

## LOT SCHEDULING ON A SINGLE MACHINE TO MINIMIZE TOTAL WEIGHTED COMPLETION TIME

FEIFENG ZHENG<sup>1</sup>, NA LI<sup>1</sup>, YINFENG XU<sup>2</sup> AND MING LIU<sup>3,\*</sup>

**Abstract.** Lot scheduling is one of the important production patterns in modern manufacturing systems. Each lot processes simultaneously one or more customer orders with a total size no more than the fixed lot capacity, consuming a uniform lot processing time. In this work we consider the single machine lot scheduling environment where any order can be split and processed in consecutive lots. The objective is to minimize total weighted completion time. We first prove the NP-hardness of the considered problem, and propose a polynomial-time algorithm with approximation ratio equal to the ratio of the largest to the smallest order weight. Moreover, we explore two special cases. For the first case where the total size of all the orders is at most twice of the lot capacity, a dynamic programming algorithm is provided. In the last special case, the sizes and weights of orders are in reverse-agreeable, *i.e.*, a larger size of an order implies an equal or smaller weight. We prove that processing orders in the non-increasing sequence of their weights results in an optimal schedule, implying that the case is solvable in polynomial time. Finally, experimental results demonstrate the effectiveness of the approximation algorithm.

**Mathematics Subject Classification.** 90B36.

Received January 16, 2024. Accepted April 10, 2025.

### 1. INTRODUCTION

Lot scheduling, which processes multiple orders simultaneously at a time, is a crucial manufacturing mode in the domain of scheduling and has extensive applications in the manufacturing industry [1, 2, 4, 10, 18]. One of the applications is the burn-in operation of integrated circuit (IC) final testing at semiconductor factories. Each oven (regarded as a processing lot) has the capacity to process a large number of integrated circuits, which may be demanded by one or more customers. The duration of the burn-in processing in each lot is predetermined depending on the parameter requirements of the IC products. Some other examples of lot scheduling can be observed in glue production *via* heating vessels and stone cleaning and processing of apparel products.

In the field of single machine lot scheduling, most studies have focused on the the settings with identical lot processing time and no weight of order. Yang *et al.* [20] examined the case where orders are indivisible, aiming to minimize the total completion time. They established the NP-hardness of the problem, and devised

---

*Keywords.* Lot scheduling, single machine, order-splitting, weighted completion time, dynamic programming.

<sup>1</sup> Glorious Sun School of Business and Management, Donghua University, Shanghai 200051, P.R. China.

<sup>2</sup> School of Management, Xi'an Jiaotong University, Xi'an 710049, P.R. China.

<sup>3</sup> School of Economics & Management, Tongji University, Shanghai 200092, P.R. China.

\*Corresponding author: [mingliu@tongji.edu.cn](mailto:mingliu@tongji.edu.cn)

a binary integer programming approach along with four heuristic algorithms for solving the problem. Among these algorithms, BFN (Best Fit Non-decreasing) demonstrated the best performance. Zheng and Jin [22] also investigated the model addressed by Yang *et al.* [20], and presented an enhanced heuristic algorithm that is of improved performance *via* numerical experiments.

Some researchers introduced the function of order splitting into the single machine lot scheduling problem. Hou *et al.* [6] showed that with the splitting ability, the single machine lot scheduling problem to minimize total completion time of orders can be solved by the SPT (Shortest Processing Time first) rule in polynomial time. Kovalyov [9] demonstrated that when modeling the problem of Hou *et al.* [6] as a classic batching machine scheduling problem, its optimal solution becomes evident. Mor and Mosheiov [14] tackled a single machine lot scheduling model with order splitting, incorporating common due-dates/due-windows. For the objective of minimizing total processing cost that is associated with earliness and tardiness, they presented dynamic programming algorithms running in polynomial time to solve the problem. Shen and Geng [19] proved the strong NP-hardness through a polynomial reduction for the problem where orders have deadlines. They also claimed that the model with agreeable due date and deadline as well as order size is polynomial solvable. Liu *et al.* [12] considered two models of single machine lot scheduling involving due dates, aiming to minimize the maximum lateness and total tardiness, respectively. They revealed that the EDD (Earliest Due-Date first) rule and SPT rule can optimally solve the two problems, respectively, under the condition that the due date and order size are agreeable, *i.e.*, larger due dates imply larger order sizes.

Mor [13] addressed a new variant where lot processing times are position-dependent in any schedule, *i.e.*, the model with non-identical processing times of lots. They investigated three minimization objectives: total completion time, makespan, and a linear combination of the first two. It was demonstrated that all the three models can be solved in polynomial time. Mor *et al.* [16] explored another variant where the option of rejection is considered in processing orders. They mainly constructed a pseudo-polynomial dynamic programming algorithm to efficiently solve the problem.

Some recent works have also paid attention to the single machine lot scheduling problem where orders are of different weights. Zhang *et al.* [21] studied the model of minimizing total weighted (discounted) completion time where orders splitting is allowed. They showed that the WSPT (Weighted Shortest Processing Time first) rule can optimally solve the problem. Mor *et al.* [20] addressed a single machine lot scheduling problem aiming at minimizing the weighted number of tardy orders. They established the NP-hardness of the problem and presented a pseudo-polynomial dynamic programming solution. Chen *et al.* [3] extended the model addressed in Zhang *et al.* [20] to a more general case, in which processing lots are of non-uniform capacities and time lengths, and orders may be of unbounded sizes. For the objective of minimizing total weighted (discounted) completion time, they concluded that the WSPT rule is able to solve the problem optimally. Mosheiov and Sarig [17] investigated a single machine lot scheduling problem with the minimization objective of maximum weighted tardiness of orders, and constructed a modified Lawler's algorithm to solve it. We summarize the majority of relevant studies in Table 1, in which the models are depicted using the classical three-field notation introduced by Graham *et al.* [5]. With regard to the models with order splitting, we observe that in literature there are two categories of calculating completion time of a split order according to its delivery mode after processing. In the former category, if an order is split and processed in two consecutive lots, then each part of the order is immediately delivered to the customer upon its completion. Therefore, the completion time of the order is a weighted summation of the two lots' completion times. More precisely, suppose that order  $O_j$  is processed in the  $[r]$ -th and  $[r+1]$ -th lots (*i.e.*,  $L_{[r]}$  and  $L_{[r+1]}$ ) with proportions of  $\delta$  and  $(1-\delta)$ , respectively, where  $0 < \delta < 1$ . Then the completion time of the order is defined as  $C_j = \delta rP + (1-\delta)(r+1)P$ . Note that lots  $L_{[r]}$  and  $L_{[r+1]}$  complete at time points  $rP$  and  $(r+1)P$ , respectively. We call this calculation method *combined calculation* (abbr. *comb*) in Table 1 [3, 6, 12, 17, 20–22].

In the latter category, different parts of one split order are required to be sent to the customer in a single delivery upon the completion of processing the last part of the order. The completion of the order is equal to the end of the lot that processes its last part. Assume that order  $O_j$  is processed in lots  $L_{[r]}$  and  $L_{[r+1]}$ , the completion time of the order is defined as  $C_j = (r+1)P$  [15, 16, 19]. In this work, we consider the latter

TABLE 1. Summary of related study on lot scheduling.

Related work	Problem	Complexity	Solution method
[6]	1  lot, comb, split  $\sum_j C_j$	$O(n \log n)$	SPT
[20, 22]	1  lot, no-split  $\sum_j C_j$	NP-hard	BFN & IBFN
[9]	1  lot, comb, split  $\sum_j C_j^A$	$O(n \log n)$	LWFB
[21]	1  lot, comb, split  $\sum_j w_j C_j$	$O(n \log n)$	WSPT
	1  lot, comb, split  $\sum_j w_j C_j (1 - e^{-dC_j})$	$O(n \log n)$	WSPT
[15]	1  lot, split  $\sum_j U_j$	$O(n \log n)$	AMA
	1  lot, split  $\sum_j w_j U_j$	NP-hard	DP
	1  lot, no-split  $\sum_j U_j$	NP-hard	DP
[13]	1  lot, comb, split, $P_{ir}$   $\sum_j C_j$	$O(n^3)$	LOS-LP
	1  lot, comb, split, $P_{ir}$   $\sum_j w_j C_j$	$O(n^3)$	WLOS-LP
	1  lot, comb, split, $P_{ir}$   $C_{\max}$	$O(n^3)$	LP
	1  lot, comb, split, $P_{ir}$   $\delta C_{\max} + (1 - \delta) \sum_j C_j$	$O(n^3)$	LOS-LP
[14]	1  lot, split, $d$   $\alpha \sum_j E_j + \beta \sum_j E_j + n\gamma d$	NP-hard	DP
	1  lot, split, $D$   $\alpha \sum_j E_j + \beta \sum_j E_j + n\gamma d_1 + n\delta D$	NP-hard	DP
[16]	1  lot, comb, split, $\sum_j R_j$   $C_{\max}$	NP-hard	DP
	1  lot, comb, split, $\sum_j R_j$   $\sum_j C_j$	NP-hard	DP
	1  lot, comb, split, $\sum_j R_j$   $\sum_j w_j C_j$	NP-hard	DP
[3]	1  lot, comb, split, $k_r, u_r$   $\sum_j w_j C_j$	$O(n \log n)$	WSPT
	1  lot, comb, split, $k_r, u_r$   $\sum_j w_j D_j$	$O(n \log n)$	WSPT
[19]	1  lot, split, $\bar{d}_j$   $\sum_j U_j$	NP-hard	-
	1  lot, split, agreeable  $\sum_j U_j$	$O(n \log n)$	-
	1  lot, split, $\sigma_j = \sigma$   $\sum_j U_j$	$O(n \log n)$	-
[17]	1  lot, split, $\bar{d}_j, d_j = d$   $\sum_j U_j$	$O(n \log n)$	-
	1  lot, comb, split  $T_{\max}^W$	-	MAL
	1  lot, comb, no-split  $T_{\max}^W$	NP-hard	-
[12]	1  lot, comb, split  $\sum_j L_j$	-	MILP
	1  lot, comb, split  $L_{\max}$	-	MILP
	1  lot, comb, split, agreeable  $\sum_j L_j$	$O(n \log n)$	SPT
	1  lot, comb, split, agreeable  $L_{\max}$	$O(n \log n)$	EDD
This work	1  lot, split  $\sum_j w_j C_j$	NP-hard	LOS
	1  lot, split, $\sum_j \sigma_j \leq 2k$   $\sum_j w_j C_j$	NP-hard	DP
	1  lot, split, reverse-agreeable  $\sum_j w_j C_j$	$O(n \log n)$	LWF

**Notes.**  $T_{\max}^W = \max_j w_j T_j$ ; SPT: Smallest Processing Time first; BFN: Best Fit Non-decreasing; IBFN: Improved Best Fit Non-decreasing; LWFB: Largest Weight Full Batch; WSPT: Weighted Shortest Processing Time first; AMA: Adapted Moore’s Algorithm; DP: Dynamic Programming; LOS-LP: Least Order Size first rule plus Linear Programming; WLOS-LP: Weighted Least Order Size first rule plus Linear Programming; LP: Linear Programming; MLA: Modified Lawler’s Algorithm; LPT: Largest Processing Time; MILP: Mixed Integer Linear Programming; EDD: Earliest Due-Date first; LOS: Least Order Size first; LWF: Largest Weight First; notation “-” means that no solution method is provided.

category of calculating  $C_j$  since it is an easier going delivery mode in operational management. The objective is to minimize total weighted completion time. Denote the problem under consideration as 1|lot, split|  $\sum_j w_j C_j$ . It shall be noted that Zhang *et al.* [21] studied the model closest to the one in this work, while they adopted the former category of calculating completion time of an order.

In this paper, we first investigate the complexity of the considered problem and propose a polynomial approximation algorithm to solve it. We then focus on two special cases of the problem. In the first case, we assume that the total size of all orders is less than  $2k$ , where  $k$  is lot capacity. We propose a dynamic programming algorithm for the first case. In the second case, we assume that the size and weight of any order are reverse-agreeable, *i.e.*,

a larger order size implies a smaller weight. We prove that processing orders in the non-increasing sequence of their weights yields an optimal schedule. This implies that the second case can be solved in polynomial time. The main contributions of this work are as follows:

- (1) For the general problem  $1|lot, split|\sum_j w_j C_j$ , we establish that it is NP-hard, and then provide a polynomial approximation algorithm named *LOS* (Least Order Size first).
- (2) For special model  $1|lot, split, \sum_j \sigma_j \leq 2k|\sum_j w_j C_j$ , we propose a dynamic programming algorithm with complexity  $O(nk)$ .
- (3) For the second special model  $1|lot, split, reverse-agreeable|\sum_j w_j C_j$ , we prove that the *LWF* (Largest Weight First) rule produces an optimal processing schedule.

The remaining of this work is organized as follows. In Section 2, we provide a comprehensive explanation of fundamental notations and problem formulation. In Section 3, we prove that the considered problem is NP-hard and present a polynomial approximation algorithm. Section 4 makes a further analysis of two special cases. Section 5 conducts numerical experiments in order to verify the effectiveness of the proposed polynomial time approximation algorithm. Finally, in Section 6 we conclude this work and suggest the prospect of future research.

## 2. PROBLEM DESCRIPTION

The general problem  $1|lot, split|\sum_j w_j C_j$  considered in this work can be described as follow. There are  $n$  customer orders to be processed on a single machine in the lot processing mode. More than one order can be simultaneously processed in a lot with a total size no more than the lot capacity of  $k$ . Each lot has a uniform processing time of  $P$ , and all the orders in the lot are of identical completion time equal to the end of the lot. Each order  $O_j$  is of size  $\sigma_j$  ( $0 < \sigma_j \leq k$ ) and weight  $w_j$  for  $1 \leq j \leq n$ .

We consider the scenario where orders can be split and processed in consecutive lots. As any order is of size no more than the lot capacity  $k$ , each split order is processed in at most two consecutive lots. We adopt the previously mentioned second category of calculating completion time of split order [15, 16, 19]. More precisely, Given a processing schedule, assume that one split order  $O_j$  is processed in the  $[r]$ -th and  $[r + 1]$ -th lots. The completion time of the order is equal to  $C_j = (r + 1)P$ . Note that the first lot starts at time zero, and there is no idleness between any adjacent processing lots in any reasonable schedule. Our goal is to minimize total weighted completion time, *i.e.*,  $\sum w_j C_j$ . We consider a general model and two special models. Using the classical three-field notation, we denote the three models P1 to P3 as follows:

- P1:  $1|lot, split|\sum_j w_j C_j$ ;  
 P2:  $1|lot, split, \sum_j \sigma_j \leq 2k|\sum_j w_j C_j$ ;  
 P3:  $1|lot, split, reverse-agreeable|\sum_j w_j C_j$ .

In model P3, *reverse-agreeable* means the size and weight of order are reverse-agreeable, *i.e.*, a larger size of order implies a smaller weight.

Fundamental notations in this work are summarized in Table 2.

## 3. THE GENERAL PROBLEM P1

In this section, we investigate the complexity of the general problem  $1|lot, split|\sum_j w_j C_j$ , and propose an approximation algorithm for the problem.

### 3.1. Problem complexity

For problem  $1|lot, split|\sum_j w_j C_j$ , there exists the following conclusion on its complexity.

**Theorem 3.1.** *Problem  $1|lot, split|\sum_j w_j C_j$  is NP-hard in the ordinary sense.*

TABLE 2. Fundamental notations.

Symbol	Definition
$O_j$	Order $j$ , $j = 1, 2, 3, \dots, n$ where $n$ is the number of orders
$L_{[r]}$	The $r$ -th processing lot in a schedule, $r = 1, 2, \dots, N$
$P$	Uniform processing time of any lot
$k$	Capacity of any lot
$w_j$	Weight of order $O_j$
$w_{\max}$	Maximum weight of order
$w_{\min}$	Minimum weight of order
$\sigma_j$	Size of order $O_j$ , $\sigma_j \leq k$
$I$	Input instance of orders
$\mathcal{B}$	Schedule produced by algorithm LOS for instance $I$
$\mathcal{B}^*$	An optimal schedule for instance $I$
$C_j(\mathcal{B})$	Completion time of order $O_j$ in $\mathcal{B}$
$C_j(\mathcal{B}^*)$	Completion time of order $O_j$ in $\mathcal{B}^*$

*Proof.* We prove the theorem by reducing the well-known **the Knapsack problem**, which is NP-complete, to the considered problem.

**The Knapsack problem:** There are  $n$  items with positive weights  $a_1, \dots, a_n$  and values  $v_1, \dots, v_n$ , and a Knapsack with capacity  $W (> 0)$ . The aim is to select a subset of items with maximum total value, while their total weight does not exceed  $W$ . Mathematically, to maximize  $\sum_i v_i x_i$ , subject to  $\sum_i a_i x_i \leq W$ ,  $x_i \in \{0, 1\}$ ; where  $a_i$ ,  $v_i$ , and  $W$  are known integers.

Based on the Knapsack problem, we construct the following instance for the lot scheduling problem. Assume without loss of generality that any lot is of unit capacity and unit processing time. The  $n$  orders are of a total size strictly larger than 1 but at most 2 in the instance. Thus any reasonable processing schedule consists of two processing lots with completion times equal to 1 and 2, respectively. Let  $TW$  and  $TW_1$  represent the total weight of all the orders and that of the orders assigned to the first processing lot, respectively. Thus, the total weight of orders in the second lot is  $TW - TW_1$ .

For this instance, the objective value is equal to the total weight of orders in the first lot plus twice that of orders in the second lot, *i.e.*,  $\sum_j w_j C_j = TW_1 + 2(TW - TW_1) = 2TW - TW_1$ . It is intuitive that minimizing the objective value is equivalent to maximizing the total order weight  $TW_1$  of the first lot. We observe that packing orders into the first processing lot reduces to the Knapsack problem, in which the Knapsack capacity  $W = 1$ , the item weight  $a_j = \sigma_j$  and item value  $v_j = w_j$ . Notice that if  $W < \sum_j a_j / 2$  in the original Knapsack problem, then resetting a new value of  $W = \sum_j a_j / 2$  does not change the nature of the problem.

As is well known, the Knapsack problem is an NP-complete problem [8]. Therefore, our problem  $1|lot, split|\sum_j w_j C_j$  is NP-hard in the ordinary sense. It completes the proof.  $\square$

It should be noted that whether problem P1 is strongly NP-hard or pseudo-polynomially solvable remains open.

### 3.2. Approximation algorithm

We present an approximation algorithm, which is called *LOS* (Least Order Size first), for the problem  $1|lot, split|\sum_j w_j C_j$ . That is, algorithm LOS processes orders in the non-decreasing sequence of their sizes. The detailed description of the algorithm is as follows.

#### Algorithm LOS

- (1) Sort the orders in the non-decreasing sequence of their sizes, *i.e.*,  $\sigma_1 \leq \sigma_2 \leq \dots \leq \sigma_n$ .
- (2) Compute the number of processing lots  $N = \lceil \frac{\sum_{j=1}^n \sigma_j}{k} \rceil$ .

(3) Assign sorted orders to processing lots one by one.

Notice that with the ability of order splitting, all the processing lots except the last one are full in the LOS schedule. Next we give an analysis on the approximation ratio of algorithm LOS.

For notation convenience, denote by  $w_{\max}, w_{\min}$  the maximum and minimum weight of an order, respectively. Given any order input instance, let  $\mathcal{B}$  be the schedule generated by algorithm LOS, and  $\mathcal{B}^*$  an optimal offline schedule. Denote by  $F(\mathcal{B})$  and  $F(\mathcal{B}^*)$  the objective value of the two schedules, respectively. Denote by  $C_j(\mathcal{B}), C_j(\mathcal{B}^*)$  the completion time of order  $O_j$  in schedules  $\mathcal{B}$  and  $\mathcal{B}^*$ , respectively.

**Theorem 3.2.** *For problem 1|lot, split| $\sum_j w_j C_j$ , algorithm LOS has the approximation ratio of  $w_{\max}/w_{\min}$ , that is,*

$$\frac{F(\mathcal{B})}{F(\mathcal{B}^*)} \leq \frac{w_{\max}}{w_{\min}}.$$

*Proof.* We prove the theorem in two steps. We first prove that LOS completes the  $[j]$ -th order, denoted by  $O_{[j]}$ , earliest over all the feasible processing schedules. This conclusion can be obtained by the following reasoning.

LOS handles orders in the non-decreasing sequence of their sizes, i.e.,  $\mathcal{B} = (O_1, O_2, \dots, O_n)$  satisfying  $\sigma_1 \leq \sigma_2 \leq \dots \leq \sigma_n$ . Therefore,  $O_{[j]} = O_j$  holds for the schedule  $\mathcal{B}$ . Assume otherwise in another processing schedule  $\delta = (O'_{[1]}, O'_{[2]}, \dots, O'_{[n]})$ , at least one order breaches the LOS rule. Let  $O'_{[j]}$  be the first order with  $O'_{[j]} \neq O_j$  where  $1 \leq j < n$ , i.e.,  $O'_{[j]}$  is not the  $[j]$ -smallest order in the schedule  $\delta$ . By the above argument, the size of order  $O'_{[j]}$  satisfies  $\sigma'_{[j]} > \sigma_j$  holds.

Since  $O'_{[j]}$  is the first order breaching the LOS rule in schedule  $\delta$ , we claim that  $O_j$  must be processed in some position later than  $O'_{[j]}$  in the schedule. Assume that  $O_j$  is in the  $[k]$ -th position of  $\delta$ , i.e.,  $O_j = O'_{[k]}$  with  $j < k \leq n$ . We claim that swapping the processing positions of the two orders  $O'_{[j]}$  and  $O'_{[k]} (= O_j)$  while keeping the other orders unchanged in the schedule  $\delta$  can result in smaller or equal completion time of the  $[j]$ -th order for  $1 \leq j \leq n$ .

More precisely, order  $O'_{[k]}$  is in the  $[j]$ -th position while  $O'_{[j]}$  is in the  $[k]$ -th position after swapping. As  $\sigma'_{[k]} < \sigma'_{[j]}$ , the  $[j]$ -th order (i.e.,  $O'_{[k]}$ ) as well as all the orders  $O'_{[j+1]}, \dots, O'_{[k-1]}$  between the swapped ones is completed in the same or an earlier lot during swapping. Notice that each order  $O'_{[u]}$  for  $u = j + 1, \dots, k - 1$  remains in the  $[u]$ -th position during swapping. Order  $O'_{[j]}$  after swapping has the same completion time as  $O'_{[k]}$  before swapping. That is, the  $[k]$ -th order keeps its completion time unchanged during swapping because the total size of the first  $[k]$  orders remains unchanged. In sum, the completion time of the  $[u]$ -th order for  $u = j, j + 1, \dots, k - 1$  either decreases or remains the same during the swapping. All the remaining orders  $O'_{[1]}, \dots, O'_{[j-1]}, O'_{[k]}, O'_{[k+1]}, \dots, O'_{[n]}$  keep their completion times unchanged during the swapping.

For other orders breaching the LOS rule in schedule  $\delta$ , they can be similarly swapped and discussed as order  $O'_{[j]}$  does, and the same conclusion can be derived. Therefore, we conclude that in the LOS schedule, the  $[j]$ -th order for  $1 \leq j \leq n$  has the smallest completion time over all the feasible processing schedules. It follows that  $C_{[j]}(\mathcal{B}) \leq C_{[j]}(\mathcal{B}^*)$ .

Next we bound the ratio of  $F(\mathcal{B})/F(\mathcal{B}^*)$ . For the LOS schedule, together with  $w_{[j]} \leq w_{\max}$  for  $1 \leq j \leq n$ ,

$$F(\mathcal{B}) = \sum_{j=1}^n w_{[j]} C_{[j]}(\mathcal{B}) \leq w_{\max} \sum_{j=1}^n C_{[j]}(\mathcal{B}).$$

For the optimal schedule, combining  $C_{[j]}(\mathcal{B}) \leq C_{[j]}(\mathcal{B}^*)$  and  $w_{\min} \leq w_{[j]}$  for  $1 \leq j \leq n$ ,

$$F(\mathcal{B}^*) = \sum_{j=1}^n w_{[j]} C_{[j]}(\mathcal{B}^*) \geq \sum_{j=1}^n w_{[j]} C_{[j]}(\mathcal{B}) \geq w_{\min} \sum_{j=1}^n C_{[j]}(\mathcal{B}).$$

Therefore,  $\frac{F(\mathcal{B})}{F(\mathcal{B}^*)} \leq \frac{w_{\max}}{w_{\min}}$  holds.

The proof of the theorem is completed. □

Obviously, the analysis with approximation ratios shows that this bound is tight when the order weights are all equal. It is noteworthy that for unequal weights, it remains an open question whether this bound is tight.

For the corresponding unweighted case, *i.e.*,  $w_j = 1$  ( $1 \leq j \leq n$ ), we have  $w_{\max} = w_{\min} = 1$ , and thus the following corollary is derived by the above theorem.

**Corollary 3.1.** *For problem,  $1|lot, split|\sum_j C_j$ , an optimal schedule exists in which orders are sequenced in the LOS rule.*

#### 4. TWO SPECIAL MODELS

In this section, we mainly analyze the two special models, P2:  $1|lot, split, \sum_j \sigma_j \leq 2k|\sum_j w_j C_j$  and P3:  $1|lot, split, reverse-agreeable|\sum_j w_j C_j$ , and provide polynomial algorithms for generating optimal solutions.

##### 4.1. Problem P2

We consider problem P2, *i.e.*,  $1|lot, split, \sum_j \sigma_j \leq 2k|\sum_j w_j C_j$  where  $\sum_j \sigma_j \leq 2k$  means that the total size of all orders is less than twice of the lot capacity. With a limited total size of orders at most  $2k$ , we claim that there are at most two processing lots in an optimal solution. Moreover, the objective function of minimizing total weighted completion time is equivalent to maximizing total weighted completion time (or equivalently total weight of orders) in the first processing lot. The latter problem reduces to the classic 0–1 Knapsack problem.

Based on the above observation, in the following we propose a dynamic programming algorithm (DP) to solve problem P2 for the case where the lot capacity  $k$  and any order size  $\sigma_j$  ( $1 \leq j \leq n$ ) are all integers. For the considered problem, we focus on the case with integer lot capacity and order sizes. Notice that for the case where at least one order is with a non-integer size, we can equally amplify the sizes of all the orders as well as the lot capacity  $k$  to make all the order sizes and lot capacity become integers.

At each recursive process of the DP, it decides whether to assign order  $O_j$  to the first processing lot. There are two selections, that is, processing and completing the order either in the first lot or in the second lot. Let  $f(j, \mathcal{R})$  denote the total weight of orders being assigned to the first processing lot, given the first  $j$  orders and a (remaining) lot capacity of  $\mathcal{R}$ . Note that the value of  $\mathcal{R}$  varies in  $[0, k]$ . The dynamic programming recursive equation can be obtained as follow.

$$f(j, \mathcal{R}) = \begin{cases} \max \{f(j - 1, \mathcal{R}), f(j - 1, \mathcal{R} - \sigma_j) + w_j\}, & \mathcal{R} \geq \sigma_j \\ f(j - 1, \mathcal{R}), & \mathcal{R} < \sigma_j. \end{cases} \tag{1}$$

Note that  $f(j - 1, \mathcal{R} - \sigma_j)$  is not defined for the case where  $\mathcal{R} < \sigma_j$ . We explain the two items inside the braces of the above formula (for the case where  $\mathcal{R} \geq \sigma_j$ ). On the decision of order  $O_j$ , it is either assigned to the first processing lot or not.

**Case 1.** Order  $O_j$  is not assigned to the first lot (and thus assigned to the second lot). In this case, the recursive sub-problem will be transformed to assigning the first  $j - 1$  orders into a processing lot with capacity of  $\mathcal{R}$ , *i.e.*, maximizing  $f(j - 1, \mathcal{R})$ . Since  $O_j$  is in the second lot,  $f(j, \mathcal{R}) = f(j - 1, \mathcal{R})$  in this case;

**Case 2.** Order  $O_j$  is assigned to the first lot. Then the recursive sub-problem becomes assigning the first  $j - 1$  orders into a lot with the remaining capacity of  $\mathcal{R} - \sigma_j$  (*i.e.*, maximizing  $f(j - 1, \mathcal{R} - \sigma_j)$ ). The objective value of the current sub-problem is  $f(j, \mathcal{R}) = f(j - 1, \mathcal{R} - \sigma_j) + w_j$  in this case.

The value of  $f(j, \mathcal{R})$  is the maximum in Cases 1 and 2. The boundary conditions are:  $f(0, \mathcal{R}) = 0$  for  $0 \leq \mathcal{R} \leq k$ , and  $f(j, 0) = 0$  for  $1 \leq j \leq n$ . That is, the total weighted completion time is 0 if there is no order to be processed or there is no available space in the first lot to contain any order. We have the following conclusion on the time complexity of the DP algorithm.

**Theorem 4.1.** *The computational complexity of DP is  $O(nk)$ .*

*Proof.* Given the boundary conditions, the recursive function (*i.e.*, Formula (1)) is calculated in  $n$  steps, in each of which a single order  $O_j$  is assigned. When solving  $f(j, \mathcal{K})$  for assigning order  $O_j$ , the values of both  $f(j-1, \mathcal{K})$  and  $f(j-1, \mathcal{K} - \sigma_j)$  are used. So, it is necessary to access the assignment results for the previous order  $O_{j-1}$  over all possible remaining lot capacities  $\mathcal{K}$  ( $0 \leq \mathcal{K} \leq k$ ), which causes  $O(k)$ -time. As there are totally  $n$  orders, the computational complexity of DP is  $O(nk)$ . It completes the proof.  $\square$

*Illustrative example.* Below we use DP to solve an instance with six orders. In the instance, the lot capacity  $k = 10$  and lot processing time  $P = 3$ . The six orders are of sizes  $\{5, 1, 2, 4, 3, 2\}$  and weights  $\{3, 1, 6, 3, 3, 5\}$ .

In the initialization,  $f(0, \mathcal{K}) = 0$  for  $0 \leq \mathcal{K} \leq 10$ , and  $f(j, 0) = 0$  for  $1 \leq j \leq 6$ .

The first recursive step is to assign the first order  $O_1$ , and we have by formula (1) that,

$$f(1, \mathcal{K}) = \max\{f(0, \mathcal{K}), f(0, \mathcal{K} - 5) + 3\} = \max\{0, 0 + 3\} = 3$$

for  $\mathcal{K} = 5, 6, 7, 8, 9, 10$ ; and

$$f(1, \mathcal{K}) = \max\{f(0, \mathcal{K}), f(0, \mathcal{K} - 5) + 3\} = f(0, \mathcal{K}) = 0$$

for  $\mathcal{K} = 1, 2, 3, 4$ , since  $f(0, \mathcal{K} - 5)$  is an infeasible function and not defined when  $\mathcal{K} < 5$ . The next recursive step is to assign the second order  $O_2$ . Similarly,

$$f(2, \mathcal{K}) = \max\{f(1, \mathcal{K}), f(1, \mathcal{K} - 1) + 1\} = \max\{3, 3 + 1\} = 4$$

for  $\mathcal{K} = 6, 7, 8, 9, 10$ ;

$$f(2, \mathcal{K}) = \max\{f(1, \mathcal{K}), f(1, \mathcal{K} - 1) + 1\} = \max\{3, 0 + 1\} = 3$$

for  $\mathcal{K} = 5$ ; and

$$f(2, \mathcal{K}) = \max\{f(1, \mathcal{K}), f(1, \mathcal{K} - 1) + 1\} = \max\{0, 0 + 1\} = 1$$

for  $\mathcal{K} = 1, 2, 3, 4$ .

With the similar reasoning, DP generates an optimal solution as follows: Orders  $O_1, O_2, O_3, O_6$  are processed in the first lot, and orders  $O_4$  and  $O_5$  are processed in the second lot. The maximum value of  $f(6, 10) = 15$  is obtained. Hence, the completion times of the orders are  $C_1 = C_2 = C_3 = C_6 = 3$  and  $C_4 = C_5 = 6$ . The minimum total weighted completion time for the problem P3 is  $\sum_j w_j C_j = (3 + 1 + 6 + 5) \times 3 + (3 + 3) \times 6 = 81$ .

## 4.2. Problem P3

In this subsection, we consider the other special case problem P3. *i.e.*,  $1|lot, split, reverse-agreeable|\sum_j w_j C_j$ , the reverse-agreeable delineates a scenario wherein any order with a smaller size is intrinsically linked to a larger weight. This phenomenon is particularly evident in the context of micro-manufacturing. In practice, a recognized correlation exists between product size and manufacturing difficulty and cost. Despite their small sizes, miniaturized products have a significant impact on the performance of electronic devices. The implementation of more advanced process technologies (*e.g.*, 5 nm, 3 nm) facilitates enhanced levels of integration, reduced power consumption, and improved operational speeds. As a result, these miniaturized products are often assigned higher weights (or priorities) when evaluating their importance and market value. We refer to the above correlation as *reverse-agreeable* in this work.

For this problem, we mainly prove that the LWF (Largest Weight First) rule is an optimal solution approach. That is, the processing sequence of the orders satisfies  $w_1 \geq w_2 \geq \dots \geq w_n$  in the optimal (*i.e.*, LWF) schedule.

**Theorem 4.2.** *For problem  $1|lot, split, reverse-agreeable|\sum_j w_j C_j$ , scheduling orders in the non-increasing sequence of their weights is optimal.*

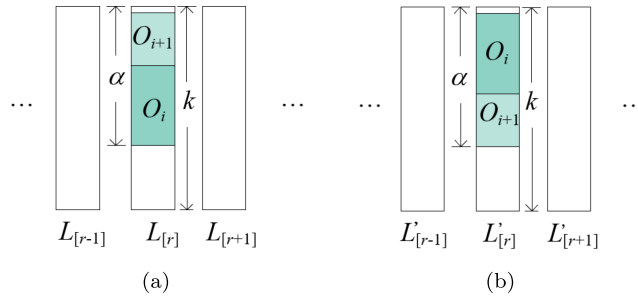


FIGURE 1. Illustration of processing orders  $O_i$  and  $O_{i+1}$  in Case 1. (a) Schedule  $\mathcal{L}$ . (b) Schedule  $\mathcal{L}'$ .

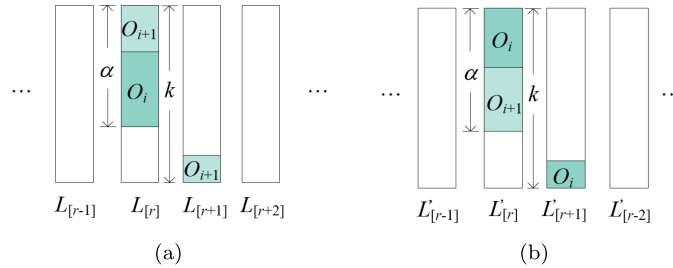


FIGURE 2. Illustration of processing  $O_i$  and  $O_{i+1}$  in Case 2.1. (a) In the schedule  $\mathcal{L}$ . (b) In the schedule  $\mathcal{L}'$ .

*Proof.* We prove this theorem by the idea of order interchange. Suppose that there are two schedules  $\mathcal{L} = (L_{[1]}, L_{[2]}, \dots, L_{[r]}, L_{[r+1]}, L_{[r+2]}, \dots)$  and  $\mathcal{L}' = (L'_{[1]}, L'_{[2]}, \dots, L'_{[r]}, L'_{[r+1]}, L'_{[r+2]}, \dots)$ . The latter schedule  $\mathcal{L}'$  is derived from the former schedule  $\mathcal{L}$  by interchanging the  $i$ th and  $(i + 1)$ st orders (*i.e.*, orders  $O_i$  and  $O_{i+1}$ ), while keeping all the other orders in the same processing sequence. Assume that in  $\mathcal{L}$ ,  $w_i < w_{i+1}$  holds, implying  $\sigma_i \geq \sigma_{i+1}$ .

One can observe that all the orders except  $O_i$  and  $O_{i+1}$  are of the same completion times in the two schedules. Therefore, the difference of the objective value between the two schedule lies in the total weighted completion times of orders  $O_i$  and  $O_{i+1}$ . Let  $C_i(\mathcal{L}), C_i(\mathcal{L}')$  be the completion time of order  $O_i$  in schedules  $\mathcal{L}$  and  $\mathcal{L}'$ , respectively. Define  $F(\mathcal{L}) = w_i C_i(\mathcal{L}) + w_{i+1} C_{i+1}(\mathcal{L})$ , and  $F(\mathcal{L}') = w_i C_i(\mathcal{L}') + w_{i+1} C_{i+1}(\mathcal{L}')$ .

We are now ready to figure out the gap of the objective value between the two schedules  $\mathcal{L}$  and  $\mathcal{L}'$ , or equivalently between  $F(\mathcal{L})$  and  $F(\mathcal{L}')$ . Assume without loss of generality that order  $O_i$  is processed in either lot  $L_{[r]}$  or lots  $L_{[r]}$  and  $L_{[r+1]}$  for some  $r \geq 1$ . Let  $\alpha$  be the remaining capacity in lot  $L_{[r]}$  immediately before assigning orders  $O_i$  and  $O_{i+1}$  in both schedules  $\mathcal{L}$  and  $\mathcal{L}'$  (see Fig. 1). There are three cases with regard to the total size of the two orders.

**Case 1.**  $\sigma_i + \sigma_{i+1} \leq \alpha$ .

In this case, both orders are processed in lot  $L_{[r]}$ , and swapping the two orders has no impact on their completion times (see Figs. 1a and 1b). That is,  $F(\mathcal{L}) = F(\mathcal{L}')$  holds.

**Case 2.**  $t < \sigma_i + \sigma_{i+1} \leq k + \alpha$ .

The case condition implies that the two orders are processed in the two consecutive lots  $L_{[r]}$  and  $L_{[r+1]}$ .

**Case 2.1.**  $\sigma_i \leq \alpha$ . Combining the assumption  $\sigma_i \geq \sigma_{i+1}$  and the case condition  $\sigma_i \leq \alpha$ , we have  $\sigma_{i+1} < \alpha$ .

In schedule  $\mathcal{L}$ , lot  $L_{[r]}$  contains order  $O_i$  and a part of  $O_{i+1}$ , and  $L_{[r+1]}$  processes the remaining part of order  $O_{i+1}$  (see Fig. 2a). Therefore,  $C_i(\mathcal{L}) = Pr$ ,  $C_{i+1}(\mathcal{L}) = P(r + 1)$ , and

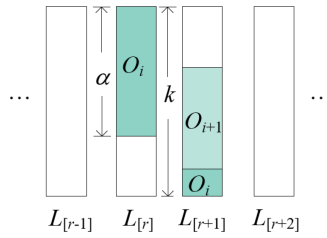


FIGURE 3. Illustration of processing  $O_i$  and  $O_{i+1}$  in schedule  $\mathcal{L}$  in Case 2.2.

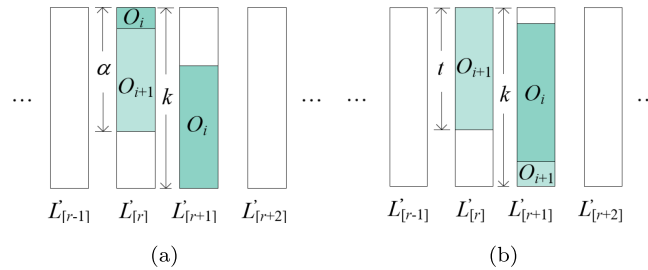


FIGURE 4. Illustration of processing  $O_i$  and  $O_{i+1}$  in schedule  $\mathcal{L}'$  in Case 2.2. (a) Case 2.2.1. (b) Case 2.2.2.

$$F(\mathcal{L}) = w_i C_i(\mathcal{L}) + w_{i+1} C_{i+1}(\mathcal{L}) = w_i Pr + w_{i+1} P(r + 1).$$

For schedule  $L'_{[r+1]}$ , since  $\sigma_{i+1} < \alpha$ , order  $O_{i+1}$  is processed in lot  $L'_{[r]}$ , while  $O_{i+1}$  is split and processed in the two lots  $L'_{[r]}$  and  $L'_{[r+1]}$  (see Fig. 2b). Thus,  $C_{i+1}(\mathcal{L}') = Pr$ ,  $C_i(\mathcal{L}') = P(r + 1)$ , and

$$F(\mathcal{L}') = w_i C_i(\mathcal{L}') + w_{i+1} C_{i+1}(\mathcal{L}') = w_i P(r + 1) + w_{i+1} Pr.$$

Together with the assumption  $w_i < w_{i+1}$ , we have

$$\begin{aligned} F(\mathcal{L}) - F(\mathcal{L}') &= w_i Pr + w_{i+1} P(r + 1) - (w_i P(r + 1) + w_{i+1} Pr) \\ &= (w_{i+1} - w_i) P > 0. \end{aligned}$$

**Case 2.2.**  $\sigma_i > \alpha$ . In this case, order  $O_i$  is split and processed in both lots  $L_{[r]}$  and  $L_{[r+1]}$ , while  $O_{i+1}$  is processed in  $L_{[r+1]}$  in schedule  $\mathcal{L}$  (see Fig. 3).

Thus,  $C_i(\mathcal{L}) = C_{i+1}(\mathcal{L}) = P(r + 1)$ , and

$$F(\mathcal{L}) = w_i C_i(\mathcal{L}) + w_{i+1} C_{i+1}(\mathcal{L}) = w_i P(r + 1) + w_{i+1} P(r + 1).$$

For  $F(\mathcal{L}')$ , there are two sub-cases by whether  $\sigma_{i+1} \leq \alpha$ .

**Case 2.2.1.**  $\sigma_{i+1} \leq \alpha$ . In this case, lot  $L'_{[r]}$  contains  $O_{i+1}$  and a part of  $O_i$ , while lot  $L'_{[r+1]}$  processes the remaining part of  $O_i$  in schedule  $\mathcal{L}'$  (see Fig. 4a).

We have  $C_{i+1}(\mathcal{L}') = Pr$ ,  $C_i(\mathcal{L}') = P(r + 1)$ , and

$$F(\mathcal{L}') = w_i C_i(\mathcal{L}') + w_{i+1} C_{i+1}(\mathcal{L}') = w_i P(r + 1) + w_{i+1} Pr.$$

Therefore,

$$\begin{aligned} F(\mathcal{L}) - F(\mathcal{L}') &= w_i P(r + 1) + w_{i+1} P(r + 1) - (w_i P(r + 1) + w_{i+1} Pr) \\ &= w_{i+1} P > 0. \end{aligned}$$

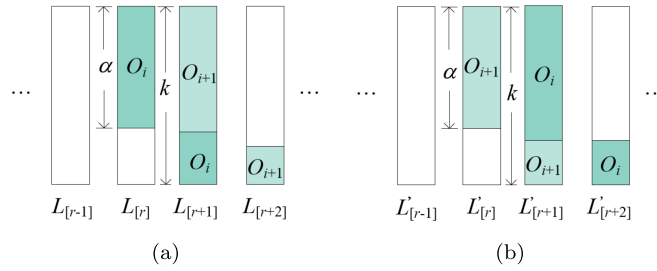


FIGURE 5. Illustration of processing  $O_i$  and  $O_{i+1}$  in Case 3. (a) In schedule  $\mathcal{L}$ . (b) In schedule  $\mathcal{L}'$ .

**Case 2.2.2.**  $\sigma_{i+1} > \alpha$ . In this case, Order  $O_{i+1}$  is split and process in both lots  $L'_{[r]}$  and  $L'_{[r+1]}$ , and  $O_i$  is processed in lot  $L'_{[r+1]}$  in schedule  $\mathcal{L}'$  (see Fig. 4b). Similarly, we have  $C_{i+1}(\mathcal{L}') = C_i(\mathcal{L}') = P(r+1)$ , and

$$F(\mathcal{L}') = w_i C_i(\mathcal{L}') + w_{i+1} C_{i+1}(\mathcal{L}') = w_i P(r+1) + w_{i+1} P(r+1).$$

It is easy to see that  $F(\mathcal{L}) = F(\mathcal{L}')$  in this case.

**Case 3.**  $\sigma_i + \sigma_{i+1} > \alpha + k$ .

The case condition implies that  $\sigma_i > \alpha$  and  $\sigma_{i+1} > \alpha$  due to  $\sigma_i, \sigma_{i+1} \leq k$ . Moreover, the two orders are processed in the three consecutive lots  $L_{[r]}, L_{[r+1]}$  and  $L_{[r+2]}$ , as shown in Figure 5a.

For schedule  $\mathcal{L}$ , we have  $C_i(\mathcal{L}) = P(r+1)$ ,  $C_{i+1}(\mathcal{L}) = P(r+2)$ , and

$$F(\mathcal{L}) = w_i C_i(\mathcal{L}) + w_{i+1} C_{i+1}(\mathcal{L}) = w_i P(r+1) + w_{i+1} P(r+2).$$

For schedule  $\mathcal{L}'$ , order  $O_{i+1}$  is processed in lots  $L'_i$  and  $L'_{i+1}$ , and  $O_i$  is processed in lots  $L'_{i+1}$  and  $L'_{i+2}$  due to  $\sigma_i, \sigma_{i+1} > \alpha$  (see Fig. 5b).

Thus,  $C_{i+1}(\mathcal{L}') = P(r+1)$ ,  $C_i(\mathcal{L}') = P(r+2)$ , and then

$$F(\mathcal{L}') = w_i C_i(\mathcal{L}') + w_{i+1} C_{i+1}(\mathcal{L}') = w_i P(r+2) + w_{i+1} P(r+1).$$

In this case,

$$\begin{aligned} F(\mathcal{L}) - F(\mathcal{L}') &= w_i P(r+1) + w_{i+1} P(r+2) - (w_i P(r+2) + w_{i+1} P(r+1)) \\ &= (w_{i+1} - w_i) P > 0. \end{aligned}$$

In all of the above cases,  $F(\mathcal{L}) - F(\mathcal{L}') \geq 0$  holds. It indicates that schedule  $\mathcal{L}'$  is better than or the same as schedule  $\mathcal{L}$ , *i.e.*, swapping orders  $O_i$  and  $O_{i+1}$  may induce the decrease of the objective value. Hence, we conclude that an optimal schedule is to process orders in the non-increasing sequence of their weights. The proof of the theorem is completed.  $\square$

### 5. NUMERICAL EXPERIMENTS

In this section, the LOS algorithm is analyzed and compared with the exact solution that is obtained by the Gurobi solver with a computational time constraint set to a maximum of 3600s. Please refer to Appendix A for the corresponding MILP model. The principal purpose of this study is to provide evidence to support the efficacy of the LOS algorithm. We execute all numerical experiments on a personal computing device equipped with an Intel(R) Core(TM) i5-1035G1 CPU operating at 1.00 GHz, with a turbo frequency of 1.19 GHz, and furnished with 16.0 GB of Random Access Memory (RAM), under the Windows 10 Operating System.

For each input instance under consideration, we designate the parameters  $k = 10$  and  $P = 5$ . The size  $\sigma_j$  of order  $O_j$  is stochastically determined within the interval  $[\lfloor \frac{k}{3} \rfloor, k]$ , while the weight is assigned within the range

TABLE 3. Computational results.

$(n, k, P)$	Gurobi		LOS		$Gap(\%)$	Average		Max	
	$TWC_G$	$time_G(s)$	$TWC_L$	$time_L(s)$		$\rho_e$	$\rho$	$\rho_e$	$\rho$
(3, 10, 5)	170	0.4959	170	0.0013	0.00%	1.00	1.31	1.00	1.50
(4, 10, 5)	264	0.5571	271	0.0015	2.65%	1.09	1.60	1.26	2.00
(5, 10, 5)	358	0.6229	373	0.0014	4.19%	1.04	1.69	1.08	2.00
(6, 10, 5)	469	2.0048	495	0.0007	5.54%	1.05	1.61	1.11	2.00
(7, 10, 5)	611	3.7850	650	0.0006	6.38%	1.06	1.73	1.11	2.00
(8, 10, 5)	763	8.4876	814	0.0007	6.68%	1.07	1.92	1.15	2.00
(9, 10, 5)	941	11.1161	997	0.0007	5.95%	1.06	1.96	1.11	2.00
(10, 10, 5)	1113	14.2856	1199	0.0007	7.73%	1.08	1.96	1.11	2.00
(11, 10, 5)	1333	26.1982	1409	0.0007	5.70%	1.06	1.96	1.09	2.00
(12, 10, 5)	1527	59.7203	1605	0.0019	5.11%	1.05	1.96	1.06	2.00
(13, 10, 5)	1841	137.1255	1995	0.0012	8.37%	1.08	1.96	1.15	2.00
(14, 10, 5)	1948	169.1549	2083	0.0009	6.93%	1.07	1.96	1.12	2.00
(15, 10, 5)	2255	213.8481	2402	0.0012	6.52%	1.06	2.00	1.10	2.00
(16, 10, 5)	2621	316.2551	2802	0.0008	6.91%	1.07	2.00	1.08	2.00
(17, 10, 5)	2935	381.9742	3116	0.0009	6.17%	1.06	2.00	1.07	2.00
(18, 10, 5)	3244	495.6360	3470	0.0008	6.97%	1.07	2.00	1.08	2.00
(19, 10, 5)	3625	630.2207	3861	0.0008	6.51%	1.07	1.96	1.07	2.00
(20, 10, 5)	3960	1608.9299	4209	0.0008	6.29%	1.06	1.96	1.08	2.00
(21, 10, 5)	4489	3600.0000	4815	0.0013	7.26%	1.07	2.00	1.09	2.00
Average	–	404.2325	–	0.0010	5.89%	1.06	1.87	1.10	1.97

[5, 10]. In order to reduce the variability of the experimental results, we use the average of the results of ten randomly generated instances for each of the settings.

Table 3 presents the computational outcomes for the MIP model and the heuristic algorithm applied to problem  $1|lot, split| \sum_j w_j C_j$ . The first column indicates instances (*e.g.*, (3, 10, 5) in the first row represents the instance with  $n = 3$  orders, lot capacity  $k = 10$ , and lot processing  $P = 5$ ). Subsequent columns delineate the objective value and computational duration of the Gurobi solution model, in that order. The heuristic algorithm's objective value and execution time are detailed in the fourth and fifth columns, respectively, with temporal measurements expressed in seconds (hereinafter denoted as "s"). The relative gap, referred to as the  $Gap$  is ascertained by the formula  $\frac{TWC_L - TWC_G}{TWC_G} 100\%$  and is recorded in the sixth column of Table 3, where  $TWC_L$  and  $TWC_G$  denote the objective value of the heuristic algorithm and the objective value of Gurobi, respectively. We denote the ratio of the approximate solution (LOS) to the exact solution (Gurobi) as  $\rho_e = \frac{TWC_L}{TWC_G}$ . The approximation rate of the LOS algorithm, as described in Section 3.2, is given by  $\rho = \frac{w_{\max}}{w_{\min}}$ . Columns 7 and 8 show the average values of the ratios  $\rho_e$  and  $\rho$  based on the 10 randomly generated instances for each setting. The final two columns in the table provide the worst-case ratios for these instances.

We observe by Table 3 that the heuristic algorithm runs significantly faster than Gurobi. Regarding the objective value, the relative error varies from a maximum of 8.37% to a minimum of 0.00%, with an average discrepancy of 5.89%. The approximation ratio  $\rho_e$  is close to 1, which greatly highlights the effectiveness of the algorithm.

## 6. CONCLUSION

In this work, we investigate the single machine lot scheduling problem to minimize total weighted completion time of orders. We first prove that the problem is NP-hard and provide a polynomial approximation algorithm. We then consider two special cases. We assume that the total size of orders is no more than twice of the lot

capacity in the first case. In the two case the size and weight of order are in reverse-agreeable. All the two special cases are shown polynomial solvable.

Subsequently, the effectiveness of the heuristic algorithm is empirically substantiated through a series of numerical experiments. The results obtained in this work may help one manufacturer make efficient decisions on processing customer orders, thus reducing production costs and improving the performance of operation management.

In future research, it is meaningful to investigate the exact complexity status (whether it admits a pseudo-polynomial time algorithm or is strongly NP-hard) of the problem  $1|lot, split|\sum_j w_j C_j$ . Whether this bound is tight for the heuristic algorithm LOS remains an open question with different weights. Additionally, one of future researches is to extend the obtained results to the multi-processor lot scheduling environment. Another interesting direction is to relax the constraint of total size of orders in the first special case, or remove the assumption of reverse-agreeable order size and order weight in the two special case, and explore optimal or approximation solutions.

#### FUNDING

This work was partially supported by the National Natural Science Foundation of China (NSFC) under Grants 72271051, 72021002, 71832001 and 72071144, and the Fundamental Research Funds for the Central Universities under Grants 2232018H-07 and CUSF-DH-T-2025026.

#### DATA AVAILABILITY STATEMENT

The research data associated with this article are included in the article.

#### REFERENCES

- [1] A. Allahverdi, A survey of scheduling problems with no-wait in process. *Eur. J. Oper. Res.* **255** (2016) 665–686.
- [2] M. Allahverdi, Minimizing total tardiness in a two-machine flowshop with uncertain and bounded processing times. *RAIRO-Oper. Res.* **57** (2023) 1353–1375.
- [3] Y. Chen, Y.X. Cheng and G.Q. Zhang, Single machine lot scheduling with non-uniform lot capacities and processing times. *J. Comb. Optim.* **43** (2022) 1359–1367.
- [4] M. Geurtsen, J.B. Didden, J. Adan, Z. Atan and I.J.B.F. Adan, Production, maintenance and resource scheduling: a review. *Eur. J. Oper. Res.* **305** (2023) 501–529.
- [5] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete. Math.* **5** (1979) 287–326.
- [6] Y.-T. Hou, D.-L. Yang and W.-H. Kuo, Lot scheduling on a single machine. *Inf. Process. Lett.* **114** (2014) 718–722.
- [7] A.R. Karlin, M.S. Manasse, L. Rudolph and D.D. Sleato, Competitive snoopy caching. *Algorithmica* **3** (1988) 79–119.
- [8] H. Kellerer, U. Pferschy, D. Pisinger, H. Kellerer, U. Pferschy and D. Pisinger, Introduction to NP-Completeness of Knapsack problems, in *Knapsack Problems*. Springer Berlin Heidelberg, Berlin, Heidelberg (2004) 483–493.
- [9] M.Y. Kovalyov, A batching machine model for lot scheduling on a single machine. *Found. Comput. Decis.* **43** (2018) 37–40.
- [10] M. Li and G.G. Wang, A review of green shop scheduling problem. *Inf. Sci.* **589** (2022) 478–496.
- [11] M. Liu, T. Lin, F. Chu, F.F. Zheng and C.B. Chu, A new and general stochastic parallel machine ScheLoc problem with limited location capacity and customer credit risk. *RAIRO-Oper. Res.* **57** (2023) 1179–1193.
- [12] M. Liu, Z. Liu, F. Zheng and C. Chu, Mathematical models and optimal algorithms for Lot scheduling considering job splitting and due dates in green logistics. *Asia-Pac. J. Oper. Res.* **41** (2024) 2350040.
- [13] B. Mor, Single-machine lot scheduling with variable lot processing times. *Optim. Lett.* **53** (2021) 321–334.
- [14] B. Mor and G. Mosheiov, A note on the single machine CON and CONW problems with lot scheduling. *Comb. Optim.* **42** (2021) 327–338.
- [15] B. Mor, G. Mosheiov and D. Shapira, Lot scheduling on a single machine to minimize the (weighted) number of tardy orders. *Inf. Process. Lett.* **164** (2020) 106009.
- [16] B. Mor, G. Mosheiov and D. Shapira, Single machine lot scheduling with optional job-rejection. *J. Comb. Optim.* **41** (2021) 1–11.

- [17] G. Mosheiov and A. Sarig, A note on lot scheduling on a single machine to minimize maximum weighted tardiness. *J. Comb. Optim.* **45** (2023) 128.
- [18] B. Nurit, B. Mor, Y. Schlissel and D. Shapira, Lot scheduling involving completion time problems on identical parallel machines. *Oper. Res-Ger.* **23** (2023) 12.
- [19] H.J. Shen and Z.C. Geng, A Note on single-machine lot scheduling with splittable jobs to minimize the number of tardy jobs. *Chin. Q. J. Math.* **37** (2022) 412.
- [20] D.-L. Yang, Y.-T. Hou and W.-H. Kuo, A note on a single-machine lot scheduling problem with indivisible orders. *Comput. Oper. Res.* **79** (2017) 34–38.
- [21] E. Zhang, M. Liu, F.F. Zheng and Y.F. Xu, Single machine lot scheduling to minimize the total weighted (discounted) completion time. *Inf. Process. Lett.* **142** (2019) 46–51.
- [22] F.F. Zheng and K.Y. Jin, An improved heuristic for single machine lot scheduling problem. *IFAC-PapersOnLine.* **52** (2019) 217–222.



**Please help to maintain this journal in open access!**

This journal is currently published in open access under the Subscribe to Open model (S2O). We are thankful to our subscribers and supporters for making it possible to publish this journal in open access in the current year, free of charge for authors and readers.

Check with your library that it subscribes to the journal, or consider making a personal donation to the S2O programme by contacting [subscribers@edpsciences.org](mailto:subscribers@edpsciences.org).

More information, including a list of supporters and financial transparency reports, is available at <https://edpsciences.org/en/subscribe-to-open-s2o>.

## APPENDIX A.

In this section, we first introduce some notations and decision variables and then present mathematical models for the considered problem.

### Indexes

- $j, j'$ : indexes of orders,  $j, j' = 1, \dots, n$ ;
- $r$ : index of lots,  $r = 1, \dots, N$  (the value of  $N$  is determined later).

### Parameters

- $n$ : number of orders;
- $k$ : lot capacity;
- $\sigma_j$ : size of order  $O_j$ ;
- $w_j$ : weight of order  $O_j$ ;
- $N$ : number of lots. It is crucial to note that, given the splittable orders, the equation  $N \equiv \lceil \frac{\sum_{j=1}^n \sigma_j}{k} \rceil$  holds;
- $O$ : set of orders,  $O = \{1, \dots, n\}$ , indexed by  $j, j'$ ;
- $R$ : set of lots,  $R = \{1, \dots, N\}$ , indexed by  $r$ ;
- $P$ : lot processing time.

### Decision variables

- $x_{jr}$ : binary variable, equals 1 if order  $O_j$  is assigned to the  $r$ -th lot, 0 otherwise;
- $z_{jr}$ : continuous variable, denotes the proportion of  $O_j$  that is assigned to the  $r$ -th lot;
- $v_{jr}$ : continuous variable, denotes the proportion of  $O_j$  that has not been processed on the completion of the  $r$ -th lot. Let  $r = 0$  be a virtual lot. Obviously,  $v_{j0} = \sigma_j$  and  $v_{jN} = 0, \forall j \in O$ .

**Mathematical model of problem 1|lot, split| $\sum_j w_j C_j$**

$$\min \sum_{j \in O} \{w_j C_j\} \tag{A.1}$$

$$\text{s.t. } \sum_{j \in O} z_{jr} \leq k, \quad \forall r \in R \tag{A.2}$$

$$v_{jr} = v_{j,r-1} - z_{jr}, \quad \forall j \in O, r \in R \tag{A.3}$$

$$C_j \geq P \cdot \left[ \sum_{r \in N} r \cdot \frac{z_{jr}}{\sigma_j} \right], \quad \forall j \in O, r \in R \tag{A.4}$$

$$z_{jr} \leq \sigma_j \cdot x_{jr}, \quad \forall j \in O, r \in R \tag{A.5}$$

$$z_{jr} > x_{jr} - 1, \quad \forall j \in O, r \in R \tag{A.6}$$

$$x_{jr} + x_{j,r+1} + x_{j'r} + x_{j',r+1} \leq 3, \quad \forall j \neq j' \in O, r \in R \setminus \{N\} \tag{A.7}$$

$$z_{j,r+1} - v_{jr} \leq \sigma_j \cdot (1 - x_{jr}), \quad \forall j \in O, r \in R \setminus \{N\} \tag{A.8}$$

$$z_{j,r+1} - v_{jr} \geq -\sigma_j \cdot (1 - x_{jr}), \quad \forall j \in O, r \in R \setminus \{N\} \tag{A.9}$$

$$x_{jr} \in \{0, 1\}, 0 \leq z_{jr} \leq \sigma_j, 0 \leq v_{jr} \leq \sigma_j, \quad \forall j \in O, r \in R \tag{A.10}$$

$$v_{jN} = 0, v_{j0} = \sigma_j, C_j \geq 0, \quad \forall j \in O. \tag{A.11}$$

The objective (A.1) aims to minimize total weighted completion time. The constraints (A.2) stipulate that within the  $r$ -th lot, the accumulated size of the assigned orders must not surpass the predefined uniform lot capacity. Constraints (A.3) ensure that the residual proportion of order  $O_j$  from the  $(r - 1)$ -th lot is appropriately allocated to the  $r$ -th lot. Constraints (A.4) define the completion time of order  $O_j$ . Constraints (A.5) and (A.6) establish the relationship between  $z_{jr}$  and  $x_{jr}$ , such that  $z_{jr} > 0$  if and only if  $x_{jr} = 1$ . Constraints (A.7) ensure the continuity of the processing of any order. Constraints (A.8) and (A.9) assert that when  $x_{jr} = 1$  and  $v_{jr} > 0$  (indicating that order  $O_j$  is not entirely completed in the  $r$ -th lot),  $z_{j,r+1} = v_{jr}$  (implying that the order is included in the subsequent lot). Finally, constraints (A.10) and (A.11) define the domains of variables.