Article

Robust Scheduling of Two-Agent Customer Orders with Scenario-Dependent Component Processing Times and Release Dates

Chin-Chia Wu [1], Jatinder N. D. Gupta [2], Win-Chin Lin [1], Shuenn-Ren Cheng [3], Yen-Lin Chiu [1], Juin-Han Chen [4] and Long-Yuan Lee [5,*]

1 Department of Statistics, Feng Chia University, Taichung 40724, Taiwan; cchwu@fcu.edu.tw (C.-C.W.); m0905335@o365.fcu.edu.tw (W.-C.L.); linwc@fcu.edu.tw (Y.-L.C.)
2 College of Business, University of Alabama in Huntsville, Huntsville, AL 35899, USA; guptaj@uah.edu
3 Department of Esports Technology Management, Cheng Shiu University, Kaohsiung 83347, Taiwan; k0252@gcloud.csu.edu.tw
4 Department of Industrial Engineering & Management, Cheng Shiu University, Kaohsiung 83347, Taiwan; k0411@gcloud.csu.edu.tw
5 Department of Leisure and Sport Management, Cheng Shiu University, Kaohsiung 83347, Taiwan
* Correspondence: k0595@gcloud.csu.edu.tw

Abstract: Although some uncertainty factors can occur in many practical environments, customer order scheduling problems involving two agents in such uncertain environments have not received attention in the current literature. Motivated by this observation, we address a two-agent customer order scheduling problem where various customer orders have scenario-dependent component processing times and release dates in order to find an appropriate schedule to minimize the maximum of the total completion time of the customer orders that belong to one agent and are subject to a constraint with the other agent. In order to solve this problem, a lower bound and six dominant properties are derived and used to propose a branch-and-bound algorithm to find an exact optimal solution. Afterward, three local search heuristics and two variants of a simulated annealing hyper-heuristic are proposed and empirically evaluated in order to find approximate solutions. Finally, we conclude the paper with a summary of our findings and some directions for future research.

Keywords: customer order scheduling; two agents; scenario-dependent component processing times; release dates; simulated annealing; hyper-heuristic

MSC: 90B35; 68M20

check for updates

Citation: Wu, C.-C.; Gupta, J.N.D.; Lin, W.-C.; Cheng, S.-R.; Chiu, Y.-L.; Chen, J.-H.; Lee, L.-Y. Robust Scheduling of Two-Agent Customer Orders with Scenario-Dependent Component Processing Times and Release Dates. *Mathematics* **2022**, *10*, 1545. https://doi.org/10.3390/math10091545

Academic Editor: Ripon Kumar Chakrabortty

Received: 30 March 2022
Accepted: 2 May 2022
Published: 4 May 2022

1. Introduction

Recent calls to enhance the practical relevance of scheduling research has resulted in the consideration of the following three problem types: *Customer Order Scheduling Problems* (COSPs), first introduced in the studies by Ahmadi and Bagchi [1,2] and Gupta et al. [3]; *multi-agent scheduling problems* (MASPs), first defined by Gupta et al. [3], Baker and Smith [4], Wu et al. [5], and Agnetis et al. [6]; and *scheduling problems with scenario-dependent processing times* (SDPTSPs), first formalized by Daniels and Kouvelis [7,8]. More recently, Framinan et al. [9] provided a comprehensive and unified picture of *COSPs*. Agnetis et al. [10] provided a comprehensive survey of the *multi-agent scheduling problems* (MASPs); meanwhile, Perez-Gonzalez and Framinan [11] reviewed a survey paper on multi-criteria order scheduling problems. More recently, Braga-Santos et al. [12] proposed an efficient size-reduction algorithm to solve a COSP in order to minimize the total tardiness. Taking the total completion time as the measurement criterion, de Athayde Prata et al. [13] applied an innovative discrete differential evolution algorithm with differential mutations to solve a COSP with sequence-dependent setups. Pinto Antonioli et al. [14] addressed a mixed-integer linear programming as well as two adaptation heuristics and two meta-heuristics to minimize the total tardiness COSP with a sequence-dependent setup time.

However, *scheduling problems with scenario-dependent processing times* (SDPTSPs) have not received much attention in the literature. SDPTSPs deal with a situation where the uncertain job processing times depend on the specific scenario that may arise in practice, thus giving rise to discrete job processing times called scenario-dependent processing times. In such situations, because of the uncertainty associated with the realization of any specific scenario, optimizing the given objective function among all possible scenarios is desired (Kouvelis and Yu [15]). Applications of SDPTSPs in several manufacturing environments, flexible manufacturing situations, and other practical circumstances are discussed by Daniels and Kouvelis [8]). For more examples, we refer readers to a few existing research efforts by Yang and Yu [16], Monch et al. [17], Aloulou and Della Croce [18], Aissi et al. [19], Kasperski and Zielinski [20], Wang et al. [21], Liu et al. [22], Wang et al. [23,24], Wu et al. [25], Xing et al. [26], Ren et al. [27], and Wu et al. [28], among others.

A review of the existing scheduling literature reveals that the research on COSPs involving multiple facilities and MASPs considered the job processing times or release dates as fixed integers. This is at odds with real-life environments. In fact, there are many significant uncertainty factors in practical production environments. For example, employees might become unable to work, machines might break, the working environments might change, and some external complex factors might provoke job cancellations or a changed tool quality. In such situations, the worst-case criterion for a system might be more important than the average-case criterion (Kouvelis and Yu [15]). Thus, inspired by these observations and in order to extend the application of scheduling research to the solving of practical problems, we address a two-agent COSP with this paper, for the first time in the scheduling literature. More specifically, we address scenario-dependent component processing times and release dates that minimize the maximum of the total completion time of the customer orders that belong to one agent and which are subject to the following constraint—that the maximum total completion time of the customer orders of the other agent should be no greater than an upper bound for all scenarios and all possible schedules. Table 1 summarizes the most recent contributions published concerning the COSP.

Table 1. Summary of most recent contributions published concerning the COSP.

Problem Setting	Algorithm	Objective Form	References
COSP without setup times	A size-reduction algorithm	$\sum_{i=1}^{n} T_i(\sigma)$	Braga-Santos et al. [12]
COSP with sequence-dependent setups	A differential evolution	$\sum_{i=1}^{n} C_i(\sigma)$	de Athayde Prata et al. [13]
COSP with sequence-dependent setups	Hybrid matheuristics	$\sum_{i=1}^{n} T_i(\sigma)$	Pinto Antonioli et al. [14]
COSP without setup times	Iterated greedy; BB	$max_s\{\sum_{i=1}^{n} T_i^s(\sigma)\}$	Wu et al. [28]
COSP without setup timeswith two agents	Simulated Annealing Hyper-heuristic; BB	$max_{s\in\{1,2\}}\left\{\sum C_i^{(s)x}(\sigma)\right\}$ s.t. $max_{s\in\{1,2\}}\left\{\sum C_i^{(s)y}(\sigma)\right\} \leq U$	This paper (2022)

In view of the fact that the proposed problem is an NP-hard one, we propose a branch-and-bound method to find an exact solution. For more details on branch-and-bound, the reader could refer to Zegordi et al. [29] and Wu et al. [25]. On the other hand, we also propose some simulated annealing algorithms (SA) and two variants of simulated annealing hyper heuristics to find near-optimal solutions. For details on SA or SAHH, readers can refer to Kirkpatrick et al. [30], Bıçakcı et al. [31], Azimi [32], Anagnostopoulos and Koulinas [33], Bai et al. [34], and Liang et al. [35].

The several contributions of this article include: (1) The presentation of a new COSP with two agents as well as scenario-dependent component processing times and release dates. (2) The proposal of a branch-and-bound algorithm along with six properties and a lower bound to find the optimal solution. (3) The proposal of three heuristics and two variants of simulated annealing hyper heuristics in order to find near-optimal solutions. (4) The execution of simulation tests to evaluate the performances of all proposed methods.

The rest of this study is organized as follows. Section 2 describes the notations, defines the problem, proposes lower bound and six dominance properties, and outlines a branch-and-bound algorithm to optimally solve the problem. Section 3 discusses three local search heuristics and two variants of simulated annealing hyper heuristics to obtain optimal or near-optimal solutions. In Section 4, we empirically investigate the performance of the proposed heuristics in generating optimal or near-optimal solutions for the considered problem. Finally, we conclude the paper in Section 5.

2. Problem Definition and Properties

Throughout this paper, we use the following notations.

Notations

$N = \{1,\ 2,\ \ldots,\ n\}$: represents a set of n jobs (orders);
$\Phi = \{x,\ y\}$: denotes the set of two agents, x and y;
$O_i^x(O_i^y)$: denotes a customer order i from agent x (y), $i \in N$;
$\Omega_N = \{O_1^\Phi,\ O_2^\Phi,\ \cdots,O_n^\Phi\}$: presents a set of n customer orders, where O_i^Φ (or in brief, order i) presents a customer order i from agent x or agent y;
$O^x = \bigcup_{i\in N} O_i^x,\ O^y = \bigcup_{i\in N} O_i^y$;
$M = \{1,\ 2,\ \ldots,\ m\}$**:** represents a set of indexes of m parallel machines,$M_1, M_2, .., M_m$;
s: scenario of customer order parameters, $s = 1, 2$;
σ, σ': denote two full schedules of n customer orders;
δ, δ': denote two partial schedules of n customer orders;
$t_{iv}^{(s)x}$(or $t_{iv}^{(s)y}$): is the component processing time of a customer order O_i^Φ of agent x (or agent y) on a machine $M_v, v \in M, s = 1, 2$;
$r_i^{(s)x}\left(or\ r_i^{(s)y}\right)$: denotes the ready (or release) time of a customer order i of agent x (or agent y), $i \in N$;
[ ]: presents the position of a customer order in a schedule;
$C_{[k]}^{(s)x}(\sigma)$ (or $C_{[k]}^{(s)y}(\sigma)$): denotes the completion time of a customer order, say i, of agent x (or agent y) scheduled in the k-th position in σ, $i \in N$, where
$C_{[k]}^{(s)x}(\sigma) = \max_{v\in M}\left\{\max\left\{C_{[k-1]}^{(s)x}(\sigma), r_i^{(s)x}\right\} + t_{iv}^{(s)x}\right\}$ or
$C_{[k]}^{(s)y}(\sigma) = \max_{v\in M}\left\{\max\left\{C_{[k-1]}^{(s)y}(\sigma), r_i^{(s)y}\right\} + t_{iv}^{(s)y}\right\}$, $s = 1, 2$.

2.1. Problem Definition

Let DPm denote the m dedicated machines used to produce m components of each job, and $sdpt$ represent the scenario-dependent processing times of the components. Following the standard three-field classification of Perez-Gonzalez and Framinan [11] and Framinan et al. [9], to define the $DPm \to 0\,|\,r_i$, *2-agent, sdpt* $|\,(max_{s\in\{1,2\}}\left\{\sum C_i^{(s)x}(\sigma)\right\}$ $/\,max_{s\in\{1,2\}}\left\{\sum C_i^{(s)y}(\sigma)\right\} \leq U)$ problem considered in this paper, consider a set of n customer orders belonging to n different customers, in which a customer order has m components to be processed on m parallel machines, and where a given component of each customer order is processed on the pre-specified dedicated machine. These orders are from two agents, agent x and agent y. Due to the fact that some factors of significant uncertainties are present, it is assumed that a customer order has scenario-dependent component processing times $t_{iv}^{(s)x}(t_{iv}^{(s)y})$ on machine $M_v, v \in M$, and a scenario-dependent ready time $r_i^{(s)x}$(or $r_i^{(s)y}$), $s = 1, 2$. Furthermore, the orders of agent y have a common upper limit U on their completion time.

With the above description and notations, the problem considered in this paper, the $DPm \to 0\,|\,r_i$, *2-agent, sdpt* $|\ (max_{s\in\{1,2\}}\left\{\sum C_i^{(s)x}(\sigma)\right\}/max_{s\in\{1,2\}}\left\{\sum C_i^{(s)y}(\sigma)\right\} \leq U)$ problem, is one of finding an appropriate schedule, among all possible schedules, to minimize the maximum of the total completion time of all customer orders that belong to agent x and are subject to the following constraint—that the maximum of the total completion time of the customer orders belonging to agent y should be no greater than an upper bound (U) for scenario $s = 1, 2$. In other words, we wish to determine a robust optimal schedule to minimize $max_{s\in\{1,2\}}\left\{\sum C_i^{(s)x}(\sigma)\right\}$, and which is subject to $max_{s\in\{1,2\}}\left\{\sum C_i^{(s)y}(\sigma)\right\} \leq U$. Given that Wu et al. [5] and Lenstra et al. [36] respectively showed that the problems $1\left|r_j\right| \sum C_j^x : \sum C_j^y$ and $1\left|r_j\right| \sum C_j \leq U$ are NP-hard, it follows that the problem considered in this paper is also an NP-hard problem. This can be summarized as follows.

Theorem 1. *The DPm $\rightarrow$ 0 | r_i, 2-agent, sdpt | ($max_{s\in\{1,2\}}\left\{\sum C_i^{(s)x}(\sigma)\right\}$/$max_{s\in\{1,2\}}\left\{\sum C_i^{(s)y}(\sigma)\right\} \leq U$) problem is NP-hard.*

Below is an illustrative example with a feasible solution to the problem under study. Consider a two-agent order schedule σ, with two orders and two scenarios to be processed on two machines; then, let $(t_{11}^{(1)x}, t_{12}^{(1)x}, r_1^{(1)x}) = (5,\ 6, 0)$ $(t_{11}^{(2)x}, t_{12}^{2(1)x}, r_1^{(2)x}) = (7,3,1)$ for order x, while $(t_{11}^{(1)y}, t_{12}^{(1)y}, r_1^{(1)y}) = (2, 1, 2)$ $(t_{11}^{(2)y}, t_{12}^{(2)y}, r_1^{(2)y}) = (3, 2, 2)$ for order y. Let $\sigma = (O_1^x, O_2^y)$, and let $U = 12$.

The completion times for each order scheduled in σ for scenario 1 are calculated as follows:

$$C_{[1]}^{(1)x}(\sigma) = max\left\{r_1^{(1)x} + t_{11}^{(1)x},\ r_1^{(1)x} + t_{12}^{(1)x}\right\} = max\{0+5,\ 0+6\} = 6$$

$$C_{[2]}^{(1)y}(\sigma) = max\{\max\{5, 2\} + 2, \max\{6, 2\} + 1\} = max\{7,\ 7\} = 7$$

The completion times for each order scheduled in σ for scenario 2 are calculated as follows:

$$C_{[1]}^{(2)x}(\sigma) = max\left\{r_1^{(2)x} + t_{11}^{(2)x},\ r_1^{(2)x} + t_{12}^{(2)x}\right\} = max\{1+7,\ 1+3\} = 8$$

$$C_{[2]}^{(2)y}(\sigma) = max\{\max\{8, 2\} + 3, \max\{4, 2\} + 2\} = max\{11,\ 6\} = 11$$

Since $max_{s\in\{1,2\}}\left\{\sum C_i^{(s)y}(\sigma)\right\} = \max\left\{C_{[2]}^{(1)y}(\sigma), C_{[2]}^{(2)y}(\sigma)\right\} = \max\{7,\ 11\} \leq 12 = U$, the objective function is given by the following equation:

$$max_{s\in\{1,2\}}\left\{\sum C_i^{(s)x}(\sigma)\right\} = \max\left\{C_{[1]}^{(1)x}(\sigma), C_{[1]}^{(2)x}(\sigma)\right\} = \max\{6,\ 8\} = 8$$

2.2. Proposed Lower Bound

To solve this intractable problem, we will derive several properties and a lower bound to be used in a branch-and-bound algorithm (Ogan and Azizoglu [37]) to accelerate the process finding an exact solution. Assume that δ is a specified partial schedule with q customer orders, while δ' denotes the undetermined set containing $(n-q)$ customer orders. For the simplification of the notation, we set $(n-q) = n_x + n_y$, where $n_x(n_y)$ denotes the number of unscheduled customer orders for agent x (agent y). Furthermore, we set $\theta_v^{(s)}$ to be the completion time of the last customer order on $M_v, v \in M$, for $s = 1,\ 2$, in δ, and $\theta^{(s)} = \max_{v\in M}\left\{\theta_v^{(s)}\right\}$, i.e., $C_{[q]}^{(s)}(\delta) = \theta^{(s)}$. To obtain the proposed lower bound, we only choose n_x x-agent orders to schedule on the $(q+1)$-*th*, $(q+2)$-*th*, … , and $(q+n_x)$-th positions appended immediately after the q-th position. From the previous definitions, therefore, it follows that:

$$C_{[q+1]}^{(s)x}(\sigma) = max_{v\in M}\left\{max\left\{C_{[q]}^{(s)}, r_{[q+1]}^{(s)x}\right\} + t_{[q+1]v}^{(s)x}\right\} > \theta^{(s)} + \sum\nolimits_{v\in M} t_{[q+1]v}^{(s)x}/m$$

$$C_{[q+2]}^{(s)x}(\sigma) = max_{v\in M}\left\{max\left\{C_{[q+1]}^{(s)}, r_{[q+2]}^{(s)x}\right\} + t_{[q+2]v}^{(s)x}\right\} > \theta^{(s)} + \left\{\sum_{v\in M} t_{[q+1]v}^{(s)x} + \sum_{v\in M} t_{[q+1]v}^{(s)x}\right\}/m.$$

Similarly, we have the following:

$$C_{[q+n_x]}^{(s)x}(\sigma) = max_{v\in M}\left\{max\left\{C_{[q+n_x-1]}^{(s)}, r_{[q+n_x]}^{(s)x}\right\} + t_{[q+n_x]v}^{(s)x}\right\} > \theta^{(s)} + \sum_{j=1}^{n_x}\sum_{v\in M} t_{[q+j]v}^{(s)x}/m.$$

Therefore, the total completion times for n_x x-agent orders can be summarized as follows:

$$\sum_{j=1}^{n_x} C_{[q+j]}^{(s)x}(\sigma) > n_x\theta^{(s)} + \sum_{j=1}^{n_x}(n_x - j + 1)\sum_{v\in \mathrm{M}} t_{[q+j]v}^{(s)x}/m, \text{ for } s = 1,\ 2.$$

For the schedule σ, we have the equation below:

$$\sum_{j=1}^{n} C_{[j]}^{(s)x}(\sigma) > \sum_{j=1}^{q} C_{[j]}^{(s)x}(\sigma) + n_x\theta^{(s)} + \sum_{j=1}^{n_x}(n_x - j + 1)\sum_{v\in M} t_{[q+j]v}^{(s)x}/m, \text{ for } s = 1,\ 2.$$

Therefore, we have the following equation:

$$max_{s=1,2}\left\{\sum_{j=1}^{q} C_{[j]}^{(s)x}(\sigma) + \sum_{j=1}^{n_x} C_{[q+j]}^{(s)x}(\sigma)\right\} > \sum_{s=1}^{2}\sum_{j=1}^{q} C_{[j]}^{(s)x}(\sigma) + n_x\theta^{(s)} + \sum_{j=1}^{n_x}(n_x - j + 1)\sum_{v\in M} t_{[q+j]v}^{(s)x}/2m.$$

Using the fact that the maximum value is larger than the mean value and a lemma, as defined by Hardy et al. [38], the proposed lower bound (*lbdd*) can be summarized as follows.

Proposition 1. $lbdd = \sum_{s=1}^{2}\sum_{j=1}^{q} C_{[j]}^{(s)x}(\sigma) + n_x\theta_{min}^{(s)} + \sum_{j=1}^{n_x}(n_x - j + 1)t_{(q+j)*}^{(s)x}/2m$, *where* $t_{q+j*}^{(s)x} = \sum_{v\in M} t_{[q+j]v}^{(s)x}$, *and* $t_{(q+1)*}^{(s)x} \le \ldots \le t_{(q+n_x)*}^{(s)x}$ *is the non-decreasing order of* $\left\{t_{q+1*}^{(s)x}, \ldots, t_{q+n_x*}^{(s)x}\right\}$.

2.3. Dominance Properties

Consider two order sequences , $\sigma = (\delta, O_i^{\Phi}, O_j^{\Phi}, \delta')$ and $\sigma' = (\delta, O_j^{\Phi}, O_i^{\Phi}, \delta')$, in which δ, δ' are two subsequences. To prove that "σ is no worse than sequence σ'", we need to confirm that for $s = 1, 2$, $C_i^{(s)x}(\sigma) + C_j^{(s)x}(\sigma) < C_j^{(s)x}(\sigma') + C_i^{(s)x}(\sigma')$ and $C_j^{(s)x}(\sigma) \le C_i^{(s)x}(\sigma')$. Furthermore, recall that $\theta_v^{(s)}$ is the completion time of the last job in the δ subsequence of schedule σ or σ' on machine $M_v, v \in M$, $s = 1, 2$. With these defined conditions, we describe the following six properties to curtail the search for an optimal solution. The details of proofs of Property 1 are provided, while other properties are omitted because they are similar to those of Property 1.

Property 1. *For any two customer orders* O_i^{Φ} *and* O_j^{Φ} *from agent x to be scheduled adjacent to each other, if* $\forall$ *s = 1, 2,* $max_{v\in M}\left\{t_{iv}^{(s)x}\right\} \le max_{v\in M}\left\{t_{jv}^{(s)x}\right\}$, $r_j^{(s)x} > r_i^{(s)x} \ge max_{v\in M}\left\{\theta_v^{(s)}\right\}$, *and* $r_j^{(s)x} \le r_i^{(s)x} + max_{v\in M}\left\{t_{iv}^{(s)x}\right\}$, *then sequence* σ *is not worse than sequence* σ'.

Proof. Suppose that there are q customer orders scheduled in the subsequence δ of σ(or σ'), and that the complete time of the last job is $C_{[q]}^{(s)}$. According to the definition:

$$C_i^{(s)x}(\sigma) = max_{v\in M}\left\{max\left\{C_{[q]}^{(s)x}, r_i^{(s)x}\right\} + t_{iv}^{(s)x}\right\}, \tag{1}$$

$$C_j^{(s)x}(\sigma) = max_{v\in M}\left\{max\left\{C_i^{(s)x}(\sigma), r_j^{(s)x}\right\} + t_{jv}^{(s)x}\right\}, \tag{2}$$

$$C_j^{(s)x}(\sigma') = max_{v\in M}\left\{max\left\{C_{[q]}^{(s)x}, r_j^{(s)x}\right\} + t_{jv}^{(s)x}\right\}, \tag{3}$$

$$C_i^{(s)x}(\sigma') = max_{v\in M}\left\{max\left\{C_j^{(s)x}(\sigma'), r_i^{(s)x}\right\} + t_{iv}^{(s)x}\right\}. \tag{4}$$

Applying the given condition $r_j^{(s)x} > r_i^{(s)x} \geq max_{v\in M}\left\{\theta_v^{(s)}\right\}$ to (1) and (3), we have the following:

$$C_i^{(s)x}(\sigma) = r_i^{(s)x} + max_{v\in M}\left\{t_{iv}^{(s)x}\right\},\ C_j^{(s)x}(\sigma') = r_j^{(s)x} + max_{v\in M}\left\{t_{jv}^{(s)x}\right\},\ \text{for } s = 1,\ 2.$$

Further applying the given condition $r_j^{(s)x} \leq r_i^{(s)x} + max_{v\in M}\left\{t_{iv}^{(s)x}\right\}$ to $C_i^{(s)x}(\sigma)$:

$$C_j^{(s)x}(\sigma) = r_i^{(s)x} + max_{v\in M}\left\{t_{iv}^{(s)x}\right\} + max_{v\in M}\left\{t_{jv}^{(s)x}\right\}, \tag{5}$$

and further applying $r_j^{(s)x} > r_i^{(s)x}$ to $C_j^{(s)x}(\sigma')$, we have the following equation:

$$C_i^{(s)x}(\sigma') = r_j^{(s)x} + max_{v\in M}\left\{t_{jv}^{(s)x}\right\} + max_{v\in M}\left\{t_{iv}^{(s)x}\right\}. \tag{6}$$

Hence, from Equations (5) and (6), we have the following:

$$C_i^{(s)x}(\sigma') - C_j^{(s)x}(\sigma) = r_j^{(s)x} - r_i^{(s)x} > 0.$$

Combine Equations (1), (2), and (5) with inequality (6), we have the following: $[C_j^{(s)x}(\sigma') + C_i^{(s)x}(\sigma')] - [C_i^{(s)x}(\sigma) + C_j^{(s)x}(\sigma)] = 2(r_j^{(s)x} - r_i^{(s)x}) + \left(\max_{v\in M}\left\{t_{jv}^{(s)x}\right\} - \max_{v\in M}\left\{t_{iv}^{(s)x}\right\}\right) > 0$, indicating that sequence σ is not worse than sequence σ'. □

Property 2. *For any two customer orders O_i^{Φ} and O_j^{Φ} from agent x to be scheduled adjacent to each other, if* $\forall$ *s = 1, 2,* $r_i^{(s)x} \geq max_{v\in M}\{\theta_v^s\}$, *and* $r_j^{(s)x} > r_i^{(s)x} + max_{v\in M}\left\{t_{iv}^{(s)x}\right\}$, *then sequence σ is not worse than sequence σ'.*

Property 3. *For any two customer orders O_i^{Φ} and O_j^{Φ} from agent x to be scheduled adjacent to each other, if* $\forall$ *s = 1,2,* $max\left\{r_i^{(s)x},\ r_j^{(s)x}\right\} \leq min_{v\in M}\{\theta_v^s\}$, *and* $max_{v\in M}\left\{\theta_v^s + t_{iv}^{(s)x}\right\} < max_{v\in M}\left\{\theta_v^s + t_{jv}^{(s)x}\right\}$, *then sequence σ is not worse than sequence σ'.*

Property 4. *For any two customer orders O_i^{Φ} and O_j^{Φ} from an agent, x to be scheduled adjacent to each other, if* $\forall$ *s* $= 1,2$, $max\left\{r_i^{(s)x},\ r_j^{(s)x}\right\} \leq min_{v\in M}\{\theta_v^s\}$, *and* $max_{v\in M}\left\{\theta_v^s + t_{iv}^{(s)x}\right\} < max_{v\in M}\left\{\theta_v^s + t_{jv}^{(s)x}\right\}$, *then sequence σ is not worse than sequence σ'.*

Property 5. *If* $\exists\ O_i^y \in \delta'$ *and* $\forall$ *s* $= 1,\ 2,\ \sum_{k\in\delta} C_{[k]}^{(s)y} > U,$ *then* $\sigma = (\delta, \delta')$ *can be eliminated from the search for an optimal (robust) schedule. (Note that δ denotes the scheduled part, while δ' denotes the unscheduled part.)*

Property 6. *If* $\exists\ O_i^y \in \delta'$ *and* $\forall$ *s* $= 1,\ 2, \sum_{k\in\delta} C_{[k]}^{(s)y} + max_{v\in M}\left\{max\left\{\theta_v^{(s)}, r_i^{(s)y}\right\} + t_{iv}^{(s)y}\right\} > U,$ *then* $\sigma = (\delta, \delta')$ *can be eliminated from the search for an optimal (robust) schedule.*

2.4. Proposed Branch-and-Bound Algorithm

The lower bound described above, the six dominance properties, and the best near-optimal solution obtained from the proposed heuristics in the next section are used to describe the proposed branch-and-bound algorithm (B&B) by using the depth-first method for the small-sized job case in this problem. The details of the B&B algorithm are provided as follows.

Algorithm 1 A branch-and-bound method

```
00: Input a set $M = \{1, 2, \ldots, m\}$ of $m$ parallel machines and a set $N = \{1, 2, \ldots, n\}$ of $n$ orders with scenario-dependent component processing times $t_{iv}^{(s)x}(t_{iv}^{(s)y})$ and ready times $r_i^{(s)x}$ (or $r_i^{(s)y}$), $i$ = 1,2, … ,$n$}; criterion function: $Minimize\ \ max_{s\in\{1,2\}}\left\{\sum C_i^{(s)x}(\sigma)\right\}$, $subject\ to\ max_{s\in\{1,2\}}\left\{\sum C_i^{(s)y}(\sigma)\right\} \le U$.
01: Input the best heuristic solution with an upper bound.
02: Start to branch from the root node by appending each order to create a new node.
03: For each active node, (i) find the lower bound based on proposition 1 in $\pi^c$; then, (ii) evaluate if the lower bounds $\geq$ the incumbent upper bound and cut those nodes and all nodes below them in the branching tree.
04: Delete the unwanted nodes from the branching tree by property 1 to property 6.
05: To determine if the node is complete or not, we do the following:
    (i)   find its criterion function equal to $max_{s\in\{1,2\}}\left\{\sum C_i^{(s)x}(\sigma)\right\}$;
    (ii)  evaluate if this criterion function is smaller than the upper bound, then update it with the new one and keep the corresponding schedule;
    (iii) For the remaining nodes, branch from the node with the minimum lower bound to create a set of new active nodes.
06: Repeat step 02 through step 05 until all nodes have been visited, and set the final complete schedule as σ.
07: Output the final complete schedule σ as an optimal solution.
```

3. Simulated Annealing Hyper-Heuristic

In this section, we propose three heuristics and two variants of a simulated annealing hyper-heuristic with different cooling temperatures for finding optimal or near-optimal solutions to the problem. To investigate the effect of the customer orders' component scenario-dependent processing times, we consider three types of combinations of the processing times $t_{iv}^{\Phi s}$ for each customer order, s = 1, 2; i.e.:

(i) $0.5max_{v\in M}\left\{r_i^{(1)\Phi} + t_{iv}^{(1)\Phi}\right\} + 0.5max_{v\in M}\left\{r_i^{(2)\Phi} + t_{iv}^{(2)\Phi}\right\}$;

(ii) $0.5min_{v\in M}\left\{r_i^{(1)\Phi} + t_{iv}^{(1)\Phi}\right\} + 0.5min_{v\in M}\left\{r_i^{(2)\Phi} + t_{iv}^{(2)\Phi}\right\}$;

and

(iii) $0.5\left(r_i^{(1)\Phi} + \sum_{v\in M} t_{iv}^{(1)\Phi}\right) + 0.5(r_i^{(2)\Phi} + \sum_{v\in M} t_{iv}^{(2)\Phi})$.

Based on these three formulas, we schedule customer orders using the smallest processing time (SPT) first rule to obtain three initial schedules, S_{MM}, S_{mm}, and S_{mean}. To ensure good quality for the approximate solutions, we consider the two following policies: (1) We improve each of the S_{MM}, S_{mm}, and S_{mean} by using a pairwise interchange method and record these three heuristics as Mpi, mpi, and meanpi, respectively. (2) We use these three improved heuristics as three initial seeds in a simulated annealing super-heuristic algorithm. They are termed as SAHM, SAHm, and SAHmean. We propose a simulated annealing hyper-heuristic algorithm (SAH), in which the low-level heuristics may be based on the proposed eight candidate mutation operators. They are termed as HL_1, HL_2, ..., and HL_8.

The frequency (coded as f_i) of the improvement of a low-level heuristic HL_i is recorded when the accumulated performance of HL_i is obtained in each cycle. The selection probability $P(HL_i)$ for HL_i is defined as $f_i / \sum_{i=1}^{8} f_i$. These low-level heuristics are selected randomly based on the values of the selection probabilities ($P(HL_1)$, $P(HL_2)$, … , $P(HL_8)$). For the first cycle (or I_{cmax}=1), the selection probabilities of these low-level heuristics are set to be equal, i.e., $f_i = 1$, i = 1, … , 8. For $I_{cmax} > 1$, the probabilities are determined according to the corresponding low-level heuristic additive performance. To ensure the diversity of all low-level heuristics in the list, we set the minimum value $f_i = \max\{f_i, 1\}$. With these definitions, the details of the eight proposed low-level heuristics are described below:

HL$_1$: Two-order swap heuristic.
HL$_2$: One step to the right heuristic.
HL$_3$: Two steps to the right heuristic.
HL$_4$: Two randomly selected orders, O_i, O_j, where $1 \leq I < j < n$, and swap both orders with their immediately succeeding orders.
HL$_5$: Two randomly selected orders, O_i, O_j, where $1 < i < j \leq n$, and swap both orders with their closest preceding orders.
HL$_6$: Two randomly selected orders, O_i, O_j, where $1 \leq i < i + 1 < j \leq n$, swap the order in front O_j with its immediately succeeding order, and swap another O_i with its closest preceding order.
HL$_7$: Two randomly selected orders O_i, O_j, where $1 < i < j < n$, swap the order in front O_j with its closest preceding order, and swap another O_i with its immediately succeeding order.
HL$_8$: The current order of jobs is reversed.

For any two schedules σ and σ_t, we set $RTC(\sigma) = \max_{s=1,2} \sum_{i \in O^x} C_{[i]}^{(s)x}(\sigma)$ and $RTC(\sigma_t) = \max_{s=1,2} \sum_{i \in O^x} C_{[i]}^{(s)x}(\sigma_t)$ as the values of their objective function. Let I$_{\text{cmax}}$ denote the maximum number of cycles for SAHs, and (T_i, T_f, Cf, Nr) denote the initial temperature, the final temperature, the cooling factor, and the maximum number of iterations per cycle. Then, we describe below the steps of the SAHs algorithms.

Algorithm 2 The steps of the SAHs procedure

```
01: Input $I_{cmax}$, $Nr$, $T_i$, $T_f$, and Cf
02: Input each initial sequence $\sigma$ and its objective function $RTC(\sigma)$
03: Set $T = T_i$; $f_k = 1$, $k = 1, \ldots, 8$; $i\_no = 0$
04: Do while {$i\_no \leq I_{cmax}$ and $T > T_f$}
05:     num = 0
05:     Do while {num $\leq Nr$}
06:         Randomly select an $HL_k$ by the roulette wheel according to their probabilities
07:         Utilize it to form a new sequence $\sigma_t$ and to find the $RTC(\sigma_t)$ of $\sigma_t$
08:         If $L_t = f(s_t)$, $RTC(\sigma_t) < RTC(\sigma)$, replace $\sigma$ with $\sigma_t$; otherwise, replace $\sigma$ with $\sigma_t$ using the probability
        $\exp(-(RTC(\sigma_t) - RTC(\sigma))/RTC(\sigma)T)$
09:             num = num + 1 and $f_k = f_k + 1$
10:     End do
11:     Update $f_k / \sum_{k=1}^{8} f_k$ for each $HL_k$; $T = T \times$ Cf; $i\_no = i\_no + 1$
12: End do
13: Output final sequence $\sigma$ and its objective function $RTC(\sigma)$.
```

In addition to the three heuristics, Mpi, mpi, and meanpi, developed to solve the studied problem, we employ two variants of the SAH algorithms labeled as (SAHM, SAHm, SAHmean) and (SAHMb, SAHmb, SAHmeanb), respectively. Both variants use the three initial schedules S$_{MM}$, S$_{mm}$, S$_{mean}$, respectively, similar to the seeds in the above SAH algorithm procedure. However, SAHMb, SAHmb, and SAHmeanb use the modified temperature formula from Bai et al. [34] as $T = T/(1 + \beta T)$, where $\beta = \left(T_i - T_f\right) * Nr/(I_{cmax} * Nr * T_i * T_f)$, to fit the studied problem.

4. Computational Results

This section presents the results of extensive simulation experiments, performed to determine the effectiveness of the branch-and-bound algorithm, three local heuristics (Mpi, mpi, meanpi), and six SAHs algorithms (SAHM, SAHm, SAHmean, SAHMb, SAHmb, SAHmeanb) in solving the considered problem. The instances generated are described as follows. The processing times of customer orders were generated from the uniform distribution $U(1, 100)$ for scenario 1 and from $U(1, 200)$ for scenario 2. Following the design of Reeves [39], the order release (ready) times were generated from the uniform distribution

$U(0, 100n\lambda)$ for scenario 1 and from the uniform distribution $U(0, 200n\lambda)$ for scenario 2, where n is the number of orders and λ is a control variable. Three different types of problem instances were created according to the values of λ at 0.25, 0.5, and 0.75. To generate a feasible problem instance, we first generated n_y orders of agent y to put before n_x orders, computed the total completion times for scenario 1 and scenario 2, and then set the value of the upper bound at $U = 2.5 \times \max_{s=1,2}\left\{\sum_{i\in O^y} C_{[i]}^{(s)y}\right\}$, where 2.5 was a tested number that ensured a feasible instance.

4.1. Tuning the Parameters in SAH Algorithms

To obtain appropriate values of related parameters in the SAH algorithms, the number of customer orders was set at n = 10 and the number of machines was set at $m = 3$, where $(n_x, n_y) = (5, 5)$. The component processing time ($t_{iv}^{(1)\Phi}$) of an order was generated from a uniform distribution $U(1, 50)$, while the processing time $t_{iv}^{(2)\Phi}$ was generated from another uniform distribution $U(1, 100)$. Meanwhile, the ready time ($r_i^{(1)\Phi}$) of order I for scenario one ($s = 1$) was generated from a uniform distribution $U(1, 50\cdot\lambda\cdot n)$, while the ready time ($r_i^{(2)\Phi}$) of order I for scenario two ($s = 2$) was generated from another uniform distribution $U(1, 100\cdot\lambda\cdot n)$, where λ was taken as 0.25. The value of the upper bound was set at $2.5 \times \max_{s=1,2}\left\{\sum_{i\in O^y} C_{[i]}^{(s)y}\right\}$, where 2.5 was a tested number that ensured a feasible instance. One hundred problem instances were tested for each combination of parameters. The final temperature was set at $T_f = 10^{-8}$ to tune all the values of the parameters in the SAH algorithms.

The average error percentage (*AEP*) was used as the index of performance; i.e., $AEP = [(HA - O^*)/O^*] \times 100[\%]$, where *HA* is the total completion time of the orders belonging to agent x obtained by using three heuristics (Mpi, mpi, meanpi) or six SAHs (SAHM, SAHm, SAHmean, SAHMb, SAHmb, SAHmeanb), and O^* is the total completion time of the orders belonging to agent x obtained by using the branch-and-bound algorithm. For simplicity, all the determined parameters of SAHM will be used in the other SAHs.

The proposed SAHs have four parameters, namely initial temperature (*Ti*), cooling factor (*Cf*), the runs of low-level loops for local improvement (*Nr*), and the runs of top-level loops (I_{cmax}). With *Cf* = 0.95, *Nr* = 1000, and I_{cmax} = 15, a simulation was conducted under a variation in *Ti*, from 10^{-5} to 10^6, with multiples of 10 times each. As shown in Figure 1a, the *AEP* reached the lowest point when *Ti* was equal to 0.1; hence, the optimal setting for *Ti* was 0.1.

With *Ti* = 0.1, *Nr* = 1000, and I_{cmax} = 15, a simulation was conducted under a variation in Cf, from 0.90 to 0.99, with increments of 0.1 each. As shown in Figure 1b, the AEP approximated zero (below 0.01%) when *Cf* was equal to 0.91, indicating an effective reduction of AEP; hence, the optimal setting for Cf was 0.91.

With *Ti* = 0.1, *Cf* = 0.91, and I_{cmax} = 15, a simulation was conducted under a variation in *Nr*, from 200 to 1000, with increments of 50 each. As shown in Figure 1c, the AEP reached its minimum when *Nr* was equal to 850, also indicating an effective reduction of AEP with an increase in *Nr*; hence, the optimal setting for *Nr* was 850.

Finally, with *Ti* = 0.1, *Cf* = 0.91, and *Nr* = 850, a simulation was conducted under a variation in I_{cmax}, from 10 to 30, with increments of two each. As shown in Figure 1d, the AEP was the lowest when I_{cmax} was equal to 18, 22, 24, 26, 28, and 30; hence, the optimal setting for I_{cmax} was 22.

After previous tested results, we adopted the parameters *Ti* = 0.1, *Cf* = 0.91, *Nr* = 850, and I_{cmax} = 22 used in SAH algorithms for the following tested problems.

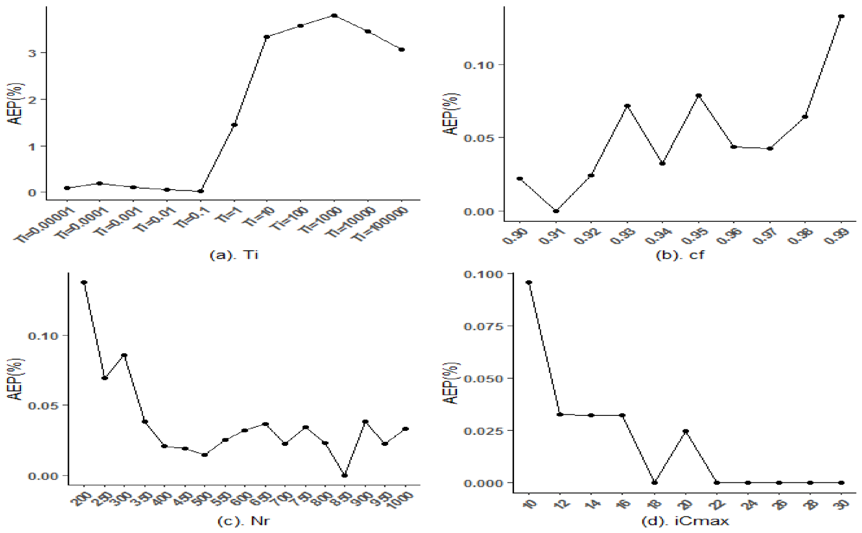

Figure 1. Behavior of related parameter tuning in SAHs algorithms (n = 10), (**a**) initial temperature; (**b**) cooling factor; (**c**) times of local improvement; (**d**) No. of cycles.

4.2. Small-n Experimental Results Analysis

For a small number of orders, the number of jobs is set at n = 8 and 10, and the machine number is set at m = 2, 3, and 4. To evaluate their impacts on the algorithms, the numbers of x-agent order and y-agent order (n_x, n_y) are set at (2, 6), (4, 4), and (6, 2) for n = 8, and at (3, 7), (5, 5), and (7, 3) for n = 10. A set of 100 instances were randomly generated for each combination of n, m, n_x, and λ. Consequently, a total of 5400 problem instances were tested. The algorithms were set to skip to the next set of data if the number of nodes exceeded 10^8.

The impacts of the number of orders (n), machine number (m), the number of x-agent orders (n_x), and the control parameter of the range of release dates (λ) on the performance of the branch-and-bound algorithms, three heuristics, and the six SAH algorithms are shown in Table 2.

Table 2. A summary of the performance of the branch-and-bound algorithm.

n	m	Node	CPU_Time
8	2	3540	0.0299
	3	4649	0.0442
	4	4707	0.0489
10	2	141,867	0.7474
	3	140,400	0.8877
	4	187,833	1.3859
	n_x		
8	2	3913	0.0409
	4	2891	0.0331
	6	6092	0.0491
10	3	127,915	0.8616
	5	117,367	0.7827
	7	224,819	1.3766
	λ		
8	0.25	4047	0.0406
	0.50	4001	0.0388
	0.75	4848	0.0438
10	0.25	112,313	0.7410
	0.50	144,859	0.9204
	0.75	212,928	1.3595

To measure the performance of the branch-and-bound algorithm, we recorded the average number of nodes and the average execution times (in seconds) for n = 8 and 10. Table 2 presents its capability, and the mean nodes and CPU times increase as n dramatically increases from 8 to 10 (columns 3 and 4 of Table 1). As demonstrated in Table 2, the mean nodes and CPU times increase as m increases regardless of the number of jobs n. When the number of x-agent orders (n_x in Table 2) becomes equivalent to half of the number of total orders (n), the mean nodes and CPU times are lower than other values of n_x independent of the number of jobs n. As the range of release dates becomes larger (λ increases from 0.25 to 0.75), the search space for the studied problem becomes wider; thus, the nodes and CPU times increase, especially for n = 10.

For the three heuristics, Mpi, mpi, and meanpi, and the six SAH algorithms, we record the *AEP* of the objective function; the $AEP = [(HA - O^*)/O^*] \times 100[\%]$, where HA and O^* are the values of the objective function obtained by running heuristics/SAHs and the branch-and-bound algorithm, respectively. The results are summarized in Table 3.

Table 3. Performance of heuristics and SAH algorithms for a small n.

		AEP(%)								
n	**m**	**Mpi**	**mpi**	**meanpi**	**SAHM**	**SAHm**	**SAHmean**	**SAHMb**	**SAHmb**	**SAHmeanb**
8	2	10.33	10.96	11.41	0.0000	0.0064	0.0000	0.0297	0.0315	0.0238
	3	10.89	11.64	11.76	0.0001	0.0009	0.0004	0.0332	0.0435	0.0312
	4	11.21	11.71	11.89	0.0000	0.0001	0.0000	0.0311	0.0261	0.0298
10	2	15.84	16.29	16.59	0.0029	0.0046	0.0026	0.0482	0.0603	0.0538
	3	16.35	16.38	16.74	0.0007	0.0055	0.0062	0.0435	0.0320	0.0498
	4	16.83	17.37	17.31	0.0048	0.0014	0.0048	0.0783	0.0515	0.0545
	n_x									
8	2	8.23	8.77	8.88	0.0001	0.0001	0.0004	0.0698	0.0717	0.0694
	4	14.03	15.21	15.56	0.0000	0.0073	0.0000	0.0242	0.0294	0.0154
	6	10.17	10.33	10.62	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
10	3	12.94	13.54	13.48	0.0077	0.0115	0.0136	0.1567	0.1314	0.1442
	5	18.17	18.93	19.09	0.0007	0.0000	0.0000	0.0133	0.0124	0.0139
	7	17.92	17.57	18.07	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	λ									
8	0.25	17.62	18.76	19.25	0.0001	0.0073	0.0004	0.0698	0.0700	0.0641
	0.50	9.78	10.27	10.51	0.0000	0.0001	0.0000	0.0175	0.0306	0.0172
	0.75	5.04	5.28	5.30	0.0000	0.0000	0.0000	0.0068	0.0006	0.0035
10	0.25	26.45	27.02	27.60	0.0069	0.0097	0.0108	0.1340	0.0967	0.1247
	0.50	14.07	14.49	14.37	0.0002	0.0018	0.0014	0.0244	0.0349	0.0219
	0.75	8.50	8.53	8.68	0.0012	0.0000	0.0014	0.0116	0.0123	0.0115
mean		13.58	14.06	14.28	0.0014	0.0032	0.0023	0.0440	0.0408	0.0405

Concerning the impact of the machine number (m) on the three heuristics and six SAH algorithms, the results in Table 3 show that there is not much difference in the performance of the AEP for heuristics or SAHs. As for the impacts of the number of x-orders (n_x) on the three heuristics and six SAH algorithms, it was observed that the three heuristics performed the worst when $n_x = n_y$, and the AEPs decreased (except for the SAHm at n = 8) as n_x increased for the six SAHs algorithms. It can also be observed in Table 2 that the AEP declines as the value of λ increases from 0.25 to 0.75 for almost all nine heuristics/algorithms (except for the SAHM at n = 10). This means that the wider the release dates of job orders, the better the solution quality resulting from these nine heuristics/algorithms. When the three heuristics and six SAH algorithms are divided into three groups, the best group is (SAHM, SAHm, SAHmean) with the lowest AEPs at (0.0014, 0.0032, 0.0023), the second is

(SAHMb, SAHmb, SAHmeanb) with AEPs at (0.0440, 0.0408, 0.0405), and the worst is (Mpi, mpi, meanpi) with the *AEP*s at (13.58, 14.06, 14.28). Figures 2 and 3 depict this peculiarity.

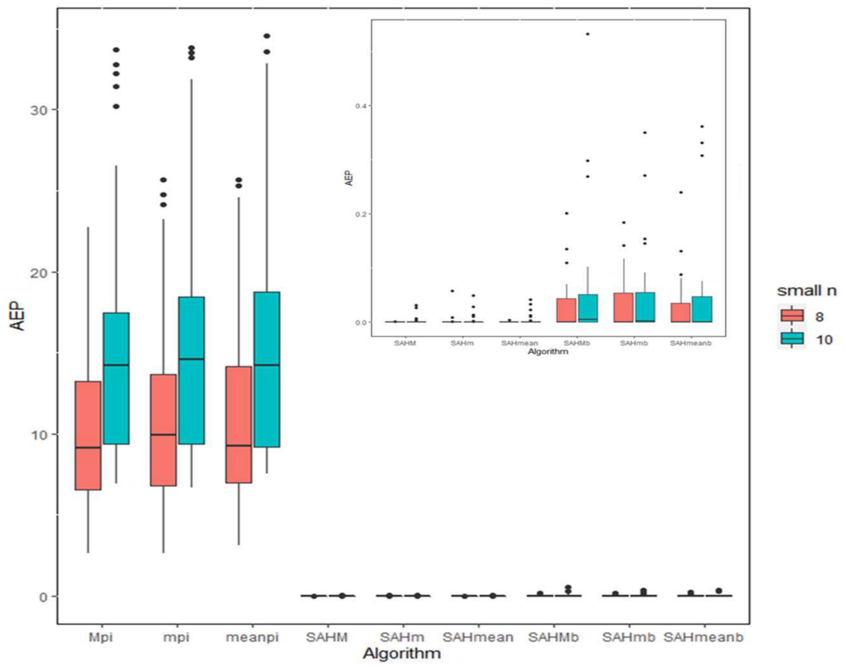

Figure 2. Boxplot for n = 8 and n = 10.

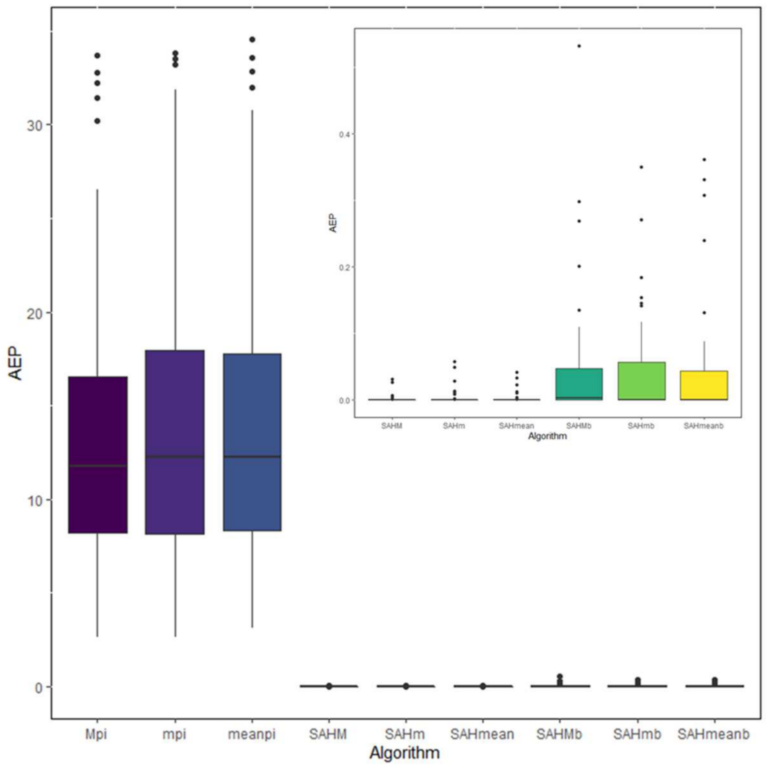

Figure 3. Boxplot for small n.

To further analyze the statistically significant differences of the solution quality among the three heuristics and six SAH algorithms, we made use of the SAS software in conducting an ANOVA (analysis of variance) on the *AEP*s for the variables, algorithms, n, m, n_x, and λ. However, we found that the normality assumption of the linear model did not hold.

Therefore, the nonparametric Friedman test was utilized to differentiate the performance of the nine heuristics/SAH algorithms. Based on the sum of ranks of the AEP calculated for each of the 54 ($n \cdot m \cdot n_x \cdot \lambda$ = 2·3·3·3) blocks of the 100 test problem instances, the Friedman test showed that the p-value was less than 0.0001 (with a chi-square value of 374.4 and eight degrees of freedom). These test results verify that the AEP samples do not follow the same distribution at the level of significance $\alpha = 0.05$. To further compare the pairwise differences among the three heuristics and six SAH algorithms, we applied the WNMT test (Wilcoxon–Nemenyi–McDonald–Thompson procedure; Holland et al. [40]).

Table 4 (column 3) shows the sums of the rank of the AEPs across the 54 blocks for the nine heuristics/SHA algorithms. The WNMT test shows that (SAHM, SAHm, SAH-mean, SAHMb, SAHmb, SAHmeanb) is a better performing group, with lower rank sums at (148.5, 152.5, 154.5, 228.5, 232.5, 217.5), and the group (Mpi, mpi, meanpi) is a worse performing group, with rank sums at (410.0, 432.0, 454.0). The test confirms that the performances of the two groups among the heuristics and SAH algorithms are statistically different at $\alpha = 0.05$. SAHM is the best overall performer for the problem instances containing a small number of customer orders.

Table 4. The rank-sum of heuristics and SAHs algorithms.

Heuristic/Algorithm	**No. of Obs.**	**Rank Sum (Ri, $i = 1, 2, \ldots, 9$)**	
		Small n	**Large n**
Mpi	54	410.0	453.0
mpi	54	432.0	461.0
meanpi	54	454.0	382.0
SAHM	54	148.5	157.0
SAHm	54	152.5	138.0
SAHmean	54	154.5	140.0
SAHMb	54	228.5	249.0
SAHmb	54	232.5	223.0
SAHmeanb	54	217.5	227.0

Note that the critical value of the WNMT test is approximated at 88.4. (Holland et al. [40]); i.e., two algorithms are significantly different if $|Ri - Rj| > 88.4$.

Concerning the usage of the eight low-level heuristics and the variation of probabilities of calling them for the SAHM, as displayed in Figure 4, HL_7 was called the most often, followed by HL_6. However, HL_8 was rarely called.

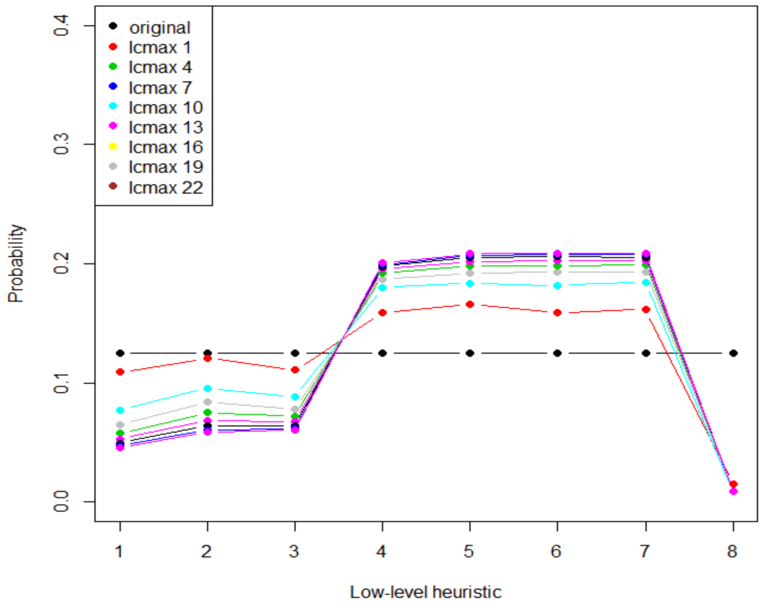

Figure 4. Variation of probabilities for low-level heuristics.

4.3. Large-n Experimental Results Analysis

In the set of experiments with a large number of orders, we set n = 80 and 100, and the machine number at m = 10, 20, and 30. We set (n_x, n_y) at (20, 60), (40, 40), and (60, 20) for n = 80, and (30, 70), (50, 50), and (70, 30) for n = 100 to evaluate their impacts on the performance of the heuristics and the SAH algorithms. A set of 100 instances were randomly generated for each case. Consequently, 5400 problem instances were tested. We recorded the mean relative percentage deviance (RPD) for each of the three heuristics and six SAH algorithms. The RPD was calculated as RPD $= 100[(H_k - A_*)/A_*][\%]$, where H_k was obtained from each heuristic/algorithm, and A_* was the smallest value among the H_ks from the three heuristics and six SAH algorithms. Table 5 summarizes the impacts of n_x, m, and λ.

Table 5. Summary of performance of heuristics and SAH algorithms for large n.

		RPD (%)								
n	m	**Mpi**	**mpi**	**meanpi**	**SAHM**	**SAHm**	**SAHmean**	**SAHMb**	**SAHmb**	**SAHmeanb**
80	10	51.8	51.8	51.6	0.4	0.7	0.4	0.6	0.6	0.6
	20	52.5	52.5	52.1	0.4	0.4	0.4	0.6	0.6	0.6
	30	52.4	52.4	52.2	0.4	0.4	0.4	0.7	0.6	0.6
100	10	54.0	54.0	53.9	0.7	0.7	0.7	0.8	0.8	0.8
	20	54.5	54.5	54.2	0.6	0.6	0.7	0.8	0.8	0.8
	30	54.5	54.5	54.3	0.6	0.6	0.6	0.8	0.8	0.8
	n_x									
80	20	37.5	37.5	37.3	0.6	0.8	0.6	0.6	0.5	0.5
	40	54.2	54.2	53.9	0.5	0.5	0.5	0.8	0.8	0.8
	60	65.0	65.0	64.7	0.1	0.1	0.1	0.5	0.5	0.5
100	30	41.9	42.0	41.8	0.8	0.9	0.8	0.6	0.6	0.6
	50	55.4	55.4	55.1	0.7	0.7	0.8	0.9	0.9	0.9
	70	65.7	65.7	65.4	0.4	0.4	0.4	0.9	0.9	0.9
	λ									
80	0.25	82.1	82.0	81.9	0.6	0.5	0.5	0.9	0.8	0.8
	0.50	42.5	42.5	42.0	0.4	0.7	0.5	0.6	0.6	0.6
	0.75	32.2	32.2	32.0	0.2	0.2	0.2	0.4	0.4	0.4
100	0.25	85.5	85.6	85.3	0.9	1.0	1.0	1.2	1.2	1.2
	0.50	43.8	43.8	43.5	0.7	0.7	0.7	0.7	0.7	0.7
	0.75	33.7	33.7	33.5	0.3	0.3	0.3	0.5	0.5	0.5
Total mean		53.3	53.3	53.1	0.6	0.6	0.6	0.7	0.7	0.7

Regarding the impact of machine number (m), the number of x-orders, and λ on the three heuristics and six SAH algorithms, the trends of the RPD for the nine heuristics/algorithms are quite similar to those of the small-sized number of orders, except that the RPDs for the six SAH algorithms are now very close, and the difference in RPD between any of the two SAHs is no greater than 0.1 (Table 5).

Moreover, from Table 5 and Figure 5, it can easily be seen that the better group is (SAHM, SAHm, SAHmean, SAHMb, SAHmb, SAHmeanb), with lower the RPDs at (0.6, 0.6, 0.6, 0.7, 0.7, 0.7), and that the worse group is (Mpi, mpi, meanpi), with RPDs at (53.3, 53.3, 53.1).

To explore the statistical significance of the differences, a Friedman test was used to differentiate the performance of the nine heuristics and SAH algorithms. Based on the sum of ranks of the RPD calculated for each of the 54 blocks of 100 test problem instances, the Freidman test showed that the p-value was less than 0.0001 (with a chi-square value at 331.1 and eight degrees of freedom). These test results verify that the RPD samples do not follow the same distribution.

For the nine heuristics/algorithms, Table 4 (column 4) shows the sums of the rank of the RPD across the 54 blocks. The WNMT test shows that (SAHM, SAHm, SAHmean, SAHmb) is the best group with lower rank sums at (157.0, 138.0, 140.0, 223.0), and the group (Mpi, mpi, meanpi) is the worst group with rank sums at (453.0, 461.0, 382.0). This test confirms that the two groups of performances are statistically different at α = 0.05. The SAHm is the best overall performer for a large-sized number of orders.

Overall, the performance of the proposed simulated annealing hyper-heuristic algorithms (SAHM, SAHm, and SAmean) appear to be experimentally validated for the considered problem.

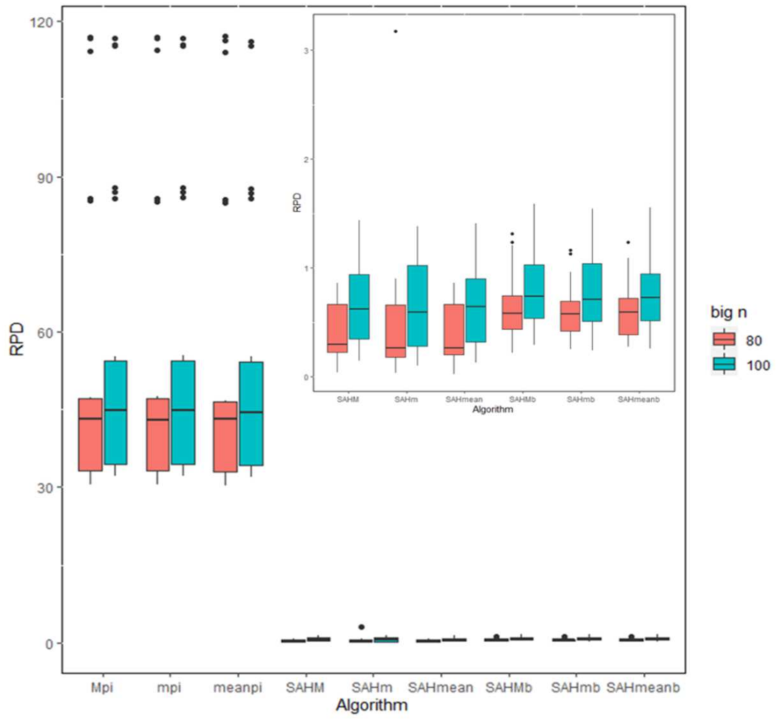

Figure 5. Boxplots of algorithms for n = 80 and n = 100.

5. Conclusions

A two-agent customer order scheduling with scenario-dependent component processing times and ready (release) times is considered in this study. The contributions are as follows: We proposed a lower bound and established six dominance properties in order to infuse them into a branch-and-bound algorithm to optimally solve this intractable problem. To obtain approximate solutions, we then developed three heuristics, Mpi, mpi, and meanpi, and six simulated annealing hyper-heuristic algorithms, SAHM, SAHm, SAHmean, SAHMb, SAHmb, and SAHmeanb. The computer simulation's experimental results showed that SAHM, SAHm, and SAHmean performed better than the other SHA algorithms and the three heuristics. Overall, although the performance of SAHmb for large-sized problem instances is not statistically and significantly different from that of the SAHM, SAHm, and SAHmean, the results suggest that SAHM, SAHm, and SAHmean can be used to solve the problem because they are very efficient and can quickly find solutions that are very close to the optimal solutions.

For a future study, we may consider performing a sensitivity analysis of the proposed algorithms. Table 2 shows that B&B can only solve up to n = 10 problem instances due to the weak ability of its properties or a lower bound. Therefore, another future study might create more powerful properties to increase the efficacy of the B&B.

Author Contributions: Conceptualization, C.-C.W., J.N.D.G., W.-C.L. and S.-R.C.; methodology, C.-C.W., J.N.D.G., W.-C.L., J.-H.C. and L.-Y.L.; software, W.-C.L., Y.-L.C. and J.-H.C.; validation, S.-R.C., J.-H.C. and L.-Y.L.; formal analysis, C.-C.W., W.-C.L. and Y.-L.C.; investigation, C.-C.W., J.N.D.G., W.-C.L. and J.-H.C.; resources, C.-C.W. and S.-R.C.; data curation, C.-C.W., W.-C.L., Y.-L.C.

and J.-H.C.; writing—original draft preparation, C.-C.W., J.N.D.G. and W.-C.L.; writing—C.-C.W., J.N.D.G., W.-C.L. and L.-Y.L.; visualization, C.-C.W., W.-C.L. and Y.-L.C.; supervision, C.-C.W., J.N.D.G., W.-C.L. and S.-R.C.; project administration, C.-C.W., J.-H.C. and L.-Y.L.; funding acquisition, C.-C.W., S.-R.C. and L.-Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research article was supported in part by the Ministry of Science and Technology of Taiwan, MOST 110-2221-E-035- 082- MY2.

Data Availability Statement: The corresponding author will provide the relative data sets upon request.

Acknowledgments: The authors would like to thank the editor and the three referees for their positive comments and useful suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ahmadi, R.; Bagchi, U. *Scheduling of Multi-Job Customer Orders in Multi-Machine Environments*; ORSA/TIMS: Philadelphia, PA, USA, 1990.
2. Ahmadi, R.; Bagchi, U. *Coordinated Scheduling of Customer Orders*; Working Paper; John E. Anderson Graduate School of Management, University of California: Los Angeles, CA, USA, 1993.
3. Gupta JN, D.; Ho, J.C.; van der Veen, J.A. Single machine hierarchical scheduling with customer orders and multiple job classes. *Ann. Oper. Res.* **1997**, *70*, 127–143. [CrossRef]
4. Baker, K.R.; Smith, J.C. A multiple-criterion model for machine scheduling. *J. Sched.* **2003**, *6*, 7–16. [CrossRef]
5. Wu, C.-C.; Wu, W.-H.; Chen, J.-C.; Yin, Y.; Wu, W.-H. A study of the single-machine two-agent scheduling problem with release times. *Appl. Soft Comput.* **2013**, *13*, 998–1006. [CrossRef]
6. Agnetis, A.; Mirchandani, P.B.; Pacciarelli, D.; Pacifici, A. Scheduling problems with two competing agents. *Oper. Res.* **2004**, *52*, 229–242. [CrossRef]
7. Daniels, R.L.; Kouvelis, P. *Robust Scheduling to Hedge against Processing Time Uncertainty in Single-Stage Production*; Working Paper; Fuqua School of Business, Duke University: Durham, NC, USA, 1992.
8. Daniels, R.L.; Kouvelis, P. Robust scheduling to hedge against processing time uncertainty in single-stage production. *Manag. Sci.* **1995**, *41*, 363–376. [CrossRef]
9. Framinan, J.M.; Perez-Gonzalez, P.; Fernandez-Viagas, V. Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures. *Eur. J. Oper. Res.* **2019**, *273*, 401–417. [CrossRef]
10. Agnetis, A.; Billaut, J.-C.; Gawiejnowicz, S.; Pacciarelli, D.; Soukhal, A. *Multiagent Scheduling: Models and Algorithms*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 10, pp. 978–983.
11. Perez-Gonzalez, P.; Framinan, J.M. A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems. *Eur. J. Oper. Res.* **2014**, *235*, 1–16. [CrossRef]
12. Braga-Santos, S.; Barroso, G.; Prata, B. A size-reduction algorithm for the order scheduling problem with total tardiness minimization. *J. Proj. Manag.* **2022**, *7*, 167–176. [CrossRef]
13. de Athayde Prata, B.; Rodrigues, C.D.; Framinan, J.M. A differential evolution algorithm for the customer order scheduling problem with sequence-dependent setup times. *Expert Syst. Appl.* **2022**, *189*, 116097. [CrossRef]
14. Pinto Antonioli, M.; Diego Rodrigues, C.; de Athayde Prata, B. Minimizing total tardiness for the order scheduling problem with sequence-dependent setup times using hybrid matheuristics. *Int. J. Ind. Eng. Comput.* **2022**, *13*, 223–236. [CrossRef]
15. Kouvelis, P.; Yu, G. *Robust Discrete Optimization and Its Applications*; Springer Science and Business Media: Berlin/Heidelberg, Germany, 1997.
16. Yang, J.; Yu, G. On the robust single machine scheduling problem. *J. Comb. Optim.* **2002**, *6*, 17–33. [CrossRef]
17. Monch, L.; Balasubramanian, H.; Fowler, J.W.; Pfund, M.E. Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal release dates. *Comput. Oper. Res.* **2005**, *32*, 2731–2750. [CrossRef]
18. Aloulou, M.A.; Della Croce, F. Complexity of single machine scheduling problems under scenario-based uncertainty. *Oper. Res. Lett.* **2008**, *36*, 338–342. [CrossRef]
19. Aissi, H.; Aloulou, M.A.; Kovalyov, M.Y. Minimizing the number of late jobs on a single machine under due date uncertainty. *J. Sched.* **2011**, *14*, 351–360. [CrossRef]
20. Kasperski, A.; Zielinski, P. Robust discrete optimization under discrete and interval uncertainty: A survey. In *Robustness Analysis in Decision Aiding, Optimization, and Analytics*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 113–143.
21. Wang, D.J.; Liu, F.; Jin, Y. A multi-objective evolutionary algorithm guided by directed search for dynamic scheduling. *Comput. Oper. Res.* **2017**, *79*, 279–290. [CrossRef]
22. Liu, F.; Wang, S.; Hong, Y.; Yue, X. On the Robust and Stable Flowshop Scheduling Under Stochastic and Dynamic Disruptions. *IEEE Trans. Eng. Manag.* **2017**, *64*, 539–553. [CrossRef]
23. Wang, J.B.; Liu, F.; Wang, J.J. Research on m-machine flow shop scheduling with truncated learning effects. *Int. Trans. Inf. Res.* **2019**, *26*, 1135–1151. [CrossRef]

24. Wang, D.J.; Liu, F.; Jin, Y. A proactive scheduling approach to steel rolling process with stochastic machine breakdown. *Nat. Comput.* **2019**, *18*, 679–694. [CrossRef]
25. Wu, C.-C.; Gupta, J.N.D.; Cheng, S.-R.; Lin, B.M.; Yip, S.-H.; Lin, W.-C. Robust scheduling for a two-stage assembly shop with scenario-dependent processing times. *Int. J. Prod. Res.* **2021**, *59*, 5372–5387. [CrossRef]
26. Xing, L.; Liu, Y.; Li, H.; Wu, C.-C.; Lin, W.-C.; Chen, X. A Novel Tabu Search Algorithm for Multi-AGV Routing Problem. *Mathematics* **2020**, *8*, 279. [CrossRef]
27. Ren, T.; Zhang, Y.; Cheng, S.-R.; Wu, C.-C.; Zhang, M.; Chang, B.-Y.; Wang, X.-Y.; Zhao, P. Effective Heuristic Algorithms Solving the Jobshop Scheduling Problem with Release Dates. *Mathematics* **2020**, *8*, 1221. [CrossRef]
28. Wu, C.C.; Bai, D.; Zhang, X.; Cheng, S.R.; Lin, J.C.; Wu, Z.L.; Lin, W.C. A robust customer order scheduling problem along with scenario-dependent component processing times and due dates. *J. Manuf. Syst.* **2021**, *58*, 291–305. [CrossRef]
29. Zegordi, S.H.; Yavari, M. A branch and bound algorithm for solving large-scale single-machine scheduling problems with non-identical release dates. *Eur. J. Ind. Eng.* **2018**, *12*, 24–42. [CrossRef]
30. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
31. Bıçakcı, P.S.; Derya, T.; Kara, İ. Solution approaches for the parallel machine order acceptance and scheduling problem with sequence-dependent setup times, release dates and deadlines. *Eur. J. Ind. Eng.* **2021**, *15*, 295–318. [CrossRef]
32. Azimi, Z.N. Comparison of metaheuristic algorithms for Examination Timetabling Problem. *J. Appl. Math. Comput.* **2004**, *16*, 337. [CrossRef]
33. Anagnostopoulos, K.P.; Koulinas, G.K. A genetic hyper heuristic algorithm for the resource constrained project scheduling problem. In Proceedings of the IEEE Congress on Evolutionary Computation, Barcelona, Spain, 27 September 2010; pp. 1–6.
34. Bai, R.; Blazewicz, J.; Burke, E.K.; Kendall, G.; McCollum, B. A simulated annealing hyper-heuristic methodology for flexible decision support. *4OR* **2012**, *10*, 43–66. [CrossRef]
35. Liang, X.-X.; Zhang, B.; Wang, J.-B.; Yin, N.; Huang, X. Study on flow shop scheduling with sum-of-logarithm-processing-times-based learning effects. *J. Appl. Math. Comput.* **2019**, *61*, 373–388. [CrossRef]
36. Lenstra, J.K.; Rinnooy Kan, A.R.; Brucker, P. Complexity of machine scheduling problems. *Ann. Discret. Math.* **1977**, *1*, 343–362.
37. Ogan, D.; Azizoglu, M. A branch and bound method for the line balancing problem in U-shaped assembly lines with equipment requirements. *J. Manuf. Syst.* **2015**, *36*, 46–54. [CrossRef]
38. Hardy, G.; Littlewood, J.; Polya, G. *Inequalities*; Cambridge Mathematical Library Series; Cambridge University Press: Cambridge, UK, 1967.
39. Reeves, C. Heuristics for scheduling a single machine subject to unequal job release times. *Eur. J. Oper. Res.* **1995**, *80*, 397–403. [CrossRef]
40. Hollander, M.; Wolfe, D.A.; Chicken, E. *Nonparametric Statistical Methods*; John Wiley & Sons: Hoboken, NJ, USA, 2013; Volume 751.