall j \in J \tag{9}$$

$$x_{jkt} \in \{0,1\} \qquad \forall j \in J, k \in K, t \in P \tag{10}$$

$$w_k \in \{0,1\} \qquad \forall k \in K. \tag{11}$$

The objective function (1) minimizes the weighted sum of the fixed location cost, job round-trip transportation cost, and total tardiness penalty cost of all jobs. Constraint (2) restricts the number of locations selected to at most the number of machines. Constraints (3) ensure that each job is processed exactly once on one machine. Constraints (4) indicate that a job can only be assigned to a location that houses a machine. Constraints (5) guarantee that one machine can process at most one job at any given time. Constraints (6) define the completion time of each job. Constraints (7) compute the tardiness of each job. Constraints (8)–(11) define the domains of variables.

The main advantage of TI is that its linear relaxation generally yields stronger lower bounds (LB) than other integer programming formulations (Dyer and Wolsey, 1990). Its main drawback is that the large number of variables and constraints weaken the computational efficiency (Akker et al., 2000). As a result, the efficiency of TI decreases quickly as the processing time of jobs increases, and this has limited its application to some extent. To overcome the inherent disadvantage of TI, enormous efforts have been devoted to developing methods including column generation (Akker et al., 2000), Lagrangian relaxation (Avella et al., 2005), and dual relaxation (Pan and Liang, 2017).

3.3.2. *Linear ordering formulation*

The linear ordering (LO) formulation was first considered by Chenery and Watanabe (1958) and was later applied to single-machine scheduling problems (Grötschel et al., 1984; Dyer and Wolsey, 1990; Li et al., 2021, 2022b). LO defines a binary variable v_{jk} to indicate whether job j is assigned to the machine at location k, and a binary sequence variable x_{ij} takes value 1 if job j is processed after job i, either immediately or not. LO is shown as follows:

$$(\text{LO}) \qquad \min \quad q_1 \sum_{k\in K} c_k w_k + 2q_2 \sum_{j\in J}\sum_{k\in K} e_{jk} v_{jk} + q_3\theta \sum_{j\in J} T_j \tag{12}$$

s.t. (2), (8), (9), (11), and to

$$\sum_{k\in K} v_{jk} = 1 \qquad \forall j \in J \tag{13}$$

$$\sum_{j\in J} v_{jk} \le n w_k \qquad \forall k \in K \tag{14}$$

$$C_j \ge p_j + \sum_{k\in K} r_{jk} v_{jk} \qquad \forall j \in J \tag{15}$$

$$C_j \ge C_i + p_j - L(3 - x_{ij} - v_{jk} - v_{ik}) \qquad \forall i,j \in J, i<j, k \in K \tag{16}$$

$$C_i \ge C_j + p_i - L(2 + x_{ij} - v_{jk} - v_{ik}) \qquad \forall i,j \in J, i<j, k \in K \tag{17}$$

$$T_j \ge C_j + \sum_{k\in K} r_{jk} v_{jk} - d_j \qquad \forall j \in J \tag{18}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in J, i<j \tag{19}$$

$$v_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K. \tag{20}$$

Constraints (13) correspond to (3). Constraints (14) perform the same function as constraints (4). Constraints (15) indicate that if job j is assigned to location k, its completion time is at least the sum of its processing time and release date to location k. Constraints (16) and (17) are the job sequence constraints, indicating that if job j is processed after job i on the same machine, the completion time of job j must be greater than or equal to the completion time of job i plus the processing time of job j. L is a large positive number, which is set to H. Constraints (18) compute the tardiness of each job. Constraints (19) and (20) define the variable domains.

We add valid inequalities adapted from similar literature to strengthen the LO formulation. The performances of these valid inequalities are assessed in the computational experiments in Section 5. These inequalities are formulated as follows.

$$v_{jk} \le w_k \qquad \forall j \in J, k \in K \tag{21}$$

$$x_{ij} + x_{jr} - x_{ir} \le 1 \qquad \forall i,j,r \in J, i<j<r \tag{22}$$

$$x_{ij} + x_{jr} - x_{ir} \ge 0 \qquad \forall i,j,r \in J, i<j<r \tag{23}$$

$$x_{ij} + x_{jr} + x_{rs} - x_{is} \le 2 \qquad \forall i,j,r,s \in J, i<j<r<s \tag{24}$$

$$x_{ij} + x_{js} - x_{rs} - x_{ir} \le 1 \qquad \forall i,j,r,s \in J, i<j<r<s \tag{25}$$

$$x_{ir} + x_{js} - x_{jr} - x_{is} \le 1 \qquad \forall i,j,r,s \in J, i<j<r<s \tag{26}$$

$$x_{ir} + x_{rs} - x_{js} - x_{ij} \le 1 \qquad \forall i,j,r,s \in J, i<j<r<s \tag{27}$$

$$x_{is} + x_{jr} - x_{js} - x_{ir} \le 1 \qquad \forall i,j,r,s \in J, i<j<r<s \tag{28}$$

$$x_{is} - x_{rs} - x_{jr} - x_{ij} \le 0 \qquad \forall i,j,r,s \in J, i<j<r<s. \tag{29}$$

Constraints (21) are the stronger counterpart of constraints (14). Constraints (22) and (23) are adapted from the 3-subsequence elimination constraints of Ríos-Mercado and Bard (2003). Constraints (22) indicate that if we have a sequence $i \to j \to r$, then the sequence $i \to r$ must hold, i.e., $x_{ij} + x_{jr} \le 1 + x_{jr}$ holds. Similarly, constraints (23) state that if we have a sequence $i \to r \to j$, then $i \to j$ holds and $x_{ir} + (1 - x_{jr}) \le 1 + x_{ij}$ must hold as well. Constraints (24)–(29) can be concluded from the 4-subsequence elimination constraints based on the sequences shown in Table 1, according to Ríos-Mercado and Bard (2003).

Table 1
4-subsequence elimination constraints [(Ríos-Mercado and Bard, 2003)].

Sequences	Constraints
$i \to j \to r \to s \Rightarrow i \to s$	$x_{ij} + x_{jr} + x_{rs} \le 2 + x_{is}$
$i \to j \to s \to r \Rightarrow i \to r$	$x_{ij} + x_{js} + (1 - x_{rs}) \le 2 + x_{ir}$
$i \to r \to j \to s \Rightarrow i \to s$	$x_{ir} + (1 - x_{jr}) + x_{js} \le 2 + x_{is}$
$i \to r \to s \to j \Rightarrow i \to j$	$x_{ir} + x_{rs} + (1 - x_{js}) \le 2 + x_{ij}$
$i \to s \to j \to r \Rightarrow i \to r$	$x_{is} + (1 - x_{js}) + x_{jr} \le 2 + x_{ir}$
$i \to s \to r \to j \Rightarrow i \to j$	$x_{is} + (1 - x_{rs}) + (1 - x_{jr}) \le 2 + x_{ij}$

3.3.3. Sequence-based formulation

We present a two-index model based on an additional binary sequence variable x_{ij} that takes value 1 if job j is processed immediately after job i on the same machine. The binary variable y_{jk} equals 1 if job j is the first job processed on a machine at location k. This enables us to omit the location variable w_k from the LO model. Binary variable v_{jk} has the same meaning as in LO. The sequence-based (SB) model is presented as follows:

$$\text{(SB)} \quad \min \quad q_1 \sum_{j\in J}\sum_{k\in K} c_k y_{jk} + 2q_2 \sum_{j\in J}\sum_{k\in K} e_{jk} v_{jk} + q_3\theta \sum_{j\in J} T_j \tag{30}$$

s.t. (8), (9), (13), (15), (18), (19), (20), and to

$$\sum_{j\in J}\sum_{k\in K} y_{jk} \le m \tag{31}$$

$$\sum_{j\in J} y_{jk} \le 1 \qquad \forall k \in K \tag{32}$$

$$y_{jk} \le v_{jk} \qquad \forall j \in J, k \in K \tag{33}$$

$$\sum_{k\in K} y_{jk} + \sum_{i\in J} x_{ij} = 1 \qquad \forall j \in J \tag{34}$$

$$\sum_{i\in J} x_{ji} \le 1 \qquad \forall j \in J \tag{35}$$

$$x_{ji} + x_{ij} \le 1 \qquad \forall j \in J, i \in J \setminus \{j\} \tag{36}$$

$$C_j \ge C_i + \sum_{k\in K} p_j - L(1 - x_{ij}) \qquad \forall j \in J, i \in J \setminus \{j\} \tag{37}$$

$$x_{ij} + v_{ik} \le 1 + v_{jk} \qquad \forall j \in J, i \in J \setminus \{j\}, k \in K \tag{38}$$

$$x_{ij} + v_{jk} \le 1 + v_{ik} \qquad \forall j \in J, i \in J \setminus \{j\}, k \in K \tag{39}$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j \in J \tag{40}$$

$$y_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K. \tag{41}$$

Constraint (31) corresponds to constraint (2). Constraints (32) ensure that at most one job can be the first processed at each location. Constraints (33) indicate that a job cannot be the first processed at a location if it is not assigned to it. Constraints (34) state that a job is the first one on a machine or follows another job. Constraints (35) indicate that a job can be followed by at most one job. Constraints (36) mean that job i cannot be both the predecessor and the successor of job j. Constraints (37) indicate that if job j is the successor of job i on a machine, the completion time of job j is at least the sum of the completion time of i and the processing time of j. Constraints (38) and (39) link the assignment variables to the scheduling variables, so that if two jobs are consecutively processed, they must be processed at the same location. In particular, constraints (38) restrict a job j to be processed at location k if $x_{ij} = 1$ and job i is assigned to location k; constraints (39) state that if jobs i and j are processed consecutively at the same location and job j is processed at location k, then job i must also be assigned to location k. Constraints (41) define variable y_{jk}.

4. Solution methods

The studied discrete DPSL-DD problem simultaneously determines machine locations, job assignments to machines, and job sequences on machines, and is a combination of two NP-hard problems, i.e., the facility location problem (Korupolu et al., 1998) and the parallel machine scheduling problem (Adamopoulos and Pappis, 1998). As a result, the DPSL-DD is also NP-hard. As preliminary experiments on CPLEX indicate that the performances of the previous formulations on medium- or large-sized instances dropped dramatically with large optimality gaps and long computation time, effective solution methods are highly needed to handle practical-sized instances. Thus we develop a logic-based Benders decomposition (LBBD) to exactly solve the studied DPSL-DD based on the good decomposition property of the problem.

4.1. Logic-based benders decomposition

Benders decomposition (Benders, 1962) has been successfully applied to large-sized MIPs. The original problem is decomposed into a master problem (MP) and a subproblem (SP) in a classic Benders decomposition method. Cuts are generated based on the dual information of the SP, which is required to be a linear program (LP). Hooker and Ottosson (2003) further generalized the classic Benders decomposition by allowing the SP to take any form instead of just LP, with cuts generated based on logic information. This method, known as LBBD, has shown its good performance in solving operating room planning (Roshanaei et al., 2017), vehicle routing (Fachini and Armentano, 2020), and machine scheduling problems (Emde et al., 2019; Sun et al., 2019; Li et al., 2022a).

We first decompose the studied DPSL-DD problem, where the MP determines the machine location and job assignment variables. Then, with these variables fixed, the SP solves a series of single-machine scheduling problems with total tardiness minimization. The MP is a relaxation of the original problem, and its objective value is the problem's lower bound (LB). Based on the solution to the MP, the SP solves the sequencing of jobs at each location, and the objective value is the upper bound (UB). Cuts will be generated after SP is solved and added to the MP at the next iteration. As the iterations continue, the LB increases, and the UB decreases. When the LB equals the UB, an optimal solution is obtained. The termination condition is met when an optimal solution is obtained or a time limit is reached. Next, we present the MP, SP, and Benders cuts.

4.1.1. Master problem

For the MP, let ψ be a lower bound of the total tardiness of all jobs. The MP can be formulated as follows:

$$MP: \quad \min \quad q_1 \sum_{k\in K} c_k w_k + 2q_2 \sum_{j\in J}\sum_{k\in K} e_{jk} v_{jk} + q_3\theta\psi \tag{42}$$

s.t.

$$\sum_{k\in K} w_k \le m \tag{43}$$

$$\sum_{k\in K} v_{jk} = 1 \qquad \forall j \in J \tag{44}$$

$$\sum_{j\in J} v_{jk} \le n w_k \qquad \forall k \in K \tag{45}$$

$$w_k \in \{0,1\} \qquad \forall k \in K \tag{46}$$

$$v_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K \tag{47}$$

$$CUTS. \tag{48}$$

The MP minimizes the weighted sum of the total location, transportation, and surrogate of the tardiness penalty costs. Note that ψ underestimates the real tardiness, and its value is increased by adding cuts (48). Solving the MP determines the values of the machine location variables w_k and job assignment variables v_{jk}.

The MP can yield a bad relaxation since no information about the total tardiness penalty cost is given. The following section is devoted to strengthening the MP.

4.1.2. Strengthening the master problem

To improve the computational efficiency, we propose several inequalities to strengthen the MP. These help the MP generate better lower bounds and reduce the number of cuts added. Define π_k as the total tardiness of jobs at location k, and let M_k be an upper bound on the total tardiness at each location k. M_k is computed by assigning all the n jobs to each location and sequencing them using the EDD rule. We add to the MP inequalities (21) and:

$$\psi \geq \sum_{k \in K} \pi_k \tag{49}$$

$$\psi \geq \sum_{j \in J} T_j \tag{50}$$

$$\pi_k \leq M_k w_k \qquad \forall k \in K \tag{51}$$

$$\pi_k \geq \sum_{j \in J} p_j v_{jk} + \min_{j \in J, h \in K} r_{jh} - \max_{j \in J, h \in K} (d_j - r_{jh}) \qquad \forall k \in K \tag{52}$$

$$\pi_k \geq \sum_{j \in J} T_j v_{jk} \qquad \forall k \in K \tag{53}$$

$$T_j \geq \sum_{k \in K} (p_j + 2r_{jk}) v_{jk} - d_j \qquad \forall j \in J \tag{54}$$

$$T_j \geq 0 \qquad \forall j \in J \tag{55}$$

$$\pi_k \geq 0 \qquad \forall k \in K. \tag{56}$$

Constraint (49) indicates that the total tardiness is no less than the tardiness generated on each machine. Constraint (50) means that the total tardiness is at least the total tardiness of all jobs. Inequalities (51) limit that the total tardiness at each location k is not greater than the upper bound M_k. Inequalities (52) indicate that the total tardiness generated at location k is not smaller than the total processing time of jobs assigned to location k, plus the minimum release date for all jobs to all locations, minus the maximum modified due date of all jobs, i.e., $\max_{j \in J} (d_j - r_{jk})$. Constraints (53) indicate that the total tardiness at each location k is at least the sum of the tardiness of the jobs that are assigned to it. Inequalities (54) ensure that if job j is assigned to location k, its tardiness is greater than or equal to the total of its processing time and round trip transportation time minus its due date. We can formulate the SP to generate cuts once the MP is solved with these inequalities.

As constraints (53) are nonlinear, we linearize them as follows.

We define a non-negative auxiliary variable τ_{jk} and a parameter $\hat{M}_{jk}$ indicating an upper bound on tardiness of job j on location k. We set $\hat{M}_{jk} = max(max_{j \in J} r_{jk} + \sum_{j \in J} p_j + r_{jk} - d_j, 0)$. Then we replace constraints (53) with the following inequalities.

$$\tau_{jk} \leq \hat{M}_{jk} v_{jk} \qquad \forall j \in J, k \in K \tag{57}$$

$$\tau_{jk} \geq T_j - \hat{M}_{jk}(1 - v_{jk}) \qquad \forall j \in J, k \in K \tag{58}$$

$$\pi_k \geq \sum_{j \in J} \tau_{jk} \qquad \forall k \in K \tag{59}$$

$$\tau_{jk} \geq 0 \qquad \forall j \in J, k \in K. \tag{60}$$

Inequalities (57) indicate that τ_{jk} is at most $\hat{M}_{jk}$ if job j is assigned to location k. Inequalities (58) limit τ_{jk} to be at least T_j if job j is assigned to location k, otherwise it equals 0 since it is non-negative. Based on (57) and (58), inequalities (59) perform the same function as (53).

Proposition 1. *Inequalities (52) provide lower bounds for the MP.*

Proof. Inequalities (52) define lower bounds for the total tardiness at each location k. To prove its validity, we provide the proof with three cases.

Case 1: $\sum_{j \in J} v_{jk} = 0$, i.e., no job is assigned to location k.

In this case, the right-hand side of (52) becomes $\min_{j \in J, h \in K} r_{jh} - \max_{j \in J, h \in K} (d_j - r_{jh})$. According to the problem assumption 4), $d_j \geq 2 \min_{h \in K} r_{jh} + p_j, \forall j \in J$. Thus we can get $\min_{h \in K} r_{jh} \leq d_j - \min_{h \in K} r_{jh}$ for each job j. Then we can further conclude that $\min_{h \in K} r_{jh} \leq \max_{j \in J, h \in K} (d_j - r_{jh})$, which means that the right-hand side of inequalities (52) is non-positive. Since $\pi_k \geq 0, \forall k \in K$, the inequality is satisfied.

Case 2: $\sum_{j \in J} v_{jk} = 1$, i.e., only one job is assigned to location k.

Assume only job i is assigned to location k, then we have $\pi_k = T_i \geq p_i + r_{ik} - (d_i - r_{ik})$. Also, (52) can be re-written as $\pi_k \geq p_i + \min_{j \in J, h \in K} r_{jh} - \max_{j \in J, h \in K} (d_j - r_{jh})$. Since $r_{ik} \geq \min_{j \in J, h \in K} r_{jh}$ and $d_i - r_{ik} \leq \max_{j \in J, h \in K} (d_j - r_{jh})$ are valid, we can further obtain that $\pi_k = T_i \geq p_i + r_{ik} - (d_i - r_{ik}) \geq p_i + \min_{j \in J, h \in K} r_{jh} - \max_{j \in J, h \in K} (d_j - r_{jh})$, i.e., inequality (52) is satisfied.

Case 3: $\sum_{j \in J} v_{jk} \geq 2$, i.e., at least two jobs are assigned to location k.

Consider an optimal solution in which all the assigned jobs are sequenced at location k. In this sequence, we denote the first- and last-processed jobs as i and j, respectively.

We can first obtain $\pi_k \geq T_j$ since the total tardiness π_k is the sum of the tardiness of jobs assigned to it. The completion time C_j of job j is at least the release date of job i plus the total processing time of all the assigned jobs, i.e., $C_j \geq r_{ik} + \sum_{j \in J} p_j v_{jk}$, and then the tardiness of job j can be computed as:

$$T_j = max\left(C_j + r_{jk} - d_j, 0\right) \geq r_{ik} + \sum_{j \in J} p_j v_{jk} - (d_j - r_{jk}) \tag{61}$$

Therefore we can further obtain that:

$$\begin{aligned} \pi_k \geq T_j &\geq r_{ik} + \sum_{j \in J} p_j v_{jk} - (d_j - r_{jk}) \\ &\geq \sum_{j \in J} p_j v_{jk} + \min_{j \in J, h \in K} r_{jh} - \max_{j \in J, h \in K} (d_j - r_{jh}) \end{aligned} \tag{62}$$

To conclude, inequality (52) is valid. $\square$

4.1.3. Subproblems

Given a feasible solution to the MP, machine location and job assignment variables are known. The SP corresponds to m single-machine scheduling problems with release dates to minimize the total tardiness $(1|r_j|\sum T_j)$. We define S_k as the set of jobs assigned to location k. For jobs in S_k, we define a binary variable y_{ij} equal to 1 if job j is processed immediately after job i. The SP for location k can be formulated as follows:

$$SP_k: \quad \min \sum_{j \in S_k} T_j \tag{63}$$

s.t.

$$y_{ij} + y_{ji} \leq 1 \qquad \forall i, j \in S_k \tag{64}$$

$$C_j \geq C_i + p_j - L(1 - y_{ij}) \qquad \forall i, j \in S_k \tag{65}$$

$$C_j \geq p_j + r_{jk} \qquad \forall j \in S_k \tag{66}$$

$$T_j \geq C_j + r_{jk} - d_j \qquad \forall j \in S_k \tag{67}$$

$$C_j \geq 0 \qquad \forall j \in S_k \tag{68}$$

$$T_j \geq 0 \qquad \forall j \in S_k \tag{69}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i, j \in S_k. \tag{70}$$

The objective (63) minimizes the total tardiness of jobs assigned to location k. Constraints (64) indicate that one job cannot simultaneously be the predecessor and successor of another. Constraints (65) and (66) are job completion time constraints. Constraints (67) compute job tardiness, while (68)–(70) define the variables.

The above $1|r_j|\sum T_j$ problem is NP-hard (Pinedo and Rammouz, 1988), unlike its makespan minimization version, which is solvable in polynomial time. Among the exact algorithms to solve it, the dynamic programming-based (DP) approach of Tanaka and Fujikuma (2012) is efficient. Tanaka et al. (2009) proposed an exact algorithm based on the successive sublimation DP method (Ibaraki and Nakamura, 1994) for a single-machine scheduling problem without machine idle time, and Tanaka and Fujikuma (2012) extended it to a single-machine scheduling problem with release date, which can solve instances of $1|r_j|\sum w_j T_j$ with up to 80 jobs within a reasonable computation time. Note that the efficient DP method of Tanaka has been used in other studies, including Tanaka and Araki (2013), Tanaka and Sato (2013) and Şen and Bülbül (2015).

We apply the DP of Tanaka and Fujikuma (2012) to solve the SP. Once the SP is solved, Benders cuts are generated based on the obtained solution of the SP.

4.1.4. Benders cuts

It can be observed that if machine locations and job-machine assignments are given, at least one feasible solution exists. Therefore, feasibility cuts were not generated. We now describe the generation of the optimality cuts. Denote the total tardiness at location k as π'_k, which is obtained by solving the SP. We add the following optimality cuts:

$$\pi_k \geq \pi'_k \left(\sum_{j \in S_k} v_{jk} - |S_k| + 1 \right) \qquad \forall k \in K \tag{71}$$

Cuts (71) indicate that if a superset of jobs of S_k is assigned to the machine at location k in subsequent iterations, the tardiness π_k of location k is at least π'_k. With this cut, the tardiness π_k will get much higher and the solver will reject the solution in the subsequent iterations. Note that cut (71) becomes nonbinding when at least one job is removed from S_k in subsequent iterations. The MP is solved by a branch-and-cut (B&C) procedure, and cuts (71) are dynamically added to each branching node of the search tree upon finding a feasible solution. This process terminates when an optimal solution is obtained or the time limit is reached.

Theorem 1. *Optimality cut (71) is valid.*

Proof. Let set S_k contain all the jobs assigned to location k, i.e., $S_k = \{j|v_{jk} = 1, \forall j \in J\}$. Let set S_k^h be the set containing all the jobs that are assigned to location k in the subsequent iteration h. Thus, we consider two cases.

Case 1: $S_k^h \cap S_k = S_k$. This case means that in a subsequent iteration h, all the jobs in set S_k are assigned to location k, i.e., $\sum_{j\in S_k} v_{jk} = |S_k|$. By adding this into cut (71) we can obtain $\pi_k \geq \pi'_k$, which indicates that the cut is valid.

Case 2: $S_k^h \cap S_k \neq S_k$. This case means that at least one job $j^* \in S_k$ is removed in the set S_k^h, i.e., $v_{j^*k} = 0$. Therefore we can obtain $\sum_{j\in S_k} v_{jk} < |S_k|$. Thus the right-hand side of cut (71) becomes non-positive and we can obtain $\pi_k \geq 0$ based on constraint (56). This indicates that the cut (71) will not remove new feasible solutions in the subsequent iterations. □

To conclude, the cut will limit the total tardiness π_k to at least π'_k when the same set of jobs as S_k is assigned to location k in the subsequent iterations and will not remove new feasible solutions. Thus the cut (71) is valid. The cut (71) is similar to that made in some existing papers, e.g., scheduling problem by Zhang et al. (2021) and order acceptance and scheduling problem by Naderi and Roshanaei (2020).

As analyzed in Section 5.5, both the MP and SP become difficult to solve when the instances get considerably large. To handle those challenging instances, a new matheuristic is proposed.

4.2. Matheuristic

Matheuristics (MH) are mathematical programming-based heuristics (Maniezzo et al., 2010) that have been widely used for the solution of integrated optimization problems, e.g., Yıldırım and Çatay (2014) and Avci and Yildiz (2019). A novel feature of an MH is that the general heuristic is built based on a mathematical programming model, which plays a central role in its performance (Fischetti and Fischetti, 2016). This idea has been widely applied to solve optimization problems such as the knapsack (Lahyani et al., 2019), health care planning (Nikzad et al., 2021), inventory-routing (Bertazzi et al., 2019; Schenekemberg et al., 2020), and scheduling (Fanjul-Peyro et al., 2017; Dang et al., 2021). We propose an MH based on a piecewise MILP model for the studied DPSL-DD problem. Our MH reduces the problem complexity by solving an approximate MILP model with predetermined job processing sequences on the machine at each location, thus excluding the binary sequencing variables from the approximate MILP model. MH first solves a series of $1|r_j|\sum T_j$ problems for each location. Then the original problem is reformulated with an approximate MILP in which the processing sequences of jobs on each machine are fixed to the given sequences. The approximate model is solved, providing the machine locations and job assignments. Finally, MH solves a series of $1|r_j|\sum T_j$ problems to improve the solution further. We now present the three steps of MH.

Step 1 obtains predetermined sequences for each location. For every location $k \in K$ and all jobs in the set J, a $1|r_j|\sum T_j$ problem is solved using the DP method of Tanaka and Fujikuma (2012). Let J_k be an ordered list of jobs for location k, in which the obtained processing sequences order the n jobs in J_k by the DP. We have also tried two other methods to sequence jobs at each location. One is to sort all jobs in nondecreasing order of their modified due dates, i.e., $d_j - r_j$, which tends to first process jobs with tighter due dates and far from their assigned location. Another is to sort all jobs in nondecreasing order of $d_j - p_j - r_j$, which is based on the modified Lawler's algorithm (Lawler, 1977). Because the DP method in Tanaka and Fujikuma (2012) can handle $1|r_j|\sum T_j$ problems with up to 80 instances, we performed preliminary experiments to evaluate the performances of the rules. The results indicate that for instances with fewer than 100 jobs, the sequence obtained by solving a $1|r_j|\sum T_j$ for each location yields the best results when solving the approximate model in Step 2. However, when instances get extensively large, the time consumption of the DP method increases sharply. We use the DP method for instances with fewer than 100 jobs to balance efficiency and effectiveness and apply the $d_j - p_j - r_j$ rule for larger instances.

Step 2 solves the approximate MILP model with predetermined processing sequences. Based on the ordered lists $J_k, k \in K$, we define a continuous variable Q_{jk} indicating the completion time for job j on the machine at location k. Let $Q_{j-1,k}$ be the completion time of the job processed right before job j at location k. The approximate MILP model, denoted as AP, is formulated as follows:

$$\text{(AP)} \qquad \min \quad q_1 \sum_{k\in K} c_k w_k + 2q_2 \sum_{j\in J}\sum_{k\in K} e_{jk} v_{jk} + q_3\theta \sum_{j\in J} T_j \tag{72}$$

s.t. (2), (8), (9), (11), (13), (14), (18), (20), (21), and to

$$C_j \geq Q_{jk} - L(1 - v_{jk}) \qquad \forall j \in J, k \in K \tag{73}$$

$$Q_{jk} \geq (r_{jk} + p_j)v_{jk} \qquad \forall j \in J, k \in K \tag{74}$$

$$Q_{jk} \geq Q_{j-1,k} + p_j v_{jk} \quad \forall j \in J_k \setminus \{1\}, k \in K. \tag{75}$$

Constraints (73) state that the completion time of job j is at least Q_{jk} if job j is assigned to location k. Constraints (74) indicate that if job j is assigned to location k, its completion time at that location cannot be less than the time to arrive at the location, i.e., the release date r_{jk}, plus processing time p_j. Constraints (75) provide the sequence rules for jobs at each location to ensure that the completion time of job j processed on a machine at location k is greater than or equal to the completion time of its predecessor in the ordered list J_k, plus the processing time of job j. Compared to the models in Section 3, this approximate model requires no binary sequence variables, which greatly reduces the complexity. By solving the approximate model, we obtain the values of machine location variables w_k and job assignment variables v_{jk}, and consequently, a feasible solution.

Step 3 solves a series of $1|r_j|\sum T_j$ problems to improve the solution. With variables w_k and v_{jk} fixed in Step 2, we further improve the

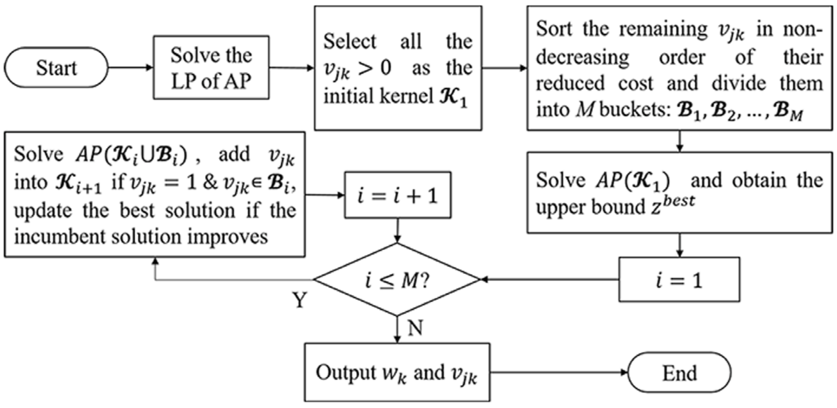

Start
Solve the LP of AP
Select all the $v_{jk} > 0$ as the initial kernel $\mathcal{K}_1$
Sort the remaining v_{jk} in non-decreasing order of their reduced cost and divide them into M buckets: $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_M$
Solve $AP(\mathcal{K}_1)$ and obtain the upper bound z^{best}
$i = 1$
$i \leq M$?
Y
N
Solve $AP(\mathcal{K}_i \cup \mathcal{B}_i)$, add v_{jk} into $\mathcal{K}_{i+1}$ if $v_{jk} = 1$ & $v_{jk} \in \mathcal{B}_i$, update the best solution if the incumbent solution improves
$i = i + 1$
Output w_k and v_{jk}
End

Fig. 3. Flow chart using KS to solve AP.

feasible solution by solving a $1|r_j|\sum T_j$ problem for each location with $w_k = 1$. The DP of Tanaka and Fujikuma (2012) is again used to solve these single-machine scheduling problems. Then, we obtain a near-optimal solution to the original problem.

Preliminary experiments show that the MH is very efficient on small and medium-sized instances, but performs poorly on large-sizes ones with hundreds of jobs and dozens of candidate locations. The reason is that the model AP cannot be efficiently solved using CPLEX within a reasonable time. So We design a tailored kernel search (KS) to substitute this part and improve the efficiency of MH. A combined approach MH-KS is proposed.

The KS algorithm was first designed by Angelelli et al. (2010) for a multi-dimensional knapsack problem and later applied in facility location (Filippi et al., 2021), nurse routing (Gobbi et al., 2019), and some other MILP problems (Guastaroba and Speranza, 2014; Guastaroba et al., 2017). The key idea of KS is to find the subset of the most promising binary decision variables as the kernel and solve the restricted mixed binary program (RMBP) with the kernel to be optimized. The remaining binary variables are fixed and split into several buckets to be iteratively added to the RMBP. The MH-KS method is described with the following two steps.

Step 1 builds the initial kernel. First of all, the linear relaxation of AP is firstly solved. If all the variables w_k and v_{jk} are integers in the solution, an optimal solution is obtained, and the process terminates. Otherwise, all the assignment variables v_{jk} taking positive values are selected as the initial kernel $\mathcal{K}_1$, and the remaining ones are sorted in non-decreasing order of their reduced costs. They are divided into several buckets with a given length L_b. We define the number of variables in the kernel $\mathcal{K}_1$ as N^k. According to the relevant literature (Guastaroba et al., 2017), for each bucket $\mathcal{B}_i$, its length is set to $L_b = N^k$, thus the number of buckets is $M = \lceil \frac{nl-N^k}{N^k} \rceil$, where nl indicates the total number of likely variables. Then, we solve the problem with $AP(\mathcal{K}_1)$, i.e., solving the AP with only the variables in the initial kernel $\mathcal{K}_1$ to be optimized, while those in the buckets are fixed to zero. The objective function value obtained is set as the initial upper bound z^{best}.

Step 2 solves a series of RMBP and update the solutions. For $i = 1, 2, \ldots, M$, $AP(\mathcal{K}_i \cup \mathcal{B}_i)$ is solved iteratively. In each iteration, the kernel $\mathcal{K}_i$ is updated with the variables in the bucket $\mathcal{B}_i$, and all variables taking value 1 will be added to the new kernel $\mathcal{K}_{i+1}$. If the new objective function value is better than the upper bound z^{best}, z^{best} is updated, and the current solution is recorded. The iteration continues until all the M buckets are utilized one by one, and the values of variables w_k and v_{jk} are determined.

The flow chart using KS to solve AP is shown in Fig. 3.

5. Computational experiments

To evaluate the performances of the proposed formulations and algorithms, we conducted numerical experiments on 720 randomly generated instances. The experiments are designed based on the principles proposed in Johnson (2002). In particular, these principles were used as guidance in generating instance data, conducting experiments, and presenting results. The instances were divided into several groups according to their sizes. From the small-sized ones to the extra-large sized ones. The size of the extra-large sized instances, with up to 300 jobs, 100 locations, and 50 machines, matches that of similar problems in the literature.

5.1. Data generation

Instances were randomly generated by varying the number of jobs, candidate locations, and machines, i.e., (n, l, m). The largest-sized instances solved by current studies, to our knowledge, are limited to 300 jobs and 20 machines for a parallel machine scheduling problem with total weighted tardiness minimization (Amorim et al., 2017). In the instances used in Wang et al. (2020a), the number of jobs ranged from six to 50, and the number of machines was not larger than 10. In Şen and Bülbül (2015), the experiments were conducted based on instances with up to five machines and 200 jobs. In Fang and Lin (2013), the maximum number of jobs and machines were 50 and five, respectively.

In our paper, we generated instances with a wide range of sizes. Specifically, we first generate instances with $n \in \{10, 20, \ldots, 100\}$, $l \in \{4, 6, 8, 10\}$, and $m = l/2$. These instances are divided into three groups, i.e., the small-sized ($n \in \{10, 20\}$), the medium-sized ($n \in \{30, 40, 50\}$), and the large-sized ones ($n \in \{60, 70, \ldots, 100\}$). We further generate some extra-large sized instances with $n \in \{150, 200, 250, 300\}$, $l \in \{20, 40, \ldots, 100\}$, and $m = l/2$. The coordinates of job storage and candidate locations were randomly generated from a uniform distribution in the interval $[0, 100]$. The distance D_{jk} between job j and location k was calculated as the Euclidean distance, rounded down to the nearest integer. Following the method of Liu et al. (2019), the fixed cost was calculated as $c_k = \hat{c} + \bar{n}_k \cdot \bar{c}$, where $\hat{c} = 20$, $\bar{c} = 10$, and $\bar{n}_k$ is the number of jobs whose distance to location k was less than a predetermined threshold $\bar{d}$. For the value of $\bar{d}$, we sorted the distances between jobs and locations in non-decreasing order to obtain the vector d, with $n \times l$ elements. Then $\bar{d}$ was set to be the $\lfloor \frac{1}{3} \cdot n \cdot l \rfloor$th element in d. The processing time p_j was uniformly generated in the interval $[1, 100]$. Based on Shim and Kim (2007) and Ching-Fang Liaw and Chen (2003), the due dates were generated from a uniform distribution with range $[\bar{P}(1 - TF - RDD/2), \bar{P}(1 - TF + RDD/2)]$, where $\bar{P} = \sum_{j \in J} p_j / m$, the tightness factor of the due date $TF = \{0.2, 0.4, 0.6\}$, and the range factor $RDD = \{0.2, 0.4, 0.6, 0.8\}$. For each instance, the weights of the fixed location cost, the job transportation cost, and the total tardiness penalty

Table 2
Performance of valid inequalities for the LO formulation.

n	Linear Relaxation						CPLEX					
	LP Bound			Time (s)			Gap (%)$^{(opt)}$			Time (s)		
	LO	LO+(21)	LO+all	LO	LO+(21)	LO+all	LO	LO+(21)	LO+all	LO	LO+(21)	LO+all
10	596.23	730.53	730.53	0.01	0.01	0.02	0.17$^{(47)}$	0.11$^{(47)}$	0.17$^{(46)}$	22.57	23.76	29.53
20	1057.84	1359.17	1359.17	0.03	0.03	0.17	5.50$^{(12)}$	5.37$^{(12)}$	7.98$^{(6)}$	696.12	693.64	839.55
30	1491.90	1939.73	1939.73	0.06	0.05	1.10	10.90$^{(1)}$	11.09$^{(0)}$	19.08$^{(0)}$	898.89	900.02	900.36
40	1969.52	2549.65	2549.65	0.07	0.08	4.66	16.67$^{(0)}$	16.46$^{(0)}$	43.96$^{(0)}$	900.03	900.03	901.31
50	2428.05	3224.81	3224.81	0.11	0.11	24.05	18.11$^{(0)}$	18.21$^{(0)}$	64.78$^{(0)}$	900.05	900.04	903.59
Average	1508.71	1960.78	1960.78	0.06	0.06	6.00	10.27$^{(60)}$	10.25$^{(59)}$	27.19$^{(52)}$	683.53	683.50	714.87

Table 3
Performance of valid inequalities for the LBBD.

Jobs	Gap (%)$^{(opt)}$				Time (s)$^{(opt)}$			
	LBBD	LBBD+(21)	LBBD+(51)	LBBD+(53)	LBBD	LBBD+(21)	LBBD+(51)	LBBD+(53)
10	0.00$^{(48)}$	0.00$^{(48)}$	0.00$^{(48)}$	0.00$^{(48)}$	0.16	0.14	0.17	0.16
20	0.46$^{(46)}$	0.43$^{(46)}$	0.43$^{(46)}$	0.38$^{(46)}$	47.51	54.02	45.98	47.35
30	3.41$^{(34)}$	3.27$^{(34)}$	3.04$^{(34)}$	3.08$^{(34)}$	288.92	289.38	288.04	287.12
40	8.64$^{(23)}$	8.49$^{(24)}$	7.98$^{(24)}$	7.97$^{(24)}$	482.91	482.89	479.35	480.98
50	10.40$^{(16)}$	10.08$^{(15)}$	9.19$^{(15)}$	9.19$^{(15)}$	634.81	642.39	633.80	636.63
Average	4.58$^{(167)}$	4.46$^{(167)}$	4.13$^{(167)}$	4.12$^{(167)}$	290.86	293.76	289.47	290.45

cost are set to $q_1 = q_2 = q_3 = 1$, and the tardiness penalty coefficient θ was generated from a uniform distribution in the interval $[0.1, 0.5]$. A total of $(10 \times 4 + 5 \times 4) \times 3 \times 4 = 720$ instances were generated for numerical experiments, and are available along with detailed results at https://www.dmu-yantongli.com/instances.

We coded all the models and algorithms in C++ linked with CPLEX 12.10. The models are solved using the default B&C of CPLEX. A time limit of 900 s was imposed on each run. More specifically, in the MH-KS, a time limit of 150 s for resolution is set to each reduced subproblem of the KS. All experiments were run on a computer with an Intel Xeon CPU E5-2690 v3 at 2.60 GHz with 32 GB RAM. Next, we firstly evaluate the performances of the valid inequalities and then present the computational results of all the solution methods.

5.2. Performance of valid inequalities

We now demonstrate the effectiveness of adding the proposed valid inequalities for LO and the MP of LBBD by conducting experiments on the instances with $n = \{10, 20, \dots, 50\}$. All models are solved by using CPLEX 12.10 with a time limit of 900 s.

5.2.1. Performance of valid inequalities for the LO formulation

We first report the performance of the valid inequalities for the LO formulation. The results are presented in Table 2, where the first column shows the number of jobs. The following six columns are the LP bounds and the execution time for LO, LO+(21), and LO plus all valid inequalities from (21) to (29). The last six columns correspond to the optimality gaps and the computation time for the three variants. The superscript (opt) indicates the number of optimal solutions obtained. The value of gaps is calculated as

$$Gap\ (\%) = \frac{UB - LB}{UB} \times 100, \tag{76}$$

where UB and LB are the upper and lower bounds, respectively. Note that each entity in the table is the average of 48 instance solutions.

From Table 2, we can find that in terms of the LP relaxations, LO+(21) and LO+all reach the same LP bound, with an average of 1960.78, which is better than that of LO (1508.71). This indicates that the LO+(21) and LO+all obtain better LP bounds than LO and the improvements are due to the inclusion of constraints (21). The average time consumed by LO+(21) is 0.06s, which is as fast as LO. LO+all is inferior to LO and LO+(21) as it takes 6.00s on average. Regarding the solution quality directly obtained using CPLEX, LO+(21) and LO are of the same level as the average optimality gaps obtained by LO+(21) is slightly lower than that of LO (10.25% vs. 10.27%). In contrast, LO obtains one more optimal solution than LO+(21) does (60 vs. 59). Taking both the LP bound and CPLEX performance into consideration, we can conclude that LO+(21) outperforms LO and LO+all, which can reach a good balance between solution quality and computational efficiency.

Therefore, for the LO formulation, we use LO+(21) in the remainder of the paper and, for simplicity, still denote it as LO in the computational experiments.

Table 4
Performance of strengthening techniques for the LBBD.

Jobs	Gap (%)		Time (s)		(opt)	
	LBBD	LBBD-N	LBBD	LBBD-N	LBBD	LBBD-N
10	0.00	0.00	0.15	0.17	48	48
20	0.37	0.74	45.96	100.09	46	45
30	3.11	4.88	289.30	405.77	34	27
40	7.92	12.23	483.32	585.41	23	19
50	9.26	13.61	637.53	714.60	16	11
60	8.95	12.98	659.52	772.56	14	7
70	11.93	16.52	676.84	721.26	12	10
80	11.11	15.90	728.56	791.62	11	7
90	14.94	20.04	749.02	787.41	9	7
100	15.94	22.26	724.22	817.32	10	5
Average (Sum)	8.35	11.92	499.44	569.62	223	186

5.2.2. Performance of valid inequalities for the LBBD

The performance of valid inequalities for the LBBD is shown in Table 3, in which MP and SP are solved directly using CPLEX. The computational results of LBBD combining (57)–(60) is still denoted as LBBD+(53). The table shows that the three variants obtain the same optimal solutions (167 out of 240). All the inequalities (21), (51) and (53) can improve the solutions compared with the basic LBBD. The performances of LBBD+(51) and LBBD+(53) are similar, and both of them outperform LBBD+(21) with a lower average gap and shorter average computation time. Therefore, we add all the three inequalities to the LBBD and, for simplicity, still denote it as LBBD in the computational experiments.

To evaluate the performance of all the strengthening techniques for the master problem, we present the computational results of the LBBD with and without the strengthening techniques on 480 instances with up to 100 jobs in Table 4. In the table, LBBD and LBBD-N represent the computational results of LBBD with and without the strengthening techniques of Section 4.1.2, respectively. Specifically, LBBD-N only uses the inequalities (49) and (56) to ensure the connectivity between MP and SP. The last row of the table shows the average gap, the average computation time, and the total number of optimal solutions, respectively. From Table 4, we see that the strengthening techniques applied in LBBD significantly improve the computational performance. The average gap of the 480 instances obtained by LBBD is 8.35%, increasing to 11.92% without the strengthening techniques. LBBD is also more efficient than LBBD-N as their average computation times are 499.44 s and 569.62 s, respectively. As for the optimal solutions, LBBD solves 223 instances to optimality, while LBBD-N only solves 186 instances optimally. The computational results show that the strengthening techniques make LBBD effective and efficient, obtaining more optimal solutions and a lower average gap in a shorter computation time.

We next report detailed computational results for each subset of the instances.

5.3. Computational results for small-sized instances

The results for small-sized instances with up to 20 jobs and 10 locations are presented in Table 5. We report the optimality gaps and computation time obtained by each model and algorithm. Since both MH and MH-KS cannot provide a lower bound, the best lower bound obtained by TI, LO, SB, and LBBD is used to evaluate its quality. It can be seen from Table 5 that models TI, LO, and SB can give feasible solutions to all 96 small-sized instances. They solve to optimality 75, 59, and 39 instances, respectively. TI is the best among the three models in the average gap (1.88%) and computation time (271.10s). It can obtain optimal solutions for all the instances with 10 jobs in less than 30 s. SB is generally the worst of the three models, obtaining a gap of 18.83% in 624.30 s. LO is slightly inferior to TI, but it is much better than SB. As the number of jobs increases from 10 to 20, the relative gaps and computation time increase for all models. SB loses the capability of obtaining optimal solutions within the time limit, as it provides no optimal solution. It is interesting to observe that the relative gap decreases as the number of locations increases. For example, the average gap of TI decreases from 7.24% to 1.04% as the number of locations increases from four to ten for instances with 20 jobs. The reason might be that more jobs will be assigned to each location for instances with fewer locations, resulting in a longer total processing time on each machine, which will inherently reduce the efficiency of TI. LBBD and MH provide near-optimal solutions for all small-sized instances among the proposed solution methods. LBBD solves 94 out of 96 instances to optimality in 23.05 s, obtaining a tiny average gap (0.18%) within one second. It solves all instances with ten jobs to optimality. The average gap of MH is 0.28%, much better than the models, and it is also the most efficient one, with an average computation time of 1.31 s. Compared to MH, its variant MH-KS is not satisfactory since it only obtains 36 optimal solutions with an average gap of 3.66%. However, its computational efficiency is the best among all the methods, with an average computation time about 0.88 s.

In summary, the results on small-sized instances show that TI performs best among the three models. LO is inferior to TI but is much better than SB. LBBD provided the most optimal solutions, with smaller gaps, in a shorter computation time, and MH obtained near-optimal solutions with high efficiency. MH-KS performs poorly in solution quality, but it solves the instances in the shortest computation time. The results clearly show that LBBD significantly outperformed the other models and MH on small-sized instances.

5.4. Computational results for medium-sized instances

We present results for medium-sized instances in Table 6, where the symbol "–" indicates that a model or method cannot provide a feasible solution for at least one of the 12 instances of the same size. We see from Table 6 that TI and SB fail to provide feasible solutions for instances with more than 30 jobs, and the gaps exceed 70% for some instances for which a feasible solution is obtained. LO can provide feasible solutions for all medium-sized instances, but no instance is solved optimally. On the other hand, LBBD solves 73 out of the 144 medium-sized instances to optimality with an average gap of 6.76% over all medium-sized instances. MH and MH-KS provide fewer optimal solutions (22 and 12) with a slightly higher average gap (7.19% and 9.91%). Regarding computation time, all three models reach the time limit in most instances, while LBBD, MH, and MH-KS take less time. MH-KS is significantly the most efficient method among the three as the average computation time is 65.28 s, approximately one-seventh that of LBBD and one-fifth that of MH. We note that the performances of MH and MH-KS depend on the number of locations. Taking instances with 40 jobs as an example, when the number of locations is four, the average computation time of MH is 39.01 s. However, it increases sharply to 516.48 s when there are 10 locations. At the same time, the average gap decreases from 10.68% to 6.55%. For MH-KS, we can find that more locations will bring a lower average gap and longer computation time for instances with the same number of jobs. To summarize, LO consistently performs best among the three models. The developed solution methods LBBD, MH, and MH-KS outperform LO. Comparing LBBD, MH, and MH-KS, LBBD provides more optimal solutions and a slightly better gap. MH-KS obtains a slightly inferior gap in a much shorter computation time.

5.5. Computational results for large-sized instances

We report the results for large-sized instances, whose solution is more challenging than in small- and medium-sized instances. The results are shown in Table 7, from which we can find that TI and SB lose power in such challenging instances. LO provides feasible solutions for all instances with only one solved optimally. The average gap is 21.41%. LBBD solves 56 instances to optimality, with an average gap of 12.58%. Moreover, LBBD optimally solves ten instances with 100 jobs, which indicates that it significantly outperforms LO. MH and MH-KS

Table 5
Computational results for small-sized instances.

Instances			Gap (%)$^{(opt)}$						Time (s)					
n	*l*	*m*	TI	LO	SB	LBBD	MH	MH-KS	TI	LO	SB	LBBD	MH	MH-KS
10	4	2	$0.00^{(12)}$	$0.00^{(12)}$	$3.71^{(8)}$	$0.00^{(12)}$	$0.23^{(10)}$	$4.12^{(6)}$	23.12	7.97	500.79	0.19	0.11	0.24
	6	3	$0.00^{(12)}$	$0.00^{(12)}$	$0.91^{(11)}$	$0.00^{(12)}$	$0.21^{(10)}$	$5.18^{(3)}$	10.00	3.09	292.27	0.12	0.19	0.35
	8	4	$0.00^{(12)}$	$0.00^{(12)}$	$0.97^{(11)}$	$0.00^{(12)}$	$0.16^{(9)}$	$2.76^{(8)}$	13.53	5.62	241.46	0.11	0.31	0.49
	10	5	$0.00^{(12)}$	$0.45^{(11)}$	$2.85^{(9)}$	$0.00^{(12)}$	$0.25^{(9)}$	$3.53^{(6)}$	19.16	78.38	359.84	0.19	0.39	0.72
20	4	2	$7.24^{(3)}$	$7.87^{(0)}$	$32.88^{(0)}$	$0.71^{(11)}$	$0.34^{(8)}$	$3.15^{(4)}$	804.62	900.01	900.01	79.46	0.58	0.40
	6	3	$4.40^{(7)}$	$6.80^{(3)}$	$35.17^{(0)}$	$0.76^{(11)}$	$0.50^{(7)}$	$3.06^{(4)}$	517.28	707.48	900.01	94.31	3.22	0.58
	8	4	$2.38^{(8)}$	$3.20^{(4)}$	$34.55^{(0)}$	$0.00^{(12)}$	$0.33^{(7)}$	$4.26^{(3)}$	396.90	604.89	900.01	6.25	1.86	2.42
	10	5	$1.04^{(9)}$	$3.60^{(5)}$	$39.57^{(0)}$	$0.00^{(12)}$	$0.21^{(7)}$	$3.22^{(2)}$	384.17	562.18	900.02	3.82	3.85	1.81
Average			$1.88^{(75)}$	$2.74^{(59)}$	$18.83^{(39)}$	$0.18^{(94)}$	$0.28^{(67)}$	$3.66^{(36)}$	271.10	358.70	624.30	23.05	1.31	0.88

Table 6
Computational results for medium-sized instances.

Instances			Gap (%)$^{(opt)}$						Time (s)					
n	l	m	TI	LO	SB	LBBD	MH	MH-KS	TI	LO	SB	LBBD	MH	MH-KS
30	4	2	20.13$^{(0)}$	11.25$^{(0)}$	34.67$^{(0)}$	4.34$^{(7)}$	4.75$^{(1)}$	11.44$^{(1)}$	902.01	900.02	900.02	436.64	2.16	0.78
	6	3	19.91$^{(0)}$	10.74$^{(0)}$	45.43$^{(0)}$	2.82$^{(9)}$	2.13$^{(3)}$	8.09$^{(2)}$	903.23	900.02	900.02	235.31	10.99	2.31
	8	4	28.46$^{(0)}$	11.23$^{(0)}$	45.03$^{(0)}$	2.71$^{(9)}$	2.99$^{(4)}$	4.97$^{(2)}$	904.11	900.02	900.03	242.62	166.72	32.73
	10	5	–	11.12$^{(0)}$	48.89$^{(0)}$	2.56$^{(9)}$	2.64$^{(4)}$	3.54$^{(3)}$	–	900.02	900.03	242.64	252.44	82.41
40	4	2	43.48$^{(0)}$	16.40$^{(0)}$	43.44$^{(0)}$	10.03$^{(6)}$	10.68$^{(1)}$	14.07$^{(0)}$	903.85	900.02	900.02	475.71	39.01	3.23
	6	3	58.11$^{(0)}$	16.15$^{(0)}$	–	8.64$^{(7)}$	8.72$^{(3)}$	9.93$^{(1)}$	906.93	900.02	–	384.24	277.29	39.70
	8	4	–	17.49$^{(0)}$	62.73$^{(0)}$	7.16$^{(5)}$	7.58$^{(2)}$	11.18$^{(2)}$	–	900.03	900.06	528.19	510.18	93.68
	10	5	–	15.90$^{(0)}$	65.56$^{(0)}$	5.84$^{(5)}$	6.55$^{(1)}$	8.43$^{(1)}$	–	900.03	900.05	545.16	516.48	211.56
50	4	2	73.09$^{(0)}$	19.26$^{(0)}$	54.01$^{(0)}$	11.56$^{(3)}$	13.12$^{(1)}$	16.90$^{(0)}$	905.98	900.03	900.04	742.49	156.07	3.21
	6	3	–	19.91$^{(0)}$	–	10.68$^{(3)}$	11.20$^{(0)}$	13.06$^{(0)}$	–	900.04	–	686.71	463.55	25.16
	8	4	–	16.81$^{(0)}$	74.08$^{(0)}$	8.00$^{(5)}$	8.54$^{(1)}$	9.28$^{(0)}$	–	900.05	900.07	543.19	800.23	70.72
	10	5	–	16.84$^{(0)}$	–	6.81$^{(5)}$	7.42$^{(1)}$	8.02$^{(0)}$	–	900.06	–	577.73	693.16	217.95
Average			–	15.26$^{(0)}$	–	6.76$^{(73)}$	7.19$^{(22)}$	9.91$^{(12)}$	–	900.03	–	470.05	324.02	65.28

Table 7
Computational results for large-sized instances.

Instances			Gap (%)$^{(opt)}$						Time (s)					
n	l	m	TI	LO	SB	LBBD	MH	MH-KS	TI	LO	SB	LBBD	MH	MH-KS
60	4	2	–	16.96$^{(0)}$	–	10.21$^{(4)}$	11.44$^{(0)}$	16.70$^{(0)}$	–	900.04	–	639.00	101.90	5.60
	6	3	–	15.61$^{(1)}$	–	9.20$^{(3)}$	10.58$^{(1)}$	11.80$^{(0)}$	–	890.53	–	700.92	504.13	16.98
	8	4	–	17.82$^{(0)}$	–	8.84$^{(4)}$	10.26$^{(0)}$	11.94$^{(0)}$	–	900.07	–	613.15	670.48	149.67
	10	5	–	17.53$^{(0)}$	–	7.56$^{(3)}$	9.08$^{(0)}$	10.99$^{(0)}$	–	900.09	–	685.02	823.58	371.74
70	4	2	–	20.26$^{(0)}$	–	13.21$^{(3)}$	15.93$^{(0)}$	17.70$^{(0)}$	–	900.06	–	675.67	368.34	9.13
	6	3	–	20.68$^{(0)}$	–	12.44$^{(3)}$	13.91$^{(0)}$	15.19$^{(0)}$	–	900.07	–	677.49	611.58	70.39
	8	4	–	20.44$^{(0)}$	–	11.72$^{(3)}$	13.05$^{(1)}$	15.44$^{(0)}$	–	900.09	–	675.55	765.22	187.43
	10	5	–	19.49$^{(0)}$	–	10.34$^{(3)}$	11.62$^{(0)}$	12.81$^{(0)}$	–	900.10	–	678.63	908.75	427.82
80	4	2	–	24.83$^{(0)}$	–	16.14$^{(3)}$	17.33$^{(0)}$	21.32$^{(0)}$	–	900.07	–	675.59	539.56	64.79
	6	3	–	19.18$^{(0)}$	–	10.92$^{(3)}$	12.12$^{(0)}$	15.08$^{(0)}$	–	900.10	–	750.17	616.00	83.73
	8	4	–	16.20$^{(0)}$	–	8.05$^{(3)}$	9.18$^{(1)}$	10.52$^{(1)}$	–	900.13	–	714.06	843.16	245.03
	10	5	–	17.34$^{(0)}$	–	9.34$^{(2)}$	10.19$^{(0)}$	12.11$^{(0)}$	–	900.15	–	774.41	895.03	362.90
90	4	2	–	24.50$^{(0)}$	–	16.94$^{(3)}$	17.53$^{(0)}$	20.22$^{(0)}$	–	900.10	–	675.64	380.74	48.77
	6	3	–	23.17$^{(0)}$	–	14.95$^{(2)}$	15.72$^{(0)}$	17.87$^{(0)}$	–	900.13	–	756.85	647.95	117.07
	8	4	–	25.12$^{(0)}$	–	16.31$^{(3)}$	16.47$^{(0)}$	18.30$^{(0)}$	–	900.15	–	738.11	769.86	236.71
	10	5	–	21.28$^{(0)}$	–	11.57$^{(1)}$	11.71$^{(0)}$	13.73$^{(0)}$	–	900.18	–	825.47	854.96	441.14
100	4	2	–	28.64$^{(0)}$	–	16.96$^{(2)}$	18.70$^{(0)}$	22.60$^{(0)}$	–	900.12	–	750.87	538.73	67.07
	6	3	–	28.27$^{(0)}$	–	15.29$^{(1)}$	17.98$^{(0)}$	21.04$^{(0)}$	–	900.17	–	825.37	880.58	178.59
	8	4	–	26.34$^{(0)}$	–	16.54$^{(3)}$	16.72$^{(0)}$	18.85$^{(0)}$	–	900.20	–	711.30	921.77	377.70
	10	5	–	24.59$^{(0)}$	–	14.98$^{(4)}$	15.06$^{(0)}$	15.71$^{(0)}$	–	900.23	–	609.34	852.63	542.69
Average			–	21.41$^{(1)}$	–	12.58$^{(56)}$	13.73$^{(3)}$	16.00$^{(1)}$	–	899.64	–	707.63	674.75	200.24

obtain significantly fewer optimal solutions than LBBD. MH-KS solves the large-sized instances with an average computation time of 200.24 s, much shorter than LBBD and MH.

5.6. Computational results for extra-large sized instances

As Table 7 indicates that TI and SB fail to obtain any feasible solution on large-sized instances, only LO, LBBD, and MH are tested in the group of extra-large sized instances. Note that this group involves instances with 150 to 300 jobs, we sort the jobs on each location with non-decreasing order of $(d_j - p_j - r_{jk})$, as stated in Section 4.2. Also, to further improve the efficiency of MH-KS, we solve the LR of MH before the first step of MH-KS and define a potential location parameter $\bar{w}_k = 1$ if the LR value of $w_k \geq 0$, 0 otherwise. Then, constraints (77) are added to the later steps of MH-KS to limit the scope of location selection. The number of buckets enumerated is also limited to $min(3, M)$ to make sure that the computation time limit is also 900 s.

$$w_k \leq \bar{w}_k \qquad \forall\, k \in K. \tag{77}$$

The extra-large sized instances are even more challenging than the large-sized instances. The obtained LB is generally weak. We only compare the obtained UB of LO, LBBD, MH, and MH-KS. The results are reported in Table 8. In the table, the obtained UB of LO, LBBD, MH, and MH-KS is denoted as UB_{LO}, UB_{LBBD}, UB_{MH}, and $UB_{MH\text{-}KS}$, respectively. To show the improvements between different solution methods, we define $\%impr1$ as the percentage of improvement of LBBD to LO, i.e., $\%impr1 = (UB_{LO} - UB_{LBBD})/UB_{LO} \times 100\%$. Similarly, we define $\%impr2$ and $\%impr3$ as the percentage of improvement of MH to LBBD, and MH-KS to LBBD, respectively.

From Table 8, we can find that the four methods can find feasible solutions for all the extra-large sized instances. At the same time, LO can only provide feasible solutions with poor quality. None of the instances are solved to optimality by all the solution methods. Based on the objective values, LBBD can obtain much better solutions than the LO. The average value of $\%impr1$ is 88.87% for all the instances within this group, and some even exceed 94%. We can observe from the value of $\%impr2$ that MH and LBBD are not dominated by one another, and LBBD is slightly superior given the average value of $\%impr2$ (7.19%). We can also find that $\%impr3 \geq 0$ for each group of the extra-large sized instances. The average value of $\%impr3$ is 23.94%, indicating that MH-KS is significantly superior to LBBD on extra-large sized instances. To conclude, LBBD and MH are much better than LO, and MH-KS further outperforms LBBD with lower UB.

From the above discussions, we conclude the following:

(1) LO performs best among the three proposed models. It solves more instances to optimality, obtains smaller gaps, and provides feasible solutions for all the 720 instances. The other two models obtain

Table 8
Computational results for extra-large sized instances.

n	l	m	UB_{LO}	UB_{LBBD}	UB_{MH}	$UB_{MH\text{-}KS}$	%impr1	%impr2	%impr3
150	20	10	40261.40	11692.34	11947.57	11611.38	70.96	−2.18	0.69
	40	20	96039.19	16120.69	15518.49	14198.61	83.21	3.74	11.92
	60	30	141469.26	17931.53	17785.13	16216.13	87.32	0.82	9.57
	80	40	90718.68	17372.90	16054.97	15049.68	80.85	7.59	13.37
	100	50	129863.49	23457.83	19680.76	18709.89	81.94	16.10	20.24
200	20	10	276871.63	17089.20	15837.39	15488.27	93.83	7.33	9.37
	40	20	224597.83	20536.07	20410.18	18053.97	90.86	0.61	12.09
	60	30	327399.83	30565.70	28423.74	23714.29	90.66	7.01	22.42
	80	40	332045.67	33661.73	29204.88	24499.63	89.86	13.24	27.22
	100	50	354959.67	36656.78	34485.45	26117.47	89.67	5.92	28.75
250	20	10	535414.58	27757.49	23663.98	22418.87	94.82	14.75	19.23
	40	20	444038.92	35105.02	26592.47	24407.78	92.09	24.25	30.47
	60	30	468511.42	42779.94	35690.25	28404.67	90.87	16.57	33.60
	80	40	484964.92	50088.13	38915.78	31012.63	89.67	22.31	38.08
	100	50	549007.42	58017.50	51004.73	35187.61	89.43	12.09	39.35
300	20	10	816275.58	30916.83	28615.33	27048.10	96.21	7.44	12.51
	40	20	769325.17	45323.93	40691.91	31793.08	94.11	10.22	29.85
	60	30	727535.50	65248.27	62660.46	39134.33	91.03	3.97	40.02
	80	40	721580.75	68298.22	53912.37	41811.93	90.53	21.06	38.78
	100	50	806877.33	85082.58	126810.19	49960.01	89.46	−49.04	41.28
Average			416887.91	36685.13	34895.30	11627.32	88.87	7.19	23.94

fewer optimal solutions and cannot provide a feasible solution for a large subset of medium- and large-sized instances.

(2) The proposed models gradually lose power as the size of an instance increases, while LBBD and MH obtain many more optimal solutions or a much better gap.

(3) LBBD and MH outperform LO in solution quality and computational efficiency. LBBD obtains significantly more optimal solutions, and MH provides a very competitive gap. LBBD performs consistently well in solving instances of all sizes. MH is slightly inferior to LBBD in solution quality but superior to LO. MH-KS solves all the instances with the highest efficiency and performs the best on extra-large sized instances.

5.7. Details of LBBD

To better describe the proposed LBBD algorithm, we run the 480 instances with $n \in \{10, 20, \ldots, 100\}$ and record some detailed information about the process, which is shown in Fig. 4. In the figure, (a) reports the average gap, the three lines in (b) represent the computation time of the SP, the MP, and the total algorithm, respectively, (c) indicates the number of iterations between the MP and SP, and (d) is the number of Benders cuts generated and added to the MP. The horizontal $\boldsymbol{x}$-axis shows the number of jobs, which means that each entity is the average of 48 instance solutions.

The figure shows that the LBBD is efficient and effective when the instances are small. When $n = 10$, the average gap and the computation time are nearly zero, and the number of iterations and cuts are tiny, indicating that the LBBD can obtain optimal solutions for instances with ten jobs efficiently and quickly. When the instances become larger, the proposed DPSL-DD problem becomes more complicated, leading to higher average gaps, longer computation time, and more iterations and Benders cuts. We can find in the figure that when $n = 20$, the increase is modest, but it rises sharply when $n \geq 30$. The average gaps depicted in Fig. 4a generally maintain an upward trend, which means that the larger the instances are, the lower the solution qualities will be. From Fig. 4b, we can find that the total computation time increases when the instances become larger and stays stable when $n \geq 80$. This indicates that the instances with over 80 jobs (114 of 144) can hardly be optimally solved within 900 s. The time consumed by MP increases when the instances are small and start to decrease when $n \geq 50$, while the SP costs more time when the instances become larger. At first, the MP consumes more time than the SP, but this is inversed when $n \geq 60$. This might be the result of the time limit set on computation.

The computation time of MP and SP impacts the number of iterations and cuts, shown in Fig. 4c and Fig. 4d, respectively. The number of iterations and Benders cuts increases when the instance size increases. However, when the instances are large enough ($n \geq 60$), the SP takes more time in each iteration, decreasing the number of iterations and cuts.

5.8. Analysis of the impact of due dates

We analyze the impact of due dates on LO, LBBD, and MH. We grouped the 480 instances with up to 100 jobs and 10 candidate locations and computed the average gap and computation time for each combination of TF and RDD. Different values of TF and RDD indicate different tightness and ranges of due dates. The results are reported in Table 9, which shows average values over 40 instances with a specific combination of TF and RDD.

Table 9 shows that all the three solution approaches can obtain relatively good gaps when due dates are loose ($TF = 0.2$). LBBD significantly outperforms the other approaches for this group of instances. It obtains optimal solutions for more than 80% of these instances (131 out of 160), generates an average gap of 0.47%, and consumes 197.72s. LBBD performs especially well for instances with large values of RDD when $TF = 0.2$. As the value of TF increases from 0.2 to 0.6, due dates become tighter. We observe that the relative gaps increase in this case, which is true for all three methods. For example, the largest gap of LBBD increases from 1.58% for $TF = 0.2$ to 24.94% for $TF = 0.6$. In terms of the impact of RDD, the average gaps obtained by LO and LBBD decrease with the increase of RDD, indicating that a wider range of due dates leads to a simpler problem, except for $RDD = 0.8$. For instance, for $TF = 0.4$, the average gap obtained by LO decreases from 19.24% to 11.55% as the value of RDD increases from 0.2 to 0.8. The reason might be that when RDD is small, due dates are highly intensive; there may be many similar potential solutions for the MILP to determine, which increases its computational complexity. We can obtain the same trends from the results of the computation time.

5.9. Sensitivity analysis

In this section, we conduct a sensitivity analysis to better study the impacts of the three cost components' weights on the computational results. We choose 12 instances with $n = \{10, 20, 30, 40\}$, $l = 8$, $m = 4$, and $(TF, RDD) = \{(0.2, 0.2), (0.4, 0.4), (0.6, 0.6)\}$. For each run, we choose one of the three weights $q_i, (i = 1, 2, 3)$ and set $q_i = 2, 3, 5,$

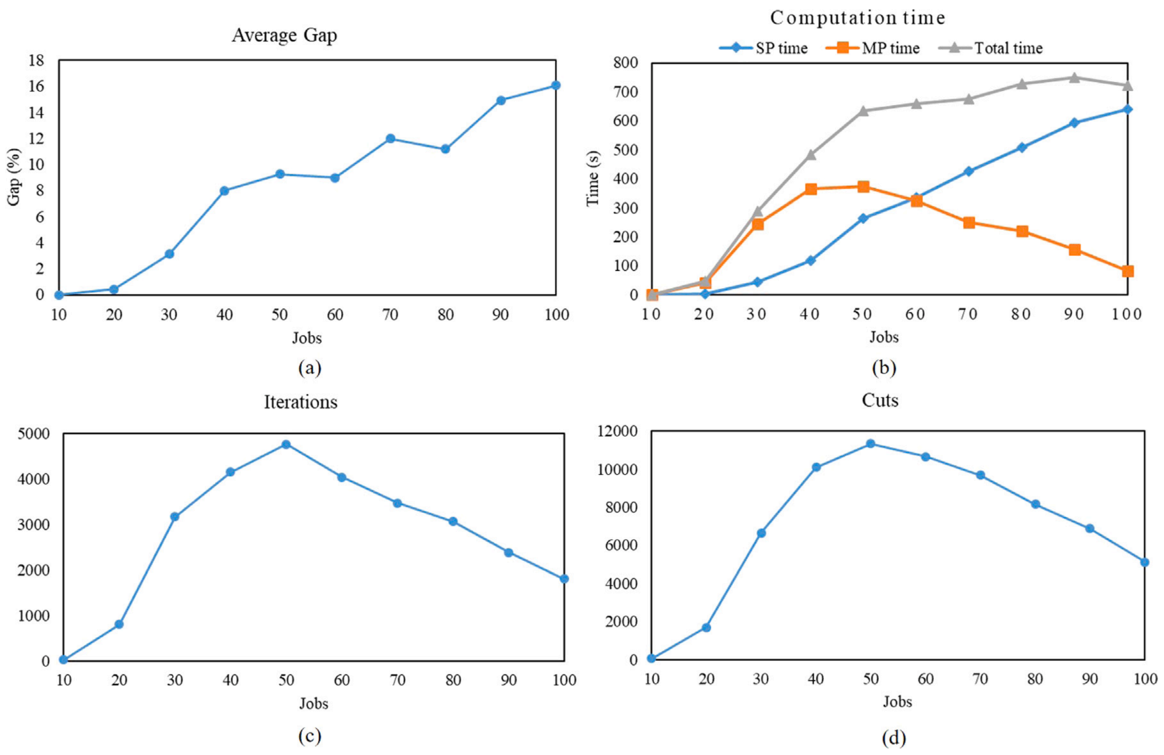


Fig. 4. Details of the LBBD.

Table 9
Impact of due dates.

Gap (%)												
TF	0.2				0.4				0.6			
RDD	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
LO	10.03$^{(4)}$	7.31$^{(6)}$	3.90$^{(4)}$	4.41$^{(5)}$	19.24$^{(6)}$	15.53$^{(7)}$	15.40$^{(6)}$	11.55$^{(4)}$	23.69$^{(5)}$	23.45$^{(6)}$	33.32$^{(4)}$	22.15$^{(3)}$
LBBD	1.58$^{(21)}$	0.23$^{(32)}$	0.04$^{(39)}$	0.01$^{(39)}$	9.77$^{(12)}$	7.12$^{(14)}$	5.45$^{(18)}$	3.40$^{(13)}$	15.41$^{(9)}$	16.13$^{(12)}$	24.94$^{(6)}$	16.16$^{(8)}$
MH	1.90$^{(8)}$	1.16$^{(6)}$	1.55$^{(10)}$	3.74$^{(3)}$	9.26$^{(8)}$	7.24$^{(11)}$	7.26$^{(9)}$	5.47$^{(6)}$	15.09$^{(9)}$	15.89$^{(10)}$	24.38$^{(5)}$	15.99$^{(7)}$

Time (s)												
TF	0.2				0.4				0.6			
RDD	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
LO	810.47	768.96	810.64	808.88	766.05	748.68	769.26	810.16	794.57	766.95	810.16	834.06
LBBD	449.40	248.67	42.87	49.92	656.56	594.10	511.01	619.29	700.93	633.04	767.33	720.21
MH	233.35	268.76	233.29	303.71	447.13	402.39	453.32	453.99	557.67	559.28	668.99	636.25

respectively, while the other two weights remain unchanged. To better understand the impacts on the solutions, we output the number of optimal solutions obtained, the value of the fixed location cost, the transportation cost, and the total tardiness penalty cost, and denote them as *OPT*, *Location*, *Transportation*, *Tardiness*, respectively. LBBD solves the chosen instances, and the results are reported and analyzed.

5.9.1. Impact of fixed location cost

From Table 10 we can observe that with the increase of q_1, the fixed location cost plays a more critical role in the objective. Therefore, the value of *Location* decreases from 423.33 to 290.00 as the value of q_1 increases from one to five. From the values of *OPT* and Gap, we can find that when $q_1 \leq 3$, the solution quality decreases slightly. However, when q_1 increases from three to five, the number of optimal solutions decreases from nine to six. The average gaps grow from 2.63% to 4.49%, indicating that the problem becomes more challenging to solve when the weight of the fixed location cost gets higher.

5.9.2. Impact of transportation cost

The impact of transportation cost is reported in Table 11. From the table, we can find that the value of *Transportation* decreases when the weight q_2 gets higher. At the same time, the solution quality becomes better. As q_2 increases from one to five, the number of optimal solutions increases from 10 to 12 and the average gap drops from 2.67% to 0.00%. The average computation time decreases significantly from 174.28 s to 3.49 s. The reason might be as follows: since *Transportation* takes a large proportion of the objective, its change greatly influences the decision-making of the problem. When q_2 becomes exceptionally high, jobs will be assigned to locations as close as possible, making the problem less complicated and thus easier to be solved.

5.9.3. Impact of tardiness penalty cost

From Table 12 we can find that the tardiness penalty cost has a significant impact on the computational complexity of solving the instances. When $q_3 = 1$, 10 of the 12 instances are optimally solved, the average gap is 2.67%, and the average computation time is 174.28s. However, when $q_3 = 5$, only six instances are optimally solved. The

Table 10
Impact of fixed location cost.

q_1	Obj	LB	*OPT*	Gap (%)	Time (s)	*Location*	*Transportation*	*Tardiness*
1	1781.633	1701.456	10	2.67	174.28	423.33	1087.50	959.92
2	2190.617	2098.725	10	2.57	170.11	382.50	1136.17	1008.75
3	2560.008	2452.326	9	2.63	324.34	355.83	1177.00	1146.75
5	3210.792	2993.983	6	4.49	468.07	290.00	1344.67	1689.17

Table 11
Impact of transportation cost.

q_2	Obj	LB	*OPT*	Gap (%)	Time (s)	*Location*	*Transportation*	*Tardiness*
1	1781.633	1701.456	10	2.67	174.28	423.33	1087.50	959.92
2	2863.85	2817.059	10	1.05	150.98	425.83	1077.33	1020.00
3	3941.933	3933.567	11	0.15	103.56	425.83	1074.83	1043.75
5	6087.767	6087.767	12	0.00	3.49	434.17	1069.17	1087.33

Table 12
Impact of tardiness penalty cost.

q_3	Obj	LB	*OPT*	Gap (%)	Time (s)	*Location*	*Transportation*	*Tardiness*
1	1781.633	1701.456	10	2.67	174.28	423.33	1087.50	959.92
2	2051.6	1838.944	7	5.95	425.60	425.00	1103.33	902.42
3	2308.433	1951.323	6	8.79	450.37	425.00	1105.83	898.50
5	2821.375	2184.778	6	12.29	452.51	425.00	1115.83	889.33

average gap and computation time are 12.29% and 452.51s, respectively, indicating that the solution quality worsens. However, we can notice that the value of *Tardiness* gets minor changes when q_3 gets larger. The reason might be that, in our LBBD algorithm, the total tardiness penalty cost is determined by the SP, which has been proved to be NP-hard. Therefore, the objective is sensitive to the weight q_3. Its increase makes the problem more complicated, consumes much more computation time, and thus leads to poor solutions with large objective values.

5.10. Sequential versus integrated planning

We also compare the results obtained by the proposed integrated methods to a sequential approach for solving the DPSL-DD problem. We first present a sequential heuristic (SH) method for the studied DPSL-DD problem, which solves the problem by sequentially solving location and scheduling SPs. In the first stage of SH, a MILP model corresponding to a location problem is proposed to minimize the fixed location and transportation costs. In the second stage, a parallel-machine scheduling problem with release dates is solved by adapting the dispatching rule proposed by Lin Yang-Kuei (2013). Next, we describe the two stages of SH.

5.10.1. First stage: location subproblem

The first stage of SH determines the location of machines by solving the following MILP model:

$$\min \quad q_1 \sum_{k\in K} c_k w_k + 2q_2 \sum_{j\in J}\sum_{k\in K} e_{jk} v_{jk} \tag{78}$$

s.t.

(2), (11), (13), (14), (20), (21).

We aim to minimize the objective function (78), i.e., the sum of the fixed location cost and transportation cost. After the location SP is solved, the location variable w_k is determined and fixed. We have a parallel-machine scheduling problem with release dates to minimize the weighted sum of transportation cost and tardiness penalty cost.

5.10.2. Second stage: scheduling subproblem

In the second stage, as the location variable w_k is fixed, we need to solve a parallel-machine scheduling problem with release dates to minimize the weighted sum of transportation cost and total tardiness penalty cost. We adapt the dispatching rule ATCR proposed by Lin Yang-Kuei (2013) to solve the second-stage problem.

Let U be the set of unscheduled jobs and t_k be the sum of the processing times of the jobs assigned to location $k, k \in M$. A machine k^* is randomly selected from set M, and jobs are ordered by the priority parameter,

$$I_{jk^*} = \frac{E_j}{p_j} \exp\left(-\frac{\max(d_j - p_j - \max(r_{jk^*}, t^*_k), 0)}{k_1 \bar{p}}\right) \times \exp\left(-\frac{\max(r_{jk^*} - t^*_k, 0)}{k_3 \bar{p}}\right), \tag{79}$$

where E_j is the weight of each job, $\bar{p}$ is the average processing time of the remaining unscheduled jobs, and k_1 and k_3 are scaling parameters determined empirically. We consider the following values of k_1 and k_3:

$$k_1 = \{0.2, 0.8, 1.2, 1.6, 2.0, 2.8, 3.6, 4.4, 5.2, 6.0\}$$
$$k_3 = \{0.001, 0.0025, 0.004, 0.005, 0.025, 0.04, 0.05, 0.25, 0.4, 0.6, 0.8, 1.0, 1.2\}.$$

We select the job j^* with the largest value of $I_{j^*k^*}$. Then a machine k^{**} is selected to minimize the weighted sum of transportation cost and tardiness of job j^*, i.e., $2q_2 e_{j^*k^{**}} + q_3\theta(C_{j^*k^{**}} + r_{j^*k^{**}} - d_{j^*})$. Therefore, job j^* is assigned to machine k^{**} at the last position of the current processing sequence. Then U and t_k are updated. This process iterates until all jobs are scheduled, i.e., $U = \emptyset$. Once a feasible schedule is formed, an adjacent pairwise interchange procedure is performed to improve the solution further.

5.10.3. Results and discussion

We report the results obtained by SH and compare the results to those of the integrated methods LO, LBBD, and MH. In the implementation of SH, parameters are set to $E_j = 1$ ($\forall j \in J$), $k_1 = 0.2$, and $k_3 = 0.004$ based on preliminary experiments. Results are presented in Table 13. We use the best lower bound provided by all methods to compute the gap between the solutions provided by MH and SH.

We compare the results obtained by SH with those of LO, LBBD, and MH. From Table 13 we can see that no instance is solved to optimality by SH, and it generates an average gap of 15.75% over all instances, which is slightly lower than that of LO (15.83%). This is primarily due to the poor performance of LO on large-sized instances, as we can see that the LO performs much better on small-sized instances in both the number of optimal solutions and the average gap. LBBD

Table 13
Sequential versus integrated planning.

Instance size	n	Gap (%)				Time (s)			
		LO$^{(opt)}$	LBBD$^{(opt)}$	MH$^{(opt)}$	SH$^{(opt)}$	LO	LBBD	MH	SH
Small-sized	10	0.11$^{(47)}$	0.00$^{(48)}$	0.21$^{(38)}$	6.00$^{(0)}$	23.76	0.15	0.25	0.05
	20	5.37$^{(12)}$	0.37$^{(46)}$	0.34$^{(29)}$	7.36$^{(0)}$	693.64	45.96	2.38	0.08
Medium-sized	30	11.09$^{(0)}$	3.11$^{(34)}$	3.13$^{(12)}$	10.49$^{(0)}$	900.02	289.30	108.08	0.09
	40	16.49$^{(0)}$	7.92$^{(23)}$	8.38$^{(7)}$	16.33$^{(0)}$	900.03	483.32	335.74	0.12
	50	18.21$^{(0)}$	9.26$^{(16)}$	10.07$^{(3)}$	17.78$^{(0)}$	900.04	637.53	528.25	0.15
Large-sized	60	16.98$^{(1)}$	8.95$^{(14)}$	10.34$^{(1)}$	16.07$^{(0)}$	897.68	659.52	525.02	0.19
	70	20.22$^{(0)}$	11.93$^{(12)}$	13.63$^{(1)}$	19.68$^{(0)}$	900.08	676.84	663.47	0.23
	80	19.39$^{(0)}$	11.11$^{(11)}$	12.21$^{(1)}$	17.99$^{(0)}$	900.11	728.56	723.44	0.25
	90	23.52$^{(0)}$	14.94$^{(9)}$	15.36$^{(0)}$	22.35$^{(0)}$	900.14	749.02	663.38	0.28
	100	26.96$^{(0)}$	15.94$^{(10)}$	17.11$^{(0)}$	23.44$^{(0)}$	900.18	724.22	798.43	0.29
Average		15.83$^{(60)}$	8.35$^{(223)}$	9.08$^{(92)}$	15.75$^{(0)}$	791.57	499.44	434.84	0.17

obtains significantly smaller gaps than SH and provides many more optimal solutions. MH outperforms SH by providing smaller gaps for every set of instances and solving more instances to optimality. These results demonstrate the benefits of the proposed integrated approaches, including the exact LBBD method and the tailored MH. SH is generally fast but provides inferior solutions, while integrated approaches take more time but provide significantly better solutions.

In summary, results for small-sized instances demonstrate the benefits of the proposed integrated approaches. Results on large-sized instances show the consistently good performance of LBBD and MH, which both outperform the sequential method SH.

6. Conclusions and discussion

We investigated the parallel-machine ScheLoc problem with delivery time and due date. A novel feature of the problem is that jobs must be delivered back to their original locations upon completion of processing. The objective is to minimize the weighted sum of the fixed location cost, job transportation cost, and total tardiness penalty cost. To this end, we proposed three MILP formulations adapted from similar scheduling problems for the studied problem. We also developed an LBBD method that decomposes the problem into an MP and several SP, which are solved iteratively, and cuts are dynamically added to MP upon finding a feasible solution. To handle large-sized instances, we developed a matheuristic, which solves an approximate MILP model with predetermined job processing sequences on each machine. Considering the difficulty of extra-large sized instances, we further designed a variant MH-KS that combined MH with a kernel search framework. We conducted extensive computational experiments on 720 randomly generated instances with up to 300 jobs, 100 locations, and 50 machines. Results showed that the proposed models addressed small-sized instances and drastically lost efficiency when the instance size increased. The linear ordering model outperformed the other two models. The proposed LBBD method consistently outperformed the proposed models by providing significantly more optimal solutions and smaller gaps in less computation time. The developed MH provided similar gaps compared to LBBD and outperformed the MILP models. MH-KS solved the instances with the highest efficiency and outperformed all other methods on extra-large sized instances, posing a good trade-off between solution quality and computational efficiency. The LBBD algorithm was analyzed in detail, including the average gaps, the computation time, the number of iterations, and the number of Benders cuts. We observed that the average gap and the computation time increased for larger instances. When the instances were large enough ($n \geq 60$), the time consumed on SP was considerably longer, and it forced the number of iterations and Benders cuts to decrease. We analyzed the impacts of the due dates on the performance of different methods. Results indicate that instances with tighter and more intensive due dates are generally more challenging. A sensitivity analysis was conducted to study the cost components' impacts on the computational results. The problem became more complicated when the weights of fixed location cost and tardiness penalty cost increased. At the same time, it was easier to solve as the weight of the transportation cost grew.

We further showed the advantages of our methods by applying the integrated approaches over an adapted sequential method. We visualized the solutions obtained by different methods on an illustrative example detailed in Appendix. Results showed that an optimal solution shows a balance among all cost components. A better single cost component, e.g., fixed location cost, transportation cost, or tardiness penalty cost, does not indicate an overall optimal solution.

The scheduling and location problem is an interesting and complex combinatorial optimization problem that remains to be solved. Machine-dependent processing time is worth considering to make the solutions more applicable to heterogeneous machines in manufacturing industries. Disruption risk at the candidate locations is worth considering because this risk may vary by machine location.

CRediT authorship contribution statement

Chuang Zhang: Writing – original draft, Methodology, Software, Formal analysis, Validation, Investigation, Data curation. **Yantong Li:** Conceptualization, Methodology, Software, Writing – review & editing, Visualization, Supervision, Project administration. **Junhai Cao:** Resources, Supervision, Project administration. **Zhen Yang:** Writing – review & editing. **Leandro C. Coelho:** Writing – review & editing, Visualization.

Acknowledgments

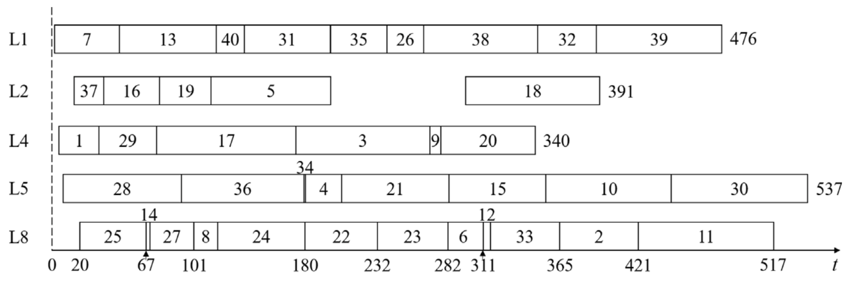
We thank an associate editor and four anonymous referees for their valuable suggestions on an earlier version of this paper.

This research is supported by the National Key R&D Program (#2019YFB1703800), the Fundamental Research Funds for the Central Universities of China (3132022267), and the China Postdoctoral Science Fund (2022M710018).

Appendix. An illustrative example

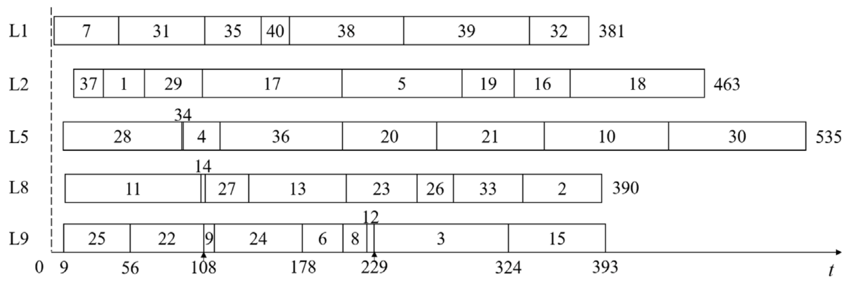
To further examine the solutions provided by different methods, we visualize the solutions obtained by LO, LBBD, MH, and SH using an illustrative example. We choose a medium-sized instance with 40 jobs, 10 locations, and five machines, $TF = 0.2$, $RDD = 0.4$, $q_1 = q_2 = q_3 = 1$, and $\theta = 0.2$. The data are given in Table A.14. The release date r_{jk} and transportation cost e_{jk} are calculated using the Euclidean distance D_{jk}, and we set $u_{jk} = f_{jk} = 1$ ($r_{jk} = e_{jk} = D_{jk}$).

Figs. 5, 6, 7, and 8 represent the solutions obtained by LO, LBBD, MH, and SH, respectively. In each figure, we visualize the selected locations and the corresponding schedule at each location and give the values of each cost component, i.e., the location cost, transportation



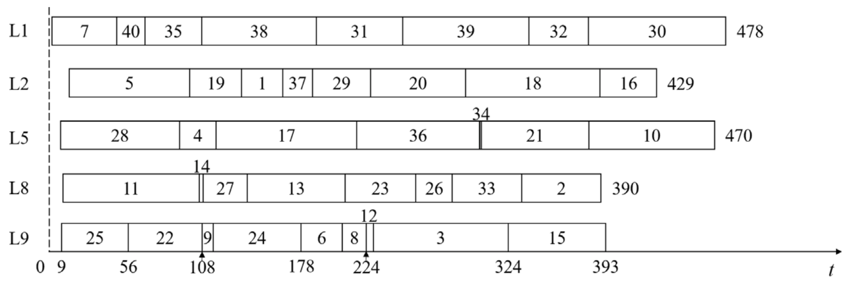
Fixed cost = 740 , Transportation cost = 1710, Total tardiness = 652 , Objective = 2580.4, Status = 6.99%, Time = 900.03s

Fig. 5. Illustrative solution by LO.



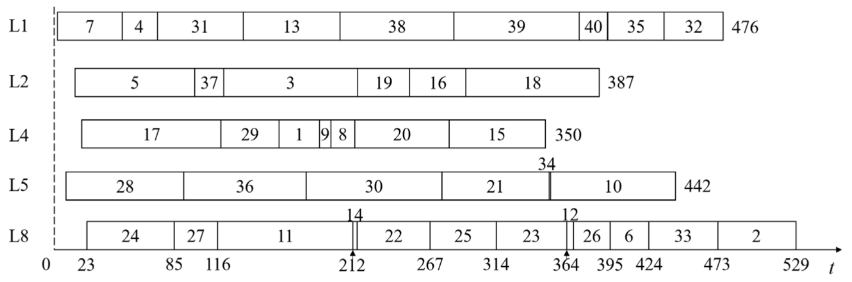
Fixed cost = 780, Transportation cost = 1702, Total tardiness = 385, Objective = 2559, Status = Optimal, Time = 1.86s

Fig. 6. Illustrative solution by LBBD.



Fixed cost =780 , Transportation cost = 1702, Total tardiness = 405, Objective = 2563, Status = 0.16%, Time = 12.00s

Fig. 7. Illustrative solution by MH.



Fixed cost =740 , Transportation cost = 1762, Total tardiness =1012 , Objective =2704.4, Status = 5.38%, Time = 0.23s

Fig. 8. Illustrative solution by SH.

cost, tardiness penalty cost, and the objective value and corresponding computation time. Note that the objective value is computed by summing the products of the cost components and their corresponding weights.

From Figs. 5–8, we observe that all the approaches give solutions without machine idle time, which means that the solutions are generally acceptable in terms of machine utilization. Among the approaches, LBBD obtains the optimal solution in 1.56 s. MH is the second-best model, with a gap as low as 0.16% and computation time of 12.00 s. SH provides a feasible solution in a very short time, with a gap of 5.38%. The integrated LO model gives a gap of 6.99%, higher than that of SH. However, the UB, i.e., the objective value of LO is smaller than SH.

The objective value of LO is 2580.4, which is only 21.4 larger than the optimal solution obtained by LBBD. However, its gap is 6.99%,

Table A.14
Parameters for illustrative example.

Jobs	p_j	d_j	Release date r_{jk}									
			1	2	3	4	5	6	7	8	9	10
1	29	294	94	27	59	5	45	103	53	61	47	46
2	56	370	77	59	62	65	64	80	85	9	23	38
3	95	372	62	46	36	41	34	69	54	25	12	38
4	26	255	29	79	5	65	20	38	31	63	52	78
5	85	297	104	14	74	32	65	110	81	39	33	11
6	29	342	49	70	39	67	46	53	66	30	30	56
7	46	264	2	107	35	93	49	10	56	80	74	101
8	17	343	65	42	37	37	34	72	53	27	12	35
9	8	322	72	34	41	27	34	80	52	33	17	32
10	89	410	54	56	18	40	7	62	26	53	39	59
11	96	338	73	60	58	64	61	76	81	10	21	39
12	5	364	47	70	37	66	44	52	64	31	29	56
13	69	330	45	89	49	86	60	45	78	45	48	73
14	3	289	63	78	58	79	65	65	85	28	37	58
15	69	397	66	40	34	31	27	73	46	36	20	39
16	40	418	114	12	83	34	73	121	87	50	44	21
17	99	259	78	41	42	20	28	86	34	61	45	54
18	95	404	106	13	76	32	67	112	82	41	35	12
19	37	366	108	18	79	38	71	114	87	39	36	10
20	67	359	85	41	50	19	35	94	38	67	51	57
21	76	366	83	71	52	49	39	92	26	91	75	86
22	52	316	52	63	36	59	41	57	62	26	22	50
23	50	323	67	81	63	83	70	68	90	31	41	60
24	62	289	56	59	38	56	42	61	62	23	19	46
25	47	316	64	48	40	46	40	70	60	20	9	36
26	26	349	51	96	59	94	70	51	88	49	55	79
27	31	269	68	97	72	99	82	68	101	47	57	76
28	84	262	58	57	23	38	8	67	21	60	45	63
29	41	258	86	28	51	7	38	95	47	56	41	44
30	97	364	30	76	10	63	23	38	39	55	45	72
31	61	308	18	96	33	86	48	20	61	63	59	87
32	42	387	15	93	20	79	35	24	43	71	63	89
33	49	372	83	70	72	77	76	85	96	21	36	47
34	1	348	67	52	32	32	17	76	23	63	47	62
35	40	342	16	98	34	88	49	18	61	65	61	89
36	87	265	38	87	20	69	24	47	20	80	67	90
37	21	257	96	16	67	29	59	103	75	34	26	11
38	81	356	20	93	31	83	45	23	59	59	55	83
39	89	364	16	101	38	92	53	16	65	67	64	92
40	20	341	12	118	45	104	59	11	63	91	85	112
Fixed cost c_k			120	100	210	160	170	110	110	190	200	140

indicating that LO yields a bad lower bound. The fixed location cost by LO is 740, which is less than LBBD. However, the overall objective value of LO is worse than that of LBBD, which indicates that a lower location cost does not mean a better overall solution. From Figs. 6 and 7, we can find that LBBD and MH select the same locations to deploy machines, which indicates that the MILP model used in MH is a good approximation of the original problem. However, MH has greater tardiness than LBBD, perhaps because the constraints based on the ordered lists J_k affect MH to optimize job assignments. In this problem, the due dates are generally loose ($TF = 0.2$), and the tardiness penalty coefficient is small ($\theta = 0.2$). As a result, the tardiness penalty cost may not be high, and the objective value is mainly determined by the fixed location cost and transportation cost. Through the statistical analysis, we find that 27 of 40 jobs are assigned to the nearest location in the solution of both LBBD and MH versus 22 and 23 for LO and SH, respectively. That can partly explain why the solutions of SH and LO have higher transportation costs than LBBD and LO. To conclude, the cost components in the solutions obtained by different methods may differ greatly. LBBD provides the globally optimal solution with the minimum total cost.

References

Adamopoulos, G.I., Pappis, C.P., 1998. Scheduling under a common due-data on parallel unrelated machines. European J. Oper. Res. 105 (3), 494–501.

Akbarinasaji, S., Mckendall, A.R., 2017. Heuristics for the integrated single machine scheduling and location problem. Int. J. Ind. Syst. Eng. 27 (2), 196–209.

Akker, V.D., J., M., Hurkens, C.A.J., Savelsbergh, M.W.P., 2000. Time-indexed formulations for machine scheduling problems: Column generation. INFORMS J. Comput. 12 (2), 111–124.

Amorim, R., Thomaz, M., de Freitas, R., 2017. Solving large instances applying metaheuristics for classical parallel machine scheduling problems under tardiness and earliness penalties. In: 2017 XLIII Latin American Computer Conference. CLEI, pp. 1–10.

Angelelli, E., Mansini, R., Grazia Speranza, M., 2010. Kernel search: A general heuristic for the multi-dimensional knapsack problem. Comput. Oper. Res. 37 (11), 2017–2026.

Avci, M., Yildiz, S.T., 2019. A matheuristic solution approach for the production routing problem with visit spacing policy. European J. Oper. Res. 279 (2), 572–588.

Avella, P., Boccia, M., D'Auria, B., 2005. Near-optimal solutions of large-scale single-machine scheduling problems. INFORMS J. Comput. 17 (2), 183–191.

Baker, K.R., Trietsch, D., 2013. Principles of Sequencing and Scheduling. John Wiley & Sons, Hoboken, New Jersey.

Behnamian, J., Ghomi, S.M.T.F., 2016. A survey of multi-factory scheduling. J. Intell. Manuf. 27 (1), 231–249.

Benders, J.F., 1962. Partitioning procedures for solving mixed-variables programming problems. Numer. Math. 4, 238–252.

Bertazzi, L., Coelho, L.C., De Maio, A., Laganà, D., 2019. A matheuristic algorithm for the multi-depot inventory routing problem. Transp. Res. E 122, 524–544.

Chan, L.M.A., Simchi-Levi, M.D., 1998. Parallel machine scheduling, linear programming, and parameter list scheduling heuristics. Oper. Res. 46 (5), 729–741.

Chen, Z.L., Pundoor, G., 2006. Order assignment and scheduling in a supply chain. Oper. Res. 54 (3), 555–572.

Chenery, H.B., Watanabe, T., 1958. International comparisons of the structure of production. Econometrica 26 (4), 487–521.

Ching-Fang Liaw, C.-Y.C., Chen, M., 2003. Scheduling unrelated parallel machines to minimize total weighted tardiness. Comput. Oper. Res. 30 (12), 1777–1789.

Dang, Q.V., Diessen, T.V., Martagan, T., Adan, I., 2021. A matheuristic for parallel machine scheduling with tool replacements. European J. Oper. Res. 291 (2), 640–660.

Dkhil, H., Yassine, A., Chabchoub, H., 2017. Multi-objective optimization of the integrated problem of location assignment and straddle carrier scheduling in maritime container terminal at import. J. Oper. Res. Soc. 69 (2), 247–269.

Dyer, M.E., Wolsey, L.A., 1990. Formulating the single machine sequencing problem with release dates as a mixed integer program. Discrete Appl. Math. 26 (3), 255–270.

Elvikis, D., Hamacher, H.W., Kalsch, M.T., 2009. Simultaneous scheduling and location (ScheLoc): the planar ScheLoc makespan problem. J. Sched. 12 (4), 361–374.

Emde, S., Polten, L., Gendreau, M., 2019. Logic-based benders decomposition for scheduling a batching machine. Comput. Oper. Res. 113, 104777.

Fachini, R.F., Armentano, V.A., 2020. Logic-based benders decomposition for the heterogeneous fixed fleet vehicle routing problem with time windows. Comput. Ind. Eng. 48, 106641.

Fang, K.T., Lin, B.M.T., 2013. Parallel-machine scheduling to minimize tardiness penalty and power cost. Comput. Ind. Eng. 64 (1), 224–234.

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

Filippi, C., Guastaroba, G., Huerta-Muñoz, D., Speranza, M., 2021. A kernel search heuristic for a fair facility location problem. Comput. Oper. Res. 132, 105292.

Fischetti, M., Fischetti, M., 2016. Matheuristics. In: Martí, R., Panos, P., Resende, M.G. (Eds.), Handbook of Heuristics. Springer, Cham, Switzerland, pp. 1–33.

Garmdare, H.S., Lotfi, M.M., Honarvar, M., 2017. Integrated model for pricing, delivery time setting, and scheduling in make-to-order environments. J. Ind. Eng. Int. 14 (1), 1–10.

Gobbi, A., Manerba, D., Mansini, R., Zanotti, R., 2019. A kernel search for a patient satisfaction-oriented nurse routing problem with time-windows. IFAC-PapersOnLine 52 (13), 1669–1674.

Grötschel, M., Jűnger, M., Reinelt, G., 1984. A cutting plane algorithm for the linear ordering problem. Oper. Res. 32 (6), 1195–1220.

Guastaroba, G., Savelsbergh, M., Speranza, M., 2017. Adaptive kernel search: A heuristic for solving mixed integer linear programs. European J. Oper. Res. 263 (3), 789–804.

Guastaroba, G., Speranza, M., 2014. A heuristic for BILP problems: The single source capacitated facility location problem. European J. Oper. Res. 238 (2), 438–450.

Hennes, H., Hamacher, H.W., 2002. Integrated Scheduling and Location Models: Single Machine Makespan Problems. Technical Report, University of Kaiserslautern, Shaker Verlag, Aachen.

Heßler, C., Deghdak, K., 2017. Discrete parallel machine makespan ScheLoc problem. J. Comb. Optim. 34, 1159–1186.

Hooker, J.N., Ottosson, G., 2003. Logic-based benders decomposition. Math. Program. 96 (1), 33–60.

Ibaraki, T., Nakamura, Y., 1994. A dynamic programming method for single machine scheduling. European J. Oper. Res. 76 (1), 72–82.

Jin, X., Li, K., Sivakumar, A.I., 2013. Scheduling and optimal delivery time quotation for customers with time sensitive demand. Int. J. Prod. Econ. 145 (1), 349–358.

Johnson, D.S., 2002. A theoretician's guide to the experimental analysis of algorithms. In: Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges. pp. 215–250.

Kalsch, M.T., 2009. Scheduling-Location (ScheLoc) Models, Theory and Algorithms. (Ph.D. thesis). University of Kaiserslautern.

Kalsch, M.T., Drezner, Z., 2010. Solving scheduling and location problems in the plane simultaneously. Comput. Oper. Res. 37 (2), 256–264.

Korupolu, M., Plaxton, C.G., Rajaraman, R., 1998. Analysis of a local search heuristic for facility location problems. J. Algorithms 37 (1), 146–188.

Kramer, A., Dell'Amico, M., Feillet, D., Iori, M., 2020. Scheduling jobs with release dates on identical parallel machines by minimizing the total weighted completion time. Comput. Oper. Res. 123, 105018.

Kramer, R., Kramer, A., 2021. An exact framework for the discrete parallel machine scheduling location problem. Comput. Oper. Res. 132, 105318.

Krumke, S.O., Le, H.M., 2020. Robust absolute single machine makespan scheduling-location problem on trees. Oper. Res. Lett. 48 (1), 29–32.

Lahyani, R., Chebil, K., Khemakhem, M., Coelho, L.C., 2019. Matheuristics for solving the multiple knapsack problem with setup. Comput. Ind. Eng. 129, 76–89.

Lawler, E.L., 1977. A ‘‘Pseudopolynomial’’ algorithm for sequencing jobs to minimize total tardiness. Ann. Discrete Math. 1 (08), 331–342.

Ławrynowicz, M., Filcek, G., 2020. A comparison of evolutionary and simulated annealing algorithms for bi-criteria location-scheduling problem. In: 2020 15th Conference on Computer Science and Information Systems. FedCSIS, pp. 251–255.

Ławrynowicz, M., Jozefczyk, J., 2019. A memetic algorithm for the discrete scheduling-location problem with unrelated machines. In: 2019 24th International Conference on Methods and Models in Automation and Robotics. MMAR, pp. 158–163.

Lee, K., Choi, B.-C., Leung, J., Pinedo, M., Briskorn, D., 2012. Minimizing the total weighted delivery time in container transportation scheduling. Nav. Res. Logist. 59 (3–4), 266–277.

Li, Y., Côté, J.-F., Callegari-Coelho, L., Wu, P., 2022a. Novel formulations and logic-based Benders decomposition for the integrated parallel machine scheduling and location problem. INFORMS J. Comput. 34 (2), 1048–1069.

Li, Y., Côté, J.-F., Coelho, L.C., Wu, P., 2021. Novel efficient formulation and matheuristic for large-sized unrelated parallel machine scheduling with release dates. Int. J. Prod. Res. 1–20.

Li, Y., Li, Y., Cheng, J., Wu, P., 2022b. Order assignment and scheduling for personal protective equipment production during the outbreak of epidemics. IEEE Trans. Autom. Sci. Eng. 19 (2), 692–708.

Lin Yang-Kuei, L.C.-W., 2013. Dispatching rules for unrelated parallel machine scheduling with release dates. Int. J. Adv. Manuf. Technol. 67, 269–279.

Liu, M., Liu, X., 2019a. Distributionally robust parallel machine ScheLoc problem under service level constraints. IFAC-PapersOnLine 52 (13), 875–880.

Liu, M., Liu, R., 2019b. Risk-averse scheduling-location (ScheLoc) problem. In: 2019 International Conference on Industrial Engineering and Systems Management. IESM, pp. 1–6.

Liu, M., Liu, X., Zhang, E., Chu, F., Chu, C., 2019. Scenario-based heuristic to two-stage stochastic program for the parallel machine ScheLoc problem. Int. J. Prod. Res. 57 (6), 1706–1723.

Maniezzo, V., Stützle, T., Voss, S., 2010. Matheuristics – Hybridizing Metaheuristics and Mathematical Programming. Springer, US.

Mönch, L., Shen, L., 2021. Parallel machine scheduling with the total weighted delivery time performance measure in distributed manufacturing. Comput. Oper. Res. 127, 105126.

Musavi, M.M., Bozorgi-Amiri, A., 2017. A multi-objective sustainable hub location-scheduling problem for perishable food supply chain. Comput. Ind. Eng. 113, 766–778.

Naderi, B., Roshanaei, V., 2020. Branch-Relax-and-Check: A tractable decomposition method for order acceptance and identical parallel machine scheduling. European J. Oper. Res. 286 (3), 811–827.

Nikzad, E., Bashiri, M., Abbasi, B., 2021. A matheuristic algorithm for stochastic home health care planning. European J. Oper. Res. 288 (3), 753–774.

Pan, Y., Liang, Z., 2017. Dual relaxations of the time-indexed ILP formulation for min-sum scheduling problems. Ann. Oper. Res. 249 (1–2), 197–213.

Pinedo, M., Rammouz, E., 1988. A note on stochastic scheduling on a single machine subject to breakdown and repair. Probab. Eng. Inf. Sci. 2 (1), 41–49.

Pundoor, G., Chen, Z.L., 2005. Scheduling a production-distribution system to optimize the tradeoff between delivery tardiness and distribution cost. Nav. Res. Logist. 52 (6), 571–589.

Rajabzadeh, M., Ziaee, M., Bozorgi-Amiri, A., 2016. Integrated approach in solving parallel machine scheduling and location (ScheLoc) problem. Int. J. Ind. Eng. Comput. 7 (4), 573–584.

Ríos-Mercado, R.Z., Bard, J.F., 2003. The flow shop scheduling polyhedron with setup times. J. Comb. Optim. 7, 291–318.

Roshanaei, V., Luong, C., Aleman, D.M., Urbach, D.R., 2017. Collaborative operating room planning and scheduling. INFORMS J. Comput. 29 (3), 558–580.

Schenekemberg, C.M., Scarpin, C.T., Pécora, Jr., J.E., Guimarães, T.A., Coelho, L.C., 2020. The two-echelon inventory-routing problem with fleet management. Comput. Oper. Res. 121, 104944.

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.

Shim, S.O., Kim, Y.D., 2007. Scheduling on parallel identical machines to minimize total tardiness. European J. Oper. Res. 177 (1), 135–146.

Slotnick, S.A., Sobel, M.J., 2005. Manufacturing lead-time rules: Customer retention versus tardiness costs. European J. Oper. Res. 163 (3), 825–856.

Sousa, J.P., Wolsey, L.A., 1992. A time indexed formulation of non-preemptive single machine scheduling problems. Math. Program. 54 (1), 353–367.

Sun, X.T., Chung, S.H., Chan, F.T.S., 2015. Integrated scheduling of a multi-product multi-factory manufacturing system with maritime transport limits. Transp. Res. E 79, 110–127.

Sun, D., Tang, L., Baldacci, R., 2019. A benders decomposition-based framework for solving quay crane scheduling problems. European J. Oper. Res. 273 (2), 504–515.

Tanaka, S., Araki, M., 2013. An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times. Comput. Oper. Res. 40 (1), 344–352.

Tanaka, S., Fujikuma, S., 2012. A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time. J. Sched. 15 (3), 347–361.

Tanaka, S., Fujikuma, S., Araki, M., 2009. An exact algorithm for single-machine scheduling without machine idle time. J. Sched. 12 (6), 575–593.

Tanaka, S., Sato, S., 2013. An exact algorithm for the precedence-constrained single-machine scheduling problem. European J. Oper. Res. 229 (2), 345–352.

Tian, J., Fu, R., Yuan, J., 2007. On-line scheduling with delivery time on a single batch machine. Theoret. Comput. Sci. 374 (1–3), 49–57.

Wang, X., Li, Z., Chen, Q., Mao, N., 2020a. Meta-heuristics for unrelated parallel machines scheduling with random rework to minimize expected total weighted tardiness. Comput. Ind. Eng. 145, 106505.

Wang, Y., Wu, P., Cheng, J., 2020b. A decision model and method for the bi-objective parallel machine ScheLoc problem. In: 2020 7th International Conference on Control, Decision and Information Technologies, vol. 1. CoDIT, pp. 886–890.

Wang, S., Wu, R., Chu, F., Yu, J., Liu, X., 2020c. An improved formulation and efficient heuristics for the discrete parallel-machine makespan ScheLoc problem. Comput. Ind. Eng. 140, 106238.

Wesolkowski, S., Francetić, N., Grant, S.C., 2014. Trade: Training device selection via multi-objective optimization. In: 2014 IEEE Congress on Evolutionary Computation. CEC, pp. 2617–2624.

Yeh, W.-C., Lai, P.-J., Lee, W.-C., Chuang, M.-C., 2014. Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects. Inform. Sci. 269, 142–158.

Yıldırım, U.M., Çatay, B., 2014. A parallel matheuristic for solving the vehicle routing problems. In: de Sousa, J.F., Rossi, R. (Eds.), Computer-Based Modelling and Optimization in Transportation. Springer, Cham, Switzerland, pp. 477–489.

Zhang, Y., Lin, W.-H., Huang, M., Hu, X., 2021. Multi-warehouse package consolidation for split orders in online retailing. European J. Oper. Res. 289 (3), 1040–1055.