https://doi.org/10.1016/j.cor.2021.105477

Integrated production and transportation scheduling with order-dependent inventory holding costs

Xianyan Yang[1], Zhixue Liu[1], Feng Li[1*], Zhou Xu[2]

1. School of Management, Huazhong University of Science and Technology, Wuhan, China
2. Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

Revised on 5 July 2021

Abstract In this study, we consider an integrated production and transportation scheduling problem faced by make-to-order manufacturers that adopt a commit-to-delivery business mode and cooperate with third-party logistics providers to deliver processed products to customers. The third-party logistics providers typically offer multiple shipping modes chosen by the manufacturers to deliver products, each shipping mode with a shipping time guarantee and a shipping cost function that is non-increasing in shipping time and sub-additive, non-decreasing in shipping quantity. The problem involves inventory holding costs which not only depend on the time that the products spend in temporary storage but also depend on customer types. The problem is to determine an integrated production and shipping schedule that satisfies the committed delivery due date limitations for the customers, such that the total cost of shipping and inventory holding is minimized. We investigate two cases with and without split delivery. For both cases, we first show that both of them are ordinarily NP-hard, prove that there exist no polynomial-time approximation algorithms with constant worst-case ratios, propose exact algorithms to solve them, and finally design column generation-based heuristic algorithms to find feasible solutions. The computational results demonstrate that the heuristic algorithms are capable of generating near-optimal solutions efficiently. We also consider two interesting practical variants of the problem.

Keywords: Scheduling, Production and Transportation, NP-hardness, Dynamic Programming, Column Generation

[*]Corresponding author: li_feng@hust.edu.cn

1 Introduction

In this study, we consider an integrated production and transportation scheduling problem, faced by a make-to-order manufacturer with a commit-to-delivery business mode. Make-to-order is a business production strategy that typically allows customers to customize products. Under the make-to-order production strategy, the manufacturer begins to process its products only after receiving customer orders. Commit-to-delivery is a business delivery strategy wherein the manufacturer commits a delivery due date for each customer and is responsible for shipping costs. For example, Dell Computer Company adopts the make-to-order production strategy[†] and the commit-to-delivery delivery strategy[‡], wherein customers can order customized computers online and receive them within a promised delivery due date, which customers can see at Dell's website when placing orders.

The manufacturer often cooperates with third-party logistics (3PL) providers such as FedEx or UPS to deliver its processed products to customers. A 3PL provider typically offers multiple shipping modes chosen by the manufacturer, each shipping mode associated with a shipping time guarantee and a shipping cost function that is non-increasing in shipping time and sub-additive, non-decreasing in shipping quantity. For example, UPS offers multiple shipping modes[§] , such as UPS Next Day Air Shipping, UPS 2nd Day Air, UPS 3 Day Select and UPS Ground with Freight Pricing, which guarantee the shipping times of one day, two days, three days, and more than three days, respectively. To save shipping costs, the manufacturer would like to combine the products for each customer into a single shipment because of exhibiting economies of scale. However, when some units of the products for a customer have been processed completely, but not shipped out on the same day, they must be kept in temporary storage, which incurs an inventory holding cost. The inventory holding cost should in theory depend on the time that the products spend in temporary storage. Furthermore, in some practical situations, for example, in furniture industries, customers often order a variety of products. The inventory holding cost also typically depends on customers'

[†]https://www.dell.com/en-in/shop/configure-and-buy/cp/configure-and-buy

[‡]https://www.dell.com/koa/search?q=customize%20laptop#q=customize%20laptop&t=default&sort=relevancy&layout=card&dpsalessegment:radioGroup=dhs

[§]https://www.ups.com/service-selector?loc=en_US

products, storage needs, and locations. To save inventory holding costs, when some units of the products have been processed completely, the manufacturer has an incentive to deliver the processed products as soon as possible, which may incur high shipping costs. Therefore, the manufacturer must strike a balance between shipping costs and inventory holding costs.

In this study, except for the shipping and inventory holding costs, we closely follow the problem setting proposed in Stecke and Zhao (2007), which we now describe. At the beginning of a planning horizon consisting of multiple days, the manufacturer receives orders from customers and commits a delivery due date for each customer. For example, in Dell's case, each customer usually places only one order each time. If a customer places multiple orders at different times, these orders would be thought to be different orders from different customers. See Ahmadi et al. (2005), Leung et al. (2005b), Leung et al. (2005a), Stecke and Zhao (2007), Armstrong et al. (2008), and Liu and Liu (2020) for other similar examples occurring in the lenses industry, the paper industry, the pharmaceutical industry, Dell Computer Company, the adhesive chemicals industry, and the perishable products industry, respectively. Motivated by these examples from practice, in this study, we focus on the situation where each customer places only one order. The manufacturer needs to (i) process the products ordered by these customers on a single production line with production capacity limitation and (ii) choose shipping modes from a set of available shipping modes to deliver the processed products to customers before or on the committed delivery due dates. The shipping cost function of each shipping mode has a general form, that is, it is non-increasing in shipping time and sub-additive, non-decreasing in shipping quantity. In line with Melo and Wolsey (2010) and Li et al. (2017), it is commonly assumed that the inventory holding cost per unit of an order, namely order-dependent inventory holding cost, depends on its corresponding customer type. The order-dependent inventory holding cost is also assumed to be proportional to the time that the products spend in temporary storage. The problem is to determine an integrated production and shipping schedule such that the total cost of shipping and inventory holding is minimized. We study the problem under different delivery scenarios where split delivery is allowed and where it is not allowed. When split delivery is allowed for each customer, the manufacturer could divide a customer's order into multiple smaller orders and deliver them separately using different shipping

modes. This may help the manufacturer to save more on shipping costs and inventory holding costs. When split delivery is not allowed for each customer, products for each customer are required to be shipped in the same shipment. In some practical situations where idle times between products during product processing are not allowed, as the production line incurs huge setup costs, we also consider such a case in Section 7.1. The presence or absence of idle times between products during product processing does not change the computational complexity of the problem or the method adopted to solve the problem. Therefore, in the main body of this study, we focus on the problem with idle times allowed. In Section 7.2 we briefly discuss an extension with production capacity, inventory capacity, shipping capacity, and order acceptance.

We present a unified analysis of the integrated production and transportation scheduling problem. Our study makes the following contributions. First, we present comprehensive results on computational complexity for the various cases of the problem. We show that (i) when split delivery is not allowed for each customer, the problem is ordinarily NP-hard when the length of the planning horizon is fixed; (ii) when split delivery is not allowed for each customer, the problem has no polynomial-time approximation algorithm with a constant worst-case ratio; and (iii) when split delivery is allowed for each customer, the problem is ordinarily NP-hard when the length of the planning horizon is fixed. Second, we propose exact algorithms to solve the cases with and without split delivery, and prove that they achieve pseudo-polynomial running times for the cases with a fixed length of the planning horizon. Third, we develop column generation-based heuristic algorithms for both cases with and without split delivery to find near-optimal solutions quickly. The heuristic algorithms first solve the linear programming relaxation of a Dantzig-Wolfe reformulation using the column generation approach, and subsequently solve the restricted Dantzig-Wolfe reformulation based on the generated columns to find feasible solutions.

The remainder of this paper is organized as follows: Section 2 discusses related literature. In Section 3, we describe the problem in detail. In Sections 4 and 5, we study the case without split delivery and the case with split delivery, respectively. Computational results are presented in Section 6. In Section 7, we provide analysis for different variants of the problem. Section 8 concludes the paper.

2 Literature Review

In this section, we compare our problem with those in literature, focusing on integrated production and outbound distribution scheduling (IPODS) problems. Per our understanding, IPODS problems have been an active research area since the seminal works by Lee and Chen (2001), Hall and Potts (2003), Chen and Vairaktarakis (2005), and Chen and Pundoor (2009). Chen (2010) provides an overview of IPODS literature up to and including 2010. We now review the key literature on IPODS problems over the years 2011-2021. Geismar et al. (2011) study an IPODS problem with a limited lifetime of a product and a modifiable production rate. They gain insights into optimizing the total cost of production and transportation using theoretical analysis. Ullrich (2013) proposes a genetic algorithm to solve an IPODS problem such that the total tardiness is minimized. The problem takes into account parallel machines with machine-dependent ready times, vehicles with capacities and ready times, delivery time windows, service times, and destinations. Leung and Chen (2013) design polynomial-time algorithms for three cases of an IPODS problem with fixed delivery departure times. The first is to minimize the maximum lateness of orders, the second is to minimize the number of vehicles used subject to the condition that the maximum lateness is minimum, while the last is to minimize the weighted sum of the maximum lateness and the number of vehicles used. Mensendiek et al. (2015) propose a branch-and-bound algorithm and two heuristic algorithms to solve a similar problem as the third case studied by Leung and Chen (2013). However, the problem in Mensendiek et al. (2015) involves parallel machines and its objective is to minimize the total tardiness. Li et al. (2017) consider an IPODS problem with fixed departure times and delivery time windows. They consider two cases: where split delivery is allowed and where split delivery is not allowed. For each case, they study its computational complexity by either showing the NP-hardness or proposing an exact algorithm to solve it. Devapriya et al. (2017) provide evolutionary-based heuristic algorithms for an IPODS problem of a perishable product with a limited lifetime, such that the total routing cost of used trucks is minimized. Cheref et al. (2017) focus on the NP-hardness proofs and polynomial-time algorithms of an IPODS problem in which the scheduling sequence and the delivery sequence are the same and predefined. Tang

et al. (2019) study a general IPODS problem with two-stage delivery. In this problem, products that are processed completely on parallel production lines are delivered to customer sites in two stages of shipping. They propose exact algorithms and heuristic algorithms with worst-case and asymptotic performances for variants of the problem. Li and Li (2020) propose polynomial-time exact algorithms for the cases of an IPODS problem with transportation disruption. The objective is to simultaneously maintain a low-cost schedule and control the magnitude of changes in the delivery times of products. Chevroton et al. (2021) consider an integrated scheduling and routing problem with flow shop machines and delivery routing. They propose a two-step approach to minimize the total cost of inventory holding, routing, and tardiness penalty. Bachtenkirch and Bock (2021) focus on a novel single-stage scheduling problem with jobs being delivered in batches on flexibly definable customer-dependent delivery dates. They propose a branch-and-bound procedure and a randomized adaptive search procedure to solve the problem efficiently.

Our problem differs from the above reviewed IPODS problems in the following aspects. First, our problem involves multiple shipping modes, whereas most existing IPODS problems predominantly focus on a single shipping mode. Second, in our problem, each shipping mode is associated with a shipping cost function, which is non-linear in shipping time, whereas in most existing IPODS problems, shipping cost functions do not involve shipping time.

To the best of our knowledge, only a handful of papers examine problems closely related to our problem. Stecke and Zhao (2007) study an integrated production and transportation problem for a make-to-order manufacturer with a commit-to-delivery business mode. They consider multiple shipping modes, each with a shipping cost function that is convexly non-increasing with shipping time and linearly non-decreasing with shipping quantity. They consider two cases: where split delivery is allowed and where split delivery is not allowed. For the case with split delivery, they propose exact algorithms to solve it. For the case without split delivery, they provide a heuristic algorithm to find feasible solutions. Later, Melo and Wolsey (2010) present a novel integer programming formulation to solve large-scale instances for the case without split delivery. Zhong et al. (2010) and Zhong (2015) consider a special case where split delivery is not allowed and the shipping cost function is linear in both shipping time and shipping quantity. They propose heuristic

algorithms with worst-case ratios of 2 and 5/3, respectively. Li et al. (2020) extend to the case where split delivery is allowed, the shipping cost function is non-linear in both shipping time and shipping quantity, and multiple orders from the same customer can be combined if necessary. They propose a fully polynomial time approximation scheme (FPTAS) for a special case with a planning horizon of two days and a column generation-based heuristic algorithm for the general case.

The most obvious differences between all the above reviewed papers, except for Li et al. (2020), and our problem, are as follows: (i) In the reviewed papers, the shipping cost functions are linear in shipping quantity whereas our problem involves shipping cost functions that are non-linear in shipping quantity and (ii) the reviewed papers do not take into account the inventory which plays an important role in the integrated scheduling area, whereas our problem considers this element. Li et al. (2020) is inspirational to our research and only considers the problem with split delivery. However, our problem considers two cases: where split delivery is allowed and where it is not allowed. In Li et al. (2020), the unit inventory holding costs of orders are identical, whereas in this study we consider a more general case that involves order-dependent unit inventory holding costs.

3 Problem Description

At the beginning of a planning horizon consisting of m days, $M = \{1, 2, \ldots, m\}$, a make-to-order manufacturer has received orders from n customers, $N = \{1, 2, \ldots, n\}$, one order per customer (see practice examples described in Section 1). Note that we use customers and orders interchangeably. The manufacturer processes products ordered by the customers on its own single production line and promises to deliver the processed products to customers in different locations. For each customer $i \in N$, let $q_i \in \mathbb{Z}_+$ denote the quantity of the products ordered and $d_i \in \mathbb{Z}_+$ denote its committed delivery due date before or on which the products must be received by customer i, where $\mathbb{Z}_+$ is the set of non-negative integer numbers. In line with Stecke and Zhao (2007), without loss of generality, we assume that one production day in the planning horizon is from 3:00 PM to 3:00 PM the next day (illustrated in Figure 1). Let c denote the daily production capacity, which means that total quantity of products processed in each day cannot exceed c. At the end of each

day (e.g., 3:00PM), a logistics company comes to pick up products and ship them away, and for each customer $i \in N$, there is a set of k_i different shipping modes, $K_i = \{1, \ldots, k_i\}$, which are available to deliver products. Each shipping mode $s \in K_i$ is associated with a shipping time of e_{is} days, as well as a shipping cost function $G_{is}(e_{is}, y)$, where y is the quantity of products shipped by the shipping mode s and $G_{is}(e_{is}, 0) = 0$. Without loss of generality, we assume that $0 \leq e_{i1} \leq e_{i2} \leq \ldots \leq e_{i,k_i} \leq m-1$. More specifically, if the products are shipped out by the shipping mode s on day t for customer i, then customer i receives these products on day $t + e_{is}$. To enhance the applicability of our problem, we assume that (i) $G_{is}(e_{is}, y)$ is non-increasing in shipping time e_{is}, that is, for $s_1, s_2 \in K_i$ and any $y \geq 0$, if $e_{i,s_1} \leq e_{i,s_2}$, then $G_{i,s_2}(e_{i,s_2}, y) \leq G_{i,s_1}(e_{i,s_1}, y)$; and (ii) $G_{is}(e_{is}, y)$ is sub-additive, non-decreasing in shipping quantity y, that is, $G_{is}(e_{is}, y)$ is sub-additive in y if $G_{is}(e_{is}, y_1 + y_2) \leq G_{is}(e_{is}, y_1) + G_{is}(e_{is}, y_2)$, for any $y_1 \geq 0$ and $y_2 \geq 0$. We also assume that the value of $G_{is}(e_{is}, y)$ can be calculated in $O(1)$ time for any $s \in K_i$, $y \geq 0$, $i \in N$. As described in Section 1, if the products are processed completely but not shipped out on the same day, this incurs inventory holding costs. Let h_i indicate the inventory holding cost per unit per day for order $i \in N$. The problem is to determine an integrated production and shipping schedule that satisfies the committed delivery due date of each order, such that the total shipping and inventory holding cost is minimized. From the above definition, we can see that the problem has feasible solutions if and only if the following feasibility condition (1) holds:

$$\sum_{i \in N: d_i - e_{i1} \leq t} q_i \leq c \cdot t \text{ for } t = 1, \ldots, m. \tag{1}$$

Therefore, we assume that the above feasibility condition is satisfied throughout this study except for Section 7.2.

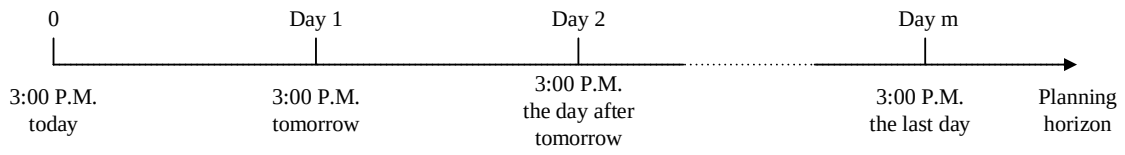

Figure 1: The planning horizon of the problem

As defined in Section 1, in this study, we consider two cases of the problem: one without split delivery, denoted as SDN, and the other with split delivery, denoted as SD. For SDN, the products for each order are required to be combined into a single shipment and be shipped by a single shipping mode. We first present an optimality property that applies to both cases.

Property 1. *For both SDN and SD, there exists an optimal solution in which the products, shipped out on day $t \in M$ for customer $i \in N$, are shipped by the single shipping mode s, where $s = argmax_{s' \in K_i}\{e_{i,s'} | e_{i,s'} + t \leq d_i\}$.*

Proof. For SDN, the products of each order are required to be shipped by a single shipping mode. For SDN, since $G_{is}(\cdot, \cdot)$ is non-increasing in shipping time, it is optimal that an order is shipped out on its production completion day by using the slowest shipping mode that meets its committed delivery due date. Thus, for SDN, this property holds.

Next we will prove that for SD an optimal solution that does not satisfy this property can be transformed to another optimal solution that satisfies this property. For SD, let π^* be an optimal solution which does not satisfy this property. Suppose that in π^* there exists a pair (i, t) such that the products, shipped out on day t for customer i, are shipped by two shipping modes s_1 and s_2. Let s be the slowest shipping mode such that the products, shipped out on day t for customer i, arrive at customer i before or on their committed delivery due date, i.e., $s = argmax_{s' \in K_i}\{e_{i,s'} | e_{i,s'} + t \leq d_i\}$. Thus, we have that $e_{i,s_1} \leq e_{is}$ and $e_{i,s_2} \leq e_{is}$. Let y_1 and y_2 be the quantities of products, shipped out on day t for customer i, are shipped by the shipping modes s_1 and s_2, respectively. From π^*, we construct a new solution π' by shipping the products, shipped out on day t for customer i, by the shipping mode s, and keeping the shipping modes unchanged for all the other products. From the definition of s, π' is a feasible solution for SD. To compare with π^*, π' increases the total cost by $G_{is}(e_{is}, y_1 + y_2) - [G_{i,s_1}(e_{i,s_1}, y_1) + G_{i,s_2}(e_{i,s_2}, y_2)]$ which must be non-negative, since π^* is optimal. Thus, we have that

$$G_{is}(e_{is}, y_1 + y_2) \geq G_{i,s_1}(e_{i,s_1}, y_1) + G_{i,s_2}(e_{i,s_2}, y_2) \geq G_{is}(e_{is}, y_1) + G_{is}(e_{is}, y_2) \geq G_{is}(e_{is}, y_1 + y_2),$$

where the second inequality holds since $e_{i,s_1} \leq e_{is}$, $e_{i,s_2} \leq e_{is}$, and $G_{is}(\cdot, \cdot)$ is non-increasing in

shipping time, and the third inequality holds since $G_{is}(\cdot,\cdot)$ is sub-additive in shipping quantity. Thus, solution π' has the same total cost as solution π^*, which implies that solution π' must also be an optimal solution in which the products, shipped out on day t for customer i, are shipped by a single shipping mode s. We repeat the above process iteratively to obtain an optimal solution which satisfies this property. □

Let s_{it} be the slowest shipping mode such that the products, shipped out on day $t \in M$ for customer $i \in N$, arrives at customer i before or on their committed delivery due date, i.e., $s_{it} = argmax_{s' \in K_i}\{e_{i,s'} | e_{i,s'} + t \leq d_i\}$. From Property 1, it is optimal that for SDN and SD, the products, shipped out on day $t \in M$ for customer $i \in N$, are shipped by the shipping mode s_{it}.

4 Solving the Problem without Split Delivery

Note that we denote the problem without split delivery as SDN. We first formulate SDN as an integer programming model in Section 4.1 and subsequently study its computational complexity in Section 4.2. For SDN, we propose an exact algorithm and a column generation-based heuristic algorithm in Section 4.3.

4.1 Integer Programming Model for SDN

Let g_{it} be the minimum shipping cost of order $i \in N$ shipped out on day $t \in M$. From Property 1, if order $i \in N$ is shipped out on day $t \in M$, then it is optimal to use the shipping mode s_{it} to ship this order. Thus, if $t + e_{i1} \leq d_i$, then let $g_{it} = G_{i,s_{it}}(e_{i,s_{it}}, q_i)$ and otherwise, let $g_{it} = +\infty$. Let decision variables x_{it} for $i \in N$ and $t \in M$ represent the quantity of the products processed in day t for customer i. Let decision variables z_{it} for $i \in N$ and $t \in M$ represent whether or not the products for customer i are shipped out on day t. Next we show an integer program for SDN (denoted as SDN-I) as follows.

$$\text{SDN-I} \qquad \min \sum_{i\in N}\sum_{t\in M} z_{it}g_{it} + \sum_{i\in N}\sum_{t\in M} h_i\left(\sum_{\tau=1}^{t} x_{i\tau} - \sum_{\tau=1}^{t} z_{i\tau}q_i\right) \tag{2}$$

$$\text{s.t.} \sum_{i\in N} x_{it} \le c, \text{ for } t \in M; \tag{3}$$

$$\sum_{t=1}^{d_i-e_{i1}} x_{it} = q_i, \text{ for } i \in N; \tag{4}$$

$$\sum_{t=1}^{m} z_{it} = 1, \text{ for } i \in N; \tag{5}$$

$$z_{it}q_i \le \sum_{\tau=1}^{t} x_{i\tau}, \text{ for } i \in N, t \in M; \tag{6}$$

$$x_{it} \in \mathbb{Z}_+, z_{it} \in \{0,1\}, \text{ for } i \in N, t \in M. \tag{7}$$

The objective function (2) used in this formulation is to minimize the total shipping and inventory holding cost. Constraints (3) state that the total quantity of products processed in a day cannot exceed daily production capacity. Constraints (4) ensure that for order i the total quantity of products processed in the first $d_i - e_{i1}$ days equals its ordered quantity to guarantee its committed delivery due date. Constraints (5) ensure that for order i there exists exactly one day on which all products for this order are shipped out. Constraints (6) state the relation between x_{it} and z_{it}. Constraints (7) are integral and binary constraints on decision variables x_{it} and z_{it}, respectively.

4.2 Computational Complexity for SDN

We next show that SDN is ordinarily NP-hard when the length of the planning horizon is fixed and prove that there is no constant factor polynomial-time approximation algorithm for SDN.

4.2.1 NP-hardness for SDN

The problem without split delivery, considered in Stecke and Zhao (2007), can be reduced to SDN when $h_i = 0$, $G_{is}(e_{is}, y) = G(e_{is}, y)$, and $G(t, y)$ is convexly non-increasing with shipping time of t and linearly non-decreasing with shipping quantity of y. From Theorem 2 in Stecke and Zhao

(2007), we know that when the length of the planning horizon is arbitrary, SDN is strongly NP-hard. Next, we investigate the computational complexity of SDN when the length of the planning horizon is fixed.

Theorem 1. *SDN is at least ordinarily NP-hard when the length of the planning horizon is fixed.*

Proof. The following shows that SDN with a fixed length of the planning horizon is at least ordinarily NP-hard by reduction from the PARTITION PROBLEM (PP), which is known to be NP-hard (Gary and Johnson, 1979).

PARTITION PROBLEM (PP). Given a set of r items, $Q = \{1, 2, \ldots, r\}$, where each item $j \in Q$ has a positive integer size a_j with $\sum_{i=1}^{r} a_i = 2A$ for some integer A, the PP is to determine whether or not there exists a subset, $Q' \subseteq Q$, such that $\sum_{i \in Q'} a_i = A$.

Given a PP instance, we construct an instance for SDN as follows: Let $n = r + 1$, $c = 2A$, $q_{r+1} = (2m-3)A$, $d_{r+1} = m-1$, $h_{r+1} = 0$, $K_{r+1} = \{1\}$, $e_{i1} = 0$, and $G_{r+1,1}(0, q_{r+1}) = B$, where B is an arbitrary integer. For $i = 1, \ldots, r$, $q_i = a_i$, $d_i = m$, $h_i = 0$, $K_i = \{1, 2\}$, $e_{i1} = 0$, $e_{i2} = m-1$, $G_{i1}(0, q_i) = mq_i$, and $G_{i2}(m-1, q_i) = q_i$. Let a threshold of the total shipping and inventory holding cost be $(m+1)A + B$.

To prove that SDN with fixed m is NP-hard when $m \geq 2$, we need to show that the instance of PP has a feasible solution if and only if the instance of SDN has a feasible solution with the total cost no more than $(m+1)A + B$.

If Part. Suppose there exists a feasible solution Q' for PP. Based on the feasible solution for PP, we construct a solution (illustrated in Figure 2) for SDN as follows: The products for orders in Q' are processed and shipped out on day 1 by the shipping mode 2. Order $r+1$ is processed immediately after the orders in Q', which implies that order $r+1$ is processed completely on day $m-1$ and shipped out on day $m-1$ by the shipping mode 1. The products for orders in $Q \setminus Q'$ are processed and shipped out on day m by the shipping mode 1. We can see that this solution is feasible for SDN and its objective value is equal to $(m+1)A + B$.

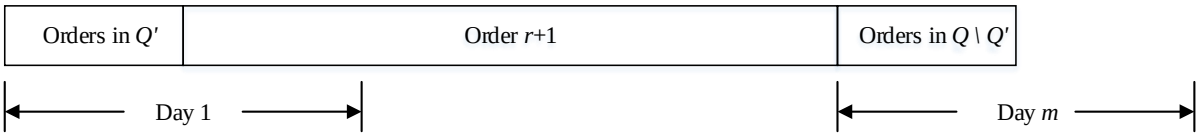

Figure 2: Constructed solution in the proof of Theorem 1

Only If Part. Suppose that there exists a feasible solution for the instance of SDN with the objective value no more than $(m+1)A+B$. As $d_{r+1} = m-1$ and $q_{r+1} = (2m-3)A \geq (d_{r+1}-1)c = (2m-4)A$, order $r+1$ must be processed in the first $m-1$ days and shipped out on day $m-1$ by the shipping mode 1. If all the orders in Q are shipped by the shipping mode 1, the total shipping cost of order $r+1$ and the orders in Q is $2mA+B$ which is larger than $(m+1)A+B$. Thus, some orders in Q are shipped by the shipping mode 2, which implies that these orders are processed in day 1. Let Q' be the set of orders in Q which are processed and shipped out on day 1. Thus, the orders in $Q \setminus Q'$ are processed on day m and shipped out on day m by the shipping mode 1. As $c = 2A$, we have $\sum_{i\in Q'} q_i + q_{r+1} = \sum_{i\in Q'} a_i + (2m-3)A \leq (m-1)c$. This means that $\sum_{i\in Q'} q_i = \sum_{i\in Q'} a_i \leq A$. As the total cost of the solution is no more than $(m+1)A+B$, the total shipping cost and inventory holding cost of all the orders is $\sum_{i\in Q'} G_{i2}(e_{i2}, q_i) + G_{r+1,1}(e_{r+1,1}, q_{r+1}) + \sum_{i\in Q\setminus Q'} G_{i1}(e_{i1}, q_i) = \sum_{i\in Q'} q_i + B + m\sum_{i\in Q\setminus Q'} q_i = \sum_{i\in Q'} a_i + B + m\sum_{i\in Q\setminus Q'} a_i \leq (m+1)A+B$. Because $\sum_{i\in Q'} a_i + \sum_{i\in Q\setminus Q'} a_i = 2A$, we have $(m-1)\sum_{i\in Q\setminus Q'} a_i \leq (m-1)A$, which together with the inequality that $\sum_{i\in Q'} a_i \leq A$, implies that $\sum_{i\in Q'} a_i = \sum_{i\in Q\setminus Q'} a_i = A$. Thus, Q' is a feasible partition of Q for PP. □

From Theorem 1, we know that SDN is at least ordinarily NP-hard, which implies that there is no exact algorithm in polynomial time, unless NP=P. In Section 4.3, we will show that SDN with fixed m is solved in pseudo-polynomial time. Thus, by Theorems 1 and 3, we can immediately obtain the following result.

Corollary 1. *SDN is ordinarily NP-hard when the length of planning horizon is fixed.*

4.2.2 No Polynomial-Time Algorithm with a Constant Worst-case Ratio

Theorem 2. *For SDN, there is no polynomial time algorithm with a constant worst-case ratio unless NP = P.*

Proof. We show the theorem using the NP-hard PARTITION PROBLEM (PP), which is given in the proof of Theorem 1. Given an instance U of PP, we construct an instance I of our problem as follows: Let $N = Q$, $n = r$, $m = 2$, $q_i = a_i$, $d_i = 2$, $h_i = +\infty$, $c = A$, $K_i = \{1, 2\}$, $e_{i1} = 0$, $e_{i2} = 1$, and $G_{is}(e_{is}, y) = 1$ for $i \in N$, $s \in K_i$, and $y \geq 0$.

To prove Theorem 2, we first show the following Results 1 and 2.

Result 1. If there is a feasible solution to the instance U of PP, then the objective value of any optimal solution for instance I is n.

Result 2. If there is no solution to the instance U of PP, then the objective value of any optimal solution for instance I is at least $n + h_{min}$, where $h_{min} = \min\{h_i | i = 1, \ldots, n\} = +\infty$.

To show Result 1, let $Q' \subseteq Q$ be a feasible solution for instance U of PP. Based on the solution for instance U, we construct a corresponding solution for instance I of SDN, as follows: All the products for orders in Q' are processed and shipped out on day 1 by the shipping mode 2. All the products for orders in $Q \setminus Q'$ are processed and shipped out on day 2 by the shipping mode 1. Clearly, the objective value of this solution is n, which is the minimum possible. Thus, this solution is optimal for instance I of SDN, and hence, the objective value of any optimal solution for instance I of SDN is n. This shows Result 1.

To show Result 2, let π be an optimal solution for instance I. In π, let Q' be the set of orders for which all the products are processed and shipped out on day 1. As $c = A$, we have $\sum_{i \in Q'} q_i \leq A$, which together with the fact that for instance U of PP there is no optimal solution obtained in polynomial time, implies $\sum_{i \in Q'} q_i \leq A - 1$. As $\sum_{i \in N} q_i = 2A$, there must exist at least one order, for which some units of the products are processed on day 1 and shipped out on day 2. This incurs an inventory holding cost, which is at least h_{min}. Therefore, the objective value of solution π is at least $n + h_{min}$. This shows Result 2.

Suppose that for instance I of SDN, there exists an approximation algorithm H with a worst-

case ratio of $b(b < \infty)$ in polynomial time, that is, $F^H/F^* \leq b$, where F^H and F^* are the objective values of the solution generated by algorithm H and solution π, respectively. Define a threshold $Z = n+h_{min}$. We will answer whether there exists a feasible solution for instance U of PP by simply comparing Z and F^H. If $F^H < Z$, then $F^* \leq F^H < Z = n + h_{min}$. By Result 2, we have that a feasible solution exists for instance U of PP. If $F^H \geq Z$, we have $F^* \geq \frac{F^H}{b} \geq \frac{Z}{b} = \frac{n+h_{min}}{b} > n$, since $h_{min} = +\infty$. According to Result 1, there exists no feasible solution for instance U. Comparing Z and F_H, we can answer whether there exists a feasible solution for instance U of PP. Thus, PP can be solved by algorithm H in polynomial time, which contradicts the fact that PP is NP-hard. □

4.3 Solution Methods for SDN

In this section, we present an exact algorithm which runs in pseudo-polynomial time for SDN with a fixed length of the planning horizon. Furthermore, we also propose a column generation-based heuristic algorithm to find a feasible solution efficiently.

4.3.1 Exact Algorithm for SDN

As shown in section 4.1, there exists an optimal solution in which each order is shipped out on its production completion day by the slowest shipping mode with its committed delivery due date satisfied. Based on this, we only need to focus on an optimal production schedule which can be obtained by the following Algorithm 1 to solve SDN.

Algorithm 1 : To solve SDN

1. For each order $i \in N$, enumerate all the possible combinations of $(x_{i1}, \ldots, x_{i,d_i-e_{i1}})$ such that $x_{i1} + \ldots + x_{i,d_i-e_{i1}} = q_i$. For each combination of $(x_{i1}, \ldots, x_{i,d_i-e_{i1}})$, let $g(x_{i1}, \ldots, x_{i,d_i-e_{i1}})$ be the total cost of shipping and inventory holding for customer i, i.e., $g(x_{i1}, \ldots, x_{i,d_i-e_{i1}}) = \sum_{\tau=1}^{t}(t-\tau)h_i x_{i\tau} + G_{i,s_{it}}(e_{i,s_{it}}, q_i)$, where $t = \text{argmax}_{\tau=1,\ldots,d_i-e_{i1}}\{x_{i\tau} > 0\}$.

2. Generate a production schedule for NSD using the following dynamic program.

   - *Value function:* Define $f(i; Q_1, \ldots, Q_m)$ as the minimum shipping and inventory cost of a partial production schedule where the first i orders, $\{1, 2, \ldots, i\}$, have been processed

and shipped, and the total quantity of the products for the first i orders processed in day t is Q_t, for $t = 1, \ldots, m$.

- *Boundary conditions:* $f(0; 0, \ldots, 0) = 0$ and $f(0; Q_1, \ldots, Q_m) = +\infty$ if $\sum_{t=1}^{m} Q_t > 0$.
- *Recursive relation:* For $i = 1, \ldots, n$; $Q_t = 0, 1, \ldots, c$, for $t = 1, \ldots, m$, such that $Q_1 + \ldots + Q_m = \sum_{j=1}^{i} q_j$,

$$f(i; Q_1, \ldots, Q_m) = \min \Big\{ f(i-1; Q_1 - x_{i1}, \ldots, Q_{d_i - e_{i1}} - x_{i,d_i-e_{i1}}, Q_{d_i-e_{i1}+1}, \ldots, Q_m) + g(x_{i1}, \ldots, x_{i,d_i-e_{i1}}) \Big| \text{ for all possible combinations of } (x_{i1}, \ldots, x_{i,d_i-e_{i1}}) \text{ such that } x_{i1} + \ldots + x_{i,d_i-e_{i1}} = q_i \text{ and } x_{i\tau} \le Q_\tau \text{ for } \tau = 1, \ldots, d_i - e_{i1} \Big\}.$$

- *Value to return:* The optimal production schedule value is determined as $\min\{f(n; Q_1, \ldots, Q_m) | Q_t \le c \text{ for } t = 1, \ldots, m \text{ such that } Q_1 + \ldots + Q_m = \sum_{i \in N} q_i\}$.

Theorem 3. *Algorithm 1 finds an optimal solution for SDN in $O(n(q_{max})^{m-1}(c^{m-1} + m))$ time, where $q_{max} = \max_{i=1,\ldots,n}\{q_i\}$.*

Proof. For every feasible state $f(i; Q_1, \ldots, Q_m)$, the recursive relation in Algorithm 1 takes into account all possible combinations of $(x_{i1}, \ldots, x_{i,d_i-e_{i1}})$. Given a combination of $(x_{i1}, \ldots, x_{i,d_i-e_{i1}})$, the total shipping and inventory holding cost contributed by the products for order i to the objective value is $g(x_{i1}, \ldots, x_{i,d_i-e_{i1}})$. From the above analysis, the recursive relation in Algorithm 1 computes the value of each $f(i; Q_1, \ldots, Q_m)$ correctly. This shows the validity of the recursive relation in Algorithm 1.

Given the values of $x_{i1}, \ldots, x_{i,d_i-e_{i1}-1}$, by the equation: $x_{i1} + \ldots + x_{i,d_i-e_{i1}} = q_i$, we can calculate the value of $x_{i,d_i-e_{i1}}$. Hence, the number of combinations of $(x_{i1}, \ldots, x_{i,d_i-e_{i1}})$ is at most $(q_{max})^{m-1}$. Given the value of $x_{i1}, \ldots, x_{i,d_i-e_{i1}}$, it takes $O(m)$ time to calculate $g(x_{i1}, \ldots, x_{i,d_i-e_{i1}})$. Thus, the time complexity of Step 1 is bounded by $O(nm(q_{max})^{m-1})$. Given the values of $Q_1, \ldots, Q_{m-1}$, by the equation: $Q_1 + \ldots + Q_m = \sum_{j=1}^{i} q_j$ shown in the recursive relation, we can calculate the value of Q_m. Therefore, together with the fact that $Q_t \le c$ for $t \in M$, there are at most nc^{m-1}

possible states of $(i; Q_1, \ldots, Q_m)$. Given a state $(i; Q_1, \ldots, Q_m)$, the recursive relation considers all possible combinations of $(x_{i1}, \ldots, x_{i,d_i - e_{i1}})$. Thus, given the state $(i; Q_1, \ldots, Q_m)$, it takes $O((q_{max})^{m-1})$ time to calculate $f(i; Q_1, \ldots, Q_m)$. Thus, the time complexity of Step 2 is bounded by $O(nc^{m-1}(q_{max})^{m-1})$. Therefore, the overall time complexity of Algorithm 1 is bounded by $O(nc^{m-1}(q_{max})^{m-1} + nm(q_{max})^{m-1})$. □

By Theorem 3, it can be seen that in a case where the number of days is fixed, Algorithm 1 can solve SDN to optimality in pseudo-polynomial time.

4.3.2 Heuristic Algorithm for SDN

From Corollary 1, we know that SDN is ordinarily NP-hard when the length of the planning horizon is fixed and it becomes strongly NP-hard when the length of the planning horizon is arbitrary. Hence, there is no exact algorithm in polynomial time for SDN, unless NP = P. From Theorem 2, we also know that there is no polynomial-time heuristic algorithm with a constant worst-case ratio for SDN. Therefore, solving SDN is difficult. In this section, we propose a column generation-based heuristic algorithm to quickly generate a near-optimal solution for SDN.

The main idea of the heuristic algorithm is as follows: We first present a Dantzig-Wolfe reformulation for SDN, which includes a large number of columns, then solve the relaxation of the formulation using a column generation approach, and finally solve the restricted formulation with the columns generated to obtain a feasible solution for SDN.

Any feasible solution for SDN consists of n subsolutions, one for each order $i \in N$. Let S_i be the set of all feasible subsolutions for order $i \in N$. Each subsolution $\pi \in S_i$ represents a production schedule for order i and specifies the quantity of products for order i processed in each day. Hence, each subsolution $\pi \in S_i$ can be represented by a vector $(x_{i1}^{\pi}, \ldots, x_{im}^{\pi})$, where x_{it}^{π} represents the quantity of the products processed in day t for customer i. Note that given the production schedule for order $i \in N$, we know that the shipping schedule for order i can be determined accordingly. More specifically, given the production schedule for order i, we can find day t in which the last part of order i is processed completely. From Property 1, each subsolution in S_i also specifies the

shipping day of order i. Thus, we can calculate the contribution of the products for each order to the objective value as follows: For order i, let f_i^π be the total shipping and inventory holding cost of subsolution $\pi \in S_i$ and $t_i^\pi = \text{argmax}_{t=1,\ldots,d_i-e_{i1}}\{x_{it}^\pi > 0\}$. Thus,

$$f_i^\pi = \sum_{\tau=1}^{t_i^\pi}(t_i^\pi - \tau)h_i x_{i\tau}^\pi + G_{i,s_{i,t_i^\pi}}(e_{i,s_{i,t_i^\pi}}, q_i). \tag{8}$$

We define binary variable θ_i^π to be 1 if subsolution $\pi \in S_i$ for order i is selected in the final solution for SDN, and 0 otherwise. SDN can be formulated as the following integer program, denoted as SDN-IP.

$$\text{SDN-IP} \qquad \min \sum_{i=1}^{n}\sum_{\pi\in S_i} f_i^\pi \theta_i^\pi \tag{9}$$

$$\text{s.t.} \sum_{\pi\in S_i} \theta_i^\pi = 1, \text{ for } i = 1,\ldots,n; \tag{10}$$

$$\sum_{i=1}^{n}\sum_{\pi\in S_i} x_{it}^\pi \theta_i^\pi \le c, \text{ for } t = 1,\ldots,m; \tag{11}$$

$$\theta_i^\pi \in \{0,1\}, \text{ for } i \in N, \pi \in S_i. \tag{12}$$

The objective function (9) minimizes the total cost of the final solution. Constraints (10) ensure that one subsolution is chosen for each order and constraints (11) ensure that the total quantity of the products processed in each day does not exceed the daily production capacity.

We relax constraints (12) by replacing them with $0 \le \theta_i^\pi \le 1$ for $i \in N, \pi \in S_i$ to make the relaxed problem (denoted as SDN-LP) relatively easier to solve. To solve SDN-LP, we use the column generation approach (Refer to Desaulniers et al. (2006) and Lübbecke (2010) for more details), which decomposes SDN-LP into a master problem (denoted as SDN-RLP) and n pricing subproblems, one for each order. The master problem is the restricted version of SDN-LP which includes only a reasonably small subset of the columns corresponding to subsolutions. More columns are added only when needed. The column generation approach solves iteratively SDN-RLP. After solving SDN-RLP in each iteration, we can obtain the dual values corresponding to constraints (10)

and (11) to update the pricing subproblems, and subsequently we solve n pricing subproblems. The pricing subproblems are new problems created to identify new promising columns. The objective functions of the pricing subproblems are the reduced costs of new columns with respect to current dual variables. In each iteration, at most n new columns with the minimum negative reduced costs (one associated with the subsolution for one pricing subproblem) generated by solving the pricing subproblems are added into SDN-RLP.

Let α_i and β_t be the dual values corresponding to constraints (10) and (11), respectively. The reduced cost $f_i^{'\pi}$ of any subsolution π for order i is given by

$$f_i^{'\pi} = f_i^{\pi} - \alpha_i - \sum_{t=1}^{m} x_{it}^{\pi} \beta_t.$$

The pricing subproblem associated with order i is to find a feasible subsolution π with a minimum reduced cost $f_i^{'\pi}$ for order i. If there exist subsolutions with negative reduced costs, one would like to find a subsolution with the minimum reduced cost and add it into SDN-RLP. If there exist no subsolutions with negative reduced costs for all pricing subproblems, SDN-LP is solved to optimality.

Next we propose the following Algorithm 2 to solve the pricing subproblem associated with order i to optimality.

Algorithm 2 : To solve the pricing subproblem for SDN

1. *Value function:* Define $f(t, Q)$ as the minimum reduced cost of a partial subsolution where the total quantity of the products for order i processed in the first t days is Q.

2. *Boundary conditions:* $f(0, 0) = -\alpha_i$; $f(0, Q) = +\infty$ if $Q > 0$.

3. *Recursive relation:* For $t = 1, \ldots, d_i - e_{i1}$, $Q = 0, 1, \ldots, \min(tc, q_i)$,

$$f(t, Q) = \min_{q'=0,1,\ldots,\min(c,Q)} \begin{cases} f(t-1, Q-q') + G_{i,s_{it}}(e_{i,s_{it}}, q_i) - \beta_t q', & \text{if } Q = q_i \text{ and } q' > 0; \\ f(t-1, Q), & \text{if } Q = q_i \text{ and } q' = 0; \\ f(t-1, Q-q') + h_i Q - \beta_t q', & \text{if } Q < q_i. \end{cases}$$

4. *Value to return:* The optimal subsolution value is determined as $f(d_i - e_{i1}, q_i)$.

Theorem 4. *Algorithm 2 finds an optimal subsolution for the pricing subproblem of SDN in* $O\Big(m(q_{max})^2\Big)$ *time, where* $q_{max} = \max_{i=1,\ldots,n}\{q_i\}$.

Proof. Suppose that there exists a partial subsolution for order i, where the total quantity of the products processed in the first t days is Q. In the recursive relation of Algorithm 2, we take into account three possible cases for the partial subsolution associated with $f(t, Q)$: (i) If $Q = q_i$ and $q' > 0$, we know that all the products for order i are shipped out on day t. Thus, the contribution of the products for order i in day t to the objective value is $G_{i,s_{it}}(e_{i,s_{it}}, q_i) - \beta q'$. (ii) If $Q = q_i$ and $q' = 0$, we know that all the products for order i have been shipped out in one of the first $t-1$ days. Thus, the contribution of the products for order i in day t to the objective value is 0. (iii) If $Q < q_i$, we know that the products for order i are not processed completely in the first t days and no products for order i are shipped out on day t. Thus, the contribution of the products for order i in day t to the objective value is $h_i Q - \beta_t q'$. This shows the validity of the recursive relation in Algorithm 2.

Given a state (t, Q), there are at most q_{max} possible values of q', and it takes $O(q_{max})$ time to calculate $f(t, Q)$. There are at most $O(m(q_{max}))$ states of (t, Q). Therefore, the time complexity of Algorithm 2 is bounded by $O(m(q_{max})^2)$. □

To accelerate the column generation method, we generate a number of initial columns. First, we generate a initial schedule for SDN as follows: we sort orders in non-decreasing order of their committed delivery due dates. The products for orders are processed in this order and shipped out on their production completion days. From condition (1), the initial schedule is a feasible solution to SDN. Then, the column corresponding to each order in the feasible solution is added into SDN-RLP. These columns together become the initial columns of SDN-RLP.

5 Solving the Problem with Split Delivery

Note that we denote the problem with split delivery as SD. Li et al. (2020) consider a case in which each customer has exactly one order. The case considered in Li et al. (2020) is a special

case of SD with $h_i = h$. From Theorems 1 and 2 in Li et al. (2020), we know that (i) when the number of days is arbitrary, SD is strongly NP-hard; (ii) when the number of days is fixed, SD is at least ordinarily NP-hard; and (iii) SD has no FPTAS, unless NP=P. However, Li et al. (2020) do not clarify whether or not there exists a pseudo-polynomial time exact algorithm for SD when the number of days is fixed.

In this section, we formulate SD as an integer programming model given in Section 5.1, propose an exact algorithm to solve SD to optimality in Section 5.2, prove that the exact algorithm achieves pseudo-polynomial running time for SD with a fixed length of planning horizon, and provide a column generation based heuristic algorithm for SD in Section 5.3.

5.1 Integer Programming Model for SD

Let decision variables x_{it} for $i \in N$ and $t \in M$ represent the quantity of the products processed in day t for customer i. Let decision variables y_{it} for $i \in N$ and $t \in M$ represent the quantity of the products shipped out on day t for customer i. Let $g_{it}(y_{it})$ be the minimum shipping cost of the y_{it} products shipped out on day $t \in M$ for customer $i \in N$. From Property 1, it is optimal that the products, shipped out on day t for customer i, are shipped by the shipping mode s_{it}. Thus, if $t + e_{i1} \leq d_i$, then let $g_{it}(y_{it}) = G_{i,s_{it}}(e_{i,s_{it}}, y_{it})$ and otherwise, $g_{it}(y_{it}) = +\infty$. Next we use the decision variables x_{it} and y_{it} to formulate SD as an integer program (denoted as SD-I) as follows.

$$\text{SD-I} \quad \min \sum_{i\in N}\sum_{t=1}^{d_i-e_{i1}} g_{it}(y_{it}) + \sum_{i\in N}\sum_{t=1}^{d_i-e_{i1}} h_i \sum_{\tau=1}^{t}(x_{i\tau}-y_{i\tau}) \tag{13}$$

$$\text{s.t.} \sum_{i\in N} x_{it} \le c, \text{ for } t \in M; \tag{14}$$

$$\sum_{t=1}^{d_i-e_{i1}} x_{it} = q_i, \text{ for } i \in N; \tag{15}$$

$$\sum_{\tau=1}^{t} y_{i\tau} \le \sum_{\tau=1}^{t} x_{i\tau}, \text{ for } i \in N; t = 1,\ldots,d_i-e_{i1}; \tag{16}$$

$$\sum_{t=1}^{d_i-e_{i1}} y_{it} = q_i, \text{ for } i = 1,\ldots,n; \tag{17}$$

$$x_{it}, y_{it} \in \mathbb{Z}_+, \text{ for } i \in N; t = 1,\ldots,d_i-e_{i1}; \tag{18}$$

The objective function (13) used in this formulation minimizes the total shipping cost and inventory holding cost. Constraints (14) state that the total quantity of products processed in a day cannot exceed the daily production capacity. Constraints (15) ensure that for customer i, the total quantity of the products processed in the first $d_i - e_{i1}$ days equals its order quantity. Constraints (16) ensure that for customer i, the total quantity of products shipped in the first t days cannot exceed that processed in the first t days. Constraints (17) ensure that for customer i, the total quantity of the products shipped in the first $d_i - e_{i1}$ days equals its order quantity. Constraints (18) are integral and binary constraints on decision variables x_{it} and y_{it}, respectively.

5.2 Exact Algorithm for SD

We now show that SD can be solved by an exact algorithm based on the dynamic programming algorithm. Given a production schedule $(x_{i1},\ldots,x_{im})$ for order $i \in N$, we use the following Algorithm 3 to determine an optimal shipping schedule for order i.

Algorithm 3 : To find an optimal shipping schedule for SD

1. *Value function:* Define $f(t,Q)$ as the minimum total cost of a partial shipping schedule, where the total quantity of products for order i shipped in the first t days is Q.

2. *Boundary conditions:* $f(0,0) = 0$; $f(0,Q) = +\infty$ for any $Q > 0$.

3. *Recursive relation:* For $t = 1, \ldots, d_i - e_{i1}$; $Q = 0, 1, \ldots, \sum_{\tau=1}^{t} x_{i\tau}$.

$$f(t,Q) = \min\Big\{ f(t-1, Q-q') + G_{i,s_{it}}(e_{i,s_{it}}, q') + h_i(\sum_{\tau=1}^{t} x_{i\tau} - Q) \\ | q' = \max(0, Q - \sum_{\tau=1}^{t-1} x_{i\tau}), \ldots, Q \Big\}.$$

4. *Value to return:* The optimal shipping schedule value is determined as $f(d_i - e_{i1}, q_i)$.

Theorem 5. *Given a production schedule $(x_{i1}, \ldots, x_{im})$ for order $i \in N$, Algorithm 3 finds an optimal shipping schedule for this order in $O(m(q_{max})^2)$ time, where $q_{max} = \max_{i=1,\ldots,n}\{q_i\}$.*

Proof. Given a production schedule $(x_{i1}, \ldots, x_{im})$ for order i, suppose that a partial shipping schedule associated with (t, Q) has been formed. In the partial shipping schedule, the total quantity of products for order i shipped in the first t days is Q. The recursive relation enumerates all possible values of q', which represents the quantity of the products for order i shipped out on day t. The total shipping and inventory holding cost of the products for order i in day t is $G_{i,s_{it}}(e_{i,s_{it}}, q') + h_i(\sum_{\tau=1}^{t} x_{i\tau} - Q)$. Hence, the recursive relation of Algorithm 3 compares all possible shipping schedules for order i and this shows the validity of the recursive relation in Algorithm 3.

Given a production schedule $(x_{i1}, \ldots, x_{im})$ for order i, in Algorithm 3, there are at most $mq_{\max}$ states of (t, Q). We can have that there are at most $q_{\max}$ values of q', and hence, it takes $O(q_{\max})$ time to calculate $f(t, Q)$. Therefore, the time complexity of Algorithm 3 is bounded by $O(m(q_{max})^2)$. □

Based on Algorithm 3, we provide the following Algorithm 4 to solve SD to optimality.

Algorithm 4 : To solve SD

1. For each order $i \in N$, enumerate all the possible combinations of $(x_{i1}, \ldots, x_{im})$, such that $x_{i1} + \ldots + x_{im} = q_i$, $x_{it} \in \{0, 1, \ldots, c\}$ for $t = 1, \ldots, d_i - e_{i1}$, and $x_{it} = 0$ for $t = d_i - e_{i1} + 1, \ldots, m$.

2. For each combination $(x_{i1}, \ldots, x_{im})$, $i \in N$, calculate its corresponding cost, $g_i(x_{i1}, \ldots, x_{im}) = f(d_i - e_{i1}, q_i)$, by using Algorithm 3.

3. Generate an optimal solution for SD using the same algorithm as Algorithm 1, except that the recursive relation in Algorithm 1 is replaced by the following.

$$f(i; Q_1, \ldots, Q_m) = \min \Big\{ f(i-1; Q_1 - x_{i1}, \ldots, Q_m - x_{im}) + g_i(x_{i1}, \ldots, x_{im}) \Big| \text{ for all possible combinations of } (x_{i1}, \ldots, x_{im}) \text{ such that } x_{i1} + \ldots + x_{im} = q_i, \; x_{it} \le Q_t \text{ for } t = 1, \ldots, d_i - e_{i1}, \text{ and } x_{it} = 0 \text{ for } t = d_i - e_{i1} + 1, \ldots, m \Big\}.$$

Theorem 6. *Algorithm 4 finds an optimal solution for SD in* $O(n(q_{max})^{m-1}(c^{m-1} + m(q_{max})^2))$ *time, where* $q_{max} = \max_{i=1,\ldots,n}\{q_i\}$.

Proof. From Theorem 5, we know that given a combination of $(x_{i1}, \ldots, x_{im})$ for order $i \in N$, one can calculate $g_i(x_{i1}, \ldots, x_{im})$. Given the values of $g_i(x_{i1}, \ldots, x_{im})$ for all possible combinations of $(x_{i1}, \ldots, x_{im})$, by a similar argument to that in proof of Theorem 3, we obtain the validity of the recursive relation in Step 3.

For each $i \in N$, there are at most $O((q_{max})^{m-1})$ possible combinations of $(x_{i1}, \ldots, x_{im})$. From Theorem 5, given a combination of $(x_{i1}, \ldots, x_{im})$, it takes $O(m(q_{max})^2)$ time to calculate the value of $g_i(x_{i1}, \ldots, x_{im})$. Hence, the time complexity of Step 2 is bounded by $O(nm(q_{max})^{m+1})$. From Theorem 3, the time complexity of Step 3 is bounded by $O(nc^{m-1}(q_{max})^{m-1})$. Therefore, the overall time complexity of Algorithm 4 is bounded by $O(n(q_{max})^{m-1}(c^{m-1} + m(q_{max})^2))$. □

By Theorem 6, it can be seen that in a case where the number of days is fixed, Algorithm 4 can solve SD to optimality in pseudo-polynomial time.

5.3 Heuristic Algorithm for Problem SD

Note that the column generation-based heuristic algorithm in Li et al. (2020) cannot be applied immediately to SD because of order-dependent inventory holding cost. More specifically, the pricing

subproblem in Li et al. (2020) does not need to take into account the inventory holding cost. However, the pricing subproblem for SD must make a balance between order-dependent inventory holding cost and shipping cost. For SD, we propose the same heuristic algorithm as that for SDN, except for the following aspects. It is easy to see that the initial feasible solution generated as shown in Section 4.3.2 is also feasible for SD. We use the same notation and definitions given in Section 4.3.2. Let a vector $(y^{\pi}_{i1}, \ldots, y^{\pi}_{im})$ be the shipping schedule in subsolution $\pi \in S_i$, where y^{π}_{it} represents the quantity of the products shipped out on day t for customer i. Given $(x^{\pi}_{i1}, \ldots, x^{\pi}_{im})$, from Algorithm 3, we know that the shipping schedule of subsolution π can be determined accordingly. Therefore, we also assume that each subsolution $\pi \in S_i$ specifies the shipping schedule for order i. Thus, the total shipping and inventory holding cost of subsolution $\pi \in S_i$ can be calculated as follows:

$$f^{\pi}_i = \sum_{t=1}^{d_i - e_{i1}} \Big(G_{i,s_{it}}(e_{i,s_{it}}, y^{\pi}_{it}) + h_i \sum_{\tau=1}^{t} (x^{\pi}_{i\tau} - y^{\pi}_{i\tau}) \Big).$$

We use the same integer program as SDN-IP given in Section 4.3.2 to formulate SD which is denoted as SD-IP. Next, we use a different algorithm (i.e., Algorithm 5) to solve the pricing subproblem associated with order i to optimality.

Algorithm 5 : To solve the pricing subproblem for SD

1. *Value function:* Define $f(t, Q, Y)$ as the minimum reduced cost of a partial subsolution, where the total quantities of the products for order i processed and shipped in the first t days are Q and Y, respectively.

2. *Boundary conditions:* $f(0, 0, 0) = -\alpha_i$; $f(0, Q, Y) = +\infty$ for any $Q > 0$ and $Y > 0$.

3. *Recursive relation:* For $t = 1, \ldots, d_i - e_{i1}$; $Q = 0, 1, \ldots, \min(tc, q_i)$; $Y = 0, 1, \ldots, Q$.

$$f(t, Q, Y) = \min \Big\{ f(t-1, Q-q', Y-y') + G_{i,s_{it}}(e_{i,s_{it}}, y') + h_i(Q - Y) - \beta_t q', \\ | q' = 0, 1, \ldots, \min(c, Q), y' = \max(0, Y - Q + q'), \ldots, Y \Big\},$$

4. *Value to return:* The optimal subsolution value is determined as $f(d_i - e_{i1}, q_i, q_i)$.

Theorem 7. *Algorithm 5 finds an optimal solution for the pricing subproblem of SD in* $O\Big(m(q_{max})^4\Big)$ *time, where* $q_{max} = \max_{i=1,\ldots,n}\{q_i\}$.

Proof. Suppose that there exists a partial subsolution for order i, where the total quantities of the products processed and shipped in the first t days are Q and Y, respectively. The recursive relation in Algorithm 5 takes into account all possible values of q' and y', where q' and y' represent the quantities of the products processed and shipped out on day t, respectively. The total shipping and inventory holding cost of the products for order i in day t is $G_{i,s_{it}}(e_{i,s_{it}}, y') + h_i(Q - Y)$. Hence, the recursive relation of Algorithm 5 compares all possible subsolutions for order i, and this shows the validity of the recursive relation in Algorithm 5.

Given each state (t, Q, Y), there are at most $(q_{max})^2$ possible combinations of values of q' and y', and hence, it takes $O((q_{max})^2)$ time to calculate $f(t, Q, Y)$. There are at most $O(m(q_{max})^2)$ states of (t, Q, Y). Therefore, the time complexity of Algorithm 5 is bounded by $O(m(q_{max})^4)$. □

6 Computational Experiments

We now evaluate the performances of the heuristic algorithms for SDN and SD based on instances generated randomly. Both the heuristic algorithms for SDN and SD are coded in C++ and implemented on a PC with 16.0GB of RAM and an Intel(R) Core(TM) I7-10510U CPU @ 1.80GHz 2.30GHz. We use CPLEX 12.6.3 to solve the master problem and the restricted integer program. To obtain efficiently feasible solutions for SDN and SD, we instruct CPLEX to stop as soon as it has found a feasible integer solution with the objective value which lies within 1% of the optimum for the restricted integer program.

To systematically evaluate the performances of our heuristic algorithms, we randomly generate two test instance sets under all-unit discount cost structure and step-wise cost structure. Except for different cost structures, both sets are generated in the same way as follows.

- Number of customers $n \in \{100, 200, 300, 400, 500\}$.
- Number of days $m \in \{5, 10, 15\}$.

- Committed delivery due date of each customer i: d_i is an integer randomly drawn from $\{1, \ldots, m\}$.
- Set of shipping modes for each customer i: $K_i = \{1, \ldots, d_i\}$.
- Shipping time of each shipping mode s: $e_{is} = s - 1$.
- Quantity of each customer i: q_i is an integer randomly drawn from $\{20, ..., 40\}$.
- Daily production capacity: c is an integer randomly drawn from $\{\hat{c}, \ldots, 1.2\hat{c}\}$, where $\hat{c} = \max_{t=1,...,m} \left\lceil \sum_{i \in N: d_i - e_{i1} \leq t} q_i / t \right\rceil$, to ensure that there exist feasible solutions.
- Shipping cost functions under the all-unit discount cost structure: for each customer $i \in N$ and $s \in K_i$, we first define a shipping cost function $G_{is}(e_{is}, y)$ (shown in Figure 3) as follows:

$$G_{is}(e_{is}, y) = \begin{cases} 0, & y = 0, \\ a_{is1} l_{is1}, & y \in (l_{is0}, l_{is1}], \\ a_{is1} y, & y \in (l_{is1}, l_{is2}], \\ a_{is2} l_{is3}, & y \in (l_{is2}, l_{is3}], \\ a_{is2} y, & y \in (l_{is3}, l_{is4}], \end{cases}$$

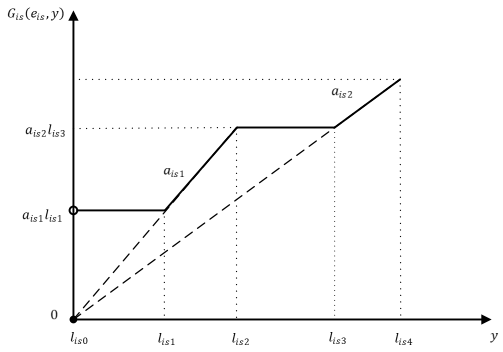

Figure 3: All-unit discount cost structure

where $0 = l_{is0} < l_{is1} < l_{is2} < l_{is3} < l_{is4} = \max\{q_i, l_{is3} + 1\}$ and the parameters a_{isr} and l_{isr} are determined as follows:

  - The discount price a_{i11} is an integer randomly drawn from $[10, 20]$, each discount price a_{is1} for $s = 2, \ldots, d_i$ is randomly drawn from $[0.9a_{i,s-1,1}, 1.0a_{i,s-1,1}]$, and each other discount price a_{is2} for $s = 1, ..., d_i$ is randomly drawn from $[0.7a_{is1}, 0.8a_{is1}]$, where $a_{is2} \leq a_{i,s-1,2}$ for $s = 2, \ldots, d_i$, so as to exhibit economies of scale.
  - The parameter $l_{isr} = rq_i/4$ for $r = 1, 2$, and by the fact that $a_{is1} l_{is2} = a_{is2} l_{is3}$, the value of l_{is3} can be calculated accordingly, so that the difference between any two consecutive breakpoints is identical.

- Shipping cost functions under step-wise cost structure: for each customer $i \in N$ and $s \in K_i$, we first define a shipping cost function $G_{is}(e_{is}, y)$ (shown in Figure 4) as follows:

$$G_{is}(e_{is}, y) = \begin{cases} 0, & y = 0, \\ b_{is1}, & y \in (l_{is0}, l_{is1}], \\ b_{is2}, & y \in (l_{is1}, l_{is2}], \\ b_{is3}, & y \in (l_{is2}, l_{is3}], \\ b_{is4}, & y \in (l_{is3}, l_{is4}], \end{cases}$$

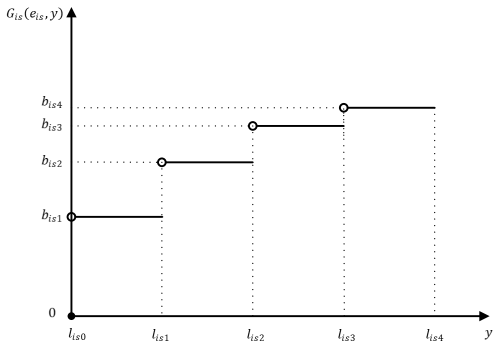

Figure 4: Step-wise cost structure

where $0 = l_{is0} < l_{is1} < l_{is2} < l_{is3} < l_{is4} \leq q_i$ and the parameters b_{isr} and l_{isr} are determined as follows:

  - b_{i11} is an integer randomly drawn from [100, 200] and b_{is1} is randomly drawn from $[0.9b_{i,s-1,1}, 1.0b_{i,s-1,1}]$ for $s = 2, \ldots, d_i$. $b_{isr} = b_{is,r-1} + (b_{is,r-1} - b_{is,r-2}) \cdot \alpha_{r-1}$ where $b_{is0} = 0$ and α_{r-1} is randomly drawn from $[90\%, 100\%]$ for $r = 2, 3, 4$, $s = 1, \ldots, d_i$, so as to exhibit economies of scale.
  - The parameter $l_{isr} = rq_i/4$ for $r = 1, \ldots, 4$, so that the difference between any two consecutive breakpoints is identical.

- Unit inventory holding cost: h_i is randomly drawn from interval $[0.1\overline{G}, 0.3\overline{G}]$, where $\overline{G} := G_{i1}(e_{i1}, q_i)/q_i$, which represents the average shipping cost if all the products for order $i \in N$ are shipped by the fastest shipping mode.

For each combination of (m, n) we generate 10 test instances. Thus, there is a total of 150 test instances for each of the two instance sets. These test instances together have a great variety and a majority of them are on a large scale, as measured by the number of orders. Let Z_{SDN} (Z_{SD}) and LB be the total cost of the solution generated by the heuristic algorithm and the lower bound obtained by solving the relaxation of the master problem for SDN (SD), respectively. The relative gap between the objective value Z_{H} of the solution generated by the heuristic algorithm and LB is

defined as $(Z_{\mathrm{H}} - LB)/LB \times 100\%$, for H = {SDN, SD}. In Tables 1 and 2, we report the average gaps in % (column "Ave(%)") , the maximum gaps in % (column "Max(%)") , and the average running times in seconds (column "Time(s)") of the heuristic algorithms for SDN and SD over the ten instances for each combination of (m, n).

The computational results in Tables 1 and 2 demonstrate that for SDN (SD), the heuristic algorithm is capable of generating near-optimal solutions, with an overall average gap of 0.26% (0.25%) for the instances under all-unit discount cost structure and 0.29% (0.28%) for the instances under step-wise cost structure, in a short running time. This implies that the heuristic algorithms are of great practical value and we may also draw the conclusion that the lower bound is extremely strong. This is probably because of the following reasons. SDN-LP and the linear programming relaxation of SD-IP (denoted as SD-LP) have the potential to yield a tight lower bound on the optimum (Lübbecke and Desrosiers (2005)). From the computational results, we find that the optimal solution for SDN-LP (SD-LP) is very close to the feasible solution generated by the heuristic algorithm for SDN (SD). More specifically, let n' denote the number of the customers whose integrated production and shipping schedules in the optimal solution for SDN-LP (SD-LP) are the same as those in the feasible solution generated by the heuristic algorithm for SDN (SD). For example, for the instances under step-wise cost structure and with $n = 100$ and $m = 5$ ($n = 100$ and $m = 10$), we observe that $n' \times 100\%/n = 90.7\%(83.1\%)$, which means that most columns chosen in the optimal solution for LP and the feasible solution generated by the heuristic algorithm are the same. For all the instances, the gaps for SDN (SD) are no more than 1.64% (1.54%) at maximum for the instances under all-unit discount cost structure and 1.24% (1.18%) at maximum for the instances under step-wise cost structure. For some instances, the larger average gaps for SDN than those for SD, are largely because of the effect of benefits from allowing split delivery. For some instances, we also see the larger average gaps for SD than those for SDN, because split delivery is allowed in SD, which makes it more difficult for the heuristic algorithm to find a high-quality solution. For most instances, the average running times for SDN and SD increase as the number of orders increases and the number of days increases. This can be generally explained as the difficulty caused by problem sizes. From Tables 1 and 2, we also observe that the average running times for SD are

larger than those for SDN, which is consistent with the result that the time complexity for solving the pricing subproblem of SD is higher than that for solving the pricing subproblem of SDN.

Table 1: Results of solution methods for SDN and SD under all-unit discount cost structure

		SDN			SD		
m	n	Ave(%)	Max(%)	Time(s)	Ave(%)	Max(%)	Time(s)
5	100	0.18	0.36	0.1	0.17	0.39	1.4
	200	0.11	0.19	0.1	0.10	0.18	2.8
	300	0.08	0.22	0.2	0.08	0.13	4.2
	400	0.08	0.13	0.2	0.07	0.15	5.7
	500	0.05	0.09	0.2	0.14	1.00	7.1
10	100	0.65	1.20	0.2	0.50	0.84	4.3
	200	0.29	0.67	0.3	0.22	0.38	8.9
	300	0.18	0.26	0.4	0.21	0.31	13.5
	400	0.16	0.24	0.5	0.15	0.29	18.3
	500	0.11	0.21	0.5	0.09	0.16	22.2
15	100	0.78	1.64	1.8	0.72	1.54	9.6
	200	0.45	0.99	1.1	0.44	0.95	17.4
	300	0.34	0.67	0.8	0.34	0.73	27.7
	400	0.28	0.41	0.9	0.28	0.65	38.7
	500	0.19	0.34	1.0	0.20	0.38	46.6
Average		0.26	0.51	0.5	0.25	0.54	15.2

Table 2: Results of solution methods for SDN and SD under step-wise cost structure

		SDN			SD		
m	n	Ave(%)	Max(%)	Time(s)	Ave(%)	Max(%)	Time(s)
5	100	0.30	1.05	0.1	0.30	1.05	1.3
	200	0.19	1.00	0.1	0.20	0.97	2.8
	300	0.14	0.91	0.2	0.15	0.89	4.2
	400	0.38	0.98	0.1	0.35	0.97	5.5
	500	0.20	0.95	0.2	0.21	0.95	6.9
10	100	0.51	1.10	0.2	0.46	1.18	4.3
	200	0.19	0.33	0.3	0.20	0.40	8.8
	300	0.16	0.35	0.4	0.19	0.61	13.3
	400	0.21	1.02	0.5	0.21	0.95	17.9
	500	0.10	0.19	0.8	0.13	0.25	22.6
15	100	0.66	1.24	2.0	0.68	1.15	9.5
	200	0.47	0.77	0.7	0.41	0.80	17.5
	300	0.25	0.41	1.1	0.26	0.51	27.2
	400	0.29	0.49	1.2	0.27	0.65	47.1
	500	0.22	0.39	1.6	0.24	0.54	48.0
Average		0.29	0.75	0.6	0.28	0.79	15.8

7 Problem Variants

In this section, we consider two interesting practical variants of SDN and SD. In Section 7.1, we consider a situation where idle times between products are not allowed during product processing for both SDN and SD. In Section 7.2, we extend SDN and SD to the case taking into consideration production capacity, inventory capacity, shipping capacity, and order acceptance.

7.1 Problems without Idle Times Allowed

In previous sections, we have examined SDN and SD, in which idle times between products are allowed during product processing. However, in some applications such as steel production, products are often processed continuously without idle time, until all products are completed, as a large cost is incurred to resume production if the production line stops temporarily. Therefore, we now consider problems in which idle times between products are not allowed during product processing. Most of our results can be easily extended to corresponding problems without idle times allowed.

First, Theorems 1 and 2 also hold for SDN without idle times allowed and the computational complexity results for SD with idle times allowed can be also applied to SD without idle times allowed. This is because in all the NP-hardness proofs, the production capacity is set in a way that no idle time is possible between products during product processing.

When idle times are not allowed, SDN is still solved by the same algorithm as Algorithm 1, except that the optimal solution value is determined as

$$f(n; Q_1, \ldots, Q_m) \text{ where } Q_t = c \text{ for } t = 1, \ldots, \lceil \sum_{i \in N} q_i/c \rceil - 1 \text{ and}$$

$$Q_{\lceil \sum_{i \in N} q_i/c \rceil} = \sum_{i \in N} q_i - (\lceil \sum_{i \in N} q_i/c \rceil - 1)c, \text{ and } Q_t = 0, \text{ for } t = \lceil \sum_{i \in N} q_i/c \rceil + 1, \ldots, m. \quad (19)$$

Using the same proof as that of Theorem 5, we can see that Theorem 5 also holds for SD without idle times allowed. We use the same algorithm as Algorithm 4, except that in Step 3, the optimal solution value is determined by (19), to solve SD without idle times allowed.

For SDN(SD) without idle times allowed, we could propose the same heuristic algorithm as that for SDN(SD), except that we replace the constraints (11) by the following constraints:

$$\sum_{i=1}^{n} \sum_{\pi \in S_i} x_{it}^{\pi} \theta_i^{\pi} = c, \text{ for } t = 1, \ldots, \lceil \sum_{i \in N} q_i/c \rceil - 1;$$

and

$$\sum_{i=1}^{n} \sum_{\pi \in S_i} x_{it}^{\pi} \theta_i^{\pi} = \sum_{i \in N} q_i - (\lceil \sum_{i \in N} q_i/c \rceil - 1)c, \text{ for } t = \lceil \sum_{i \in N} q_i/c \rceil.$$

7.2 Problems with Production Capacity, Inventory Capacity, Shipping Capacity, and Order Acceptance

In Sections 4 and 5, we have studied SDN and SD where each day has a production capacity limitation, but the inventory capacity of the warehouse and the shipping capacity of each shipping mode are not limited. In addition, in SDN and SD all the orders must be processed and shipped to their customers before or on their committed due dates. However, in many practical applications, decision makers must consider simultaneously production, inventory, and shipping capacities in

the decision-making process. For example, Atamtürk and Küçükyavuz (2008) has shown some emerging industrial applications with limited inventory capacity. In this section, we briefly discuss an extension with production capacity, inventory capacity, shipping capacity, and order acceptance for SDN and SD. In the extension, each day has a production capacity limit (denoted by c_P). The warehouse has an inventory capacity limit (denoted by c_I) such that the total quantity of the products processed completely but not shipped out at the end of each day cannot exceed c_I. Each shipping mode $s \in K_i$ for $i \in N$ has a shipping capacity limit (denoted by c_{si}) such that the total quantity of products shipped by shipping mode s cannot exceed c_{si} but each shipping mode can be applied more than once. In the extension, let p_i be the penalty per unit product per day for delay and r_i be penalty for rejecting order i, for $i \in N$. The problem is to find an integrated schedule of production and shipping such that the sum of the total cost of shipping and inventory holding and the total penalty for delay and rejection is minimized. We refer to the extension of SDN and SD as Ex-SDN and Ex-SD, respectively. Note that condition (1) may not be satisfied and Property 1 may not hold for EX-SDN and EX-SD because some orders may be delayed or rejected.

From Sections 4 and 5, we know that Ex-SDN and Ex-SD are all at least ordinarily NP-hard when the number of days is fixed and become strongly NP-hard when the number of days is arbitrary, because Ex-SDN and Ex-SD are more general forms of SDN and SD, respectively. Next we propose pseudo-polynomial time exact algorithms and column generation-based heuristic algorithms for Ex-SDN and Ex-SD, which implies that both Ex-SDN and Ex-SD are also ordinarily NP-hard.

7.2.1 Solving the Ex-SDN

To ensure that there exists a feasible solution where the products for each order can be combined into a single shipment shipped by a single shipping mode, we assume that the quantity of products for order i cannot exceed the maximum shipping capacity of shipping modes in K_i, i.e., $q_i \leq \max_{s \in K_i}\{c_{si}\}$ for $i \in N$. By the similar argument as that for SDN, it is optimal that the products for each order are shipped out on the production completion day. Based on this, to solve Ex-SDN, we only need to find an optimal production schedule which can be obtained by the following

Algorithm 6.

Algorithm 6 : To solve Ex-SDN

1. *Value function:* Define $f(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1})$ as the minimum objective value of a partial production schedule where the first i orders, $\{1, 2, \ldots, i\}$, have been considered, the total quantity of the products for the first i orders processed in day t is Q_t, and the inventory level at the end of day t is I_t.

2. *Boundary conditions:* $f(0; 0, \ldots, 0; 0, \ldots, 0) = 0$, and $f(0; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1}) = +\infty$ if any $Q_t > 0$ for $t = 1, \ldots, m$ or any $I_t > 0$ for $t = 1, \ldots, m-1$.

3. *Recursive relation:* For $i = 1, \ldots, n$; $Q_t = 0, 1, \ldots, c_P$, for $t = 1, \ldots, m$ such that $\sum_{t \in M} Q_t \leq \sum_{j=1}^{i} q_j$; $I_t = 0, 1, \ldots, \min\left\{\sum_{\tau=1}^{t} Q_\tau, c_I\right\}$, for $t = 1, \ldots, m-1$,

   $f(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1}) =$

   $$\min\Bigg\{ f(i-1; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1}) + r_i, \min\Big\{ f(i-1; Q_1 - x_{i1}, \ldots, Q_m - x_{im}; I'_1, \ldots, I'_{m-1}) + \min_{s \in K_i : c_{si} \geq q_i} \left\{ G_{is}(e_{is}, q_i) + p_i q_i \cdot \max\{t'_i + e_{is} - d_i, 0\} \right\} + \sum_{t=1}^{t'_i - 1} h_i (I_t - I'_t) \Big|$$ (i) for all possible combinations of $(x_{i1}, \ldots, x_{im})$ such that $x_{i1} + \ldots + x_{im} = q_i, x_{it} \leq Q_t$ for $t \in M$,

   $t'_i = \text{argmax}_{t \in M}\{x_{it} > 0\}$, and $\sum_{\tau=1}^{t} x_{i\tau} \leq I_t$ for $t = 1, \ldots, t'_i - 1$;

   (ii) $I_t = I'_t + \sum_{\tau=1}^{t} x_{i\tau}$ for $t = 1, \ldots, t'_i - 1$ and $I_t = I'_t$ for $t = t'_i, \ldots, m-1\Big\}\Bigg\}$.

4. *Value to return:* The optimal production schedule value is determined as

   $$\min\{f(n; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1}) | Q_t = 0, 1, \ldots, c_P, \text{ for } t = 1, \ldots, m \text{ such that } \sum_{t \in M} Q_t \leq \sum_{j=1}^{n} q_j;$$
   $$I_t = 0, 1, \ldots, \min\{\sum_{\tau=1}^{t} Q_\tau, c_I\}, \text{ for } t = 1, \ldots, m-1\}.$$

Theorem 8. *Algorithm 6 finds an optimal solution for Ex-SDN in* $O(n \cdot (c_P)^m \cdot (c_I)^{m-1} (q_{\max})^{m-1} \cdot (m + |K|_{max}))$ *time, where* $|K|_{max} = \max_{i \in N}\{|K_i|\}$ *and* $q_{max} = \max_{i \in N}\{q_i\}$.

Proof. When computing $f(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1})$, the recursive relation in Algorithm 6 takes into account two possible cases of order i: order i is rejected or not. In the case where order i is rejected, it incurs a penalty r_i for rejecting. In the case where order i is accepted, the recursive relation in Algorithm 6 takes into account all possible combinations of $(x_{i1}, \ldots, x_{im})$ that satisfies $x_{i1} + \ldots + x_{im} = q_i$, $x_{it} \leq Q_t$ for $t \in M$, and $\sum_{\tau=1}^{t} x_{i\tau} \leq I_t$ for $t = 1, \ldots, t'_i - 1$. Given a combination of $(x_{i1}, \ldots, x_{im})$, we know that order i must be processed completely on day t'_i. Furthermore, it is optimal that order i is shipped out on day t'_i. Thus, the minimum sum of the shipping cost and delay penalty for order i is $\min_{s \in K_i : c_{si} \geq q_i} \left\{ G_{is}(e_{is}, q_i) + p_i q_i \cdot \max\{t'_i + e_{is} - d_i, 0\} \right\}$ and the total inventory holding cost for order i is $\sum_{t=1}^{t'_i - 1} h_i (I_t - I'_t)$. Given combinations of $(x_{i1}, \ldots, x_{im})$ and $(I_1, \ldots, I_{m-1})$, by a inventory balance equation: $I_t = I'_t + \sum_{\tau=1}^{t'-1} x_{i\tau}$ for $t = 1, \ldots, t'_i - 1$ and $I_t = I'_t$ for $t = t'_i, \ldots, m-1$, one can calculate the value of I'_t for $t = 1, 2, \ldots, m-1$, which implies that we can track the previous feasible state $(i-1; Q_1 - x_{i1}, \ldots, Q_m - x_{im}; I'_1, \ldots, I'_{m-1})$. From the above analysis, the recursive relation in Algorithm 6 computes the value of each $f(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1})$ correctly. This shows the validity of the recursive relation in Algorithm 6.

We have that there are at most $n \cdot (c_P)^m \cdot (c_I)^{m-1}$ possible states of $(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1})$. Given the values of $x_{i1}, x_{i2}, \ldots, x_{i,m-1}$, by the equation: $x_{i1} + x_{i2} + \ldots + x_{im} = q_i$ shown in the recursive relation of Algorithm 6, we can calculate the value of x_{im}. Therefore, together with the fact that $x_{it} \leq q_i$ for $t \in M$, there are at most $(q_{\max})^{m-1}$ possible combinations of $(x_{i1}, \ldots, x_{im})$. Given a combination of $(x_{i1}, \ldots, x_{im})$, it takes $O(|K_i|)$ time to evaluate the value of $\min_{s \in K_i : c_{si} \geq q_i} \left\{ G_{is}(e_{is}, q_i) + p_i q_i \cdot \max\{t'_i + e_{is} - d_i, 0\} \right\}$ and it takes $O(m)$ to evaluate the value of $\sum_{t=1}^{t'_i - 1} h_i (I_t - I'_t)$. Therefore, to compute the value of $f(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1})$, it takes $O((m + |K_i|)(q_{\max})^{m-1})$ time. Thus, the overall time complexity of Algorithm 6 is bounded by $O(n \cdot (c_P)^m \cdot (c_I)^{m-1} \cdot (m + |K_i|)(q_{\max})^{m-1})$. □

For Ex-SDN, we propose a similar heuristic algorithm as that for SDN. We use the same notation and definition given in Section 4.3.2. For order i, S_i consists of two types of feasible subsolutions: The first type contains only one subsolution in which $x^{\pi}_{it} = 0$ for $t \in M$ and f^{π}_i is set as r_i. The subsolution implies that order i is rejected. The second type contains subsolutions which implies

that order i is accepted. Given the vector $(x_{i1}^{\pi}, \ldots, x_{im}^{\pi})$ associated with subsolution π of the second type, we can find day $t_i' = \text{argmax}_{t \in M}\{x_{it} > 0\}$ which represents the last part of order i is processed completely on day t_i'. Thus, f_i^{π} can be calculated as follows:

$$f_i^{\pi} = \min_{s \in K_i : c_{si} \geq q_i} \left\{ G_{is}(e_{is}, q_i) + p_i q_i \cdot \max\{t_i' + e_{is} - d_i, 0\} \right\} + \sum_{t=1}^{t_i'-1} h_i \sum_{\tau=1}^{t} x_{i\tau}^{\pi}.$$

Let Q_{it}^{π} be the inventory level at the end of day $t \in M$ in subsolution $\pi \in S_i$. In subsolution π of the first type, $Q_{it}^{\pi} = 0$ for $t \in M$. In subsolution π of the second type, $Q_{it}^{\pi} = \sum_{\tau=1}^{t} x_{i\tau}^{\pi}$ for $t = 1, \ldots, t_i' - 1$ and $Q_{it}^{\pi} = 0$ for $t = t_i', \ldots, m$. Ex-SDN can be formulated as the following integer program, denoted as [Ex-SDN-IP].

$$\text{[Ex-SDN-IP]} \qquad \min \sum_{i=1}^{n} \sum_{\pi \in S_i} f_i^{\pi} \theta_i^{\pi} \tag{20}$$

$$\text{s.t.} \sum_{\pi \in S_i} \theta_i^{\pi} = 1, \text{ for } i = 1, \ldots, n; \tag{21}$$

$$\sum_{i=1}^{n} \sum_{\pi \in S_i} x_{it}^{\pi} \theta_i^{\pi} \leq c_P, \text{ for } t = 1, \ldots, m; \tag{22}$$

$$\sum_{i=1}^{n} \sum_{\pi \in S_i} Q_{it}^{\pi} \theta_i^{\pi} \leq c_I, \text{ for } t = 1, \ldots, m; \tag{23}$$

$$\theta_i^{\pi} \in \{0, 1\}, \text{ for } i \in N, \pi \in S_i. \tag{24}$$

The objective function (20) minimizes the total cost of the final solution. Constraints (21) ensure that one subsolution is chosen for each order. Constraints (22) ensure that the total quantity of the products processed in each day for all the orders cannot exceed the daily production capacity. Constraints (23) ensure that the inventory level at the end of each day for all the orders cannot exceed the inventory capacity.

Next we only need to find subsolutions of the second type for each order such that these subsolutions have negative reduced costs. Let α_i, β_t, and γ_t be the dual values corresponding to constraints (21), (22), and constraints (23), respectively. The reduced cost $f_i'^{\pi}$ of any subsolution

π of the second type for order i is given by

$$f_i'^{\pi} = f_i^{\pi} - \alpha_i - \sum_{t=1}^{m} x_{it}^{\pi}\beta_t - \sum_{t=1}^{m} Q_{it}^{\pi}\gamma_t.$$

Next we propose the following Algorithm 7 to find a feasible subsolution of the second type for order i such that the subsolution has the minimal value of $f_i'^{\pi}$.

Algorithm 7 : To solve the pricing subproblem for Ex-SDN

1. *Value function:* Define $f(t, Q)$ as the minimum reduced cost of a partial subsolution where the total quantity of the products for order i processed in the first t days is Q.

2. *Boundary conditions:* $f(0, 0) = -\alpha_i$; $f(0, Q) = +\infty$ if $Q > 0$.

3. *Recursive relation:* For $t = 1, \ldots, m$, $Q = 0, 1, \ldots, \min\{t \cdot c_P, q_i\}$,

$$f(t, Q) = \min_{q'=0,1,\ldots,\min(c_P,Q)} \begin{cases} f(t-1, Q-q') + \min_{s \in K_i : c_{si} \geq q_i} \left\{ G_{is}(e_{is}, q_i) + p_i \cdot q_i \cdot \max\{e_{is} + t - d_i, 0\} \right\} \\ -\beta_t q', & \text{if } Q = q_i \text{ and } q' > 0; \\ f(t-1, Q), & \text{if } Q = q_i \text{ and } q' = 0; \\ f(t-1, Q-q') + h_i Q - \beta_t q' - \gamma_t Q, & \text{if } Q < q_i. \end{cases}$$

4. *Value to return:* The optimal solution value is determined as $f(m, q_i)$.

By the similar argument as that for Algorithm 2, we can show the validity of the recursive relation in Algorithm 7.

7.2.2 Solving the Ex-SD

In this section, we first propose an exact algorithm to solve Ex-SD to optimality and then provide a column generation-based heuristic algorithm to find a feasible solution for Ex-SD.

The main idea of the exact algorithm for Ex-SD is as follows: For each value of y_{it} which represents the quantity of products for order i shipped out on day t, the algorithm constructs an auxiliary problem, then solves this auxiliary problem, and finally based on optimal solutions for all auxiliary problems, solves Ex-SD to optimality.

In the following, we first formulate and subsequently solve the auxiliary problem. Given $y_{it} \in \{0, 1, \ldots, q_i\}$, consider an auxiliary problem which requires determining a shipping schedule for the products of customer i shipped out on day t. The objective of the auxiliary problem is to minimize the sum of the shipping cost and the penalty for delay. To obtain a shipping schedule, we need to determine (i) how many times each shipping mode of K_i is used to ship the y_{it} units of products; and (ii) how many products should be shipped by each shipping mode of K_i at one time. We refer to such an auxiliary problem as $AP_{it}(y_{it})$. We can develop an exact algorithm to solve $AP_{it}(y_{it})$ by dynamic program, as shown in Algorithm 8.

Algorithm 8 : To solve $AP_{it}(y_{it})$ with $y_{it} \in \{0, 1, \ldots, q_i\}$, $t \in M$, and $i \in N$

1. *Value function:* Define $f(s, j, Q)$ as the minimum objective value of a partial shipping schedule, where the total quantity of products for order i shipped out on day t by the first s shipping mode of K_i is Q, and the s-th shipping mode of K_i has been used j times to ship these products.

2. *Boundary conditions:* $f(0, 0, 0) = 0$; $f(0, j, Q) = +\infty$ for any $j, Q > 0$.

3. *Recursive relation:* For $s = 1, 2, \ldots, k_i$, $j = 0, 1, \ldots, Q$, $Q = 0, 1, \ldots, y_{it}$,

$$f(s,j,Q) = \min \begin{cases} \min\{f(s-1, j', Q) | j' = 0, 1, \ldots, Q\}, & \text{if } j = 0; \\ \min\{f(s-1, j', Q') + G_{is}(e_{is}, Q - Q') + p_i \cdot (Q - Q') \cdot \max\{t + e_{is} - d_i, 0\} \\ | j' = 0, 1, \ldots, Q'; Q' = Q - c_{si}, \ldots, Q\}, & \text{if } j = 1; \\ \min\{f(s, j-1, Q') + G_{is}(e_{is}, Q - Q') + p_i \cdot (Q - Q') \cdot \max\{t + e_{is} - d_i, 0\} \\ | Q' = Q - c_{si}, \ldots, Q\}, & \text{if } j \geq 2. \end{cases}$$

4. *Value to return:* The optimal shipping schedule value is determined as $\min\{f(k_i, j, y_{it}) | j = 0, 1, \ldots, y_{it}\}$.

Theorem 9. *Given $y_{it} \in \{0, 1, \ldots, q_i\}$, $t \in M$, and $i \in N$, Algorithm 8 with time complexity $O(|K_i|(q_{\max})^4)$ finds an optimal solution to $AP_{it}(y_{it})$, where $q_{\max} = \max_{i \in N}\{q_i\}$.*

Proof. When computing $f(s, j, Q)$, the recursive relation in Algorithm 8 takes into account three possible cases of the s-th shipping mode of K_i: (i) If $j = 0$, we know that the s-th shipping mode of K_i is not used to ship products, which implies that the Q units of products are shipped by some of the first $(s-1)$ shipping modes of K_i. Thus, the contribution of the products shipped by the s-th shipping mode of K_i to the objective value is 0. In this case, the recursive relation needs to take into account each possible value of j'. (ii) If $j = 1$, we know that the s-th shipping mode of K_i is used for the first time to ship the $(Q - Q')$ units of products. Thus, the contribution of the products shipped by the s-th shipping mode of K_i to the objective value is $G_{is}(e_{is}, Q - Q') + p_i \cdot (Q - Q') \cdot \max\{t + e_{is} - d_i, 0\}$. In this case, the recursive relation needs to take into account all possible values of j' and Q' for the $(s-1)$-th shipping mode of K_i. (iii) If $j \geq 2$, we know that the s-th shipping mode of K_i is used for the j-th time to ship the $(Q - Q')$ units of products. Thus, the contribution of the products shipped by the s-th shipping mode of K_i for the j-th time to the objective value is $G_{is}(e_{is}, Q - Q') + p_i \cdot (Q - Q') \cdot \max\{t + e_{is} - d_i, 0\}$. As a result, Algorithm 8 returns the minimum sum of the shipping cost and the penalty for delay over all feasible shipping schedules to $AP(y_{it})$. This shows the validity of the recursive relation in Algorithm 8.

Given y_{it}, there are at most $|K_i|(q_{\max})^2$ possible states of (s, j, Q). Given a state (s, j, Q), the recursive relation considers all possible values of j' and Q' and the total number of values of j' and Q' is at most $(q_{\max})^2$. Thus, given the state (s, j, Q), it takes $O((q_{\max})^2)$ time to calculate $f(s, j, Q)$. Therefore, the overall time complexity of Algorithm 8 is bounded by $O(|K_i|(q_{\max})^4)$. □

Based on Algorithm 8, to solve Ex-SD, we only need to find an optimal production schedule which can be obtained by the following Algorithm 9.

Algorithm 9 : To solve Ex-SD

1. For each order $i \in N$, enumerate all the possible combinations of $(y_{i1}, \ldots, y_{im})$, such that $y_{i1} + \ldots + y_{im} = q_i$ and $y_{it} \in \{0, 1, \ldots, q_i\}$ for $t \in M$.

2. For each possible value of y_{it}, let $Z(y_{it})$ be the objective value of the optimal shipping schedule generated by Algorithm 8 for $AP_{it}(y_{it})$. For each combination $(y_{i1}, \ldots, y_{im})$, calculate its sum of the total shipping cost and the total penalty for delay, $g_i(y_{i1}, \ldots, y_{im}) = \sum_{t=1}^{m} Z(y_{it})$.

3. Generate an optimal production schedule for Ex-SD using the following dynamic program.

- *Value function:* Define $f(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1})$ as the minimum objective value of a partial production schedule where the first i orders, $\{1, 2, \ldots, i\}$, have been considered, the total quantity of the products for the first i orders processed in day t is Q_t, and the inventory level at the end of day t is I_t.
- *Boundary conditions:* $f(0; 0, \ldots, 0; 0, \ldots, 0) = 0$ and $f(0; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1}) = +\infty$ if any $Q_t > 0$ for $t = 1, \ldots, m$ or any $I_t > 0$ for $t = 1, \ldots, m-1$.
- *Recursive relation:* For $i = 1, \ldots, n$; $Q_t = 0, 1, \ldots, c_P$, for $t = 1, \ldots, m$ such that $\sum_{t \in M} Q_t \leq \sum_{j=1}^{i} q_j$; $I_t = 0, 1, \ldots, \min\{\sum_{\tau=1}^{t} Q_\tau, c_I\}$, for $t = 1, \ldots, m-1$,

  $f(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1}) =$

  $\min \Bigg\{ f(i-1; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1}) + r_i, \min \Big\{ f(i-1; Q_1 - x_{i1}, \ldots, Q_m - x_{im}; I'_1, \ldots, I'_{m-1})$

  $+ g_i(y_{i1}, \ldots, y_{im}) + \sum_{t=1}^{m} h_i(I_t - I'_t) \Big|$ (i) for all possible combinations of $(x_{i1}, \ldots, x_{im})$

  such that $x_{i1} + \ldots + x_{im} = q_i$ and $x_{it} \leq Q_t$ for $t \in M$; (ii) for all possible combinations of

  $(y_{i1}, \ldots, y_{im})$ such that $y_{i1} + \ldots + y_{im} = q_i$ and $0 \leq \sum_{\tau=1}^{t} (x_{i\tau} - y_{i\tau}) \leq I_t$ for $t \in M$;

  (iii) $I_t = I'_t + \sum_{\tau=1}^{t} (x_{i\tau} - y_{i\tau})$ for $t = 1, \ldots, m-1 \Big\} \Bigg\}$.

- *Value to return:* The optimal production schedule value is determined as

  $$\min\{f(n; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1}) | Q_t = 0, 1, \ldots, c_P, \text{ for } t \in M \text{ such that } \sum_{t \in M} Q_t \leq \sum_{j=1}^{n} q_j;$$
  $$I_t = 0, 1, \ldots, \min\{\sum_{\tau=1}^{t} Q_\tau, c_I\}, \text{ for } t = 1, \ldots, m-1\}.$$

Theorem 10. *Algorithm 9 finds an optimal solution for Ex-SD in* $O\big(n \cdot (q_{\max})^{m-1}(|K|_{max}(q_{\max})^4 + (c_P)^m \cdot (c_I)^{m-1} \cdot (q_{\max})^{m-1} m)\big)$ *time, where* $q_{max} = \max_{i=1,\ldots,n}\{q_i\}$ *and* $|K|_{max} = \max_{i \in N}\{|K_i|\}$.

Proof. When computing $f(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1})$, the recursive relation in Algorithm 9 takes into account two possible cases of order i: order i is rejected or not. In the case where order i is rejected, it incurs a penalty r_i for rejecting. In the case where order i is accepted, from Theorem 9, we know that given a combination of $(y_{i1}, \ldots, y_{im})$, one can calculate the value of $g_i(y_{i1}, \ldots, y_{im})$. For every feasible state $(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1})$, the recursive relation in Algorithm 9 takes into account all possible combinations of $(y_{i1}, \ldots, y_{im})$ and $(x_{i1}, \ldots, x_{im})$. Given a combination of $(y_{i1}, \ldots, y_{im})$, the sum of the total shipping cost and the total penalty for delay is $g_i(y_{i1}, \ldots, y_{im})$. Given a combination of $(I_1, \ldots, I_{m-1})$, it can be seen that the total inventory cost contributed by products for order i to the objective value is $\sum_{t=1}^{m} h_i(I_t - I'_t)$. Given combinations of $(x_{i1}, \ldots, x_{im})$, $(y_{i1}, \ldots, y_{im})$, and $(I_1, \ldots, I_{m-1})$, by a inventory balance equation: $I_t = I'_t + \sum_{\tau=1}^{t}(x_{i\tau} - y_{i\tau})$ shown in the recursive relation of Algorithm 9, one can calculate the value of I'_t for $t = 1, 2, \ldots, m-1$, which implies that we can track the previous feasible state $(i-1; Q_1 - x_{i1}, \ldots, Q_m - x_{im}; I'_1, \ldots, I'_{m-1})$. From the above analysis, the recursive relation in Algorithm 9 computes the value of each $f(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1})$ correctly. This shows the validity of the recursive relation in Algorithm 9.

Given the values of $y_{i1}, y_{i2}, \ldots, y_{i,m-1}$, by the equation: $y_{i1} + y_{i2} + \ldots + y_{im} = q_i$ shown in Step 1, we can calculate the value of y_{im}. Therefore, together with the fact that $y_{it} \leq q_i$ for $t \in M$, there are at most $(q_{\max})^{m-1}$ possible combinations of $(y_{i1}, \ldots, y_{im})$ for each $i \in N$. Note that from Theorem 9, we have that the time complexity of Algorithm 8 is bounded by $O(|K_i|(q_{\max})^4)$. Thus, the time complexity of Step 2 is bounded by $O(n|K_i|(q_{\max})^{m+3})$. There are at most $(c_P)^m$ possible combinations of $(Q_1, Q-2, \ldots, Q_m)$. By the inequation: $I_t \leq c_I$ for $t = 1, 2, \ldots, m-1$, we have that there are at most $n \cdot (c_P)^m \cdot (c_I)^{m-1}$ possible states of $(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1})$. To compute the value of $f(i; Q_1, \ldots, Q_m; I_1, \ldots, I_{m-1})$, the recursive relation needs to consider all possible combinations of $(y_{i1}, \ldots, y_{im})$ and $(x_{i1}, \ldots, x_{im})$. By the similar argument to that for computing the time complexity of Step 2, the number of all possible combinations of $(x_{i1}, \ldots, x_{im}; y_{i1}, \ldots, y_{im})$ is at most $(q_{max})^{2(m-1)}$. Thus, the time complexity of Step 3 is bounded by $O(n \cdot (c_P)^m \cdot (c_I)^{m-1} \cdot (q_{\max})^{2m-2}m)$. Therefore, the overall time complexity of Algorithm 9 is bounded by $O\big(n \cdot (q_{\max})^{m-1}(|K_i|(q_{\max})^4 + (c_P)^m \cdot (c_I)^{m-1} \cdot (q_{\max})^{m-1}m)\big)$. □

For Ex-SD, we propose the same heuristic algorithm as that for Ex-SDN except for the following aspects. Let a vector $(y^{\pi}_{i1}, \ldots, y^{\pi}_{im})$ be the shipping schedule in subsolution $\pi \in S_i$. Given $(x^{\pi}_{i1}, \ldots, x^{\pi}_{im})$, to find an optimal shipping schedule of subsolution π, we provide an exact algorithm that is the same as Algorithm 3 except that the recursive relation of Algorithm 3 is replaced by the following:

$$f(t,Q) = \min\Big\{f(t-1, Q-q') + Z_{it}(q') + h_i(\sum_{\tau=1}^{t} x_{i\tau} - Q) | q' = \max(0, Q - \sum_{\tau=1}^{t-1} x_{i\tau}), \ldots, Q\Big\},$$

where $Z_{it}(q')$ is the objective value of the optimal solution generated by Algorithm 8 for $AP_{it}(q')$. Therefore, given $(x^{\pi}_{i1}, \ldots, x^{\pi}_{im})$, the optimal shipping schedule of subsolution π can be determined accordingly. Therefore, we also assume that each subsolution $\pi \in S_i$ specifies the shipping schedule for order i. The objective value of subsolution π of the second type can be calculated as follows:

$$f^{\pi}_i = \sum_{t=1}^{m} \min_{s \in K_i} \big\{G_{is}(e_{is}, y^{\pi}_{it}) + p_i \cdot y^{\pi}_{it} \cdot \max\{e_{is} + t - d_i, 0\}\big\}.$$

Let $Q^{\pi}_{it} = \sum_{\tau=1}^{t}(x^{\pi}_{i\tau} - y^{\pi}_{i\tau})$ for $t \in M$. We use the same integer program as Ex-SDN-IP given in Section 7.2.1 to formulate Ex-SD. Next we use a different algorithm (i.e., Algorithm 10) to find the subsolution of the second type for order i such that the subsolution has the minimal value of $f_i'^{\pi}$.

Algorithm 10 : To solve the pricing subproblem for Ex-SD

1. *Value function:* Define $f(t, Q, Y)$ as the minimum reduced cost of a partial subsolution, where the total quantities of the products for order i processed and shipped in the first t days are Q and Y, respectively.

2. *Boundary conditions:* $f(0,0,0) = -\alpha_i$; $f(0, Q, Y) = +\infty$ for any $Q > 0$ and $Y > 0$.

3. *Recursive relation:* For $t = 1, \ldots, m$; $Q = 0, 1, \ldots, \min(t \cdot c_P, q_i)$; $Y = 0, 1, \ldots, Q$.

$$f(t,Q,Y) = \min\Big\{f(t-1, Q-q', Y-y') + Z_{it}(y') + h_i(Q-Y) - \beta_t q' - \gamma_t(Q-Y),$$
$$|q' = 0, 1, \ldots, \min(c_P, Q), y' = \max\{0, Y - Q + q'\}, \ldots, Y\Big\},$$

where $Z_{it}(y')$ is the objective value of the optimal solution generated by Algorithm 8 for $AP_{it}(q')$.

4. *Value to return:* The optimal solution value is determined as $f(m, q_i, q_i)$.

By the similar argument as that for Algorithm 5, we can show the validity of the recursive relation in Algorithm 10.

8 Conclusions

This study examines an integrated production and transportation scheduling problem that make-to-order manufacturers encounter. This problem involves multiple shipping modes and nonlinear shipping cost functions. In this study, we take into account order-dependent inventory holding costs, which not only depend on the time that the products spend in temporary storage but also depend on customer types. We consider two cases: the cases with and without split delivery. We examine the computational complexity of each case, and prove that it has no polynomial-time algorithm with a constant worst-case ratio. We also propose an exact algorithm to solve each case and design a heuristic algorithm based on column generation for each case. The computational results show that the heuristic algorithm can efficiently generate near-optimal solutions for each case in a very short running time for all the instances generated randomly. Finally, we show that all the results for each case can be extended to the corresponding problem without idle times allowed. In addition, we also discuss an extension with production capacity, inventory capacity, shipping capacity, and order acceptance.

There are possible directions for future research. First, this study takes into account one order for each customer. The model with this assumption is applicable to the situations described in Section 1. In some other situations, a customer may place multiple orders within the planning horizon. For these situations, it would be worthwhile to develop efficient solution methods for the case where each customer has multiple orders with different committed delivery due dates. Second, in many applications, orders arrive dynamically over time. The order information such as quantity

and due date, cannot be known in advance until the orders are placed by customers. Hence, it would be useful to extend our analysis to the case with considering dynamically arriving orders.

Acknowledgements

The authors thank Editor-in-Chief (Prof. Francisco Saldanha da Gama), Area Editor (Prof. Mikhail Kovalyov), and the anonymous reviewers for their helpful comments and suggestions.

References

Ahmadi, R., Bagchi, U., Roemer, T. A., 2005. Coordinated scheduling of customer orders for quick response. Naval Research Logistics (NRL) 52 (6), 493–512.

Armstrong, R., Gao, S., Lei, L., 2008. A zero-inventory production and distribution problem with a fixed customer sequence. Annals of Operations Research 159 (1), 395–414.

Atamtürk, A., Küçükyavuz, S., 2008. An o (n2) algorithm for lot sizing with inventory bounds and fixed costs. Operations Research Letters 36 (3), 297–299.

Bachtenkirch, D., Bock, S., 2021. Finding efficient make-to-order production and batch delivery schedules. European Journal of Operational Research.

Chen, Z.-L., 2010. Integrated production and outbound distribution scheduling: review and extensions. Operations Research 58 (1), 130–148.

Chen, Z.-L., Pundoor, G., 2009. Integrated order scheduling and packing. Production and Operations Management 18 (6), 672–692.

Chen, Z.-L., Vairaktarakis, G. L., 2005. Integrated scheduling of production and distribution operations. Management Science 51 (4), 614–628.

Cheref, A., Agnetis, A., Artigues, C., Billaut, J. C., 2017. Complexity results for an integrated single machine scheduling and outbound delivery problem with fixed sequence. Journal of Scheduling 20 (1), 1–13.

Chevroton, H., Kergosien, Y., Berghman, L., Billaut, J.-C., 2021. Solving an integrated scheduling and routing problem with inventory, routing and penalty costs. European Journal of Operational Research.

Desaulniers, G., Desrosiers, J., Solomon, M. M., 2006. Column generation. Vol. 5. Springer Science & Business Media.

Devapriya, P., Ferrell, W., Geismar, N., 2017. Integrated production and distribution scheduling with a perishable product. European Journal of Operational Research 259 (3), 906–916.

Gary, M. R., Johnson, D. S., 1979. Computers and Intractability: A Guide to the Theory of NP-completeness. WH Freeman and Company, New York.

Geismar, H. N., Dawande, M., Sriskandarajah, C., 2011. Pool-point distribution of zero-inventory products. Production and Operations Management 20 (5), 737–753.

Hall, N. G., Potts, C. N., 2003. Supply chain scheduling: Batching and delivery. Operations Research 51 (4), 566–584.

Lee, C.-Y., Chen, Z.-L., 2001. Machine scheduling with transportation considerations. Journal of Scheduling 4 (1), 3–24.

Leung, J. Y., Li, H., Pinedo, M., 2005a. Order scheduling models: an overview. In: Multidisciplinary scheduling: theory and applications. Springer, pp. 37–53.

Leung, J. Y.-T., Chen, Z.-L., 2013. Integrated production and distribution with fixed delivery departure dates. Operations Research Letters 41 (3), 290–293.

Leung, J. Y.-T., Li, H., Pinedo, M., 2005b. Order scheduling in an environment with dedicated resources in parallel. Journal of Scheduling 8 (5), 355–386.

Li, C.-L., Li, F., 2020. Rescheduling production and outbound deliveries when transportation service is disrupted. European Journal of Operational Research 286 (1), 138–148.

Li, F., Chen, Z.-L., Tang, L., 2017. Integrated production, inventory and delivery problems: Complexity and algorithms. INFORMS Journal on Computing 29 (2), 232–250.

Li, F., Xu, Z., Chen, Z.-L., 2020. Production and transportation integration for commit-to-delivery mode with general shipping costs. INFORMS Journal on Computing 32, 1012–1029.

Liu, L., Liu, S., 2020. Integrated production and distribution problem of perishable products with a minimum total order weighted delivery time. Mathematics 8 (2), 146.

Lübbecke, M. E., 2010. Column generation. Wiley encyclopedia of operations research and management science.

Lübbecke, M. E., Desrosiers, J., 2005. Selected topics in column generation. Operations research 53 (6), 1007–1023.

Melo, R. A., Wolsey, L. A., 2010. Optimizing production and transportation in a commit-to-delivery business mode. European Journal of Operational Research 203 (3), 614–618.

Mensendiek, A., Gupta, J. N., Herrmann, J., 2015. Scheduling identical parallel machines with fixed delivery dates to minimize total tardiness. European Journal of Operational Research 243 (2), 514–522.

Stecke, K. E., Zhao, X., 2007. Production and transportation integration for a make-to-order manufacturing company with a commit-to-delivery business mode. Manufacturing & Service Operations Management 9 (2), 206–224.

Tang, L., Li, F., Chen, Z.-L., 2019. Integrated scheduling of production and two-stage delivery of make-to-order products: Offline and online algorithms. INFORMS Journal on Computing 31 (3), 493–514.

Ullrich, C. A., 2013. Integrated machine scheduling and vehicle routing with time windows. European Journal of Operational Research 227 (1), 152–165.

Zhong, W., 2015. An improved algorithm for integrated production and distribution scheduling problem with committed delivery dates. Optimization Letters 9 (3), 537–567.

Zhong, W., Chen, Z.-L., Chen, M., 2010. Integrated production and distribution scheduling with committed delivery dates. Operations Research Letters 38 (2), 133–138.