RAIRO-Oper. Res. 57 (2023) 1417–1441
https://doi.org/10.1051/ro/2023056

RAIRO Operations Research
www.rairo-ro.org

SEQUENCING SINGLE MACHINE MULTIPLE-CLASS CUSTOMER ORDER JOBS USING HEURISTICS AND IMPROVED SIMULATED ANNEALING ALGORITHMS

Win-Chin Lin[1], Xingong Zhang[2], Xinbo Liu[3], Kai-Xiang Hu[1], Shuenn-Ren Cheng[4], Ameni Azzouz[5] and Chin-Chia Wu[1,*]

Abstract. The multiple job class scheduling problem arises in contexts where a group of jobs belong to multiple classes and in which if all jobs in the same class are operated together, extra setup times would not be needed. On the other hand, the customer order scheduling problem focuses on finishing all jobs from the same order at the same time in order to reduce shipping costs. However, works on customer orders coupled with class setup times do not appear often in the literature. Hence we address here a bicriteria single machine customer order scheduling problem together with multiple job classes. The optimality criterion minimizes a linear combination of the sum of the ranges and sum of tardiness of all customer orders. In light of the high complexity of the concerned problem, we propose a lower bound formula and a property to be used in a branch-and-bound method for optimal solutions. To find approximate solutions, we then propose four heuristics together with a local search method, four cloudy theoretical simulated annealing and a cloudy theoretical simulated annealing hyperheuristic along with five low-level heuristics. The simulation results of the proposed heuristics and algorithms are analyzed.

Mathematics Subject Classification. 90B35, 68M20.

Received August 2, 2022. Accepted April 13, 2023.

1. Introduction

The sequencing (or scheduling) issue involving setup times or setup costs in many manufacturing and service environments is continually receiving more attention in the research community. Allahverdi and Soroush [5] pointed out that the efficacy of reducing setup times or costs contributes to reliable products and services being delivered to customers on time. For more details on applications involving setup times or costs, readers may refer to the five survey studies by Allahverdi *et al.* [6, 7], Allahverdi [3], Cheng *et al.* [15] and Yang and Liao [62].

Keywords. Scheduling, bicriteria, cloudy theoretical simulated annealing, hyperheuristic, multiple job classes.

[1] Department of Statistics, Feng Chia University, Taichung 40724, Taiwan.
[2] College of Mathematics Science, Chongqing Normal University, 401331 Chongqing, P.R. China.
[3] SolBridge International School of Business, Woosong University, Daejeon 34613, Republic of Korea.
[4] Department of E-sport Technology Management, Cheng Shiu University, Kaohsiung City 83347, Taiwan.
[5] Université de Tunis, Institut supérieur de gestion de Tunis, SMART Lab., Tunis, Tunisia.
*Corresponding author: cchwu@fcu.edu.tw

Setup times/costs in sequencing models have received wide discussion in the past thirty years because they play a key role in the on-time delivery of customer orders and have crucial impacts on cycle times [30,63,64]. In the multiple job class case, the jobs are divided into several classes, and a setup time is needed when a machine switches from one class of jobs to another on account of having to tune the production equipment or change tools. On the other hand, producers process a set of jobs that might come from different orders of distinct clients, and each client's order will contain multiple classes. Our research stems from an actual application involving aluminum extrusion plants. In such a manufacturing environment, if the machine has been set to process a specific set of jobs, it is desirable to run all the jobs of the same class since this will reduce the total setup time to a great extent. In particular, the measurement criterion of holding cost for an order is defined as the difference between the completion times of the first job and the last job of the order. The producer sequences all the jobs of the same order as close as possible to minimize the holding cost (or waiting times) and to reduce the shipping cost [12].

Regarding the optimization of a single objective relating to the setup time studies on a single-machine setting, readers may refer to Psaraftis [46], Monma and Potts [40], Potts and van Wassenhove [45], and Liaee and Emmons [33]. For more extensions and improvements such as optimizing the maximum completion time, the makespan, and the mean flow time, we refer readers to van der Veen and Zhang [54], Ahn and Hyun [1], Gupta [24, 25], Mason and Anderson [39], and Potts [44]. For more literature references on family sequence independent setup times, readers can refer to a comprehensive survey by Allahverdi [3] and to three recent papers by Janiak *et al.* [29], Muştu and Eren [42], Singh [51], etc.

Regarding the optimization of multiobjective criteria relating to setup time studies on single-machine settings, Dileepan and Sen [16] and Fry *et al.* [22] were the leaders to discuss three kinds of procedures, including *a priori*, *a posteriori*, and interactive procedures. Afterward, Liao [34] developed a branch and bound algorithm to solve a single-machine multiple-objective and multiple-job class model. Gupta *et al.* [26] proposed two constructive polynomial time algorithms to solve a hierarchical scheduling problem. Erel and Ghosh [18] investigated a customer order scheduling problem with a certain number of products. Lin and Kononov [35] studied another customer order scheduling problem on parallel dedicated machines. Liu [37,38] and Hsu and Liu [28] addressed three different problems of multiple jobs composing a customer order in a job shop environment. More recently, Allahverdi *et al.* [8] addressed the no-wait flowshop scheduling problem on m machines with separate setup times to minimize total tardiness with an upper bound on makespan. Allahverdi [4] provided a survey of scheduling problems on single machine, parallel machine, flowshop, job shop settings with uncertain processing or sequence independent setup times. As for the sequence-dependent setup times, readers might refer to Alimian *et al.* [2], Allali *et al.* [9], Rifai *et al.* [47], and so on.

Regarding the customer order literature studies with due dates, Blocher *et al.* [13] examined the performance of order-based dispatching rules in a general job shop to minimize the flow time and tardiness. Erel and Ghosh [18] considered a customer order scheduling model of minimizing the total order lead time in which they assumed that a setup is needed whenever production switches from one family to another. Lee [31] proposed a branch-and-bound method incorporating with some dominance properties and three lower bounds and three heuristics to solve the order scheduling problem of minimizing total tardiness. Following the same problem of Lee [31], but with a position-based learning consideration, Xu *et al.* [60] applied branch-and-bound method with new properties and lower bound, simulated annealing, and particle swarm optimization (PSO) algorithms to the problem. Adopting the same customer order scheduling model of Lee [31], but with sum-of-processing-time-based learning effect, Wu *et al.* [56] utilized a branch-and-bound algorithm, four heuristics, and three metaheuristics (the artificial bee colony, a PSO with a varying linear declining inertia weight, and a simulated annealing) to solve it. Framinan and Perez-Gonzalez [20] proposed a new constructive heuristic with a look-ahead mechanism to improve the OMDD heuristic, proposed by Lee [31] and claimed that their proposed method outperformed OMDD. Wu *et al.* [57] proposed three modified heuristics, a hybrid iterated greedy algorithm, and a particle swarm colony algorithm for an order scheduling problem to minimize the linear sum of the total flowtime and the maximum tardiness. More streamlined scheduling problems of customers ordering multiple products, readers might refer to a review paper covering all concurrent-type scheduling problems by Framinan *et al.* [21].

Recently, Antonioli *et al.* [11] considered a customer order scheduling environment in which the setup times are explicit and depend on the production sequence. They proposed a mixed-integer linear programming, several heuristics, and the hybrid matheuristic for minimizing the total tardiness criterion. In view of the fact that the single-machine scheduling issues for orders of multiple work categories have not been addressed, we introduce a bicriteria single-machine scheduling with multiple job classes and customer orders. Our goal is to find a schedule that minimizes a linear combination of the total holding cost and total tardiness cost of given orders.

The remaining part of this study is organized as follows. In Section 2, we introduce the notations and formulate the problem. In Section 3, we propose a dominant property and a lower bound for the branch-and-bound method searching for an optimal solution. In Section 4, we propose four heuristics and a cloudy theoretical simulated annealing (CSA) and a cloudy theoretical simulated annealing hyperheuristic (CSAHH) methods. In Section 5, we experimentally tune the parameters of the proposed CSA and CSAHH. In Section 6, we execute several experiments to evaluate the effectiveness and efficiency of the proposed algorithms. Conclusions and suggestions are summarized in Section 7.

2. Notations and problem formulation

First, some notation used in this study are defined as follows.

$O_M = \{O_1, O_2, \ldots, O_m\}$ denotes a set of m (customer) orders;
$B = \{B_1, B_2, \ldots, B_K\}$ denotes a set of K classes of jobs;
$\{s_1, s_2 \ldots, s_K\}$ denotes the setup times for the K classes of jobs;
$\{p_1, p_2 \ldots, p_n\}$ denotes a set of processing times of n jobs;
σ, σ' denote two full job schedules;
δ, δ' denote two partial schedules;
$\{d_1, d_2 \ldots, d_m\}$ denotes a set of due dates for m orders;
$[\ ]$ denotes the position in a given schedule;
$C_j^f(\sigma)$, $C_j^f(\sigma')$ denotes the completion time of the first job of O_j in $\sigma(\sigma')$, $j = 1, 2, \ldots, m$;
$\mathrm{HC}_j(\sigma) = C_j^l(\sigma) - C_j^f(\sigma)$ denotes the holding cost of order O_j in σ, $j = 1, 2, \ldots, m$;
$\mathrm{TC}_j(\sigma) = \max\{C_j^l(\sigma) - d_j, 0\}$ is the tardiness of order O_j, $j = 1, 2, \ldots, m$.

The considered problem can be stated as follows. Suppose there is a set of n jobs which are grouped into a set of m orders and each order includes several jobs. The jobs can be classified into K different classes and each order contains only one job from each class. Jobs are ready at time zero and will be operated on a single machine. No preemption is allowed during the processing a job. Suppose that each job J_i has a processing time and belongs to a job class. Furthermore, an order consists of at least one job from each job class. During the processing period, if a job belonging to class b is scheduled immediately following a job belonging to class a ($a \neq b$), then a setup time s_b is needed. Otherwise, it does not need a setup time. The holding cost $\mathrm{HC}_j(\sigma)$ of order j is defined by the range between the completion time of the first job in order j and the completion time of the last job from the same order. In this study, we address the problem of minimizing a linear combination of the total holding cost $\sum_{j=1}^m \mathrm{HC}_j(\sigma)$ and total tardiness cost $\sum_{j=1}^m \mathrm{TC}_j(\sigma)$ for m given orders. For convenience, let $h(\sigma) = \alpha \sum_{j=1}^m \mathrm{HC}_j(\sigma) + (1-\alpha) \sum_{j=1}^m \mathrm{TC}_j(\sigma)$. The proposed problem is NP-hard because for fixed $K = 1, m = n, \alpha = 0$, and all setup times 0, the resulting problem is an NP-hard problem (see [43]).

3. A lower bound and property

For the branch-and-bound method, we wish to determine a lower bound of a node $\sigma = (\pi, \pi^c)$, where π is the already scheduled part with n_π jobs while π^c is the set of n_{π^c} unscheduled jobs from q orders ($q \leq m$). Let t_π be the completion time of the last job J_L in π, and let $\sum_{i \in \pi} \mathrm{HC}_i$ be the total holding cost of those completed orders in π.

For the lower bound of holding cost in π^c, we sort the processing times of n jobs by the smallest processing times first rule, *i.e.*, $p_{(1)} \le p_{(2)} \le \ldots \le p_{(n)}$ is a non-decreasing order of $\{p_1, p_2, \ldots, p_n\}$, and we record the frequency number f_i for each of q order in π^c, $i = 1, 2, \ldots, q$, and set $1 \le f_{(1)} \le f_{(2)} \ldots \le f_{(q)}$ as a non-decreasing order of $\{f_i, i = 1, 2, \ldots, q\}$. Next, we calculate the lower bound of completion times of jobs in π^c as $t_\pi + \sum_{v=1}^{i} p_{(v)}$, $i = 1, 2, \ldots, n_{\pi^c}$. We note that $n_{\pi^c} < n$. Thus, we can calculate the lower bound of holding cost for q orders as follows:

$$\begin{aligned}\sum_{i\in\pi^c} \mathrm{HC}_i &= \Big\{C_{\left[n_\pi+f_{(1)}\right]}(\sigma) - C_{[n_\pi+1]}(\sigma) + C_{\left[n_\pi+f_{(1)}+f_{(2)}\right]}(\sigma) - C_{\left[n_\pi+f_{(1)}+1\right]}(\sigma) + \ldots \\ &\quad + C_{\left[n_\pi+f_{(1)}+f_{(2)}+\ldots+f_{(q)}\right]}(\sigma) - C_{\left[n_\pi+f_{(1)}+f_{(2)}+\ldots+f_{(q-1)}+1\right]}(\sigma)\Big\} \\ &= \sum_{i=1}^{q}\Big(C_{\left[n_\pi+\sum_{v=1}^{i} f_{(v)}\right]}(\sigma) - C_{\left[n_\pi+\sum_{v=0}^{i-1} f_{(v)}\right]}(\sigma)\Big),\end{aligned}$$

where it is assumed that $f_{(0)} = 1$, the range of $C_{[n_\pi+f_{(1)}]}(\sigma) - C_{[n_\pi+1]}(\sigma)$ denotes the lower bound of holding cost of the remaining order with f_1 jobs, the range of $C_{[n_\pi+f_{(1)}+f_{(2)}]}(\sigma) - C_{[n_\pi+f_{(1)}+1]}(\sigma)$ denotes the lower bound of holding cost of the remaining order with f_2 jobs and the range of $C_{[n_\pi+f_{(1)}+f_{(2)}+\ldots+f_{(q)}]}(\sigma) - C_{[n_\pi+f_{(1)}+f_{(2)}+\ldots+f_{(q-1)}+1]}(\sigma)$ denotes the lower bound of holding cost of the remaining order with f_q jobs.

For the lower bound of the tardiness cost of those q orders in π^c, we sort the due dates of q orders by the earliest due dates first rule, *i.e.*, $d_{(1)} \le d_{(2)} \le \ldots \le d_{(q)}$ is a non-decreasing order of $\{d_1, d_2, \ldots, d_q\}$, which can be determined as follows:

$$\begin{aligned}\sum_{i\in\pi^c} \mathrm{TC}_i &= \max\Big\{C_{\left[n_\pi+f_{(1)}\right]}(\sigma) - d_{(1)}, 0\Big\} + \max\Big\{C_{\left[n_\pi+f_{(1)}+f_{(2)}\right]}(\sigma) - d_{(2)}, 0\Big\} + \ldots \\ &\quad + \max\Big\{C_{\left[n_\pi+f_{(1)}+f_{(2)}+\ldots+f_{(q)}\right]}(\sigma) - d_{(q)}, 0\Big\} \\ &= \sum_{i=1}^{q}\Big(\max\{C_{\left[n_\pi+\sum_{v=1}^{i} f_{(v)}\right]}(\sigma) - d_{(i)}, 0\}\Big).\end{aligned}$$

From the above analysis, we have the following inequality:

$$\begin{aligned}&\left\{\alpha \sum_{i=1,\ldots,m} \mathrm{HC}_i + (1-\alpha) \sum_{i=1,\ldots,m} \mathrm{TC}_i\right\} \ge \alpha\left\{\sum_{i\in\pi} \mathrm{HC}_i + \sum_{i=1}^{q}\Big(C_{\left[n_\pi+\sum_{v=1}^{i} f_{(v)}\right]}(\sigma)\right. \\ &\quad \left.- C_{\left[n_\pi+\sum_{v=0}^{i-1} f_{(v)}\right]}(\sigma)\Big)\right\} + (1-\alpha)\left\{\sum_{i\in\pi} \mathrm{TC}_i + \sum_{i=1}^{q}\Big(\max\Big\{C_{\left[n_\pi+\sum_{v=1}^{i} f_{(v)}\right]}(\sigma) - d_{(i)}, 0\Big\}\Big)\right\}.\end{aligned}$$

Therefore, a lower bound on node $\sigma = (\pi, \pi^c)$ can be obtained as follows.

$$\begin{aligned}\mathrm{LB1}(\pi) &= \alpha\left\{\sum_{i\in\pi} \mathrm{HC}_i + \sum_{i=1}^{q}\Big(C_{\left[n_\pi+\sum_{v=1}^{i} f_{(v)}\right]}(\sigma) - C_{\left[n_\pi+\sum_{v=0}^{i-1} f_{(v)}\right]}(\sigma)\Big)\right\} \\ &\quad + (1-\alpha)\left\{\sum_{i\in\pi} \mathrm{TC}_i + \sum_{i=1}^{q}\Big(\max\Big\{C_{\left[n_\pi+\sum_{v=1}^{i} f_{(v)}\right]}(\sigma) - d_{(i)}, 0\Big\}\Big)\right\}.\end{aligned}$$

Apart that, let S^c be a subset of the job classes for jobs in π^c exclude the last job class in π. Another lower bound for the holding cost by sorting only the jobs in π^c and by considering the setup times is also developed as follows:

$$\sum_{i\in\pi^c} \mathrm{HC}_i = \Big\{C_{\left[n_\pi+f_{(1)}\right]}(\sigma) - C_{[n_\pi+1]}(\sigma) + C_{\left[n_\pi+f_{(1)}+f_{(2)}\right]}(\sigma) - C_{\left[n_\pi+f_{(1)}+1\right]}(\sigma) + \ldots$$

$$+ C_{[n_\pi+f_{(1)}+f_{(2)}+\ldots+f_{(q)}]}(\sigma) - C_{[n_\pi+f_{(1)}+f_{(2)}+\ldots+f_{(q-1)}+1]}(\sigma)\Big\}$$
$$= \sum_{i=1}^{q}\Big(C_{[n_\pi+\sum_{v=1}^{i} f_{(v)}]}(\sigma) - C_{[n_\pi+\sum_{v=0}^{i-1} f_{(v)}]}(\sigma)\Big) + \sum_{S^c} s_k.$$

Therefore, another lower bound on node $\sigma = (\pi, \pi^c)$ can be obtained as follows:

$$\mathrm{LB2}(\pi) = \alpha\left\{\sum_{i\in\pi}\mathrm{HC}_i + \sum_{i=1}^{q}\Big(C_{[n_\pi+\sum_{v=1}^{i} f_{(v)}]}(\sigma) - C_{[n_\pi+\sum_{v=0}^{i-1} f_{(v)}]}(\sigma)\Big)\right\} + \alpha\sum_{S^c} s_k$$
$$+ (1-\alpha)\left\{\sum_{i\in\pi}\mathrm{TC}_i + \sum_{i=1}^{q}\Big(\max\{C_{[n_\pi+\sum_{v=1}^{i} f_{(v)}]}(\sigma) - d_{(i)}, 0\}\Big)\right\}.$$

In order to find a better lower bound, we have

$$\mathrm{LB}(\pi) = \max\{\mathrm{LB1}(\pi), \mathrm{LB2}(\pi)\}.$$

It is noted that when $\pi = \emptyset$, the lower bound on node $\sigma = (\pi, \pi^c)$ can be obtained as $\sum_{i\in\pi^c}\mathrm{HC}_i = \mathrm{mo}\times(C_{[\mathrm{mk}]}(\sigma) - C_{[1]}(\sigma))$ and $\sum_{i\in\pi^c}\mathrm{TC}_i = \sum_{i=1}^{\mathrm{mo}}\max\{C_{[\mathrm{mk}\times i)]}(\sigma) - d_{(i)}, 0\}$, therefore, we have

$$\mathrm{LB}_0 = \alpha\sum_{i\in\pi^c}\mathrm{HC}_i + (1-\alpha)\sum_{i\in\pi^c}\mathrm{TC}_i$$
$$= \alpha\times\mathrm{mo}\times\big(C_{[\mathrm{mk}]}(\sigma) - C_{[1]}(\sigma)\big) + (1-\alpha)\sum_{i=1}^{\mathrm{mo}}\max\big\{C_{[\mathrm{mk}\times i]}(\sigma) - d_{(i)}, 0\big\}.$$

Additionally, a property is proposed to improve the power of the searching ability of the branch-and-bound method. Let $\sigma = (\pi, J_i, J_j, \pi^c)$ and $\sigma' = (\pi, J_j, J_i, \pi^c)$ be two schedules in which job i is scheduled before job j in σ, while job j is scheduled before job i in σ'. Moreover, let job J_L be the last job in π. We have the following property.

Property 1. Assume that jobs J_L, J_i, and J_j belong to the same job class, and $J_i \in O_u$, $J_j \in O_v$, and $u \neq v$. If both jobs J_i and J_j are the last assigned jobs of O_u and O_v, and the first assigned job of O_u is scheduled after the first assigned job of O_v, $p_i < p_j$, $d_u < d_v$, then σ dominates σ'.

Proof. The objective function is $h(\sigma) = \alpha\sum_{j=1}^{m}\mathrm{HC}_j(\sigma) + (1-\alpha)\sum_{j=1}^{m}\mathrm{TC}_j(\sigma)$. It needs to be shown that $h(\sigma) < h(\sigma')$.

(1) Because, $C_u^f(\sigma') = C_u^f(\sigma)$, and $C_v^f(\sigma') = C_v^f(\sigma)$ the $\sum_{j=1}^{m}\mathrm{HC}_j(\sigma) - \sum_{j=1}^{m}\mathrm{HC}_j(\sigma') = [\mathrm{HC}_i(\sigma) + \mathrm{HC}_j(\sigma)] - [\mathrm{HC}_j(\sigma') + \mathrm{HC}_i(\sigma')] = [C_u^l(\sigma) - C_u^f(\sigma) + C_v^l(\sigma) - C_v^f(\sigma)] - [C_v^l(\sigma') - C_v^f(\sigma') + C_u^l(\sigma') - C_u^f(\sigma')] = [C_u^l(\sigma) - C_u^l(\sigma')] + [C_v^l(\sigma) - C_v^l(\sigma')] = p_i - (p_j + p_i) + (p_i + p_j) - p_j = p_i - p_j < 0$.

(2) The rest of this proof is to show that $\mathrm{TD} = \sum_{u=1}^{m}\mathrm{TC}_u(\sigma) - \sum_{u=1}^{m}\mathrm{TC}_u(\sigma') = [\mathrm{TC}_u(\sigma) + \mathrm{TC}_v(\sigma)] - [\mathrm{TC}_v(\sigma') + \mathrm{TC}_u(\sigma')] \leq 0$, where

$$\mathrm{TC}_u(\sigma) = \max(t_\pi + p_i - d_u, 0), \mathrm{TC}_v(\sigma) = \max(t_\pi + p_i + p_j - d_v, 0)$$
$$\mathrm{TC}_v(\sigma') = \max(t_\pi + p_j - d_v, 0), \mathrm{TC}_u(\sigma') = \max(t_\pi + p_j + p_i - d_u, 0).$$

The time order $t_\pi < t_\pi + p_i < t_\pi + p_j < t_\pi + p_i + p_j$ and the given condition $d_u < d_v$ determine the TD values of ten situations. Three of these ten situations are proven as follows, and the remaining proofs of are similar.

(i) if $d_u < t_\pi + p_i$ and $d_v > t_\pi + p_i + p_j$

$$\mathrm{TD} = [(t_\pi + p_i - d_u) + 0] - [0 + (t_\pi + p_j + p_i - d_u)] = -p_j < 0.$$

(ii) if $t_\pi + p_i < d_u < t_\pi + p_j < d_v < t_\pi + p_i + p_j$

$$\text{TD} = [0 + (t_\pi + p_i + p_j - d_v)] - [0 + (t_\pi + p_j + p_i - d_u)] = d_u - d_v < 0.$$

(iii) if $d_u < t_\pi + p_i < d_v < t_\pi + p_j$

$$\text{TD} = [(t_\pi + p_i - d_u) + (t_\pi + p_i + p_j - d_v)] - [(t_\pi + p_j - d_v) + (t_\pi + p_j + p_i - d_u)] = p_i - p_j < 0.$$

Combining (1) and (2), $h(\sigma) < h(\sigma''))$ follows. □

4. Heuristics, cloudy theoretical simulated annealing algorithms and branch-and-bound

To find near-optimal solutions, in this section, we first present four heuristics, each of which was iteratively improved by a local pairwise interchange method (pi-method) several times. The pi-method means that taking a schedule with five jobs as $\sigma = (1, 2, 3, 4, 5)$, we generate two random integers $u = 2$ and $v = 4$ by randomness and then swap the 2nd job and 4th job in schedule σ to generate a new schedule as $\sigma' = (1, 4, 3, 2, 5)$. Afterwards, we propose a cloudy theoretical simulated annealing algorithm adopting found job sequences from four heuristics (without improvement by the pi-method) as their initial seeds. Finally, we propose a cloudy theoretical simulated annealing-based hyper heuristic algorithm, along with five low-level heuristics, to solve the problem. Based on the characteristics of the studied problem, multiple-class jobs and customer order of jobs, we develop four heuristics as follows.

Heuristic 1. It is modified from the idea of Gupta *et al.* [26].

Step 1. We assign m orders based on the smallest value of the sum of processing times of each order plus its setup time to yield "an order schedule".

Step 2. For each order in the yielded order schedule in Step 1, we assign job sequence based on the smallest processing time first to yield job sequence within each order. For simplification, Steps 1 and 2 are termed "OSPTLPT".

Step 3. The solution of the OSPTLPT is improved by using a pi-method iteratively. It is termed "OSPTLPT_pi".

Heuristic 2. Step 1. We assign m orders based on the smallest value of the sum of job processing times in each class to yield "a class schedule".

Step 2. In the class schedule in Step 1, we assign the job sequence based on the largest processing time first in the first class and assign the job sequence based on the smallest processing time first in other classes. For simplification, Steps 1 and 2 are termed "CSLPT".

Step 3. The solution of the CSLPT is improved by using a pi-method iteratively. It is termed "CSLPT_pi".

Heuristic 3. Step 1. We assign m orders based on the earliest order due date first rule to yield "an order schedule".

Step 2. Given the order schedule in Step 1, we assign the jobs within each order according to the non-increasing order of the sum of processing times of each job plus its setup time to yield a job sequence. For simplification, Steps 1 and 2 are termed "OEDDLPT".

Step 3. The solution of the OEDDLPT is improved by using a pi-method iteratively. It is termed "OEDDLPT_pi".

Heuristic 4. Step 1. We assign m orders based on the earliest order due date first rule to yield "an order schedule".

Step 2. Given the order schedule in Step 1, we assign the jobs in each order according to the non-decreasing order of the sum of processing times of each job plus its setup time to yield a job sequence. For simplification, Steps 1 and 2 are termed "OEDDSPT".

Step 3. The solution of the OEDDSPT is improved by using a pi-method iteratively. It is termed "OEDDSPT_pi".

There are three stopping rules used presented in used approximate methods such as the maximum number of iterative cycles, specified CPU time limit, and the maximum number of cycles between two improvements of the global best solution (refer to [50,55]). However, the stopping rule of proposed CSA or CSAHH are adopted an initial temperature T_0 and a final temperature T_f in this study. In what follows, to find the good quality of approximate solutions, we set the best solutions (job schedules) found from the four heuristics (OSPTLPT, CSLPT, OEDDLPT, and OEDDSPT) as the four initial seeds in the cloudy theoretical simulated annealing (CSA) algorithm [53]. The CSA algorithm equipped with four different seeds is recorded as algorithms CSA1, CSA2, CSA3, and CSA4. As mentioned in Torabzadeh and Zandieh [53], the CSA algorithm uses several parameters, including an initial temperature T_0, a final temperature T_f, an annealing index λ, and a number of improvement repetitions N_r. In addition, let $\delta < 10^{-8}$ (be a very small number) and η be a random number selected from a uniform distribution $U(0,1)$. Then, the procedure of the CSA algorithm is summarized as follows.

The steps of the CSA algorithm:

01: Input $T_0(<1), T_f$, λ, and N_r and set σ_0 as an initial schedule.
02: Set $T = T_0$, $\sigma = \sigma_0$, and $h(\sigma) = h(\sigma_0)$, $j = 1$
03: **Do while** $(T > T_f)$
04: Set $H_e = T$, $E_n = T$, $u_0 = 1 - T$, $E'_n = \max\{H_e + E_n - \eta/3, \delta\}$
05: Set $T' = E'_n \cdot \sqrt{-2\ln(u_0)}$
06: iterative = 1
07: **Do while** (iterative $\leq N_r$)
08: Choose integers p, q from $U[1,n]$ and change the pth position and qth position jobs in σ to create a new schedule σ_{new} with $h(\sigma_{new})$.
09: if $h(\sigma_{new}) < h(\sigma)$, replace σ by σ_{new}; otherwise, replace σ by σ_{new} if $\exp(-(h(\sigma_{new}) - h(\sigma))/T')$, where d from $U(0,1)$.
10: iterative = iterative + 1.
11: **End while**
12: $j = j + 1, T = T \times \lambda^j$
13: **End While**
14: **Output** the final best schedule σ and its corresponding objective function $h(\sigma)$.

In addition, a cloudy theoretical simulated annealing hyperheuristic algorithm (CSAHH) along with five low-level heuristics is also proposed to solve this problem. A hyperheuristic with the operators of the low-level heuristics and high-level strategy has been widely utilized in searching for near-optimal solutions (Dowsland *et al.* [17], Burke *et al.* [14], Gascón-Moreno *et al.* [23], Anagnostopoulos *et al.* [10], Wu *et al.* [58], Zhao *et al.* [65], and so on.). Instead of solving the problem directly, a low-level heuristic is directly applied for searching the solution space. The high-level policy includes the heuristic selection policy and acceptance criterion. The high-level policy is adopted to find a low-level heuristic to yield a new solution. Wu *et al.* [58] proposed five local improvement methods and separately used each of them in the cloud theory-based simulated annealing algorithm to solve a single-machine past sequence setup scheduling with two scenarios. For the diversity of neighborhood search, we modify these five local improved methods to be the five low-level heuristics (called LH_1, LH_2, LH_3, LH_4, and LH_5) embedded in the CSA algorithm. Applying a low-level heuristic on a job sequence, it first needs to choose two jobs randomly from the sequence. Then, these five low-level heuristics consist of (LH_1) swapping two chosen jobs; (LH_2) switching both jobs with their immediately succeeding jobs; (LH_3) switching both jobs with their closet-most preceding jobs; (LH_4) switching the job in front with its immediately succeeding job and

switching another with its closet-most preceding job; and (LH_5) switching the job in front with its close-most preceding job and switching another with its immediately succeeding job.

Let iCmax denote the total number of iterations in performing CSAHH. In the first run of performing the CSAHH, the probability of all five low-level heuristics contributing to the CSAHH are equally set to 1/5. We recode the accumulated performance times z_i of each LH_i in the first run and then set $z_i/\sum_{i=1}^{5} z_i$ as the selection probability of LH_i for the subsequent run. For diversity, we set $z_i = \max\{1, z_i\}$ to keep all five low-level heuristics in the pool from the first run to the end of the last run in performing CSAHH. Furthermore, the notations T_0, T_f, and λ used in CSAHH are the same as those in CSA, while Nr denotes the total number of times for five low-level heuristics called in each run. The description of the CSAHH is summarized as follows.

The procedures of the CSAHH method:

Set initial seed $\sigma = \sigma_0$, and its objective function $h(\sigma) = h(\sigma_0)$;
Input parameters Nr, iCmax, T_0, T_f; $j = 1$
Set $T = T_0$;
Set $z_k = 1, k = 1, \ldots, 5$;
For $i\_run = 1$ **to** iCmax **do**
Set $H_e = T$, $E_n = T$, $u_0 = 1 - T$, $E'_n = \max\{H_e + E_n - \eta/3, \delta\}$
Set $T' = E'_n \cdot \sqrt{-2\ln(u_0)}$
  For $id = 1$ **to** Nr **do**
    Choose an LH_k *based on the acceptance* probability among five LH_s;
    Apply it to yield a new schedule $\sigma_{k,\mathrm{new}}$ and compute its objective function $h(\sigma_{k,\mathrm{new}})$;
    Determine if $h(\sigma_{k,\mathrm{new}}) < h(\sigma)$, update σ by $\sigma_{k,\mathrm{new}}$, $z_k = z_k + 1$
    Otherwise, update σ by $\sigma_{k,\mathrm{new}}$ if $\exp(-(h(\sigma_{k,\mathrm{new}}) - h(\sigma))/T') < d$, where $d \in U(0,1)$;
  End for
  $j = j + 1$, Update $T = T_0 \times \lambda^j$
  Update $z_k/\sum_{k=1}^{5} z_k$ for each LH_k;
  Update $\sigma = \sigma_0$, and its objective function $h(\sigma) = h(\sigma_0)$
if $T < T_\mathrm{f}$, exit and go to step 18;
End for
Output σ and its $h(\sigma)$.

To solve the proposed problem, we applied the branch-and-bound (B&B) method for finding an exact solution. Performing the B&B method, we begin the root node with no job and branch the root node to obtain new nodes by adding the unscheduled jobs to the root node. We adopt a depth-search first rule and schedule the jobs from the first position to the last.

For each considered node, new jobs (nodes) selected from those unexplored jobs are one by one fabricated into a complete schedule by appending the selected jobs. A lower bound is calculated for each such node. Then, the dominance Property 1 and these lower bounds are used to remove those nodes that satisfy the dominance property or calculated lower bounds are greater than an established upper bound (obtained from the heuristics or CSA algorithm). Further branching is continued with the remaining nodes that are not considered yet. Until all possible nodes are either considered or removed, this procedure is repeated to obtain an optimal solution. On the basis of the above description, the steps of the proposed B&B method are described below.

Input: A set of $n(= Km)$ jobs with processing times $\{p_1, p_2, \ldots, p_n\}$ assigned independently into two sets: K classes $B = \{B_1, B_2, \ldots, B_K\}$ with class setup times $\{s_1, s_2, \ldots, s_K\}$ and m customer orders $O_M = \{O_1, O_2, \ldots, O_m\}$, where each O_j contains exactly one job from B_k.
Objective function: Minimize $h(\sigma) = \alpha \sum_{j=1}^{m} \mathrm{HC}_j(\sigma) + (1-\alpha) \sum_{j=1}^{m} \mathrm{TC}_j(\sigma)$.

Step 1. The search starts with an upper bound obtained from the best heuristic among all heuristics/algorithms described in Section 4. From the root node, it performs branching by appending each job as a new node.

Step 2. For each considered active node, compute the lower bounds using the equation $\mathrm{LB}(\pi)$, determinate if the lower bounds are equal to or larger than the incumbent upper bound, and remove those nodes and all beneath nodes in the branching tree.

Step 3. Applying the dominance Property 1 to purge the unwanted nodes from the branching tree.

Step 4. If the considered node is a complete schedule (say σ), then compute its objective value $h(\sigma)$ and if this objective value is smaller than the upper bound, then update the upper bound and the current solution by the σ; otherwise, branch from the node with the depth search rule on the remaining nodes to create a set of active nodes.

Step 5. Conduct Steps 2 through 4 repeatedly, until all nodes have been considered. Let the final complete schedule be σ_{opt}.

Output. The σ_{opt} is an optimal solution.

Besides, an illustrative example and node trees also were provided to explain the proposed model and the steps of the proposed Branch and Bound algorithm, respectively.

Example. The following processing times of four jobs can be classified 2 classes and 2 orders.

Job id	J_1	J_2	J_3	J_4
p_i	22	88	11	30
Class id	1	1	2	2
Oder id	1	2	1	2

Furthermore, let $\alpha = 0.25$, and let $s_1 = 5$ and $s_2 = 6$ denote the setup time for class 1 and class 2 while $d_1 = 70$ and $d_2 = 62$ denote the due date for order 1 and order 2, respectively.

Given $\sigma = (J_3, J_1, J_2, J_4)$, the completion times are computed as follows:

$$\begin{aligned} C_{[1]}(\sigma) &= s_2 + p_{[1]} = 6 + 11 = 17 && /^* \text{ for order 1,} \\ C_{[2]}(\sigma) &= C_{[1]}(\sigma) + s_1 + p_{[2]} = 17 + 5 + 22 = 44 && /^* \text{ for order 1,} \\ C_{[3]}(\sigma) &= C_{[2]}(\sigma) + p_{[3]} = 44 + 88 = 132 && /^* \text{ for order 2,} \end{aligned}$$

and

$$C_{[4]}(\sigma) = C_{[3]}(\sigma) + s_2 + p_{[4]} = 132 + 6 + 30 = 168 \qquad /^* \text{ for order 2.}$$

Thus, we have

$$\sum_{j=1}^{2} \mathrm{HC}_j(\sigma) = C_{[2]}(\sigma) - C_{[1]}(\sigma) + C_{[4]}(\sigma) - C_{[3]}(\sigma) = 44 - 17 + 168 - 132 = 63,$$

and

$$\sum_{j=1}^{2} \mathrm{TC}_j(\sigma) = \max\bigl\{0, C_{[2]}(\sigma) - d_1\bigr\} + \max\bigl\{0, C_{[4]}(\sigma) - d_2\bigr\} = \max\{0, 44 - 70\} + \max\{0, 173 - 62\} = 106.$$

Therefore, the objective function $h(\sigma)$ of $\sigma = (J_2, J_4, J_1, J_3)$ is given by

$$h(\sigma) = 0.25 \sum_{j=1}^{2} \mathrm{HC}_j(\sigma) + (1 - 0.25) \sum_{j=1}^{2} \mathrm{TC}_j(\sigma) = 95.25.$$

The best solution $h(\sigma)$ among all the proposed methods was designed as the upper bound of B&B. Figure 1 presents the details of the steps of the proposed Branch and Bound algorithm.

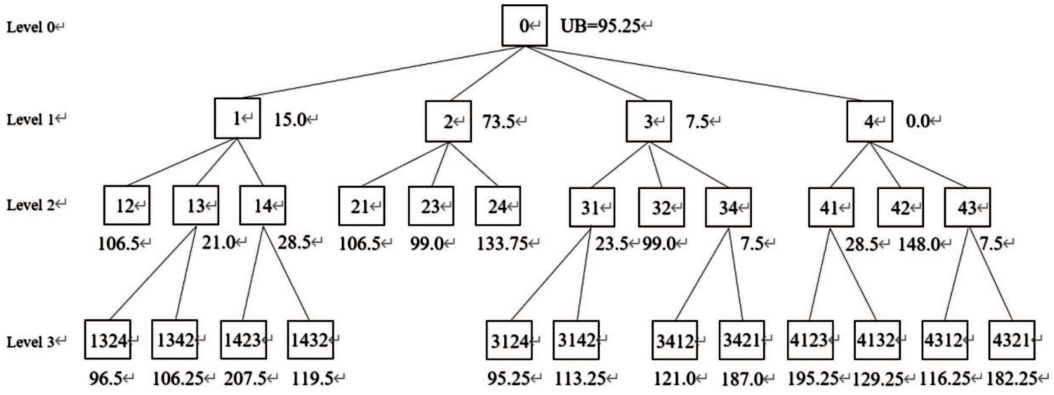

FIGURE 1. The steps of B&B for illustrative example.

Given node $\sigma = (J_1, J_4, -, -)$ in figure, the completion times for the jobs in the scheduled part are computed as follows:

$$\begin{aligned}
&C_{[1]}(\sigma) = s_1 + p_{[1]} = 5 + 22 = 27 && \text{/* for order 1,}\\
&C_{[2]}(\sigma) = C_{[1]}(\sigma) + s_2 + p_{[2]} = 27 + 6 + 30 = 63 && \text{/* for order 2,}\\
&\sum_{i\in\pi} \mathrm{HC}_i = 0, \text{ where } \pi = \{J_1, J_4\}, && \text{/ no order is completed in } \pi.
\end{aligned}$$

Regarding to the estimated completion times for those jobs in the unscheduled part are computed as follows.

At first, we sort the job processing times in nondecreasing order to obtain $p_{(1)} \le p_{(2)} \le \ldots \le p_{(4)}$ or $11 \le 22 \le 30 \le 88$, and compute the estimated completion times for the unscheduled jobs as $C_{[3]}(\sigma) = t_\pi + \sum_{v=1}^{1} p_{(v)} = 63 + 11 = 74$ and $C_{[4]}(\sigma) = t_\pi + \sum_{v=1}^{2} p_{(v)} = 63 + 11 + 22 = 96$.

Because $f_{(1)} = 1, f_{(2)} = 1$, $d_{(1)} = 62$, and $d_{(2)} = 70$, we can find

$$\begin{aligned}
\sum_{i\in\pi^c} \mathrm{TC}_i &= \max\Big\{C_{[n_\pi + f_{(1)}]}(\sigma) - d_{(1)}, 0\Big\} + \max\Big\{C_{[n_\pi + f_{(1)} + f_{(2)}]}(\sigma) - d_{(2)}, 0\Big\}\\
&= \max\{74 - 62, 0\} + \max\{96 - 70, 0\} = 38,
\end{aligned}$$

$$0.25 \times \left\{\sum_{i\in\pi} \mathrm{HC}_i + \sum_{i=1}^{2}\Big(C_{\left[n_\pi + \sum_{v=1}^{i} f_{(v)}\right]}(\sigma) - C_{\left[n_\pi + \sum_{v=0}^{i-1} f_{(v)}\right]}(\sigma)\Big)\right\} = 0.0,$$

and

$$0.75 \times \left\{\sum_{i\in\pi} \mathrm{TC}_i + \sum_{i=1}^{q}\Big(\max\Big\{C_{\left[n_\pi + \sum_{v=1}^{i} f_{(v)}\right]}(\sigma) - d_{(i)}, 0\Big\}\Big)\right\} = 0.75 \times (0 + 38) = 28.5.$$

Thus, the lower bound of node $\sigma = (J_1, J_4, -, -)$ is 28.5.

5. Exploring the parameters used in CSA algorithms

In this section, we explore the appropriate values of parameters built into the CSA and CSAHH algorithms. These parameters are the annealing index (λ), the number of improvements (Nr), the initial temperature (T_0), and the number of runs (iCmax). To choose values for these parameters, we design a scheme consisting of 9 jobs, which are grouped into three orders and three classes, and generate one hundred tested instances for each parameter combination of λ, Nr, T_0, and iCmax. The job processing times are generated uniformly from

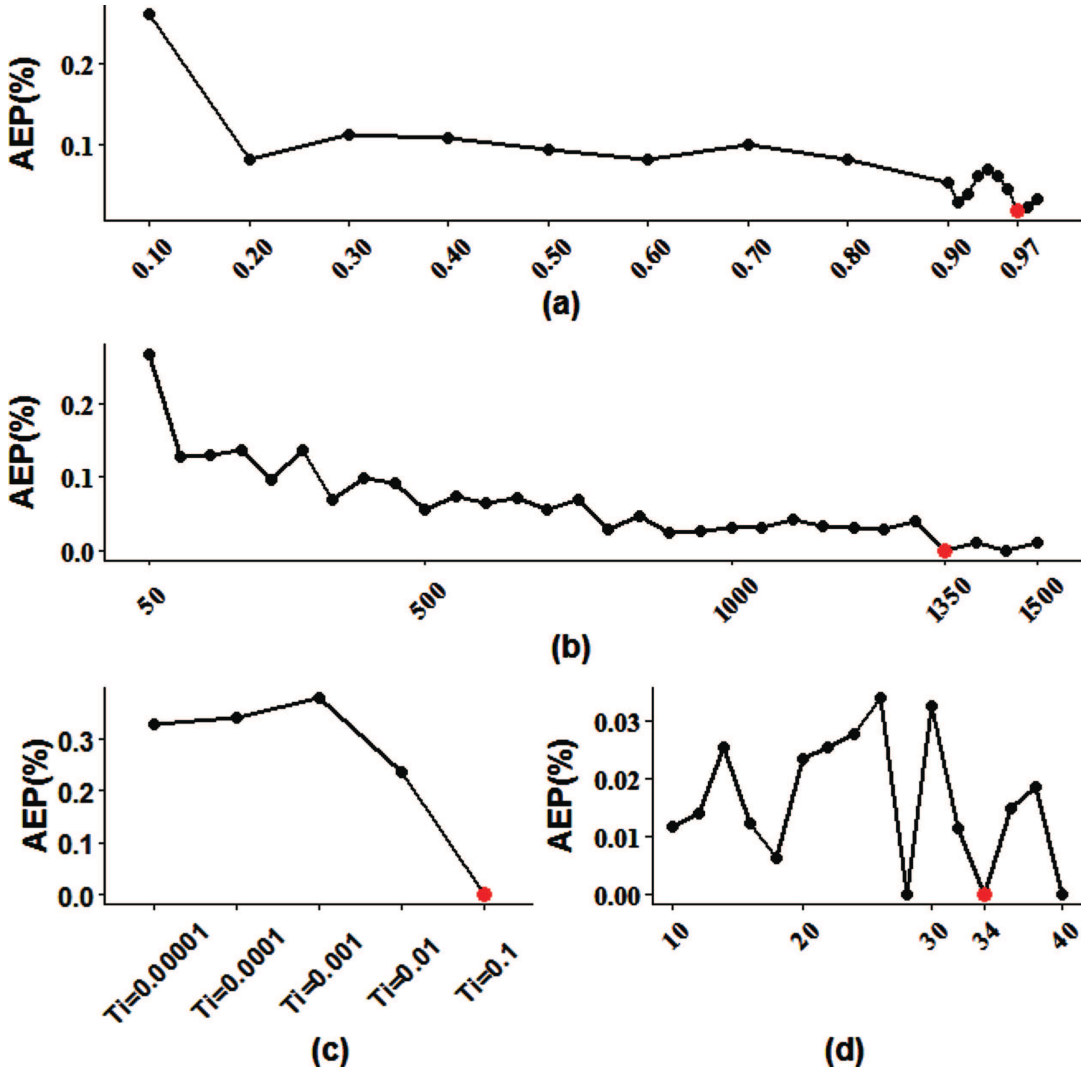

FIGURE 2. Exploring the values of parameters used in CSA and CSAHH. (a) lambda. (b) Nr. (c) Ti. (d) iCmax.

distribution $(1, 100)$, while the class setup times are generated from another uniform distribution $(1, 20)$. The customer order due dates are generated uniformly from distribution $\text{SP} \times (1 - \tau - R/2, 1 + \tau + R/2)$, where $\tau = 0.25, R = 0.25$, and $\text{SP} = \sum_{i=1}^{n} p_i$. We report the average error percentages AEP, where AEP is defined as $\text{AEP} = \text{mean}\left[\frac{H_i - \text{OB}_i}{\text{OB}_i}\right] \times 100\%$. In addition, H_i is the solution obtained from the CSAHH and OB_i is the solution obtained from the B&B method.

The parameter adjustment process is based on the one-factor at a time experimental method [41] on a generated set of problem instances. To explore the value of λ, after several trials, we fixed $\text{iCmax} = 20$, $T_0 = 0.1$, $T_\text{f} = 10^{-10}$ and $\text{Nr} = 950$ and tested the value of λ first over the interval $(0.1, 0.9)$ by an increment of 0.1 and second over the interval $(0.9, 0.99)$ by an increment of 0.01. Figure 2a shows that the AEP approaches a lower point at 0.9 on the interval $(0.1, 0.9)$ case and then approaches a lower point at 0.97 on the interval $(0.9, 0.99)$ case. Thus, we adopt $\lambda = 0.97$ in the later simulation experiments.

To explore the value of Nr, we set $\lambda = 0.97$, fix the other parameters as iCmax = 20, $T_0 = 0.1$, $T_f = 10^{-10}$, and test the value of Nr over the interval $(50, 1500)$ by an increment of 50 per trial. Figure 2b shows that the trend of AEP declines rapidly as the value of Nr increases and becomes flat when Nr is larger than 1350. Thus, we adopt Nr = 1350 in the later tests.

Following exploring parameters λ and Nr, we set $\lambda = 0.97$ and Nr = 1350 and fix iCmax = 20, and $T_f = 10^{-10}$. We test the value of T_0 from 0.00001 to 0.1 by a common rate of 0.1 per trial. Figure 2c shows that the AEP declines rapidly to a lower point at $T_0 = 0.1$.

For the value of parameter iCmax, we set $\lambda = 0.97$, Nr = 1350, and $T_0 = 0.1$ and fix $T_f = 10^{-10}$. We then explore the values of iCmax from 10 to 40 (times of runs) by an increment of 2 runs per trial. Figure 2d shows that there are three local lower points in those tests. The lowest point is located at 34. Therefore, iCmax = 34 is used in later simulation experiments.

6. Computational simulations and results

A few computational simulations were executed to determine the effectiveness of the branch-and-bound method, the four local heuristics (OSPTLPT, CLSPT, OEDDLPT, OEDDSPT) and their pairwise interchange improvement counterparts (OSPTLPT_pi, CLSPT_pi, OEDDLPT_pi, OEDDSPT_pi), a CSA algorithm fed with four seeds found from four local heuristics, coded as CSA1, CSA2, CSA3, CSA4, and the CSAHH algorithms. All heuristics and CSA or CSAHH algorithms were coded in Visual Fortran (version 6.6, Compaq) and implemented on a PC with 32.0 GB RAM in Windows 10 and a 3.00 GHz Intel(R) Core(TM) i7-9700 CPU. The problem instances generated followed the design of Leung *et al.* [32] and Lee [31]. The job processing times were generated from a uniform distribution $U(1, 100)$. The due dates of jobs were generated (following [19]) from a uniform distribution $\mathrm{SP} \times U(1-\tau-R/2, 1+\tau+R/2)$, where $\mathrm{SP} = \sum_{j=1}^{n} p_j$ is the sum of the processing times of the n jobs, and τ and R represent the tardiness factor and range of the due dates, respectively. There are six combinations of (τ, R): $(0.25, 0.25)$, $(0.25, 0.50)$, $(0.25, 0.75)$, $(0.50, 0.25)$, $(0.25, 0.50)$, and $(0.25, 0.75)$. Meanwhile, three different types of weights (α) are 0.25, 0.50 and 0.75, and two types of setup times ("setup" in tables) were employed, one followed a $U(1, 10)$ and the other followed a $U(1, 20)$.

The number of jobs was set at $n = 8$, 10 and 12 for small-sized cases. For each n, the combination of number of orders and number of classes (mo × mk) were set differently. There were 2, 2, and 4 combinations for $n = 8$, 10, 12, respectively. For each combination of the aforementioned parameters, a set of 100 problem instances were generated randomly. Thus, in total, 28 800$(= \sum_n(\tau \times R \times \alpha \times (\mathrm{mo} \times \mathrm{mk}) \times \mathrm{setup} \times 100) = (2 \times 3 \times 3 \times 2 \times 2 + 2 \times 3 \times 3 \times 2 \times 2 + 2 \times 3 \times 3 \times 4 \times 2) \times 100)$ instances were tested. To avoid too much computational time, as the number of nodes searched exceeded 10^9, the B&B method jumped to the next set of problem instances.

With regard to computational experiments on large-sized job cases, the number of jobs was set at $n = 60$ and 100. A set of 100 random problem instances were generated for each combination of parameters n, τ, R, α, mo × mk, and "setup". In total, 61 200 $= \sum_n(\tau \times R \times \alpha \times (\mathrm{mo} \times \mathrm{mk}) \times \mathrm{setup}) \times 100 = (2 \times 3 \times 3 \times 10 \times 2 + 2 \times 3 \times 3 \times 7 \times 2) \times 100$ instances were tested.

6.1. Analysis of small-n computational results

The average error percentage (AEP) is employed to evaluate the capability of searching optimal job sequences for the proposed 8 heuristics (OSPTLPT, CSLPT, OEDDLPT, OEDDSPT, OSPTLPT_pi, CSLPT_pi, OEDDLPT_pi, OEDDSPT_pi) and the five CSA algorithms (CSA1 to CSA4 and CSAHH). The AEP is defined as $100(\mathrm{mean}[(H_i - \mathrm{OB}_i)/\mathrm{OB}_i])[\%]$, where H_i is the value of $h(\sigma)$ found from each heuristic/algorithm and OB_i from the B&B method.

Tables 1 and 2 convey the performance of the B&B method. Column 3 (Tabs. 1 and 2) shows a sharp increase in the mean of the explored nodes as n rises from 8, 10 to 12. The average CPU times (in seconds), presented in Tables 1 and 2, dramatically increase as the number of jobs n increases to 12. In addition, all of the generated problem instances were solvable within 10^9 nodes. The ratio of feasible solution (marked as RF, RF = the number of tested instances solved out by B&B or by heuristics/total number instances tested per case)

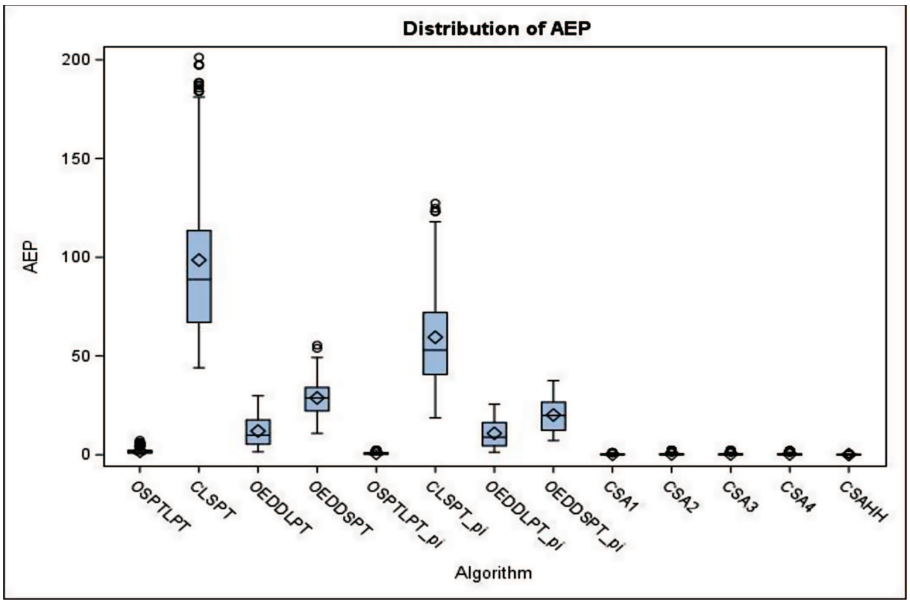

FIGURE 3. Boxplots of AEP for heuristics and CSA algorithms (small n).

is presented in Column 5 of Tables 1 and 2. Table 2 summarizes the CPU time and the number of explored nodes for each of the Factors n, τ, R, α, mo $\times$ mk, and "setup". From Columns 3 and 4 of Table 2, it is seen that as the numbers of job classes ("mk" in mo $\times$ mk) increase, the number of nodes explored and CPU times increase; the greater the number of job classes, the greater the number nodes explored. It can be observed that our proposed B&B was not so powerful when number of jobs increased to $n = 12$. But it was noted in Table 1 that on average, proposed B&B took less 700 s to solve an instance at cases mo $\times$ mk $= 4 \times 3$ and 6×2. It was also noted that overall, the average of CPU time B&B took was only 550.32 s.

With respect to the performances of the proposed 8 heuristics and five CSA algorithms, Table 3 exhibits their AEPs. Overall, the AEP values of the OSPTLPT, CSLPT, OEEDLPT, and OEEDSPT heuristics are (1.7420, 98.6349, 12.0186, 28.7837) [%] and their improvement counterparts (0.6931, 59.5241, 10.8500, 20.1509) [%], respectively, while the means of AEP are (0.2072, 0.3713, 0.3683, 0.3659, 0.0722) [%] for CSA1, CSA2, CSA3, CSA4 and CSAHH, respectively. The CSA algorithms perform better than any of 8 problem-based heuristics, and the CSAHH has the smallest AEP for small-sized jobs. The AEPs of OSPTLPT and OSPTLPT_pi are less than 2%.

Table 3 also summarizes the means of AEP for the levels of the Factors n, τ, R, α, mo $\times$ mk, and "setup". Table 3 shows that the AEP of all heuristics and algorithms increases slightly in general as n increases from 8 to 12. Furthermore, Figure 3 displays the boxplots (distributions) of AEP for the 8 heuristics and five CSA algorithms. The diamond in the boxplot represents the mean of the AEP for each heuristic/algorithm. Regarding the CPU times, it can be seen in Figure 4 that the group (CSA1, CSA2, CSA3, CSA4 and CSAHH 13) consumed more CPU time than other eight heuristics because metaheuristics are more complex than heuristics. However, all of proposed heuristics/algorithms executed all less than 1 s for small-sized instances.

To verify whether the difference among the 8 heuristics and five CSA algorithms are significant or not in a statistical sense, based on rank-sums of AEP for the 288 groups of problem instances, which are formed by the combinations of n, τ, R, α, mo $\times$ mk, and "setup", the Freidman test (a nonparametric method) was executed (in SAS 9.4). The value of the statistic is 3287.8 with degrees of freedom 12 and a p value less than 0.0001. According to the test results, it is concluded that the AEP samples do not follow the same distribution at the

TABLE 1. Performance of the branch-and-bound method.

n	τ	No. of nodes	CPU time	TS
8	0.25	18 894	0.07	3600/3600
	0.5	19 334	0.07	3600/3600
10	0.25	1 430 717	9.08	3600/3600
	0.5	1 476 083	9.11	3600/3600
12	0.25	133 527 286	1076.77	7200/7200
	0.5	135 495 545	1115.35	7200/7200
	R			
8	0.25	19 354	0.07	2400/2400
	0.5	19 157	0.07	2400/2400
	0.75	18 831	0.07	2400/2400
10	0.25	1 478 354	9.63	2400/2400
	0.5	1 456 622	8.83	2400/2400
	0.75	1 425 223	8.83	2400/2400
12	0.25	130 146 480	1095.39	4800/4800
	0.5	138 722 598	1118.56	4800/4800
	0.75	134 665 169	1074.22	4800/4800
	α			
8	0.25	17 640	0.06	2400/2400
	0.5	20 478	0.07	2400/2400
	0.75	19 224	0.07	2400/2400
10	0.25	1 342 542	9.32	2400/2400
	0.5	1 594 103	9.86	2400/2400
	0.75	1 423 554	8.11	2400/2400
12	0.25	116 449 836	969.66	4800/4800
	0.5	143 663 014	1208.03	4800/4800
	0.75	143 421 396	1110.49	4800/4800
	mo $\times$ mk			
8	2×4	24 713	0.08	3600/3600
	4×2	13 515	0.06	3600/3600
10	2×5	2 117 684	11.83	3600/3600
	5×2	789 116	6.37	3600/3600
12	2×6	258 202 915	1824.80	3600/3600
	3×4	137 835 469	1185.22	3600/3600
	4×3	72 524 072	697.13	3600/3600
	6×2	69 483 205	677.09	3600/3600
	setup			
8	$U(1, 10)$	18 676	0.07	3600/3600
	$U(1, 20)$	19 552	0.07	3600/3600
10	$U(1, 10)$	1 425 025	8.96	3600/3600
	$U(1, 20)$	1 481 775	9.23	3600/3600
12	$U(1, 10)$	134 148 216	1075.62	7200/7200
	$U(1, 20)$	134 874 615	1116.50	7200/7200
	Mean	67 623 836	550.32	

TABLE 2. Summary of the performance of the branch-and-bound method.

		No. of nodes	CPU time	TS
n	8	19 114	0.07	36 000/36 000
	10	1 453 400	9.10	36 000/36 000
	12	134 511 415	1096.06	72 000/72 000
τ	0.25	67 126 046	540.67	14 400/14 400
	0.5	68 121 627	559.97	14 400/14 400
R	0.25	65 447 667	550.12	9600/9600
	0.5	69 730 244	561.50	9600/9600
	0.75	67 693 598	539.34	9600/9600
α	0.25	58 564 964	487.18	9600/9600
	0.5	72 235 152	606.50	9600/9600
	0.75	72 071 392	557.29	9600/9600
mo × mk	2×4	24 713	0.08	3600/3600
	4×2	13 515	0.06	3600/3600
	2×5	2 117 684	11.83	3600/3600
	5×2	789 116	6.37	3600/3600
	2×6	258 202 915	1824.80	3600/3600
	3×4	137 835 469	1185.22	3600/3600
	4×3	72 524 072	697.13	3600/3600
	6×2	69 483 205	677.09	3600/3600
setup	$U(1, 10)$	67 435 033	540.06	14 400/14 400
	$U(1, 20)$	67 812 639	560.58	14 400/14 400
	Mean	67 623 836	550.32	

TABLE 3. AEP of Heuristics and CSA algorithms (small n).

		OSPT-LPT	CSLPT-LPT	OEDD-LPT	OEDD-SPT	OSPT-LPT_pi	CSLPT_pi	OEDD-LPT_pi	OEDD-SPT_pi	CSA1	CSA2	CSA3	CSA4	CSAHH	TS
n	8	1.4590	85.9356	11.5581	31.1748	0.4809	44.0226	10.3976	20.0274	0.0053	0.0456	0.0376	0.0379	0.0165	7200
	10	1.7264	94.2350	12.3529	29.4987	0.7017	55.3080	11.1419	20.3691	0.0998	0.1935	0.1875	0.1961	0.0534	7200
	12	1.8913	107.1845	12.0818	27.2305	0.7949	69.3830	10.9303	20.1036	0.3619	0.6230	0.6241	0.6147	0.1095	14 400
τ	0.25	1.8635	101.7910	12.1653	29.5252	0.7644	61.2038	10.9531	20.5706	0.2154	0.3825	0.3825	0.3774	0.0777	14 400
	0.5	1.6205	95.4789	11.8719	28.0421	0.6219	57.8445	10.7469	19.7313	0.1990	0.3600	0.3541	0.3543	0.0667	14 400
R	0.25	1.7133	99.9749	12.3658	29.2638	0.6444	60.4585	11.1662	20.5959	0.2094	0.3770	0.3708	0.3643	0.0715	9600
	0.5	1.7279	97.9485	12.0255	28.6861	0.6881	59.0508	10.8729	20.1001	0.2057	0.3685	0.3630	0.3661	0.0711	9600
	0.75	1.7848	97.9813	11.6645	28.4011	0.7468	59.0631	10.5110	19.7567	0.2065	0.3683	0.3711	0.3672	0.0741	9600
α	0.25	2.6273	62.5036	15.5628	22.1252	0.9753	37.2816	13.6272	17.3332	0.1652	0.1983	0.1946	0.1956	0.0613	9600
	0.5	1.6259	92.0304	12.2991	27.4461	0.6604	56.4631	11.2659	19.6578	0.2274	0.3578	0.3611	0.3512	0.0834	9600
	0.75	0.9728	141.3708	8.1938	36.7797	0.4436	84.8277	7.6569	23.4618	0.2290	0.5576	0.5492	0.5508	0.0720	9600
mo × mk	2×4	1.0413	78.3729	5.7800	25.7026	0.2495	38.7177	4.9993	13.5761	0.0009	0.0484	0.0376	0.0364	0.0038	3600
	4×2	1.8768	93.4984	17.3361	36.6470	0.7124	49.3275	15.7958	26.4788	0.0098	0.0428	0.0375	0.0394	0.0292	3600
	2×5	1.0080	81.6573	4.6033	21.8098	0.2754	48.3024	3.8746	11.6352	0.0024	0.0216	0.0137	0.0179	0.0079	3600
	5×2	2.4448	106.8128	20.1025	37.1877	1.1280	62.3137	18.4092	29.1030	0.1971	0.3653	0.3613	0.3743	0.0988	3600
	2×6	0.8404	79.0937	4.3077	18.5199	0.2088	51.7991	3.6799	10.3181	0.0103	0.0308	0.0251	0.0330	0.0049	3600
	3×4	1.6207	111.9398	8.6316	24.5086	0.6278	75.2289	7.6241	17.2751	0.2065	0.3373	0.3530	0.3359	0.0685	3600
	4×3	2.2520	125.7877	13.5737	29.8453	1.0385	80.0439	12.3236	23.1355	0.5458	0.8276	0.8254	0.8435	0.1215	3600
	6×2	2.8117	114.9336	21.4926	36.3970	1.3021	72.2327	19.8286	29.8699	0.7039	1.3381	1.3378	1.2909	0.2484	3600
setup	$U(1, 10)$	1.0799	103.9514	11.7035	29.1197	0.4294	61.8711	10.9413	20.5403	0.1620	0.3454	0.3414	0.3475	0.0448	14 400
	$U(1, 20)$	2.4041	93.3185	12.3337	28.4476	0.9568	57.1772	10.7587	19.7615	0.2524	0.3972	0.3953	0.3843	0.0997	14 400
	Mean	1.7420	98.6349	12.0186	28.7837	0.6931	59.5241	10.8500	20.1509	0.2072	0.3713	0.3683	0.3659	0.0722	

Notes. TS denotes the total number of instances solved out by each heuristic.

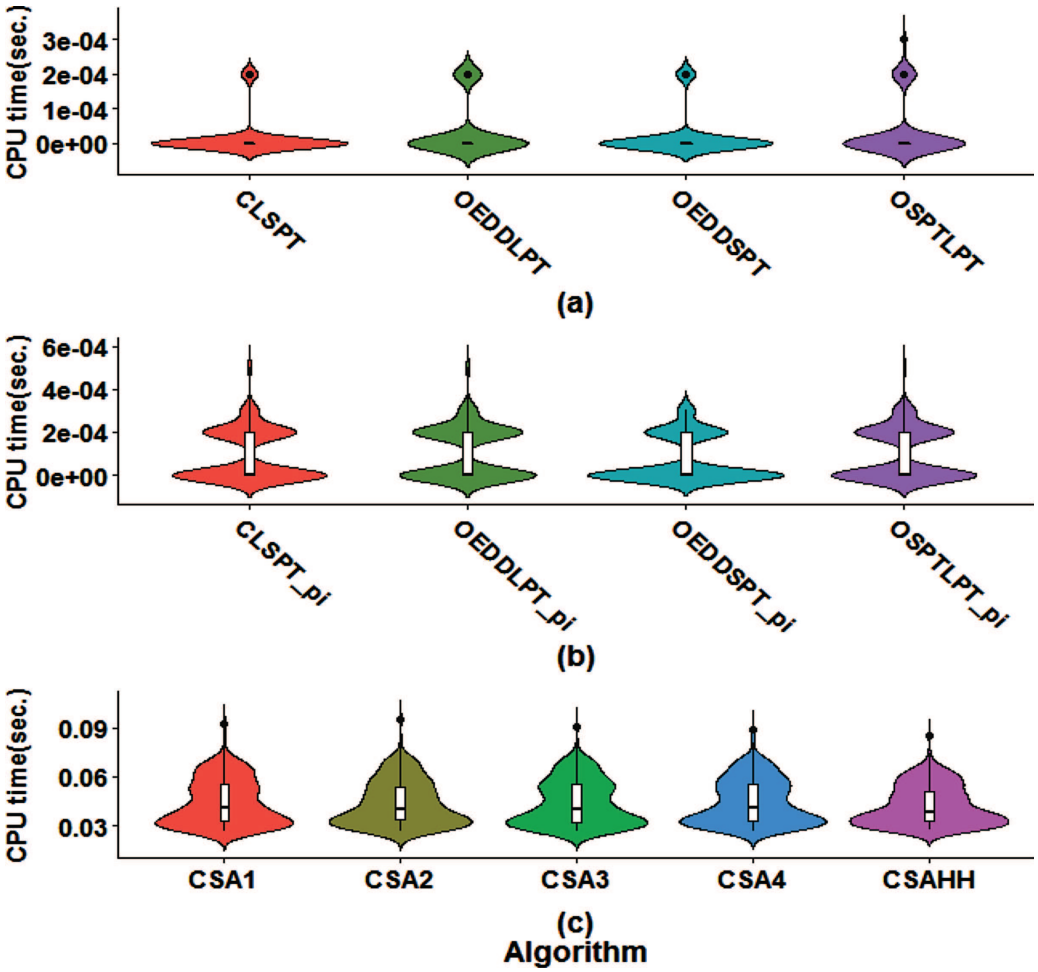

FIGURE 4. Boxplots of CPU time for heuristics and CSA algorithms (small n).

level of significance $\alpha = 0.05$. Table 4, Column 3, discloses sums of AEP ranks of the 288 instance problems for the 8 heuristics and five CSA algorithms. The rank sums of CSA1 to CSA4 and CSAHH are 623.5, 1142.5, 1102.5, 1119.5, and 552.0, respectively. The rank sums of the heuristics range from the smallest 1542.0 (OSPTLPT_pi) to the largest 3744 (CLSPT). To further analyze the performances among these heuristics and CSA algorithms, a multiple comparison procedure WNMT (Wilcoxon–Nemenyi–McDonald–Thompson, Hollander *et al.*, [27]) was utilized. There were 78 pairwise difference comparisons among the 8 heuristics and five CSA algorithms. The best performance group is CSAHH and CSA1 at a level of significance of 0.05. Any one of the CSA algorithms performs significantly better than any heuristic. The CSAHH has the smallest AEP (0.0722%) and rank-sum (552.0) for a small number of instances.

Regarding the frequencies of five low-level heuristics usage, the transmutation of probabilities of invoking them for the CSAHH are displayed in Figure 5. LH_1 was called the least, and the other four heuristics were called almost equally likely.

TABLE 4. The rank-sum of heuristics and CSA algorithms.

Heuristic/ Algorithm	No. of Obs. Small n (Big n)	Rank-sum (Ri, $i = 1, 2, \ldots, 13$)	
		Small n	Big n
OSPTLPT	288 (612)	1982.0	2646.5
OSPTLPT_pi	288 (612)	1542.0	1452.5
CSLPT	288 (612)	3744.0	7956.0
CSLPT_pi	288 (612)	3456.0	7341.5
OEDDLPT	288 (612)	2610.0	5432.0
OEDDLPT_pi	288 (612)	2304.0	4750.5
OEEDSPT	288 (612)	3168.0	6702.0
OEDDSPT_pi	288 (612)	2862.0	6037.5
CSA1	288 (612)	623.5	1758.0
CSA2	288 (612)	1142.5	4082.5
CSA3	288 (612)	1102.5	3214.5
CSA4	288 (612)	1119.5	3706.5
CSAHH	288 (612)	552.0	612.0

Notes. The critical value of the WNMT test is approximated by 309.31, 450.89 for small n and large n, respectively [27]; *i.e.*, two algorithms are different significantly on performance if $|Ri - Rj| > 309.31$ (for small n), > 450.89 (for large n).

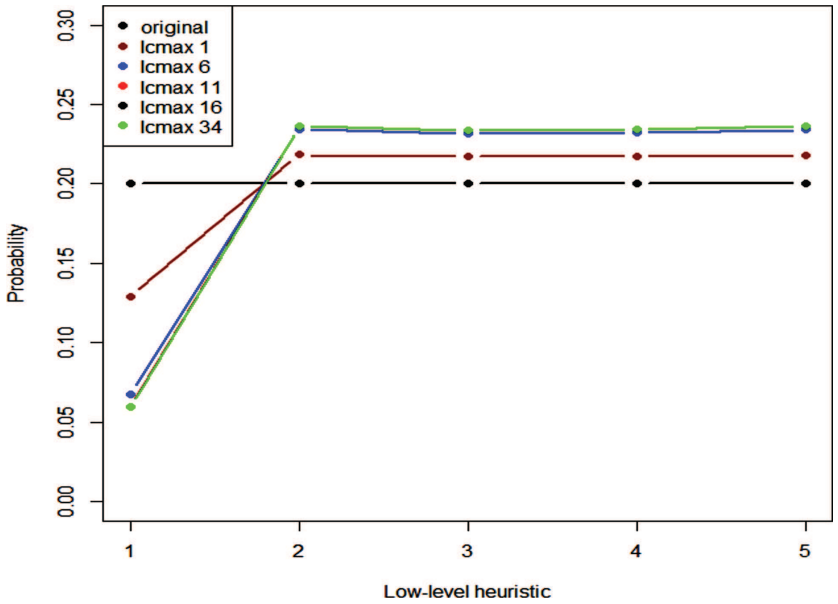

FIGURE 5. Variation of probabilities called for five low-level heuristics.

6.2. Analysis of large n experimental results

For a large-sized job instance, the optimal solution of the test instance and its optimal objective value were rarely found. Thus, an average relative percentage deviation (RPD) is reported for the performed results. RPD $= 100(\text{mean}[(H_i - B_i)/B_i])[\%]$, where H_i is the value of objective function $h(\sigma)$ for a job sequence searched by a heuristic or by a CSA algorithm and B_i is the smallest value among the values found among 13 heuristics/CSA algorithms. Regarding the performances of the proposed 8 heuristics and five CSA algorithms,

TABLE 5. RPD of Heuristics and CSA algorithms (large n).

n	τ	OSPT-LPT	CLSPT	OEDD-LPT	OEDD-SPT	OSPT-LPT_pi	CLSPT_pi	OEDD-LPT_pi	OEDD-SPT_pi	CSA1	CSA2	CSA3	CSA4	CSAHH	FS
60	0.25	2.12	171.11	12.14	16.57	1.54	159.16	11.59	15.27	1.63	4.78	4.39	4.70	0.02	18 000
	0.5	2.00	162.52	11.81	15.99	1.44	151.18	11.28	14.78	1.55	4.60	4.27	4.53	0.01	18 000
100	0.25	2.21	179.70	12.24	15.02	1.78	171.94	11.84	14.22	1.78	6.64	5.19	5.69	0.01	12 600
	0.5	2.11	172.41	11.98	14.65	1.69	164.95	11.60	13.89	1.71	6.38	5.13	5.55	0.01	12 600
	R														
60	0.25	2.07	166.77	12.08	16.38	1.50	155.13	11.54	15.13	1.59	4.70	4.34	4.63	0.01	12 000
	0.5	2.05	166.92	11.95	16.25	1.48	155.26	11.42	15.00	1.59	4.68	4.35	4.63	0.02	12 000
	0.75	2.06	166.75	11.89	16.20	1.49	155.11	11.36	14.95	1.59	4.69	4.29	4.59	0.03	12 000
100	0.25	2.14	175.97	12.12	14.84	1.72	168.36	11.73	14.05	1.75	6.51	5.17	5.63	0.01	8400
	0.5	2.15	175.92	12.13	14.85	1.73	168.31	11.73	14.06	1.73	6.57	5.16	5.61	0.01	8400
	0.75	2.18	176.27	12.09	14.82	1.75	168.65	11.70	14.03	1.76	6.45	5.14	5.61	0.01	8400
	α														
60	0.25	2.81	95.85	14.03	15.49	2.07	88.51	13.33	14.55	1.69	3.19	2.93	3.06	0.04	12 000
	0.5	2.04	148.33	12.28	15.93	1.46	138.15	11.74	14.77	1.79	4.71	4.45	4.69	0.01	12 000
	0.75	1.34	256.27	9.61	17.42	0.94	238.84	9.24	15.76	1.30	6.16	5.60	6.09	0.00	12 000
100	0.25	2.82	98.33	13.76	14.66	2.31	93.71	13.27	14.03	1.80	3.75	3.39	3.47	0.02	8400
	0.5	2.17	153.82	12.36	14.62	1.74	147.28	11.96	13.88	1.98	5.78	5.23	5.53	0.01	8400
	0.75	1.48	276.02	10.22	15.23	1.16	264.33	9.93	14.25	1.46	10.01	6.86	7.85	0.00	8400
	mo × mk														
60	2 × 30	0.28	62.25	9.59	4.20	0.64	57.23	8.85	2.78	0.42	0.94	0.55	0.74	0.09	3600
	3 × 20	0.46	108.18	3.20	6.77	0.23	100.74	2.98	5.38	0.32	1.90	1.24	1.69	0.04	3600
	4 × 15	0.66	139.65	4.60	8.71	0.37	130.92	4.31	7.46	0.54	2.66	2.00	2.49	0.03	3600
	5 × 12	0.88	160.69	5.92	10.36	0.54	151.18	5.58	9.19	0.77	3.31	2.69	3.17	0.02	3600
	6 × 10	1.14	176.54	7.33	12.00	0.75	166.30	6.94	10.88	1.02	3.80	3.33	3.83	0.02	3600
	10 × 6	2.15	210.41	12.20	17.29	1.64	198.01	11.69	16.26	1.92	5.60	5.35	5.49	0.00	3600
	12 × 5	2.68	217.13	14.43	19.52	2.10	203.83	13.87	18.50	2.34	6.19	5.96	6.28	0.00	3600
	15 × 4	3.31	218.79	17.56	22.49	2.62	204.32	16.91	21.43	2.73	6.97	6.85	7.06	0.00	3600
	20 × 3	4.12	207.57	22.33	26.88	3.15	191.20	21.44	25.64	3.08	7.62	7.59	7.56	0.00	3600
	30 × 2	4.94	155.51	30.53	34.05	3.38	137.28	29.17	32.39	3.07	7.80	7.73	7.79	0.03	3600
100	2 × 50	0.20	69.10	1.20	3.01	0.10	65.70	1.10	2.04	0.11	2.90	0.51	0.83	0.02	3600
	4 × 25	0.40	138.75	3.32	5.84	0.22	133.42	3.15	5.09	0.37	5.09	1.91	2.81	0.02	3600
	5 × 20	0.51	160.49	4.32	7.05	0.31	154.64	4.12	6.36	0.49	5.27	2.61	3.59	0.01	3600
	10 × 10	1.29	217.19	8.59	11.87	0.99	209.88	8.29	11.27	1.24	6.07	5.03	5.79	0.00	3600
	20 × 5	3.05	244.00	15.89	19.24	2.61	234.85	15.47	18.62	2.76	7.92	7.87	7.98	0.00	3600
	25 × 4	3.83	239.73	19.18	22.36	3.27	229.91	18.66	21.66	3.28	8.68	8.53	8.69	0.00	3600
	50 × 2	5.82	163.11	32.29	34.47	4.64	150.69	31.25	33.30	3.97	9.66	9.66	9.64	0.01	3600
	setup														
60	$U(1, 10)$	1.29	175.31	11.48	15.96	0.91	162.92	11.13	14.85	1.14	4.74	4.35	4.68	0.01	18 000
	$U(1, 20)$	2.83	158.32	12.47	16.60	2.07	147.42	11.75	15.20	2.04	4.64	4.30	4.55	0.03	18 000
100	$U(1, 10)$	1.36	184.74	11.66	14.49	1.08	176.67	11.40	13.82	1.25	6.67	5.24	5.80	0.00	12 600
	$U(1, 20)$	2.95	167.37	12.57	15.18	2.39	160.21	12.04	14.28	2.24	6.35	5.08	5.44	0.01	12 600
	Mean	2.10	170.62	12.03	15.68	1.59	160.63	11.55	14.62	1.65	5.44	4.67	5.03	0.01	

Table 5 presents their RPDs. Overall, the RPD values of the OSPTLPT, CSLPT, OEEDLPT, and OEEDSPT heuristics are (2.10, 170.62, 12.03, 15.68) [%] and their improvement counterparts (1.59, 160.63, 11.55, 14.62) [%], respectively, while the means of RPD are (1.65, 5.44, 4.67, 5.02, 0.01) [%] for the CSA1, CSA2, CSA3, CSA4 and CSAHH algorithms, respectively. The CSAHH algorithm performed the best according to the RPD measure, followed by OSPTLPT_pi and OSPTLPT, CSA1, CSA3, CSA4, and CSA2, and CLSPT_pi and CLSPT. It is noted that the problem-based heuristic improved with a pairwise interchange method (OSPTLSPT_pi) could be as powerful on producing (near-) optimal solutions as a metaheuristic (CSA algorithm) does.

Table 5 also summarizes the means of RPD for levels of the factors τ, R, α, mo × mk, and "setup" for $n = 60$ and 100, respectively. Table 5 shows that the RPDs of all heuristics/algorithms increase slightly in general as n increases from 60 to 100. Furthermore, Figure 6 shows the boxplots (distributions) of RPD for the 8 heuristics and five CSA algorithms.

To verify whether the difference among the 8 heuristics and five CSA algorithms are statistically significant, based on rank-sums of RPD for the 612 groups of problem instances, formed by the combinations of n, τ, R, α,

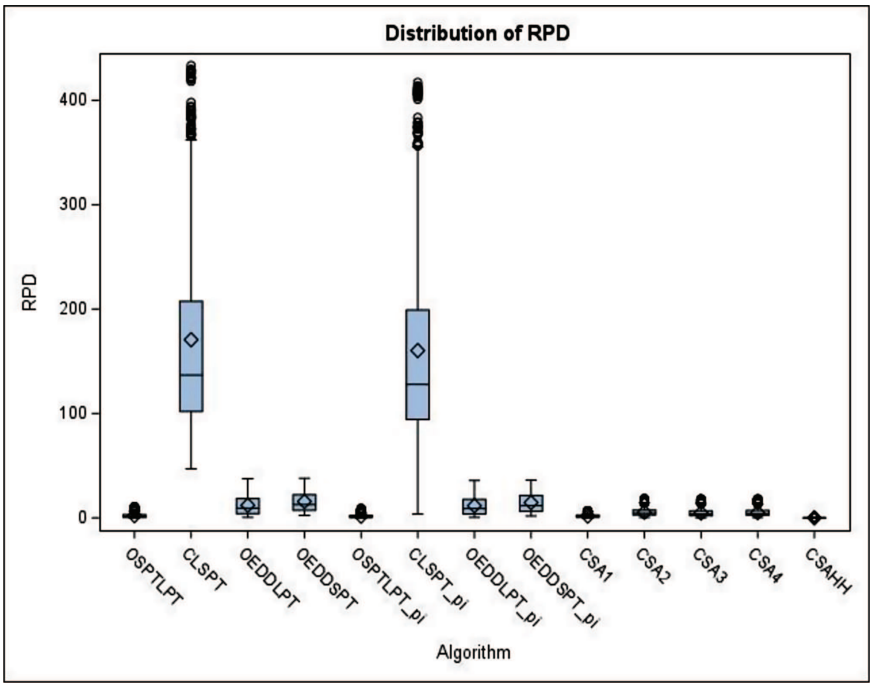

FIGURE 6. Boxplots of RPD for heuristics and CSA algorithms (large n).

mo $\times$ mk, and "setup", the Freidman test was executed (in SAS 9.4). The value of the statistic is 7039.8 with degrees of freedom 12, and the p value is less than 0.0001. Based on the test results, it is concluded that the RPD samples do not follow the same distribution at the level of significance $\alpha = 0.05$. Table 4 (Column 4) reveals sums of RPD ranks of the 612 sets of problem instances for the 8 heuristics and five CSA algorithms. The rank sums of the CSA1 to CSA4 and CSAHH algorithms are 1758.0, 4082.5, 3214.5, 3706.5, and 612.0, respectively. The rank sums of the heuristics for OSPTLPT_pi and OSPTLPT are 1452.5 and 2646.5, respectively, and those for the other heuristics range from 4750.5 (OEDDLPT_pi) to 7956.0 (CLSPT). It is observed that the heuristics OSPTLPT_pi performed better than that of any one of four CSA algorithms but worse than that of CSAHH for the large-sized job cases. To further analyze the performances among these heuristics and CSA algorithms, a multiple comparison procedure, WNMT, was used. There were 78 pairwise difference comparisons among the 8 heuristics and five CSA algorithms. The best-performing group is CSAHH, and OSPTLPT_pi and CSA1 are in the second-best group at a level of significance of 0.05. The phenomenon that the performance of OSPTLPT_pi performed better than that of CSA1 to CSA4 in this case reflects the advantage of Gupta's method [26], especially for a large number of jobs. Regardless, the CSAHH has the smallest RPD (0.01%) and rank-sum (612.0) for a large number of jobs.

Pertaining to the computational time or CPU times (in sec.) cost on searching near-optimal solutions, Figure 7 exhibits the violin plots of the CPU times of the heuristics and CSA algorithms. The differences between CPU times within groups are no more than 0.01, 0.001, and 0.00001 (s) for CSA algorithms, four heuristics-pi, and four heuristics, respectively. On average, the CSA algorithms, the four heuristics with the pi-method, and the four heuristics spent approximately 0.66, 0.03, and 0.0001 (s), respectively.

6.3. The performance of lower bound at small-n and large-n experimental results

In order further to evaluate the performance of the lower bound at small-n and large-n experimental results, we tested some additional problem instances at $n = 8$, 10, 12 for the small-n case and at $n = 60$, 100, 200

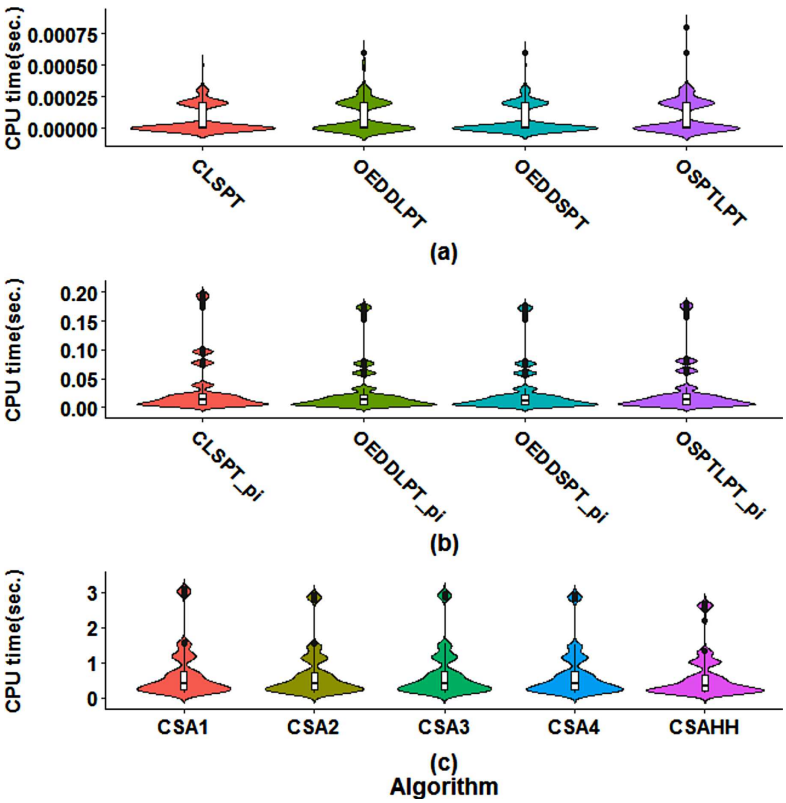

FIGURE 7. Violin plots of RPD for heuristics and CSA algorithms (large n).

TABLE 6. The performances of approximate methods and two lower bounds for small-n case.

n	α	mo*mk	τ	R	OSPTLPT Mean	CLSPT Mean	OEDDLPT Mean	OEDDSPT Mean	OSPTLPT_pi Mean	CLSPT_pi Mean
8	0.5	2*4	0.25	0.25	0.5429 (45)	88.8828 (0)	6.7286 (31)	28.5399 (0)	0.1065 (84)	42.3612 (84)
				0.5	0.6375 (34)	85.5729 (0)	6.0941 (16)	26.7472 (0)	0.1303 (83)	42.5876 (83)
				0.75	0.7834 (37)	86.2642 (0)	5.0580 (22)	27.0252 (0)	0.1929 (80)	41.4393 (80)
			0.5	0.25	0.5475 (41)	73.2011 (0)	5.4434 (15)	23.5606 (0)	0.0702 (91)	36.1633 (91)
				0.5	0.4879 (40)	75.4064 (0)	4.3950 (23)	22.4981 (0)	0.0444 (88)	37.4916 (88)
				0.75	0.6135 (36)	76.5525 (0)	4.5219 (21)	23.8052 (0)	0.1045 (84)	38.2491 (84)
10	0.5	2*5	0.25	0.25	0.6343 (28)	88.3873 (0)	5.8571 (15)	24.4361 (0)	0.0949 (85)	52.4996 (85)
				0.5	0.5631 (29)	86.9818 (0)	4.4598 (12)	21.9841 (0)	0.1221 (79)	52.1767 (79)
				0.75	0.5918 (34)	87.4039 (0)	3.7394 (22)	21.7921 (0)	0.1749 (80)	50.4236 (80)
			0.5	0.25	0.5518 (26)	75.0263 (0)	5.2612 (15)	20.7661 (0)	0.0694 (85)	45.9458 (85)
				0.5	0.5277 (32)	75.9971 (0)	5.4949 (12)	20.7852 (0)	0.0945 (83)	45.6306 (83)
				0.75	0.5370 (29)	74.7149 (0)	3.1537 (19)	18.2282 (0)	0.0630 (83)	45.2114 (83)
12	0.5	3*4	0.25	0.25	1.0283 (12)	118.6430 (0)	9.2353 (1)	25.4520 (0)	0.2685 (43)	80.7197 (43)
				0.5	1.1884 (15)	118.2541 (0)	8.7864 (2)	24.3121 (0)	0.4776 (43)	80.5796 (43)
				0.75	1.3163 (12)	119.0570 (0)	7.4413 (3)	23.6771 (0)	0.6750 (37)	80.3646 (37)
			0.5	0.25	0.8148 (12)	103.1482 (0)	8.3204 (4)	22.9192 (0)	0.2419 (53)	70.1568 (53)
				0.5	0.9771 (10)	105.8336 (0)	8.6112 (1)	23.6484 (0)	0.2658 (50)	70.7987 (50)
				0.75	0.8644 (15)	103.7016 (0)	7.8458 (3)	21.2930 (0)	0.2916 (38)	70.8135 (38)
			Total mean		0.7338	91.4203	6.1360	23.4150	0.1938	54.6452

TABLE 6. continued.

					OEDDLPT_pi	OEDDSPT_pi	CSA1	CSA2	CSA3	CSA4	CSAHH	BBLB1		BBLB2	
n	α	mo*mk	τ	R	Mean	Mean	Mean	Mean	Mean	Mean	Mean	Mean	Max	Mean	Max
8	0.5	2*4	0.25	0.25	6.2830 (47)	15.1927 (0)	**0.0000 (100)**	0.0222 (99)	0.0129 (99)	**0.0000 (100)**	0.0015 (99)	87.6228	96.1487	79.4772	92.8302
				0.5	5.6445 (42)	14.1159 (0)	**0.0000 (100)**	0.0050 (99)	**0.0000 (100)**	0.0281 (99)	**0.0000 (100)**	87.6228	96.1487	79.9772	92.4113
				0.75	4.5133 (47)	13.9273 (0)	**0.0000 (100)**	0.0281 (99)	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	87.4671	96.5728	79.2498	92.2321
			0.5	0.25	4.9297 (45)	12.5945 (0)	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	0.0117 (99)	89.4841	95.9008	82.4087	91.8015
				0.5	3.9247 (53)	11.5258 (0)	**0.0000 (100)**	**0.0000 (100)**	0.0095 (99)	**0.0000 (100)**	**0.0000 (100)**	88.9547	97.5214	82.2500	92.0372
				0.75	4.0482 (45)	12.4893 (0)	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	87.9592	96.7181	81.3294	92.6641
10	0.5	2*5	0.25	0.25	5.3638 (35)	13.6516 (0)	0.0037 (99)	**0.0000 (100)**	**0.0000 (100)**	0.0099 (99)	**0.0000 (100)**	89.6685	94.9858	81.9942	90.8235
				0.5	4.0505 (42)	11.9731 (0)	**0.0000 (100)**	0.0064 (98)	**0.0000 (100)**	0.0077 (98)	0.0114 (97)	90.4940	96.8927	83.4147	92.7928
				0.75	3.3695 (53)	11.3293 (0)	**0.0000 (100)**	**0.0000 (100)**	0.0017 (99)	0.0145 (99)	0.0033 (99)	90.3484	96.3603	83.0535	91.4468
			0.5	0.25	4.8516 (39)	11.3825 (0)	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	91.1561	97.3650	84.8346	93.5090
				0.5	5.0428 (40)	11.8828 (0)	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	91.3585	95.9970	85.0787	91.9269
				0.75	2.7022 (53)	9.4368 (0)	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	**0.0000 (100)**	90.6788	97.1270	84.6974	93.4996
12	0.5	3*4	0.25	0.25	8.5310 (6)	18.3581 (0)	0.1910 (71)	0.3812 (56)	0.3248 (61)	0.2089 (72)	**0.0092 (97)**	93.6377	97.7761	89.8419	94.8620
				0.5	8.0889 (7)	17.3148 (0)	0.1929 (71)	0.2824 (35)	0.2580 (71)	0.3282 (66)	**0.0840 (86)**	93.8367	97.6942	89.8456	95.3884
				0.75	6.7980 (11)	16.7006 (0)	0.1982 (74)	0.3754 (60)	0.2247 (68)	0.1926 (73)	**0.1060 (88)**	93.6367	97.2342	89.8376	94.4223
			0.5	0.25	7.7716 (12)	16.4848 (0)	0.1725 (62)	0.2466 (64)	0.2894 (61)	0.2633 (71)	**0.0269 (93)**	94.0859	97.2652	90.6010	94.5761
				0.5	7.9954 (9)	17.2473 (0)	0.1567 (76)	0.3415 (69)	0.3230 (69)	0.3596 (63)	**0.0465 (92)**	93.7056	97.0625	90.0952	94.6875
				0.75	7.1763 (10)	15.4370 (0)	0.0961 (74)	0.3168 (59)	0.2649 (69)	0.3991 (57)	**0.0349 (91)**	93.2371	98.0105	89.8357	94.8733
			Total mean		5.6158	13.9469	0.0562	0.1114	0.0949	0.1007	0.0186	90.8308		84.8790	

Notes. Note that (•) denotes total number of optimal solutions obtained from each approximate method.

TABLE 7. The performances of approximate methods with respective to the lower bound LB1 for big-n case.

					OSPTLPT	CLSPT	OEDDLPT	OEDDSPT	OSPTLPT_pi	CLSPT_pi
n	α	mo*mk	τ	R	Gap ratio	Gap ratio	Gap ratio	Gap ratio	Gap ratio	Gap ratio
60	0.5	10*6	0.25	0.25	70.99	399.98	89.59	97.19	70.33	379.95
				0.5	70.27	404.01	89.63	97.18	69.62	383.76
				0.75	72.45	402.42	90.18	97.62	71.79	382.54
			0.5	0.25	65.24	373.05	81.90	88.77	64.64	354.71
				0.5	66.27	371.34	82.75	89.66	65.69	353.03
				0.75	68.43	378.52	81.40	88.65	67.82	359.86
100	0.5	10*10	0.25	0.25	77.26	437.95	90.98	96.01	76.87	425.67
				0.5	77.75	439.63	92.64	97.70	77.35	427.17
				0.75	77.80	441.36	91.79	96.87	77.40	428.70
			0.5	0.25	73.59	412.58	87.63	92.39	73.21	400.90
				0.5	74.11	412.90	87.84	92.64	73.73	401.06
				0.75	74.13	414.32	86.53	91.31	73.78	402.38
200	0.5	20*10	0.25	0.25	65.74	442.47	79.73	82.22	65.50	436.10
				0.5	66.61	445.16	81.22	83.74	66.37	438.61
				0.75	65.88	445.13	80.07	82.58	65.64	438.73
			0.5	0.25	63.73	428.47	77.82	80.24	63.51	422.25
				0.5	62.75	427.29	77.32	79.73	62.53	421.07
				0.75	64.00	428.02	77.68	80.05	63.78	421.80
			Total mean		69.83	416.92	84.82	89.70	69.42	404.35

for large-n case, respectively. Other settings are included $\alpha = 0.5$, six combinations of (τ, R): $(0.25, 0.25)$, $(0.25, 0.50)$, $(0.25, 0.75)$, $(0.50, 0.25)$, $(0.25, 0.50)$, and $(0.25, 0.75)$, setup times was generated from uniformly distribution $U(1, 10)$, number of classes (mo $\times$ mk) is only one case for each n, and a set of 100 problem instances were generated randomly for each case. For the small-n case, we reported AEPs for the performance of each algorithm, the number of the optimal solutions (in (•)) obtained from each algorithm obtained, and the mean or maximum of gap ratio of the optimal solution obtained from the B&B with respect to the lower bound (LB1 or LB2) at the root node $(100(\text{mean}[(\text{OB}_i - lb(v)_i)/\text{OB}_i])[\%]$, $v = 1, 2$. For the large-n case, we reported the average gap ratio of proposed approximate methods to the lower bound LB1 $(100(\text{mean}[(H_i - lb(1)_i)/lb1_i])[\%])$, where OB_i is obtained from the B&B method, $lb(v)_i$ is the value of the lower bound 1 and lower bound 2 at the root node from each instance, and H_i is the value of objective function $h(\sigma)$ for a job sequence searched by a heuristic. To avoid too much computational time, as the number of nodes searched exceeded 10^9, the B&B

TABLE 7. continued.

n	α	mo*mk	τ	R	OEDDLPT_pi Gap ratio	OEDDSPT_pi Gap ratio	CSA1 Gap ratio	CSA2 Gap ratio	CSA3 Gap ratio	CSA4 Gap ratio	CSAHH Gap ratio
60	0.5	10*6	0.25	0.25	88.95	95.84	70.96	78.41	78.26	78.50	68.52
				0.5	88.99	95.90	70.24	78.40	78.37	78.37	67.89
				0.75	89.55	96.28	72.41	79.97	80.19	79.64	69.86
			0.5	0.25	81.29	87.54	65.23	72.12	73.00	73.49	63.06
				0.5	82.15	88.49	66.26	73.41	72.67	73.39	64.05
				0.75	80.79	87.36	68.39	75.55	75.23	76.04	64.82
100	0.5	10*10	0.25	0.25	90.58	95.20	77.25	86.32	85.63	86.40	75.85
				0.5	92.24	96.89	77.75	87.06	86.17	86.90	76.30
				0.75	91.39	96.08	77.80	86.45	85.69	86.42	76.36
			0.5	0.25	87.25	91.62	73.59	81.94	81.27	82.03	72.20
				0.5	87.46	91.85	74.11	83.06	82.20	83.02	72.72
				0.75	86.16	90.58	74.13	82.32	82.03	82.57	72.81
200	0.5	20*10	0.25	0.25	79.50	81.89	65.74	80.59	78.34	78.90	64.29
				0.5	80.98	83.40	66.61	82.29	79.42	80.62	65.09
				0.75	79.84	82.25	65.88	80.85	78.30	79.36	64.36
			0.5	0.25	77.59	79.93	63.73	78.86	76.14	77.28	62.34
				0.5	77.09	79.41	62.75	77.72	75.59	76.41	61.41
				0.75	77.46	79.74	64.00	78.20	75.74	77.17	62.61
			Total mean		84.40	88.90	69.82	80.20	79.12	79.81	68.03

method jumped to the next set of problem instances. Thus, in total, 1800 ($= n \times \tau \times R \times \alpha \times (\text{mo} \times \text{mk}) \times \text{setup} = 3 \times 2 \times 3 \times 1 \times 1 \times 1 \times 100$) instances were tested for the small-n case and for the large-n case, respectively. To avoid too much computational time, as the number of nodes searched exceeded 10^9, the B&B method jumped to the next set of problem instances. The results were reported in Table 6 for the small-n case and in Table 7 for the large-n case, respectively.

It can be observed in Table 6 that on average, the mean of AEPs or the numbers of the optimal solutions obtained from each algorithm of the group (CSA1, CSA2, CSA3, CSA4, CSAHH) were better than those of the group (OSPTLPT_pi, CSLPT_pi, OEEDLPT_pi, OEEDSPT_pi), and while the group (OSPTLPT_pi, CSLPT_pi, OEEDLPT_pi, OEEDSPT_pi) were better that those of the group (OSPTLPT, CSLPT, OEEDLPT, OEEDSPT). The last four columns in Table 6 reported that on average, the difference was only 6% between the improved lower bound LB2 and the original lower bound LB1. Regarding to the performances of proposed approximate methods to the lower bound LB1, it can be seen in Table 7 that OSPTLPT was the best than other three simple heuristics (CSLPT, OEEDLPT, OEEDSPT); OSPTLPT_pi was the best that other three improved heuristics (CSLPT_pi, OEEDLPT_pi, OEEDSPT_pi); and CSA1 was the best than other three metaheuristics (CSA2, CSA3, CSA4). It can also be observed that OSPTLPT or OSPTLPT_pi did better than the metaheuristics (CSA2, CSA3, CSA4). The reason was that the performance of CSA may be affected by the initial heuristic or seed. Table 7 also reported that CSAHH with the smallest gap ratio did the best among all proposed 13 approximate methods. It is also noted that the required CPU time of the above lower bound is less than one second.

7. CONCLUSIONS

In this study, we discuss the customer order scheduling problem with jobs belonging to different classes. To balance the costs incurred by holding the finished products and total tardiness, the optimality goal is to find a job sequence that minimizes a linear combination of the holding cost and total tardiness objective. Embedded with a dominance property, a branch-and-bound method is adopted to produce solutions for a small number of jobs; it performs well up to $n = 12$ jobs, which consist of different combinations of the number of job classes and the number of customers. For a relatively large number of jobs ($n = 60, 100$), four problem-based heuristics, each coupled with a local (pairwise interchange) iteratively improved searching method, a cloud simulated annealing

(CSA) algorithm with four different seeds (found from four heuristics), and a CSAHH algorithm were proposed for finding near-optimal solutions. It is found that the CSAHH performed best among all 8 heuristics and five CSA algorithms at the number of jobs $n = 60$ and 100. This study also observed that a well-designed problem-based heuristic improved with a simple pi-method (OSPTLSPT_pi) could be as powerful on producing (near-) optimal solutions (sequences of jobs) as a metaheuristic (CSA algorithm).

Further research can consider extending the single-machine processing environment to a two-stage production process scenario with multiple job classes and customer orders. The first stage might be with multiple machines in parallel (identical, uniform or unrelated) or in flow-shop, and the second stage, an assembly production line.

Acknowledgements. Authors thank the editor and two referees for their useful and valuable suggestions and comments. This work was supported in part by the Ministry of Science and Technology of Taiwan, MOST 110-2221-E-035-082-MY2; and in part by Chongqing Municipal Science and Technology Commission of Natural Science Fund Projects [cstc2021jcyj-msxmX0229] and the Chongqing Education Commission Natural Science Fund key Projects [KJZD-K202000501].

Conflicts of interest. It is declared that there are no conflicts of interest in this study for any of the authors.

Code availability. The corresponding author will provide the Fortran programming codes upon request.

Availability of data and material. The corresponding author will provide the relevant datasets upon request.

References

[1] B.H. Ahn and J.H. Hyun, Single facility multi-class job scheduling. *Comput. Oper. Res.* **17** (1990) 265–272.

[2] M. Alimian, V. Ghezavati, R. Tavakkoli-Moghaddam and R. Ramezanian, Solving a parallel-line capacitated lot-sizing and scheduling problem with sequence-dependent setup time/cost and preventive maintenance by a rolling horizon method. *Comput. Ind. Eng.* **168** (2022) 108041.

[3] A. Allahverdi, The third comprehensive survey on scheduling problems with setup times/costs. *Eur. J. Oper. Res.* **246** (2015) 345–378.

[4] A. Allahverdi, A survey of scheduling problems with uncertain interval/bounded processing/setup times. *J. Project Manage.* **7** (2022) 255–264.

[5] A. Allahverdi and H.M. Soroush, The significance of reducing setup times/setup costs. *Eur. J. Oper. Res.* **187** (2008) 978–984.

[6] A. Allahverdi, J.N.D. Gupta and T. Aldowaisan, A review of scheduling research involving setup considerations. *Omega* **27** (1999) 219–239.

[7] A. Allahverdi, C.T. Ng, T.C.E. Cheng and M.Y. Kovalyov, A survey of scheduling problems with setup times or costs. *Eur. J. Oper. Res.* **187** (2008) 985–1032.

[8] A. Allahverdi, H. Aydilek and A. Aydilek, No-wait flowshop scheduling problem with separate setup times to minimize total tardiness subject to makespan. *Appl. Math. Comput.* **365** (2020) 124688.

[9] K. Allali, S. Aqil and J. Belabid, Distributed no-wait flow shop problem with sequence dependent setup time: optimization of makespan and maximum tardiness. *Simul. Modell. Pract. Theory* **116** (2022) 10245.

[10] K.P. Anagnostopoulos and G.K. Koulinas, A simulated annealing hyperheuristic for construction resource levelling. *Const. Manage. Econ.* **28** (2010) 163–175.

[11] M.P. Antonioli, C.D. Rodrigues and B. de Athayde Prata, Minimizing total tardiness for the order scheduling problem with sequence-dependent setup times using hybrid matheuristics. *Int. J. Ind. Eng. Comput.* **13** (2022) 223–236.

[12] D.D. Bedworth and J.E. Bailey, Integrated Production Control Systems: Management, Analysis, Design, 2nd edition. New York, John Wiley & Sons (1987).

[13] J.D. Blocher, D. Chhajed and M. Leung, Customer order scheduling in a general job shop environment. *Dec. Sci.* **29** (1998) 951–981.

[14] E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan and R. Qu, Hyper-heuristics: a survey of the state of the art. *J. Oper. Res. Soc.* **64** (2013) 1695–1724.

[15] T.C.E. Cheng, J.N.D. Gupta and G. Wang, A review of flowshop scheduling research with setup times. *Prod. Oper. Manage.* **9** (2000) 262–282.

[16] P. Dileepan and T. Sen, Bicriterion static scheduling research for a single machine. *Omega* **16** (1988) 53–59.

[17] K.A. Dowsland, E. Soubeiga and E. Burke, A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation. *Eur. J. Oper. Res.* **179** (2007) 759–774.

[18] E. Erel and J.B. Ghosh, Customer order scheduling on a single machine with family setup times: complexity and algorithms. *Appl. Math. Comput.* **185** (2007) 11–18.

[19] M.L. Fisher, A dual algorithm for the one-machine scheduling problem. *Math. Program.* **11** (1976) 229–251.

[20] J.M. Framinan and P. Perez-Gonzalez, Order scheduling with tardiness objective: improved approximate solutions. *Eur. J. Oper. Res.* **266** (2018) 840–850.

[21] J.M. Framinan, P. Perez-Gonzalez and V. Fernandez-Viagas, Deterministic assembly scheduling problems: a review and classification of concurrent-type scheduling models and solution procedures. *Eur. J. Oper. Res.* **273** (2019) 401–417.
[22] T.D. Fry, R.D. Armstrong and H. Lewis, A framework for single machine multiple objective sequencing research. *Omega* **17** (1989) 595–607.
[23] J. Gascón-Moreno, S. Salcedo-Sanz, B. Saavedra-Moreno, L. Carro-Calvo and A. Portilla-Figueras, An evolutionary-based hyper-heuristics approach for optimal construction of group method of data handling networks. *Inf. Sci.* **247** (2013) 94–108.
[24] J.N.D. Gupta, Optimal schedules for single facility with two job classes. *Comput. Oper. Res.* **11** (1984) 409–413.
[25] J.N.D. Gupta, Single facility scheduling with multiple job classes. *Eur. J. Oper. Res.* **8** (1988) 42–45.
[26] J.N.D. Gupta, J.C. Ho and J.A.A. van der Veen, Single machine hierarchical scheduling with customer orders and multiple job classes. *Ann. Oper. Res.* **70** (1997) 127–143.
[27] M. Hollander, D.A. Wolfe and E. Chicken, Nonparametric Statistical Methods, 3rd edition. John Wiley & Sons, Hoboken, NJ, USA (2014).
[28] S.Y. Hsu and C.H. Liu, Improving the delivery efficiency of the customer order scheduling problem in a job shop. *Comput. Ind. Eng.* **57** (2009) 856–866.
[29] A. Janiak, M.Y. Kovalyov and M.C. Portmann, Single machine group scheduling with resource dependent setup and processing times. *Eur. J. Oper. Res.* **162** (2005) 112–121.
[30] S.C. Kim and P.M. Bobrowski, Scheduling jobs with uncertain setup times and sequence dependency. *Omega* **25** (1997) 437–447.
[31] I.S. Lee, Minimizing total tardiness for the order scheduling problem. *Int. J. Prod. Econ.* **144** (2013) 128–134.
[32] J.Y.T. Leung, C.Y. Lee, C.W. Ng and G.H. Young, Preemptive multiprocessor order scheduling to minimize total weighted flowtime. *Eur. J. Oper. Res.* **190** (2008) 40–51.
[33] M.M. Liaee and H. Emmons, Scheduling families of jobs with setup times. *Int. J. Prod. Econ.* **51** (1997) 165–176.
[34] C.J. Liao, Tradeoff between setup times and carrying costs for finished items. *Comput. Oper. Res.* **20** (1993) 697–705.
[35] B.M.T. Lin and A.V. Kononov, Customer order scheduling to minimize the number of tardy jobs. *Eur. J. Oper. Res.* **183** (2007) 944–948.
[36] B.M.T. Lin, P.Y. Yin and Y.S. Liu, Sequence-dependent scheduling with order deliveries. *Appl. Math. Comput.* **222** (2013) 58–71.
[37] C.H. Liu, Lot streaming for customer order scheduling problem in job shop environments. *Int. J. Comput. Integr. Manuf.* **22** (2009) 890–907.
[38] C.H. Liu, A coordinated scheduling system for customer orders scheduling problem in job shop environments. *Expert Syst. App.* **37** (2010) 7831–7837.
[39] A.J. Mason and E.J. Anderson, Minimizing flow times on a single machine with job classes and setup times. *Nav. Res. Logistics* **38** (1991) 333–350.
[40] C.L. Monma and C.N. Potts, On the complexity of scheduling with batch setup times. *Oper. Res.* **37** (1989) 798–804.
[41] D.C. Montgomery, Design and Analysis of Experiments, 5th edition. John Wiley & Sons, Inc., New York, NY, USA (2001).
[42] S. Muştu and T. Eren, The single machine scheduling problem with setup times under an extension of the general learning and forgetting effects. *Optim. Lett.* **15** (2021) 1327–1343.
[43] M. Pinedo, Scheduling: Theory, Algorithms, and Systems. Prentice Hall, Upper Saddle River, NJ, USA (2002).
[44] C.N. Potts, Scheduling two job classes on a single machine. *Comput. Oper. Res.* **18** (1991) 411–415.
[45] C.N. Potts and L.N. van Wassenhove, Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity. *J. Oper. Res. Soc.* **43** (1992) 395–406.
[46] H.N. Psaraftis, A dynamic programming approach for sequencing groups of identical jobs. *Oper. Res.* **28** (1980) 1347–1359.
[47] A.P. Rifai, S.T.W. Mara and A. Sudiarso, Multi-objective distributed reentrant permutation flow shop scheduling with sequence-dependent setup time. *Expert Syst. App.* **183** (2021) 115339.
[48] Z. Shi, L. Wang, P. Liu and L. Shi, Minimizing completion time for order scheduling: formulation and heuristic algorithm. *IEEE Trans. Autom. Sci. Eng.* **14** (2017) 1558–1569.
[49] Z. Shi, Z. Huang and L. Shi, Customer order scheduling on batch processing machines with incompatible job families. *Int. J. Prod. Res.* **56** (2018) 795–808.
[50] S.J. Shyu, B.M.T. Lin and P.Y. Yin, Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. *Comput. Ind. Eng.* **47** (2004) 181–193.
[51] A. Singh, On the intractability of preemptive single-machine job scheduling with release times, deadlines, and family setup times. *Inf. Process. Lett.* **179** (2023) 106305.
[52] T. Suma and R. Murugesan, Mathematical model and heuristic based subtask scheduling algorithm for order scheduling, in AIP Conference Proceedings 2095. Vol. 2095. AIP Publishing LLC (2019) 030026.
[53] E. Torabzadeh and M. Zandieh, Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop. *Adv. Eng. Softw.* **41** (2010) 1238–1243.
[54] J.A.A. van der Veen and S. Zhang, Low-complexity algorithms for sequencing jobs with a fixed number of job-classes. *Comput. Oper. Res.* **23** (1996) 1059–1067.
[55] C.C. Wu, W.-H. Wu, W.-H. Wu, P.-H. Hsu, Y. Yin and J. Xu, A single-machine scheduling with a truncated linear deterioration and ready times. *Inf. Sci.* **256** (2014) 109–125.
[56] C.C. Wu, W.C. Lin, X. Zhang, I.H. Chung, T.H. Yang and K. Lai, Tardiness minimisation for a customer order scheduling problem with sum-of-processing-time based learning effect. *J. Oper. Res. Soc.* (2018a) 1476–9360.

[57] C.C. Wu, S.C. Liu, T.Y. Lin, T.H. Yang, I.H. Chung and W.C. Lin, Bicriterion total flowtime and maximum tardiness minimization for an order scheduling problem. *Comput. Ind. Eng.* **117** (2018b) 152–163.
[58] C.-C. Wu, J.N.D. Gupta, S.R. Cheng, B.M.T. Lin, S.H. Yip and W.-C. Lin, Robust scheduling of a two-stage assembly shop with scenario-dependent processing times. *Int. J. Prod. Res.* **59** (2021) 5372–5387.
[59] X. Xu, Y. Ma, Z. Zhou and Y. Zhao, Customer order scheduling on unrelated parallel machines to minimize total completion time. *IEEE Trans. Autom. Sci. Eng.* **12** (2015) 244–257.
[60] J. Xu, C.-C. Wu, Y. Yin, C.L. Zhao, Y.T. Chiou and W.-C. Lin, An order scheduling problem with position-based learning effect. *Comput. Oper. Res.* **74** (2016) 175–186.
[61] J. Yang, Customer order scheduling in a two machine flowshop. *Int. J. Manage. Sci.* **17** (2011) 95–116.
[62] W.H. Yang and C.J. Liao, Survey of scheduling research involving setup times. *Int. J. Syst. Sci.* **30** (1999) 143–155.
[63] K.C. Ying, P. Pourhejazy, C.Y. Cheng and R.S. Syu, Supply chain-oriented permutation flowshop scheduling considering flexible assembly and setup times. *Int. J. Prod. Res.* **61** (2023) 258–281.
[64] Y. Zhao, X. Xu, H. Li and Y. Liu, Stochastic customer order scheduling with setup times to minimize expected cycle time. *Int. J. Prod. Res.* **56** (2018) 2684–2706.
[65] Y. Zhao, L. Leng and C. Zhang, A novel framework of hyper-heuristic approach and its application in location-routing problem with simultaneous pickup and delivery. *Oper. Res.* **21** (2021) 1299–1332.