Abstract:

We solve a delivery problem arising in e-commerce logistics. We consider a retailer with an online store and a network of stores operating in an omni-channel strategy. The fulfillment decision for an online order, which contains a number of items, involves the allocation of these items to the stores where they are available and the selection of one store for consolidation of the items into the final package to be dispatched to the customer. The transportation between the stores and the customer is handled by a third-party logistic provider which uses a concave pricing policy based on the distance between the origin and the destination, as well as on the weight of the items. We present an online problem which is defined for a set of orders placed over time, and a mixed integer programming formulation defined for each order. The main characteristics of this problem are that the solution of the formulation for each order impacts those of the subsequent orders, and the problem must be solved in real time. For the solution of the formulation, we propose an iterative matheuristic based on the solution of the set covering model and local search. Computational results on randomly generated instances are provided, which demonstrate that our algorithm is capable of producing high-quality results.
Dear Madam / Sir,

I hereby submit the manuscript entitled "Minimum Cost Delivery of Multi-item Orders in E-Commerce Logistics", co-authored by M. Hakan Akyuz, Ibrahim Muter, myself, and Gilbert Laporte to your consideration for publication in Computers & Operations Research. Your handling of the manuscript will be much appreciated.

Kind regards,

Güneş Erdoğan

11/01/2021
Highlights

We present a new model for an omni-channel e-commerce logistics problem. The model selects a consolidation point where final delivery is made for multi-item orders. A third party logistics company using a concave pricing policy performs the transportation. The problem is solved by means of an iterative metaheuristic. Results on realistic instances show the effectiveness of the methodology.
Minimum Cost Delivery of Multi-item Orders in E-Commerce Logistics

M. Hakan Akyüz, İbrahim Muter, Güneş Erdoğan, Gilbert Laporte

Abstract

We solve a delivery problem arising in e-commerce logistics. We consider a retailer with an online store and a network of stores operating in an omni-channel strategy. The fulfillment decision for an online order, which contains a number of items, involves the allocation of these items to the stores where they are available and the selection of one store for consolidation of the items into the final package to be dispatched to the customer. The transportation between the stores and the customer is handled by a third-party logistic provider which uses a concave pricing policy based on the distance between the origin and the destination, as well as on the weight of the items. We present an online problem which is defined for a set of orders placed over time, and a mixed integer programming formulation defined for each order. The main characteristics of this problem are that the solution of the formulation for each order impacts those of the subsequent orders, and the problem must be solved in real time. For the solution of the formulation, we propose an iterative matheuristic based on the solution of the set covering model and local search. Computational results on randomly generated instances are provided, which demonstrate that our algorithm is capable of producing high-quality results.

Keywords: Logistics, E-commerce, Multi-commodity flow, Order delivery

1. Introduction

E-commerce sales have shown an unbridled growth in the last decade. In the United States, the online retail industry has expanded from $144.9B in 2009 to $599.5B in 2019, which corresponds to an annual growth of approximately 15% (Statista, 2020). As a response, more and more retail organizations offer online stores to take advantage of this trend and integrate their online operations with their existing brick-and-mortar store network. However, online sales
come at the price of coordinating expensive logistics activities in order to provide a seamless delivery experience to customers. Products are often offered by free shipment, which implies that transportation costs are met by the retailer. Indeed, Lei et al. (2018) state that online retailers such as Amazon work with a thin margin which promotes cutting expenses connected to logistics operations. Therefore, retailers must find a lower cost fulfillment plan for every single order.

Traditional retailing forged a distribution network whose final level is the distribution of products to the locations of retailers. The planning of such distribution networks mainly involves inventory decisions and periodical distribution services to retail points. In addition to the functional distribution needed to sustain the retail activities, an online retailer faces important decisions prompted by orders placed online, which impact all levels of the supply chain. Xu et al. (2009) have listed the salient characteristics of e-commerce, one of which is the retailer-directed demand allocation, that is, the decision of which warehouse should fulfill a customer order still needs to be made by the company. Moreover, since all of the items in a customer order may not be available at a single warehouse, either the order is shipped to the customer in multiple deliveries, or these items must be consolidated at a single location before being delivered in a single parcel.

In this paper, we focus on the order consolidation and delivery decisions for a fashion retailer running an online store as well as brick-and-mortar stores that serve both walk-in and online customers. Online stores generally offer an extensive list of items that are available in at least one of the stores. An online order can contain multiple items available in various stores, which often leads to the shipment of multiple packages (Torabi et al., 2015; Zhang et al., 2019a; Ishfaq and Bajwa, 2019). Unlike what is done in these studies, we impose that all items of an online order be consolidated at one of the stores, which we refer to as the consolidation point, so that a single package is shipped to the customer. This is motivated by the company policy which promises a single package delivery to its customers for every online order. This renders lateral transshipments among the retailer’s stores a necessity rather than an option.

The contribution of this work is fourfold. First, we introduce and model an e-commerce logistics problem which must be solved dynamically whenever an order is placed online. We model this problem as a non-linear integer program with a piecewise linear cost function, which is an innovative feature of our work. Second, we analyze some theoretical properties of the problem. In particular, we show that even a special case of this problem with a linear objective function is NP-hard. We then introduce valid inequalities for the model that potentially tighten its linear programming (LP) relaxation. Third, we develop efficient heuristics that can solve the
problem rapidly, which is important since this is an online problem. To this end, we examine a restriction and a relaxation of the mathematical model to achieve upper and lower bounds. A restricted version of our formulation that simply enforces direct delivery of ordered items to the consolidation point proves most effective and reasonable from an operational perspective. We split the decisions in this restricted problem to solve it quickly and propose an iterative heuristic that involves i) the solution of a set covering problem that selects the stores used to fulfill an order, ii) an enumerative procedure that allocates the items of an order to the stores and selects the consolidation point, and iii) a local search procedure to improve the solution. In order to iterate these steps, we incorporate a set of constraints to the set covering problem that eliminate the previously found solutions. Fourth, we test the models and the algorithms on randomly generated instances and show that we can reach high-quality solutions within a few seconds.

The remainder of this study is organized as follows. Section 2 briefly reviews the relevant literature and positions our work. Section 3 presents the problem definition, its mathematical formulation and some theoretical properties. Section 4 provides the details of the suggested algorithms. Section 5 introduces the generation of the test bed and presents our computational results, as well as managerial insights. Finally, we conclude our paper in Section 6.

2. Literature review

In this section, we provide an overview of the relevant literature. Lateral transshipment has been practiced by companies to balance the inventory levels at the warehouses or at the stores (Mercer and Tao, 1996; Paterson et al., 2011; Coelho et al., 2012). We focus on the application of lateral transshipment solely for the consolidation of the items associated with an online order at a store where it is packaged for final delivery. As in other studies that tackle real-time allocations, we assume that the inventory level at each store is available once the items have been identified for an online customer. Furthermore, we consider an online decision making problem for each individual order where from the moment a customer places an online order, it is allocated to the stores in real time. Demand fulfillment decisions have to be immediate in our problem since both walk-in customers and online customers are treated alike and simultaneously in the brick-and-mortar stores of the retailer. A brick-and-mortar store consists of a backroom area where products are received, stored and shipped, and of a walk-in customer showroom area where products are displayed and checkout is done (Mou et al., 2018). Walk-in customers can directly purchase display products from the shelves of the showroom area. However, the backroom is shared with the online customers. Therefore, each
item of an online customer order needs to be reserved immediately in the backroom area by a member of staff who stows items in a designated space for shipment. Otherwise, when an item in a given online order is not promptly reserved at the allocated store, a walk-in customer could purchase the same item. This could result in a stock-out situation and probably lead to delays in online customer order deliveries or even to cancellation of orders.

The last-mile distribution of the orders to the customers can be performed by the fleet of the company or by a third-party logistics (3PL) provider. In this study, the distribution function is delegated to a 3PL, which not only carries the items in lateral transshipment, but also delivers the packages from the selected consolidation points to customers, namely the final delivery of the single package. Marasco (2008) presents a comprehensive survey of the literature on 3PLs, and Murphy and Poist (1998) show that the size of a company is an important indicator of whether it uses a 3PL service. The stores are visited by the 3PL at prespecified times during the day. Unlike what is commonly reported in the relevant literature, the lead time for lateral transshipment among the brick-and-mortar stores is not negligible. The expected delivery time to a customer must lie within a certain limit dictated by the retailer company and by the 3PL standards. This has to be determined at the time an online order is placed by the customer. The decision on which store to fulfill the demand and where to consolidate the items must be made within a few seconds. To sum up, our decision problem aims to find a delivery plan minimizing the transshipment and package delivery costs, which are functions of the weight of the package and of the distance between the stores, so that the package containing all the ordered items is delivered to the customer by the promised time.

The omni-channel distribution strategy integrates the operations and logistics across all sales channels (warehouses, brick-and-mortar stores, and online stores). Therefore, all customer orders are treated by the same logistics function, e.g., online orders can also be fulfilled from the stores (Agatz et al., 2008; Beck and Rygl, 2015; Ailawadi and Farris, 2017). There are many fulfillment and distribution alternatives for an online order in this distribution structure, which are explained in Hübner et al. (2016). Ishfaq and Raja (2018) suggest that using retailer stores may be more beneficial than using direct warehouse shipments for order fulfillment. Zhang et al. (2019a) empirically show that cost savings can be obtained by order consolidation instead of order splitting. Wei et al. (2020) conclude that shipment consolidation is profitable for omni-channel retailers. In our study, we adopt a delivery strategy in which the online orders are fulfilled from brick-and-mortar stores similar to in-store orders by walk-in customers.

Usually, retailers locate brick-and-mortar stores at population centres in closer proximity to customer locations than to distribution centres, which are relatively few and large in size.
Moreover, the variety of stock-keeping units available in the e-commerce sites, referred to as items, usually makes it prohibitive to allocate a distribution center for online sales. A good example is the fashion retailing sector, which is characterized by a very broad spectrum of items. Hence, the adoption of online order fulfillment through retail stores shortens delivery times and enables retailers to fulfill a broader range of orders. This requires an online retailer to make the order-fulfillment center assignment decision in real time when a customer places an order. The real-time order allocation to the fulfillment points is dependent on the full availability of inventory levels at each fulfillment center.

Several aspects of online order fulfillment problem have been investigated. These are listed and summarized in Table 1. Our work is original and fills a literature gap by uniquely studying an online fashion retailer that cooperates with a 3PL to provide home delivery. It uses a piecewise linear cost structure in the objective function considering delivery time constraints of customers and allowing lateral transshipments without splitting the final delivery package.

In the following, we present the relevant literature in more detail. We categorize the studies based on their vehicle fleet, which may be their own fleet, 3PL, or both in conjunction. Although our work does not consider uncertainty, for the sake of completeness, we also refer to several studies that incorporate stochasticity. Relevant papers include those of Mahar and Wright (2009); Mahar et al. (2012); Jasin and Sinha (2015); Lei et al. (2018); Arslan et al. (2020). Notably, only the studies that use both types of fleets in conjunction allow lateral transshipment.

2.1. Own Fleet

Li and Jia (2019) consider an offline order fulfillment and delivery problem for an online retailer that uses an own fleet system for deliveries in a two-echelon supply network, which they solve by Benders decomposition. Li et al. (2019) simultaneously consider order allocation and order routing as part of the order fulfillment problem for online retailers and present an adaptive large neighborhood search algorithm combined with a greedy heuristic for the offline problem. Jiang and Li (2020) integrate order fulfillment and vehicle routing decisions for online retailers subject to delivery time windows and synchronization constraints. The problem is solved by means of an adaptive large neighborhood search heuristic.

2.2. 3PL

By capitalizing on the delay between the order placement and inventory deployment to fulfill it, Xu et al. (2009) proposed a mathematical model to reduce the number of multiple shipments to a customer by re-allocating the not-yet-picked orders to the warehouses. This
problem is referred to as the order, shipment or package consolidation (Xu et al., 2009; Catalán and Fisher, 2012; Acimovic and Graves, 2015; Zhang et al., 2018, 2019b). Jasin and Sinha (2015) consider a stochastic order fulfillment problem with demand uncertainty to decide on the optimal policy minimizing a quasi-linear function of the transportation costs. They provide asymptotic competitive ratios of lower and upper bounds on the problem and develop two heuristics based on the linear programming relaxation of the suggested mathematical model. Arslan et al. (2020) solve a stochastic order fulfillment problem in two-stages where demand and store capacities are uncertain and inventory holding costs are handled considering replenishment decisions. A Benders decomposition algorithm is employed to solve the problem by a sample average approximation method.

2.3. Both Own Fleet and 3PL

Torabi et al. (2015) model the order fulfillment problem as a retailer demand allocation and delivery mode selection problem, allowing lateral transshipments among fulfillment centers and utilizing periodically accumulated orders to produce an offline delivery plan. This work does not explicitly consider delivery time restrictions and allows splitting the deliveries into two separate shipments. Ardjmand et al. (2018) consider simultaneous order cartonization and order delivery from fulfillment centers of an online retailer. The problem is formulated as a multi-objective mixed integer program which is solved by a genetic algorithm considering the trade-off between transportation cost and delivery time as objectives. Onal et al. (2018) assume that customer orders continuously arrive at online retailers and order picking immediately starts at warehouses. Their assumption is in line with ours in the sense that orders are immediately reserved when the fulfillment process assigns the corresponding items to a store.

Janjevic et al. (2019) describe a construction heuristic to strategically solve a two-echelon logistics network design problem with a nonlinear objective function for order fulfillment with delivery at pickup points. Their study focuses on deciding at which locations of the network to open a consolidation center and pickup points. Zhang et al. (2019a) focus on an offline order fulfillment problem with lateral transshipments where order items from customers can be consolidated at warehouses before the shipment. The authors describe a logic-based Benders decomposition method to solve the problem exactly. A shipment can consist of multiple packages if some items are incompatible with each other and cannot be placed in the same package. Ishfaq and Bajwa (2019) solve a nonlinear mixed integer programming formulation via an outer approximation algorithm to maximize the profit in an omni-channel order fulfillment problem where no delivery time restrictions exist and order splitting is possible. Additional inventory balance constraints and price-dependent demand over the planning horizon are considered for
the offline problem. Wei et al. (2020) characterize the optimal policy for fulfillment consolidation in the single- and two-warehouse systems with fixed shipment costs. They consider three types of online orders with delivery time restrictions and solve the problem by dynamic programming.
<table>
<thead>
<tr>
<th>Article</th>
<th>Delivery type</th>
<th>transportation</th>
<th>Lateral transshipment</th>
<th>Problem type</th>
<th>Order splitting</th>
<th>Delivery time</th>
<th>Formulation</th>
<th>Methodology</th>
<th>Additional constraints</th>
<th>Objective function / Cost structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mahar and Wright (2009)</td>
<td>HD</td>
<td>Both</td>
<td>No</td>
<td>Offline</td>
<td>No</td>
<td>Not explicit</td>
<td>MILP</td>
<td>Simulation</td>
<td>Improvement heuristic, local search</td>
<td>Linear / Minimize expected total cost</td>
</tr>
<tr>
<td>Xu et al. (2009)</td>
<td>HD</td>
<td>3PL</td>
<td>No</td>
<td>Both</td>
<td>Yes</td>
<td>Yes</td>
<td>MILP</td>
<td>Simulation</td>
<td>-</td>
<td>Linear / Minimize number of shipments</td>
</tr>
<tr>
<td>Mahar et al. (2012)</td>
<td>PU</td>
<td>-</td>
<td>No</td>
<td>Offline</td>
<td>No</td>
<td>Not explicit</td>
<td>MILP</td>
<td>Simulation</td>
<td>-</td>
<td>Linear / Minimize expected inventory holding and lost sales costs</td>
</tr>
<tr>
<td>Jasin and Sinha (2015)</td>
<td>-</td>
<td>3PL</td>
<td>No</td>
<td>Both</td>
<td>Yes</td>
<td>Yes</td>
<td>MILP</td>
<td>Linear programming approximation</td>
<td>Inventory constraints</td>
<td>Quasi-linear / Minimize expected transportation cost</td>
</tr>
<tr>
<td>Tavakkoli et al. (2015)</td>
<td>-</td>
<td>Both</td>
<td>Yes</td>
<td>Offline</td>
<td>Yes</td>
<td>Not explicit</td>
<td>MILP</td>
<td>Benders decomposition</td>
<td>-</td>
<td>Linear / Minimize transportation cost</td>
</tr>
<tr>
<td>Ardjmand et al. (2018)</td>
<td>HD</td>
<td>Both</td>
<td>No</td>
<td>Offline</td>
<td>Yes</td>
<td>Yes</td>
<td>MILP</td>
<td>Genetic algorithm</td>
<td>Packaging</td>
<td>Linear / Multi-objective, minimize transportation cost and delivery time</td>
</tr>
<tr>
<td>Lei et al. (2018)</td>
<td>-</td>
<td>Both</td>
<td>Yes</td>
<td>Offline</td>
<td>Yes</td>
<td>No</td>
<td>MILP</td>
<td>Linear programming approximation</td>
<td>Inventory balance, price dependent demand</td>
<td>Nonlinear / Minimize transportation cost and operational costs</td>
</tr>
<tr>
<td>Ihsan and Bajwa (2019)</td>
<td>-</td>
<td>Both</td>
<td>Yes</td>
<td>Offline</td>
<td>No</td>
<td>No</td>
<td>MINLP</td>
<td>Outer approximation</td>
<td>-</td>
<td>Nonlinear / Minimize transportation cost and operational costs</td>
</tr>
<tr>
<td>Auşjerić et al. (2019)</td>
<td>PU</td>
<td>Both</td>
<td>No</td>
<td>Offline</td>
<td>-</td>
<td>Yes</td>
<td>MINLP</td>
<td>Construction heuristic</td>
<td>-</td>
<td>Nonlinear / Minimize transportation cost and operational costs</td>
</tr>
<tr>
<td>Li and Jia (2019)</td>
<td>PU</td>
<td>OF</td>
<td>No</td>
<td>Offline</td>
<td>No</td>
<td>Yes</td>
<td>MILP</td>
<td>Benders decomposition</td>
<td>Vehicle routing</td>
<td>Linear / Minimize transportation cost</td>
</tr>
<tr>
<td>Li et al. (2019)</td>
<td>HD</td>
<td>OF</td>
<td>No</td>
<td>Offline</td>
<td>No</td>
<td>Yes</td>
<td>MILP</td>
<td>Greedy heuristic, ALNS algorithm</td>
<td>Vehicle routing</td>
<td>Linear / Minimize transportation and holding cost</td>
</tr>
<tr>
<td>Zhang et al. (2019a)</td>
<td>-</td>
<td>Both</td>
<td>Yes</td>
<td>Offline</td>
<td>No</td>
<td>No</td>
<td>MILP</td>
<td>Benders decomposition</td>
<td>-</td>
<td>Linear / Minimize transportation cost</td>
</tr>
<tr>
<td>Ardjmand et al. (2020)</td>
<td>HD</td>
<td>3PL</td>
<td>No</td>
<td>Offline</td>
<td>No</td>
<td>Yes</td>
<td>MILP</td>
<td>Benders decomposition, sample average approximation</td>
<td>Inventory balance and replenishment</td>
<td>Linear / Maximize expected revenue and operational costs</td>
</tr>
<tr>
<td>Jiang and Li (2020)</td>
<td>HD</td>
<td>OF</td>
<td>No</td>
<td>Offline</td>
<td>Yes</td>
<td>No</td>
<td>MILP</td>
<td>ALNS algorithm</td>
<td>Vehicle routing</td>
<td>Linear / Minimize transportation cost</td>
</tr>
<tr>
<td>This article</td>
<td>HD</td>
<td>3PL</td>
<td>Yes</td>
<td>Online</td>
<td>No</td>
<td>Yes</td>
<td>MINLP</td>
<td>Matheuristic, local search</td>
<td>-</td>
<td>Piecewise linear / Minimize transportation cost</td>
</tr>
</tbody>
</table>

- "HD" stands for “home delivery” and “PU” stands for “pick-up” delivery policy, either in a store or at a delivery station.
- "OF" stands for “own fleet”.

Table 1: Relevant literature and the position of our work.
3. Problem Definition and Mathematical Model

We now elaborate on the problem definition and explicitly state the assumptions, parameters, and decision variables. This paves the way for the mathematical model of the problem, which has a nonlinear objective function. We then provide a relaxation of this model and discuss its properties.

Following on the problem definition and operational characteristics given in Section 1, we establish the distinctive requirements of serving online customers compared to serving walk-in retail customers. Online customers should be able to purchase the items listed as available on the website, including in the stores that have adopted online order fulfillment practices. Hence, when a customer orders, a real-time decision is to select one of the stores for the provision of each item and to determine its journey to the store where it is to be consolidated with the other items. The delivery duration and cost can be calculated using the pricing rules of the 3PL, e.g., UPS (see UPS 2020).

The fulfillment, consolidation and delivery decisions are made under the following assumptions:

**A1:** The stores that can fulfill the demand of an item are known.

**A2:** Each item ordered by the customer is available in at least one store.

**A3:** The longest time any customer is willing to wait for delivery is known.

**A4:** Customers receive a single package for their orders rather than multiple packages, in separate deliveries.

**A5:** The set of items to be transferred between each pair of stores is known at all times.

**A6:** The 3PL’s pricing structure is concave with respect to the weight of the package and the distance from its origin to its destination.

Assumption A1 requires a continuous inventory control system, which has become a norm in practice through efficient data sharing and management software. Assumption A2 follows from the first one as a customer cannot order an item if it is not available in at least one store. Assumption A3 has some real-life implications, one of which is the delivery options that are offered to a customer at the checkout of the order. Several delivery options are listed in many retailer websites, such as priority (same day), first class (next day) or second class (two day) deliveries within a country, and we take this as an input to formulate the constraint on delivery duration. By imposing a delivery duration limit, this assumption in turn limits the number of stores through which an item can travel.
Assumption A4 implies that the items supplied from the selected stores must be consolidated in a single package at one of the stores before being delivered to a customer. One of the implications of this assumption is that the 3PL is not involved in the handling of the packages because the retailer wants to ensure the correctness of the order and impose certain quality standards before the shipment of the final package to the customer. We point out that the 3PL is responsible for carrying out the lateral transshipments, and the distribution of the online orders between the stores can be accomplished alongside these transshipments. Note that if there is a store that holds all the ordered items, then the delivery plan can become trivial when this store is also the closest to the customer and sends the package directly to the customer.

Assumptions A5 and A6 are necessary for the correct calculation of the costs. The pickup and delivery of the items from and to stores take place twice per day by the 3PL during the day, at 10h and 18h. Between two stores that are within the same city limits, the 3PL can pick up the items at 10h and deliver them at 18h, or pick them up at 18h and deliver them the next day at 10h. For two stores that are in different cities, the 3PL can pick up items at 10h and deliver them at 10h, or pick them up at 18h and deliver them at 18h, in the next day. Thus, the travel time between stores is measured by discrete time units in our setting, one unit for stores in the same city and two units for stores in different cities.

Since we treat the online customer orders similarly to those of walk-in customers, all items associated with the online orders are reserved immediately at designated areas in the selected stores, and then the 3PL picks up these items and delivers them to stores or customers. Therefore, a dynamic delivery decision must be made in real time for each order, which will affect the cost structure in the succeeding orders. The reason is that the decision of sending items between a pair of stores requires updating the concave cost structure of the 3PL, which will not increase the additional cost of shipments for the subsequent orders. We elaborate on the cost structure of the 3PL later in this section.

Consequently, we articulate the tackled problem as follows: for a given set of orders, their associated items and the available stores for each item, the Fulfillment and Consolidation Problem (FCP) is to select a store for each ordered item and consolidate the items at one of the stores before dispatching them to the customers, so that the total delivery times do not exceed a prespecified number of time periods, and the total delivery cost is minimum. We stress that the FCP must be solved in real time when the online order is placed since both walk-in and online customers are served together during work hours of the stores, which imposes a strict time limit on the solution time. The offline version of the FCP, i.e., determining when the orders can be accumulated and shipped together outside the work hours of the stores, does
not entail such restrictive time limits, and is therefore outside the scope of this paper.

The FCP is defined on a complete directed graph \( G = (R, A) \), where \( R \) is the set of nodes corresponding to the stores of the retailer, and \( A \) is the arc set made up of the links between the pairs of stores in \( R \). The set of items in an order is denoted by \( I \), and we denote by \( R_i \) the subset of \( R \) containing item \( i \in I \). Each item \( i \in I \) has a weight \( a_i \). The cost of transporting a total weight of \( \hat{a} \) on arc \((r, q)\) is determined by a concave cost function \( f_{rq}(\hat{a}) \), and it takes \( d_{rq} \) time units for the items to traverse the arc. The cost of dispatching the final consolidated package from \( r \in R \) is denoted by \( c_r \), and it takes \( \hat{d}_r \) time units to reach the customer. Finally, the order must reach the customer no later than \( T \) time units from the time at which the order is issued. The aim of the FCP is to select a node in \( R_i \) for each \( i \in I \) and a consolidation node in \( R \) to dispatch the final package, and to determine a transportation plan between the stores such that the total cost of delivery is minimized.

**Proposition 3.1.** The FCP is NP-hard.

**Proof 3.1.** The proof is given in Appendix A.

We now present a multi-commodity flow model on an extended graph \( \hat{G} = (\hat{V} \cup \hat{A}) \), where \( \hat{V} = R \cup I \) and \( \hat{A} = A_1 \cup A_2 \), with \( A_1 = \{(i, r) : i \in I, r \in R_i \} \) and \( A_2 = \{(r, q) : r, q \in R \} \).

Each arc \((i, j) \in A_2 \) is serviced by the 3PL, which takes \( d_{rq} \) time units to transport a request from \( r \in R \) to \( q \in R \). The duration parameter for \((i, r) \in A_1 \) is \( d_{ir} = 0 \), since these arcs are used to indicate whether store \( r \) is selected to supply item \( i \). Based on this definition, we define binary flow variables \( z_{ir}^i \) for \( i \in I \) and \((r, q) \in \hat{A} \), equal to 1 if and only if item \( i \) is transported on arc \((r, q) \). Finally, let \( y_r \) be a binary variable equal to 1 if and only if store \( r \in R \) is selected as the consolidation point. The resulting model is then

\[
\begin{align*}
\text{minimize} & \quad \sum_{(r, q) \in A_2} f_{rq} \left( \sum_{i \in I} a_i z_{rq}^i \right) + \sum_{r \in R} c_r y_r \\
\text{subject to} & \quad \sum_{r \in R} y_r = 1 \\
& \quad \sum_{(i, r) \in A_1} z_{ir}^i = 1, \quad i \in I \\
& \quad \sum_{q : (q, r) \in \hat{A}} z_{rq}^i - \sum_{q : (r, q) \in A_2} z_{rq}^i = y_r, \quad r \in R, i \in I \\
& \quad \sum_{q : (r, q) \in A_2} d_{rq} z_{rq}^i + \sum_{r \in R} \hat{d}_r y_r \leq T, \quad i \in I \\
& \quad y_r \in \{0, 1\}, \quad r \in R
\end{align*}
\]
Constraints (2) and (3) ensure that one store of the retailer is selected as the consolidation point, and for each \( i \in I \), one unit of flow is carried from node \( i \) to one of the stores in \( R_i \) on arcs of \( A_1 \). Flow conservation is imposed by (4) in which all the store nodes are transshipment nodes except for the one associated with the consolidation point \( r \) for which \( y_r = 1 \). Constraints (5) impose the time restriction for the delivery of each item \( i \in I \) to the customer from node \( i \).

In the left-hand side of this constraint set, the delivery duration from the selected consolidation point \( r \) denoted by \( \hat{d}_r \) is factored in the delivery time which is limited by \( T \). The first and second terms in the objective function represent the cost of carrying items between the stores of the retailer and from the consolidation point to the customer, respectively. Since there is no physical delivery of items on the arc set \( A_1 \), only the costs associated with \( A_2 \) are considered in the first term of the objective function. The solution of the problem features \(|I|\) paths, each starting from one node \( i \in I \) and ending at node \( r \) that is the consolidation point with \( y_r = 1 \).

For each \((r, q) \in \hat{A}\), \( f_{rq} \) is a piecewise concave function of the flow on arc \((r, q)\), namely \( \sum_{i \in I} a_i z^i_{rq} \). That is, the unit transportation cost decreases marginally as the total weight of the items carried on an arc \((r, q)\) increases. The structure of the cost function \( f_{rq} \) varies over the arcs \((r, q) \in A\) based on the distance between nodes \( r \in \hat{V} \) and \( q \in \hat{V} \). This piecewise concave function can be linearized by using auxiliary continuous and binary decision variables.

First, let \( l \in L \) denote the indices of weight levels each represented with the interval \([b_{l-1}, b_l]\) where \( b_{l-1} \) and \( b_l \) stand for the lower and upper weight limits of weight level \( l \), respectively. These intervals are constructed based on the pricing policy of the 3PL, and we assume that \( b_0 = 0 \) without loss of generality. We now introduce a set of continuous variables representing the weight of the flow over an arc \((r, q) \in A_2\) as \( w^l_{rq} \). Similarly, the binary variable \( v^l_{rq} \) takes a value of 1 if and only if a weight level \( l \) is selected for the flow on arc \((r, q)\). The cost of sending a unit flow on an arc \((r, q)\) which is assigned weight level \( l \) is represented by \( c^l_{rq} \). Here, the flow value \( w^l_{rq} \) on arc \((r, q)\) lies within the weight interval \([b_{l-1}, b_l]\) and the total cost is accumulated over each weight level. \( K^l_{rq} \) denotes the fixed cost of sending flow over the arc \((r, q)\) at weight level \( l \), defined as

\[
K^l_{rq} = \begin{cases} 
K^{l-1}_{rq} + (b_{l-1} - b_{l-2})c^{l-1}_{rq} & \text{if } l > 1 \\
\pi & \text{otherwise}
\end{cases} 
\quad \forall (r, q) \in A_2 
\tag{8}
\]

where \( \pi \) is a parameter indicating the fixed cost of sending for weight level \( l = 1 \) as long as \( w^1_{rq} \in [b_0, b_1] \). Note that, for the weight level \( l = 1 \), \( b_0 = 0 \) holds. The fixed cost expression
given in (8) and variable cost values \( c^l_{rq} \) corresponding to the unit cost of sending a unit flow (which decreases as the total flow on \((r,q)\) increases) is used within the following piecewise concave function to calculate the total shipment cost on arc \((r,q) \in A_2\).

\[
f_{rq}(\sum_{l \in L} u^l_{rq}) = \sum_{l \in L} (K^l_{rq} v^l_{rq} + c^l_{rq} w^l_{rq}), \quad \forall (r,q) \in A_2.
\]

Here follows the formulation (P) that we tackle in the rest of the paper:

(P) minimize \[
\sum_{(r,q) \in A_2} \sum_{l \in L} (K^l_{rq} v^l_{rq} + c^l_{rq} w^l_{rq}) + \sum_{r \in R} c_r y_r
\]
subject to \[
(2) - (7)
\]
\[
\sum_{i \in I} a^i z^i_{rq} = \sum_{l \in L} u^l_{rq} \quad \forall (r,q) \in A_2 \tag{10}
\]
\[
w^l_{rq} \leq b_l v^l_{rq} \quad \forall l \in L, (r,q) \in A_2 \tag{11}
\]
\[
w^l_{rq} \geq b_{l-1} v^l_{rq} \quad \forall l \in L, (r,q) \in A_2 \tag{12}
\]
\[
\sum_{l \in L} v^l_{rq} \leq 1, \quad \forall (r,q) \in A_2 \tag{13}
\]
\[
v^l_{rq} \in \{0,1\} \quad \forall l \in L, (r,q) \in A_2 \tag{14}
\]
\[
w^l_{rq} \geq 0 \quad \forall l \in L, (r,q) \in A_2 \tag{15}
\]

where the additional constraints (10)–(13) are used to linearize the flow values on each arc \((r,q)\). Constraints (10) set the value of the auxiliary variables \(u^l_{rq}\) equal to the total weight of flow sent from \(r\) to \(q\). Constraints (11)–(13) guarantee that for each arc \((r,q)\) only one weight level \(l\) that contains \(u^l_{rq}\) is selected, while the other weight levels are forced to zero. Constraints (14) and (15) are integrality and nonnegativity restrictions for variables \(v^l_{rq}\) and \(w^l_{rq}\), respectively.

This problem can be tackled by fixing \(y_r = 1\) for each \(r \in R\) and solving the resulting problem, which is a multi-commodity network flow problem with a piecewise concave cost function. Balakrishnan and Graves (1989) and Muriel and Munshi (2004) proposed heuristics for the uncapacitated and the capacitated versions of this problem. Not only is the exact solution of this problem challenging for each \(r \in R\), but also the large size of \(|R|\) precludes the use of this approach for the solution of (P). Proposition 3.2 presents two valid inequalities for (P).
Proposition 3.2. The following inequalities are valid for (P):
\[
\sum_{l \in L} v_{rq}^l \geq z_{rq}^i, \quad (r,q) \in A_2, i \in I
\]  
(16)
\[
\sum_{l \in L} \sum_{q \in R \setminus \{r\}} v_{rq}^l \leq 1, \quad r \in R.
\]  
(17)

Proof 3.2. The valid inequalities (16) follow from the fact that if \( z_{rq}^i = 1 \) then \( \sum_{l \in L} w_{rq}^l \geq a_i \) by (10), and consequently \( \exists l \in L : v_{rq}^l = 1 \) due to (11) and (13). The valid inequalities (17) are based on the argument about the existence of an acyclic optimal solution for the FCP, which was stated in the proof of Proposition 3.1.

4. Algorithm

In this section, we focus on the requirements of FCP and the structure of (P) in order to develop an efficient algorithm that captures their features. First, we note that FCP is an online problem so that (P) is solved whenever a new online order is placed. Let us denote the ordered set of orders by \( O = \{1,...,|O|\} \), where \(|O|\) is the number of orders arriving during work hours of the day. The set of items associated with \( o \in O \) is denoted by \( I^o \). After the solution of (P) for a given order \( o \), the total weights on the arcs are updated with the weights of the items \( I^o \) carried over the selected arcs. Formally, after solving (P) for order \( o \), the total weight on arc \( (r,q) \in A_2 \) becomes
\[
\sum_{l \in L} w_{rq}^l = \sum_{o' \in O, o' \leq o} \sum_{i \in I'} a_i z_{rq}^i.
\]

Then, when solving (P) for \( o + 1 \), we shift the intercept \( K_{rq}^i \) for each arc \( (r,q) \in A_2 \) to the total weight of items associated with the previous orders on the arcs, and minimize the incremental cost of carrying items of this order. The reason for considering only the incremental cost of delivering \( I^{o+1} \) is that it is not possible to change the previous item-retailer allocation decisions, which entail the removal of items from their locations and storage for pick-up by the 3PL. Thus, as an online optimization problem, the solution of (P) for an order, say \( o + 1 \), is influenced by the decisions made for the previous orders.

Given these descriptions for FCP, we make two caveats regarding its solution. First, being an online problem, FCP does not lend itself to optimality by the solution of (P) for each order. That is, solving (P) to optimality for each \( o \in O \) does not yield the minimum total cost for the offline version of FCP. Second, we observed in the experiments that, even for a single order, determining an optimal solution of (P) within few seconds is not viable in real-life instances of FCP.

Consequently, we develop an alternative formulation for a special case of the FCP in which the time constraints are relaxed and the stores of the retailer are connected to the consolidation
point via a single arc. Note that this is the fastest path that an order can follow due to the
discrete pick-up and delivery times of items at the stores. This ensures that the time constraints
are satisfied with this modification. Hence, we refer to the modification of FCP that sends each
item to the consolidation point via single arcs as the restricted FCP (RFCP), and its model
for a single order will be given next. We also point out that the RFCP yields an upper bound
on the optimum of (P) for a single order. However, when the RFCP is solved sequentially
for a set $O$, the solution of this problem may provide a smaller total delivery cost than that
obtained by solving (P) optimally for the same orders due to the online nature of the problem.
In fact, in Section 5, this will be exemplified on randomly generated instances, which verifies
our conjecture.

Restricting the path of each item to consist of a single arc leads to simplifications in (9)–
(15), namely the removal of constraints (5) that impose the time limit for each item. We can
also strip $\hat{G}$ of the nodes associated with $I$. Consequently, constraints (3) and (4) can be
combined to enforce a direct connection from the selected stores to the consolidation point,
e.g., using constraints (20). The resulting model is

$$
(P') \text{ minimize } \sum_{(r,q) \in A_2} \sum_{l \in L} (L^l + c_q^l w_q^l) + \sum_{r \in R} c_r y_r
$$

subject to

1) Selection of the consolidation point followed by the selection of the stores for fulfillment.
This approach first sets the store of the retailer that will send the final delivery to the
customer location, i.e., the store with the minimum $c_r$ value, as the consolidation point.
Then, a set of stores that contains all the items is selected in order to minimize the total
cost of distribution. These stores may be far away from the consolidation point, which
increases the solution cost.

ii) Selection of stores for fulfillment followed by the selection of the consolidation point.
This approach first selects a subset of stores of containing the items requested, and then
selects a store as the consolidation point that minimizes the distribution cost. The former
problem is a set covering (SC) problem. The latter problem is a 1-median facility location problem since a set of store nodes are to be connected to one of the nodes that will be the consolidation point.

The resulting solutions of both approaches can be considerably inferior to that of (P’) as the first decisions do not capture the essence of the objective function. On the other hand, our preliminary experiments indicate that the former approach (i) of first fixing the consolidation store and then solving the resulting formulation (P’) to determine the fulfillment stores does not yield an efficient algorithm. This is due to long running times to solve the resulting (P’) formulation, still being an MILP, even after reduction of the \( y_r \) variables. Therefore, we will adopt the second method (ii), and embed it within an iterative matheuristic that combines mathematical programming and local search. The first decision, namely the allocation of items to retailers, gives rise to the following set covering model:

\[
\begin{align*}
\text{(SC) minimize} & \quad \sum_{r \in R} c_r \theta_r \\
\text{subject to} & \quad \sum_{r \in R_i} \theta_r \geq 1 \quad i \in I \\
& \quad \theta_r \in \{0, 1\} \quad r \in R,
\end{align*}
\]

where the binary variable \( \theta_r \) is equal to 1 if and only if store \( r \) of the retailer is selected, and constraints (22) ensure that the selected stores contain all items. For the special case with \( c_r = 1, r \in R \), the objective becomes the minimization of the number of stores. However, the resulting solution may involve stores that are faraway from each other and from the customer. We overcome this shortcoming by assigning an approximate cost \( c_r \), which is a function of the distance from \( r \) to the customer, as the objective function coefficient. Hence, this model is to fulfill the order with a small number of stores that are close to the customer, and in turn, hopefully to each other. We underline that \( c_r \) is an input for the exact model (P), but it is used in the approximate model (SC) as a user-set parameter.

Let \( R^+ \) denote the selected stores in the solution of the (SC) model (21)–(23), which covers all items. Note that in an optimal solution of this problem, some of the orders may be covered by more than one store so that \( R_i^+ \), which is the set of selected stores covering order \( i \in I \), may have a cardinality larger than one. Moreover, the (SC) model does not specify which consolidation point is to be selected from \( R^+ \). The ensuing problem, which not only assigns orders to a particular store in \( R^+ \) but also picks one of \( R^+ \) as the consolidation point, is equivalent to (P’) defined over a relatively small set \( R^+ \). We employ the enumerative procedure
described in Algorithm 1 to minimize the delivery cost for a given $R^+$. 

**Algorithm 1:** Finding the best assignment and consolidation point given $R^+$

1. **Input:** $R^+$, $R^+_i$
2. $S = \emptyset$, $S' = \emptyset$, $\min = \infty$
3. **for** $i \in I$ **do**
   4. **if** $|R^+_i| = 1$ **then**
      5. **if** $S = \emptyset$ **then**
         6. $S = \{R^+_i\}$
      7. **else**
         8. **for** $S \in S$ **do**
            9. $S = S \cup R^+_i$
     else
      10. $S' = S$
       **for** $r \in R^+_i$ **do**
       12. **for** $S \in S'$ **do**
        13. $S' = S \cup \{r\}$
        14. $S = S \cup \{S'\}$
6. **for** $S \in S$ **do**
5. **for** $r \in R^+$ **do**
8. **if** $g(r, S) < \min$ **then**
   9. $r^* = r$
   10. $S^* = S$
   11. $\min = g(r, S)$
22. **Output:** $r^*, S^*, g(r^*, S^*)$

The first part (lines 2–15) of Algorithm 1 constructs the set of solutions, represented by a set of sets $S$, each element of which is a set of stores. The set $S$ is constructed iteratively by assigning a store to a given item at each step. For each item $i \in I$ for which $|R^+_i| = 1$, we add the store $R^+_i$ to each solution $S \in S$ (line 9). For $i \in I$ with $|R^+_i| > 1$, we first create a copy $S'$ of $S$ (line 11), and add to $S$ a solution with one of $R^+_i$ assigned to item $i$ (lines 14, 15). Given these solutions that are formed of all possible assignments of stores to orders, we start searching for the best consolidation point that results in the minimum cost. To this end, for every solution $S \in S$ and candidate consolidation point $r \in R^+$, we can find the total weight carried from each store to the consolidation point $r$, which makes the calculation of the total cost possible (denoted by the function $g(r, S)$). The total cost of a solution $S$ is calculated by explicitly enumerating each $r \in R^+$ as the consolidation point for $S$. The minimum total cost achieved among $r \in R^+$ consolidation point alternatives constitutes the objective value for every solution $S$. This calculation is performed for every solution $S \in S$. 

17
Finally, the assignment and consolidation point that attain the minimum cost are outputted by this algorithm.

The solution of (SC) and the ensuing Algorithm 1 executed in order to determine the consolidation point and store allocation leave room for improvement due to the lack of linkage between these two problems. Hence, after solving these problems sequentially, we apply a local search procedure to improve the solution. Algorithm 1 enumerates the solutions \( S \in \mathbb{S} \) and calculates their corresponding objective values. A simple move operator is applied to solutions \( S \in \mathbb{S} \). To this end, we refer to \( i \in I \) as an item selected from a solution \( S \), and \( r_i^+ \in R_i^+ \) is the store to which it is assigned. The local search operator moves \( S \) to the neighbor solution where \( i \) is assigned to a new store \( r_i \in R_i \setminus R_i^+ \). Keeping the remaining items assigned to the same stores in \( S \), we identify the consolidation point yielding the minimum cost after this move and calculate the cost improvement over \( S \). This operation is performed for each item \( i \) of solution \( S \), and we construct a new solution \( S' \) based on the best move only if it yields a lower transportation cost than \( S \). Otherwise, this move is applied to the next solution in \( S \), and the local search algorithm terminates when no improvement is possible for any \( S \in \mathbb{S} \). Applying this improvement algorithm to all solutions \( S \in \mathbb{S} \) increases the likelihood of obtaining a better solution. This type of improvement step can be considered as an intensification phase, which exists in most metaheuristics such as tabu search. Our aim is to keep the search mechanism simple in order to reduce the computational burden, considering the fact that quick decisions are to be made to meet online orders.

The move operator that we use for the LS procedure is illustrated on a test instance of 10 stores of the retailer with an order of three items in Figure 1. The order items are shown in light grey, medium grey, and dark grey to represent items 3, 9 and 10 from left to right, respectively. The stores are denoted on the upper left with the sets \( R_3, R_9 \) and \( R_{10} \). For example, item 3 is available at stores 5 and 9, and therefore, \( R_3 = \{5, 9\} \). The solution \( S^* \) returned by Algorithm 1, run after the (SC) model is solved, is \( S^* = \{5, 10, 1\} \) as the stores providing items 3, 9 and 10, respectively. The stores are represented with circles and their number is shown within. Consolidation points are shown with a hash pattern for all solutions and thus store 1 is the consolidation point in the output of Algorithm 1 with a total cost of 22.51. The solution set of the LS procedure is denoted at the bottom. Each item is sequentially assigned to all possible stores having it. For instance, the first solution of the LS procedure moves from store 5 to store 9 for item 3, yielding a total cost of 22.94. Similar steps are repeated for each item in \( S \) using the store sets \( R_3, R_9 \) and \( R_{10} \). Note that each solution is actually the outcome of the best consolidation point alternative. For \( S' = \{9, 10, 1\} \), each store is checked as playing the
role of the consolidation point, and store 1 yields a lower cost than the other two alternatives, and hence, becomes the consolidation point for this solution $S'$. The LS procedure does not always yield an improvement. Fortunately, the second solution represented with $S' = \{5, 7, 1\}$ where store 1 is the consolidation point, improves upon the initial solution $S$, and it is reported as the outcome of the LS procedure after checking the other potential solutions.

The search space induced by the move operator can be expanded by incorporating multiple items and stores to be exchanged. The main hurdle for replacing multiple stores associated with multiple items by such a move lies in the covering constraints because maintaining the feasibility of these constraints becomes more arduous for moves involving multiple items. For example, when a store $r_1$ supplies two or more items of an order, removing that store $r_1$ from the solution could harm the feasibility unless the store used for replacement, say $r_2$, has all the items provided by $r_1$. Nevertheless, we have developed a move operator, which simultaneously replaces the stores of up to two items of the current solution. We call the resulting local search LS2 in what follows.

LS2 starts with checking the same move operator for a single item in an order as in LS. Then, LS2 further searches for improving solutions where two items can be supplied from two different stores than the ones currently in the solution. In LS2, the cases for which it is no longer possible to cover all order items by newly replaced stores (e.g. exchanging two stores simultaneously), corresponds to an infeasible solution. LS2 checks for infeasibility that may arise after the move operation and only considers feasible solutions during its search. To illustrate LS2 better, consider the (SC) model solution $S^* = \{5, 10, 1\}$ in the Figure 1. In addition to the LS solution where store 5 is replaced with store 9 for item 3 of the order in $S' = \{9, 10, 1\}$, LS2 would also consider the solution $S' = \{9, 7, 1\}$ replacing both of the stores.
5 and 10 simultaneously with stores 9 and 7, respectively, for items 3 and 9 of the order. Other item pairs and their corresponding alternative stores are exchanged similarly to enumerate all combinations of neighbor solutions in LS2. Observe that exchanging the stores of two items can potentially increase the computational expense due to increased search space in LS2. For the sake of computational efficiency, we restrict LS2 to exchanges of stores with at most two items. Indeed, we demonstrate that our choice constitutes a reasonable trade-off according to our findings based on computational experiments performed for the LS2.

We also apply a diversification method that benefits from a well-known result from mathematical programming. The exact algorithm by Soyster et al. (1978) iteratively adds a series of cuts, called canonical cuts by Balas and Jeroslow (1972), and reaches an optimal solution. The basic idea is to exclude an integer solution found at previous iterations. This algorithm is able to generate good quality integer solutions within a small number of iterations. It was later improved by Wilbaut and Hanafi (2009) and then by Hanafi and Wilbaut (2011). Inspired by this approach, we add the following constraint to the (SC) model (21)-(23) in order to direct the search toward another part of the solution space:

$$\sum_{r \in R^+} \theta_r \leq |R^+| - 1.$$ \hspace{1cm} (24)

This constraint eliminates a single solution, and when added iteratively, it will forbid a set of solutions to generate a distinct set of stores in the search space. Constraint (24) is very similar to the canonical cuts and helps to search over good quality solutions within short computing times. After each new solution obtained by the (SC), we apply a local search algorithm to further improve the solution, e.g. LS and LS2.

5. Computational Experiments

In this section, we assess the performance of our algorithm on a number of test instances, and we present managerial insights. The instances were randomly generated and solved with a CPU time limit of 1,800 seconds using Gurobi 8.1.1 under its default settings to obtain a feasible solution for each formulation. Even though the Gurobi solutions are not necessarily optimal, they can be used as benchmarks for comparative purposes. The experiments were performed on a PC with i5-9350u 1.90 GHz Processors and 8 GB RAM operating within 64-bit Microsoft Windows 10 environment. We used C++ as the programming language.
5.1. Test Bed

The FCP takes into account time restrictions for the orders to be delivered, and it assumes that the items of an online order are available in at least one of the stores. Here, two issues should be handled meticulously: i) the feasibility of the test instances (i.e., an order can be fulfilled within the required time limit) and ii) the generation of non-trivial instances (i.e., no store holds all the ordered items).

To this end, an order $o$ is randomly generated in such a way that it contains $|I_o|$ items. It is assumed that the items of an order are distinct. That is, the same items of an order can be handled by treating them as different items. Considering the fact that there are typically multiple items in an order (Kunst, 2020), the number of items in order $o$ is randomly selected as a discrete value between 3 and 5, i.e., $|I_o| \in \{3, 4, 5\}$. An order with one or two items is relatively easy to handle, but the test instance becomes more challenging for more than two items in an order. On the other hand, more than five items in an order is less likely for online fashion shopping. With each item $i \in I$ is associated with a random weight $a_i$ uniformly selected in the interval $[0.2, 1.5]$, which affects the total cost of the shipment. These values are selected so as to reflect the real-life practice in the fashion industry, but our approach is generic and can handle other parameters.

For single-item orders, the order is met from the closest facility that has the item in its inventory, which is optimal due to the cost function being stepwise increasing with respect to the distance. For two-item orders, the optimal solution would consist of two facilities, one as the consolidation point and the other facility that ships to the consolidation store. The optimal solution can be found by complete enumeration in quadratic time and this type of instance is not considered in this study. Assuming that all items are available in a store, then direct shipment is optimal from the closest store. Therefore, the case where a store does not hold all the items becomes of particular interest and affects the transportation costs significantly. Nevertheless, the suggested algorithms remain valid and can be applied.

We generated instances with the number of stores $|R| \in \{10, 20, 40, 80, 120, 160, 200, 240, 300\}$. This yields $9 \times 3 = 27$ (product of the number of stores and the number of items in an order) instance combinations. For each of these combinations 10 test instances are created, and thus, 270 test instances are randomly generated for the FCP in total. With each store $r \in R$ is associated a coordinate in the plane. The store coordinates $(X_r, Y_r)$ are randomly chosen among the integer coordinates in the $[0, 1000]^2$ square. The customer location associated with each order $o$ is also randomly assigned a coordinate using the same parameters.

The 3PL charges with respect to the total weight of items in a delivery and the distance
traveled. The cost of transportation is based on a base price of $K^1$, as defined in Section 3. The main aspect of the cost structure depends on the total weight transferred between the links. The unit transportation cost per kg is a step function, the value of which decreases by a factor of $\beta$ for every added $\kappa$ kg. For every $\tau$ km of travel, the unit transportation cost per km increases by a factor of $\gamma$ as a step function. Mathematically, the transportation cost is computed as

$$ f_{rq}(\hat{a}) = \begin{cases} K^1 \times \gamma^{\lfloor \frac{d_{rq}}{\tau} \rfloor} \times \hat{a} & \text{if } \hat{a} \leq \kappa \\
 K^1 \times \gamma^{\lfloor \frac{d_{rq}}{\tau} \rfloor} \times (\sum_{k=0}^{\lfloor \hat{a}/\kappa \rfloor-1} (\kappa \times \beta^k) + (\hat{a} - \kappa \times \lfloor \hat{a}/\kappa \rfloor) \times \beta^{\lfloor \hat{a}/\kappa \rfloor}) & \text{if } \hat{a} > \kappa. \end{cases} \tag{25} $$

We emphasize that this pricing rule results in a a piecewise concave function.

Consider an example in which we aim to ship a package of total weight 2.5 kg to another store at a distance of 450 km. We will set the values of the pricing parameters as $\pi = $5.00, $\tau = 200$ km, $\gamma = 1.05$, $\kappa = 1$ kg, and $\beta = 0.9$. The cost of shipping is then

$$ \pi \times \gamma^2 \times ((1 \times \beta^0) + (1 \times \beta^1) + (0.5 \times \beta^2)) = $12.71. \tag{26} $$

Once an order $o$ is randomly generated, we initially ensure that every item $i \in I^o$ is supplied from at least one store of the retailer by assigning it to a store. Then, additional stores from the set $R$ having the items $I^o$ in order $o$ are randomly assigned to these items, and an item can exist at multiple stores. We make sure that none of the retailers can provide more than $|I^o| - 2$ of the items to avoid trivial solutions, such as providing all items from a single retailer. Our approach guarantees that every order is fulfilled, and that there is a sufficient number of alternative delivery options for each instance. We provide a pseudocode of the instance generation algorithm in Algorithm 2 in Appendix C.

### 5.2. Computational Results

Table 2 presents the results obtained by solving models (P), (P') and (SC). The first two columns give the sizes of the instances. The rows under “Stores” and “Items” give the number of stores and the number of items in the order. “UB” and “CPU(s)” denote the upper bound and the total CPU time in seconds associated with the corresponding solution method. Columns 3 to 10 are dedicated to upper bound and CPU performance of the respective models. As a remark, Algorithm 1 is run after solving the (SC) model to optimality. Each cell gives the average values of 10 randomly generated test instances. The instances with the same number of stores are grouped and their average is given in the bottom row. The last row of the Table
2 gives the overall average of each column taking into account all instances.

<table>
<thead>
<tr>
<th>Instance sizes</th>
<th>(P)</th>
<th>(P')</th>
<th>(SC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stores Items</td>
<td>UB</td>
<td>CPU(s)</td>
<td>UB</td>
</tr>
<tr>
<td>3</td>
<td>26.77</td>
<td>0.09</td>
<td>26.77</td>
</tr>
<tr>
<td>10</td>
<td>23.29</td>
<td>0.12</td>
<td>23.29</td>
</tr>
<tr>
<td>5</td>
<td>23.59</td>
<td>0.15</td>
<td>23.59</td>
</tr>
<tr>
<td>Average</td>
<td>24.55</td>
<td>0.12</td>
<td>24.55</td>
</tr>
<tr>
<td>3</td>
<td>27.56</td>
<td>0.37</td>
<td>27.56</td>
</tr>
<tr>
<td>20</td>
<td>24.16</td>
<td>0.67</td>
<td>24.16</td>
</tr>
<tr>
<td>5</td>
<td>25.62</td>
<td>0.54</td>
<td>25.62</td>
</tr>
<tr>
<td>Average</td>
<td>25.78</td>
<td>0.52</td>
<td>25.78</td>
</tr>
<tr>
<td>3</td>
<td>26.78</td>
<td>2.68</td>
<td>26.78</td>
</tr>
<tr>
<td>40</td>
<td>22.44</td>
<td>2.41</td>
<td>22.44</td>
</tr>
<tr>
<td>5</td>
<td>23.73</td>
<td>6.19</td>
<td>23.73</td>
</tr>
<tr>
<td>Average</td>
<td>24.32</td>
<td>3.76</td>
<td>24.32</td>
</tr>
<tr>
<td>3</td>
<td>27.08</td>
<td>32.73</td>
<td>27.08</td>
</tr>
<tr>
<td>80</td>
<td>22.76</td>
<td>43.81</td>
<td>22.76</td>
</tr>
<tr>
<td>5</td>
<td>24.62</td>
<td>52.36</td>
<td>24.62</td>
</tr>
<tr>
<td>Average</td>
<td>24.83</td>
<td>42.97</td>
<td>24.83</td>
</tr>
<tr>
<td>3</td>
<td>28.62</td>
<td>110.26</td>
<td>28.62</td>
</tr>
<tr>
<td>120</td>
<td>25.58</td>
<td>169.70</td>
<td>25.58</td>
</tr>
<tr>
<td>5</td>
<td>24.40</td>
<td>182.91</td>
<td>24.40</td>
</tr>
<tr>
<td>Average</td>
<td>26.20</td>
<td>154.29</td>
<td>26.20</td>
</tr>
<tr>
<td>3</td>
<td>24.18</td>
<td>368.53</td>
<td>24.18</td>
</tr>
<tr>
<td>160</td>
<td>25.36</td>
<td>519.48</td>
<td>25.36</td>
</tr>
<tr>
<td>5</td>
<td>23.80</td>
<td>571.96</td>
<td>23.80</td>
</tr>
<tr>
<td>Average</td>
<td>24.45</td>
<td>486.66</td>
<td>24.45</td>
</tr>
<tr>
<td>3</td>
<td>24.47</td>
<td>836.19</td>
<td>24.47</td>
</tr>
<tr>
<td>200</td>
<td>26.11</td>
<td>875.59</td>
<td>26.73</td>
</tr>
<tr>
<td>5</td>
<td>26.23</td>
<td>1362.22</td>
<td>26.51</td>
</tr>
<tr>
<td>Average</td>
<td>25.60</td>
<td>1024.67</td>
<td>24.57</td>
</tr>
<tr>
<td>3</td>
<td>25.6</td>
<td>1650.05</td>
<td>23.98</td>
</tr>
<tr>
<td>240</td>
<td>35.53</td>
<td>1738.88</td>
<td>23.80</td>
</tr>
<tr>
<td>5</td>
<td>35.8</td>
<td>1800.00</td>
<td>23.74</td>
</tr>
<tr>
<td>Average</td>
<td>32.31</td>
<td>1729.64</td>
<td>23.84</td>
</tr>
<tr>
<td>3</td>
<td>31.36</td>
<td>1800.00</td>
<td>25.67</td>
</tr>
<tr>
<td>300</td>
<td>24.87</td>
<td>1694.04</td>
<td>20.5</td>
</tr>
<tr>
<td>5</td>
<td>31.74</td>
<td>1743.63</td>
<td>30.49</td>
</tr>
<tr>
<td>Average</td>
<td>29.32</td>
<td>1745.89</td>
<td>25.55</td>
</tr>
<tr>
<td>Overall average</td>
<td>26.37</td>
<td>576.50</td>
<td>24.90</td>
</tr>
</tbody>
</table>

Table 2: Performance of the models (P), (P') and (SC).
There are several cases where Gurobi reaches the CPU time limit. In column (P), this can be observed for the instances with 200, 240 and 300 stores respectively in three, 21 and 27 out of 30 instances. On the other hand, these values for the (P') model are only five out of 30 instances with 300 stores. Models (P’) and (SC) enforce the items to reach the consolidation point using a single arc so that the delivery is completed using two arc connections: one for reaching the consolidation store and the other for the final delivery to the customer. By construction of instances, total duration over such two arcs does not exceed the specified time limit \( T \) (maximum delivery time). This implies that (P’) and (SC) satisfy delivery time constraints (5) and thus fastest possible delivery is provided for each customer. Nevertheless, (P) can possibly benefit from scalability and its solutions could use multiple arcs to connect with the consolidation store as long as the maximum delivery time limit is satisfied. Therefore, lower average cost values are based on better optimization results. This underpins the interest of our strategy of basing the solution algorithm on a relaxed model such as (P’). Solving (P’) is more than four times faster on average than solving (P). Indeed, the overall total cost values for (P’) are lower than those of (P) since the running times to reach the optimum are longer than the CPU time limit of 1,800 seconds for (P). Nevertheless, we observe that the running time increases prohibitively as the number of stores increases for both (P) and (P’). For an online problem such as the FCP, decisions have to be made within seconds, and we can conclude that solving either the (P) or the (P’) model exactly is not a viable option.

The results obtained by (SC) are very promising in terms of running time. Indeed, the running time requirement of the (P’) model is more than 15,000 larger than those of the (SC) model. We can observe a slight deterioration in accuracy when compared to (P’) model outcome. The (SC) model comes at an expense of 1.69% worse upper bounds than for the (P’) model in overall averages, respectively. However, the fact that the (SC) can be solved very efficiently with a minor compromise in accuracy makes it a viable method for an online decision making problem like FCP.

Recall that we improve the performance of the (SC) by applying local search procedures LS and LS2 described in Section 4. In addition to running Algorithm 1, the LS and LS2 are performed on each solution \( s \) gathered from Algorithm 1 to obtain an improved solution, if possible. These methods are respectively called the first and second improvement strategies denoted by “(SC) with LS” and “(SC) with LS2” in Table 3 where we present our improved results based on (SC) model. The third improvement strategy adds constraint (24) for five iterations. To this end, a (SC) model is solved, and Algorithm 1 is run. Constraint (24) is then added to the latest (SC) model, and the process is reiterated until a prespecified number
of iterations is reached. Clearly, this improvement strategy creates different (SC) solutions at each iteration and mimics the diversification procedures used in metaheuristics. Therefore, it increases the likelihood of finding an improved solution for the FCP. The output of the third improvement strategy is listed under the heading “(SC) with (24)”. The fourth and fifth improvement strategies respectively combine the former two with the latter and use both the LS and LS2 together with the constraints (24). These are indicated with the headings “(SC) with LS + (24)” and “(SC) with LS2 + (24)”, respectively. The LS and LS2 procedures are run as usual after Algorithm 1, and the constraints (24) are added afterwards. The iterations are repeated as in the third strategy until five iterations are completed.

For the sake of clarity, we present the percentage improvement in Table 3 as “IMP (%))”. It is calculated using the formula $100 \times \frac{UB_{base} - UB_{imp}}{UB_{base}}$ where $UB_{base}$ is the output of the (SC) model without any improvement strategy and $UB_{imp}$ is the output obtained by the corresponding improvement strategy. The contribution of the first and second improvement strategies are limited, as they improve the objective value of the (SC) model by only 0.12% and 0.37% on average, respectively. However, the LS and LS2 procedures do not bring significant additional burden to the (SC) model based method. The third improvement strategy yields better results. It outperforms the (SC) model by 1.84% on average. However, the solution time is around 2.8 times longer than that of the (SC) model on average. On the other hand, the third improvement strategy reaches even better bounds than the (P’) model within a shorter time.
<table>
<thead>
<tr>
<th>Instance sizes</th>
<th>(SC) with LS</th>
<th>(SC) with LS2</th>
<th>(SC) with (24)</th>
<th>(SC) with LS + (24)</th>
<th>(SC) with LS2 + (24)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stores</td>
<td>IMP(%) CPU(s)</td>
<td>IMP(%) CPU(s)</td>
<td>IMP(%) CPU(s)</td>
<td>IMP(%) CPU(s)</td>
<td>IMP(%) CPU(s)</td>
</tr>
<tr>
<td>Items</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.14 0.01</td>
<td>0.14 0.01</td>
<td>0.14 0.02</td>
<td>0.14 0.02</td>
<td>0.14 0.02</td>
</tr>
<tr>
<td>10</td>
<td>0 0.01</td>
<td>0.15 0.02</td>
<td>0.65 0.02</td>
<td>0.65 0.02</td>
<td>0.65 0.02</td>
</tr>
<tr>
<td>5</td>
<td>0 0.01</td>
<td>0.10 0.01</td>
<td>0.27 0.02</td>
<td>0.27 0.02</td>
<td>0.27 0.02</td>
</tr>
<tr>
<td>Average</td>
<td>0.05 0.01</td>
<td>0.13 0.01</td>
<td>0.35 0.02</td>
<td>0.35 0.02</td>
<td>0.35 0.02</td>
</tr>
<tr>
<td>20</td>
<td>0.39 0.01</td>
<td>0.53 0.01</td>
<td>0.28 0.02</td>
<td>0.44 0.02</td>
<td>0.54 0.02</td>
</tr>
<tr>
<td>4</td>
<td>0 0.01</td>
<td>0.49 0.02</td>
<td>1.82 0.02</td>
<td>1.82 0.02</td>
<td>2.01 0.03</td>
</tr>
<tr>
<td>5</td>
<td>0 0.01</td>
<td>0.24 0.02</td>
<td>3.02 0.03</td>
<td>3.02 0.03</td>
<td>4.07 0.04</td>
</tr>
<tr>
<td>Average</td>
<td>0.13 0.01</td>
<td>0.42 0.01</td>
<td>1.71 0.02</td>
<td>1.76 0.02</td>
<td>2.2 0.03</td>
</tr>
<tr>
<td>40</td>
<td>0.29 0.01</td>
<td>0.37 0.01</td>
<td>0.63 0.02</td>
<td>0.61 0.02</td>
<td>0.65 0.04</td>
</tr>
<tr>
<td>4</td>
<td>0 0.01</td>
<td>0.28 0.02</td>
<td>4.72 0.03</td>
<td>4.72 0.03</td>
<td>4.98 0.05</td>
</tr>
<tr>
<td>5</td>
<td>0 0.01</td>
<td>0.59 0.02</td>
<td>1.18 0.03</td>
<td>1.18 0.03</td>
<td>3.32 0.08</td>
</tr>
<tr>
<td>Average</td>
<td>0.18 0.01</td>
<td>0.49 0.02</td>
<td>1.84 0.03</td>
<td>1.9 0.03</td>
<td>2.66 0.17</td>
</tr>
<tr>
<td>80</td>
<td>0.47 0.01</td>
<td>0.62 0.01</td>
<td>0.38 0.02</td>
<td>0.69 0.02</td>
<td>0.76 0.1</td>
</tr>
<tr>
<td>4</td>
<td>0 0.01</td>
<td>0.18 0.02</td>
<td>3.05 0.03</td>
<td>3.05 0.03</td>
<td>3.95 0.19</td>
</tr>
<tr>
<td>5</td>
<td>0 0.01</td>
<td>0.00 0.01</td>
<td>6.08 0.04</td>
<td>6.08 0.04</td>
<td>7.66 0.92</td>
</tr>
<tr>
<td>Average</td>
<td>0.16 0.01</td>
<td>0.27 0.02</td>
<td>3.17 0.03</td>
<td>3.27 0.03</td>
<td>4.12 0.4</td>
</tr>
<tr>
<td>120</td>
<td>0.35 0.01</td>
<td>0.44 0.01</td>
<td>0.43 0.02</td>
<td>0.54 0.02</td>
<td>0.54 0.15</td>
</tr>
<tr>
<td>4</td>
<td>0 0.01</td>
<td>0.26 0.02</td>
<td>1.63 0.02</td>
<td>1.63 0.02</td>
<td>4.12 0.31</td>
</tr>
<tr>
<td>5</td>
<td>0 0.01</td>
<td>0.23 0.03</td>
<td>1.32 0.03</td>
<td>1.32 0.04</td>
<td>2.21 0.83</td>
</tr>
<tr>
<td>Average</td>
<td>0.12 0.01</td>
<td>0.31 0.02</td>
<td>1.13 0.02</td>
<td>1.16 0.03</td>
<td>2.29 0.43</td>
</tr>
<tr>
<td>200</td>
<td>0.36 0.01</td>
<td>0.46 0.02</td>
<td>0.33 0.02</td>
<td>0.5 0.02</td>
<td>0.5 0.2</td>
</tr>
<tr>
<td>4</td>
<td>0 0.01</td>
<td>0.48 0.02</td>
<td>2.25 0.02</td>
<td>2.25 0.02</td>
<td>5.36 0.37</td>
</tr>
<tr>
<td>5</td>
<td>0 0.01</td>
<td>0.36 0.03</td>
<td>4.43 0.03</td>
<td>4.43 0.03</td>
<td>6.48 3.36</td>
</tr>
<tr>
<td>Average</td>
<td>0.12 0.01</td>
<td>0.43 0.02</td>
<td>2.33 0.02</td>
<td>2.39 0.03</td>
<td>4.11 1.31</td>
</tr>
<tr>
<td>240</td>
<td>0.36 0.01</td>
<td>0.45 0.02</td>
<td>0.38 0.02</td>
<td>0.45 0.02</td>
<td>0.45 0.19</td>
</tr>
<tr>
<td>4</td>
<td>0 0.01</td>
<td>0.31 0.02</td>
<td>2.48 0.02</td>
<td>2.48 0.02</td>
<td>4.68 0.63</td>
</tr>
<tr>
<td>5</td>
<td>0 0.01</td>
<td>0.65 0.04</td>
<td>3.94 0.03</td>
<td>3.94 0.04</td>
<td>5.48 3.31</td>
</tr>
<tr>
<td>Average</td>
<td>0.12 0.01</td>
<td>0.47 0.02</td>
<td>2.27 0.02</td>
<td>2.29 0.03</td>
<td>3.54 1.38</td>
</tr>
<tr>
<td>300</td>
<td>0.24 0.01</td>
<td>0.24 0.02</td>
<td>0.27 0.02</td>
<td>0.27 0.02</td>
<td>0.27 0.35</td>
</tr>
<tr>
<td>4</td>
<td>0 0.01</td>
<td>0.49 0.02</td>
<td>1.23 0.02</td>
<td>1.23 0.03</td>
<td>2.5 0.9</td>
</tr>
<tr>
<td>5</td>
<td>0 0.01</td>
<td>0.60 0.05</td>
<td>3.44 0.04</td>
<td>3.44 0.04</td>
<td>4.53 3.33</td>
</tr>
<tr>
<td>Average</td>
<td>0.08 0.01</td>
<td>0.44 0.03</td>
<td>1.65 0.03</td>
<td>1.65 0.03</td>
<td>2.43 1.53</td>
</tr>
<tr>
<td>Overall average</td>
<td>0.12 0.01</td>
<td>0.37 0.02</td>
<td>1.84 0.02</td>
<td>1.88 0.03</td>
<td>2.74 0.59</td>
</tr>
</tbody>
</table>

Table 3: Performance of (SC) based improvement strategies.
The CPU time requirements of the improvement strategies applied on the (SC) model are significantly less than for solving (P'). These values are more than 13000, 5000, 4500, 4300 and 185 times shorter than the (P') model for the first, second, third, fourth and fifth improvement strategies based on the (SC) model, respectively. The fourth improvement strategy performs slightly better than the third one. The fifth improvement strategy using LS2 and (24) performs the best with an improvement of 2.74% over the (SC) model. We observe that the third, fourth and fifth improvement strategies yield even better solutions than the (P') model surpassing its output by 0.19%, 0.23% and 1.12%, respectively. As a note, the first, second, third, fourth and fifth improvement strategies improve the solutions obtained by (SC) model in 48, 96, 139, 147 and 179 out of 270 test instances, respectively.

We have also performed two-sided Wilcoxon signed-rank tests (Gibbons and Chakraborti, 2003) with a significance level of 0.05 on the output of each improvement strategy compared to its base (SC) model. In particular, our null hypothesis $H_0$ tests whether the median of the difference $\tilde{\mu}(UB_{base} - UB_{imp}) = 0$ or not. Here, $\tilde{\mu}(UB_{base} - UB_{imp})$ represents the median of difference between the output (upper bound) of the base (SC) model and the output of the improvement strategies as before. We observe that the difference between the outcomes of the base (SC) model and its improved versions are statistically significant, yielding a $p$-value of less than 0.001 for each improvement strategy. This demonstrates that our improvement strategies can have a tangible contribution over the base (SC) model. A similar comparison is also performed among several pairs of improvement strategies yielding a $p$-value of less than 0.001 indicating that their difference are statistically significant. To be precise, we have applied the Wilcoxon signed-rank test for the outcomes of i) first and second improvement strategies, ii) second and third improvement strategies, iii) third and fourth improvement strategies and iv) fourth and fifth improvement strategies. Our results show that the fifth improvement strategy of LS2 combined with (24) yields superior results than the fourth improvement strategy of LS combined with (24). This holds for each comparison mentioned in i), ii) iii) and iv).

Compared with the solution of (P'), the best performing improvement strategy is also the fifth one that uses “(SC) with LS2 + (24)”. We observe that in 267 out of 270 test instances the fifth improvement strategy using (SC) yields either the same or superior results compared with the (P') model in terms of accuracy. For the other instances, their results are very close to each other with a clear superiority of the fifth improvement strategy over (P') model solution. Similarly to the fifth improvement strategy, the accuracy of the (SC) model using third and fourth improvement strategies yield the same or superior results compared with the (P') model in 187 and 202 test instances out of 270.
Briefly, we observe that the (SC) model based third, fourth and fifth improvement strategies yield a better accuracy than the (P') model using a single thread by 0.19%, 0.23%, and 1.12% respectively. These results are 0.47%, 0.51% and 1.39% better when compared to (P') model using multiple threads. Our findings confirm the computational efficiency of the (SC) model based improvement strategies which clearly surpass that of the (P') model by more than 185 times in CPU times for the third, fourth and fifth improvement strategies. To sum up the above discussion, the (SC) model-based improvement strategies are observed to be viable solution approaches for the fast solution of FCP.

5.3. Cumulative Effect of Orders

We have generated an additional test instance which contains multiple customer orders. Our aim is to illustrate the cumulative effect of solving our models for each order sequentially. To this end, we have slightly modified Algorithm 2 to generate separate items and coordinates for multiple orders at steps 4 to 9 and steps 18 to 19, respectively. The rest of Algorithm 2 is unchanged.

We worked on a small size test instance having 40 stores and 20 consecutive orders. We present the results of the (P) and (P') models in Figures 2 and 3, respectively. For the sake of readability, Figures 2 and 3 are both split into 3 subfigures of different orders as presented in Figures (2a), (2b), (2c), (3a), (3b) and (3c). In the context of this illustrative example, the orders are sequentially processed, and we assume that all items are replenished at the stores as soon as the order is satisfied. To explore the cumulative effect, we have adopted this strategy that does not cause any loss of generality. Both models (P) and (P') yield different outputs starting with the order 2 and their optimal objective values start to differ with order 7. In the figures, the stores are represented by medium gray nodes, and the consolidation point, which is one of the stores of the retailer, is indicated by a light gray node. The items of the orders can be tracked above the stores (circles) for each order. As an example, the items of order 1 can be listed as \( I = \{1, 6, 7, 8, 10\} \). The flows are accumulated on the arcs as new orders are received and are shown above or below the arcs connecting stores. Each arc indicates a flow between two stores.

The solutions of the order 1 are the same for both (P) and (P'). For order 2, there are two alternative optimal solutions in which store 14 is selected as the consolidation point whilst (P) model uses store 21 and (P') model chooses store 32 to fulfill item 2. Similarly, such alternative solutions can be observed until order 7 where the cumulative cost of (P) model becomes less than that of (P'). This continues until order 13 where (P') rebounds and has a lower cumulative cost than (P) by taking advantage of existing flows over the arcs utilized before.
Figure 2: Illustrative example of cumulative effect of orders for (P) model.
Figure 3: Illustrative example of cumulative effect of orders for (P') model.
between the cumulative costs of (P) and (P') hold in favor of (P') until the last order where
the gap between the models gradually diminishes. In Figure (2c), we can see that (P) model
takes advantage of using multiple arcs to meet the customer demand with the last order 20.
(P') has disadvantage over (P) in this case (see Total Cost (TC) of order 20 for (P) and (P'))
since it connects to the consolidation store using only one arc as shown in Figure (3c).

Clearly, the cumulative effect of orders from Figures 2 and 3 implies that solving (P) to
optimality does not necessarily yield a better outcome than (P'). At first glance, this is counter-
intuitive. However, solving these models sequentially for each order ends up with suboptimal
solutions. The FCP is an online problem, and sequential solutions are not necessarily optimal
for the offline problem.

5.4. Managerial Insights

We can list the following conclusions as managerial insights gained from our work.

• The benefits of using model (P) depends on the tightness of the time limit $T$ and are
  strongly correlated with the cost structure.

• Model (P’) can be used as an efficient alternative in online decision making environments.

• The sequential solution for each order may yield suboptimal solutions when the online
  problem FCP is tackled.

• Efficient heuristic solutions for the online problem are the most viable alternative for the
  FCP.

6. Conclusions

We have considered an e-commerce logistics planning problem faced by retailers in the
fashion industry. It aims to bring together the items of given customer orders at a consolidation
point from which all items are finally sent, with the objective of minimizing the total delivery
cost. The transportation costs associated with the deliveries are concave and depends on
the distance and the total weight of the items. The resulting problem is a fulfillment and
consolidation problem (FCP) that needs to be solved online.

The FCP is a nonlinear problem that can be linearized into a MILP formulation and takes
into account the delivery time requirements of customers. The resulting MILP is difficult to
solve and is computationally expensive. As a remedy, we have developed a relaxed formulation
by ignoring the delivery time requirements, yet providing a premium service that completes all
deliveries on time for each order. Although the relaxed formulation is more efficient than the
original MILP, it still cannot be solved fast enough for this online problem. We overcome this drawback by offering a set covering (SC) model-based relaxations. The (SC) relaxation only takes into account the total cost of sending all items from a consolidation point to the customer so that all the orders are fulfilled from a store of the retailer. The (SC) model-based relaxation yields promising results within very short computing times when compared to MILP and its relaxation.

We have applied five improvement strategies to enhance the accuracy of the (SC) model-based method. The first improvement strategy uses a local search (LS) procedure to find a better solution on the outcome of (SC) model. The second improvement strategy (namely, LS2) extends the LS procedure such that up to two items are considered to exchange stores. The third strategy iteratively adds cuts to the solution of the (SC) model and acts as a diversification procedure. The fourth and fifth strategies combine the former two with the latter strategy. The last improvement strategy produces the best solutions that surpass those from both the MILP and its relaxation. In addition, the fifth improvement strategy quickly reaches solutions that are viable in an online decision making environment such as the FCP.
Appendix A. Proof of Proposition 3.1

**Proposition 3.1** The FCP is NP-hard.

**Proof** By reduction from the *Group Steiner Tree Problem* (GSTP). The GSTP is defined on an undirected graph $G = (V,E)$ with subsets of vertices $S_i \subset V, i \in \{1,...,k\}$. There is a nonnegative weight $\omega_{ij}$ associated with each edge $(i,j) \in E$, and the objective is to find a minimum weight Steiner tree $T$ that contains at least one vertex from each $S_i, i \in \{1,...,k\}$. The GSTP is NP-hard since it is a generalization of the Steiner Tree Problem, one of the 21 NP-Complete problems of Karp (1972).

Let us start from a generic instance of the GSTP and construct an instance of the FCP with $G' = (R,A)$. For each vertex $i \in V$, insert a retailer vertex into $R$. Set the number of items to be $k$ and $R_i = S_i, i \in \{1,...,k\}$, with the weight of each item being $a_i = 1$. For each edge $(i,j) \in E$, add two arcs $(i,j)$ and $(j,i)$ to $A$. Define a weight for each arc in $A$, equal to the weight of its parent edge in $E$, and define the concave cost function for transporting a weight of $\hat{a}$ on arc $(i,j) \in A$ as

$$f_{ij}(\hat{a}) = \begin{cases} 0, & \text{if } \hat{a} = 0, \\ \omega_{ij}, & \text{if } \hat{a} > 0. \end{cases} \quad (A.1)$$

Set the duration of each arc $d_{ij} = 1 \forall (i,j) \in A$. Define the cost and duration of dispatching the final package to be identical for all retailers, i.e., $c_r = 0, \hat{d}_r = 1 \forall r \in R$. Finally, set the delivery time limit $T$ to a large integer value that will allow all connected solutions to be feasible, e.g., $T = k \times |A| + 1$.

Necessity ($\Rightarrow$): Given any solution for the GSTP instance, we can construct a solution for the FCP. If the GSTP solution consists of a single edge, then the corresponding solution has a single arc, with both orientations having the same cost. If the GSTP solution contains multiple edges, we identify one of the nodes of the solution tree $T$ with degree one. For each such node, we identify the corresponding edge in $T$ and its child arc in $A$ that is directed away from the node. We remove the edge from $T$, and add the identified arc to the solution of the FCP instance. We iterate this process until all edges are removed from $T$.

Sufficiency ($\Leftarrow$): Note that there always exists an optimal solution for the resulting FCP instance that is a tree, i.e., has a connected and acyclic structure. For contradiction, assume that there exists an optimal solution that contains a circuit. Removing one arc within the circuit does not violate the connectedness of the solution, and the resulting solution has an objective function value that it less than or equal to the solution it originates from. Any acyclic solution for the FCP instance can be converted into a solution for the GSTP by simply selecting
the edges in $E$ that are the parents of the arcs in the solution.

By construction of the concave cost function, optimal solutions of both instances result in equal costs. We hence conclude that the FCP is NP-hard.
Appendix B. Pseudocode of the Instance Generation Algorithm

Algorithm 2: Instance Generation Algorithm

1. Input: $|R|, |I|, H$
2. $Io = \emptyset, Ri = \emptyset$ for $i \in I,$ $Rx/y = \emptyset$
3. //Generate $H$ order items randomly
4. $i = 0$
5. while $i < H$ do
6.   $rand = \text{random integer in } [1, |I|]$
7.   if $rand \notin Io$ then
8.     $i = i + 1$
9.   $Io = IO \cup \{rand\}$
10. //Generate retailer coordinates randomly
11. $r = 0$
12. while $r < |R|$ do
13.   $(x, y) = \text{random integer pair from } [1, 1000]^2$ for retailer $r + 1$
14.   if $(x, y) \notin Rx/y$ then
15.     $r = r + 1$
16.     $Rx/y = Rx/y \cup \{(x, y)\}$
17. //Generate order (customer) coordinates randomly
18. $(x, y) = \text{random integer pair from } [1, 1000]^2;$
19. $Ox/y = \{(x, y)\};$
20. //Assign all order items to a retailer
21. $i = 1$
22. while $i \leq |Io|$ do
23.   $rand = \text{random integer in } [1, |R|]$
24.   if $rand \notin \bigcup_{j=1}^{i} R_j$ then
25.     $R_i = R_i \cup \{rand\}$
26.     $i = i + 1$
27. //Assign extra items to a retailer randomly to ensure feasibility
28. for $r \in R$ do
29.   for $i \in I$ do
30.     $rand = \text{uniformly selected random number in } [0, 1]$
31.     if $rand < 2/3$ then
32.       if $i \in Io$ then
33.         if $r \notin R_i$ and $\sum_{j \in I^o} |\{r\} \cap R_j| > (|I^o| - 2)$ then
34.           $R_i = R_i \cup \{r\}$
35.         else
36.           $R_i = R_i \cup \{r\}$
37. Output: $Rx/y, Ri, \text{items } I^o, \text{and } Ox/y$
References


Response to Reviewers for the Second Revision of “Minimum Cost Delivery of Multi-item Orders in ECommerce Logistics” (CAOR-D-21-00045)

We thank the two anonymous reviewers for their constructive comments and the Area Editor for giving us a chance to revise our manuscript. Below is how we have handled their recommendations.

Area Editor:

As you can see from the reviewers’ reports, there are still some minor issues to be addressed. Please revise your paper accordingly.

We thank the Area Editor for giving us a chance to revise our manuscript. We have answered the recommendations of the reviewers point by point below. The resulting changes in the manuscript are made in blue colour.

Reviewer #1:

The authors have made due revisions and improvements to the paper. Generally happy to recommend an acceptance.

We thank the reviewer for the positive feedback.

Reviewer #2:

The authors have addressed most of my previous concerns very carefully. In my opinion, the revised version is worth being published, if the authors address the minor comments below.

We thank the reviewer for the positive feedback.

Minor comments:

-page 15: The expression “allocation of items to stores” suggests that items are distributed to stores. Since the items are already available at the stores, I think is more appropriate to use an expression that suggests the selection of stores from where items are being delivered.

Thanks for the comment. We have reworded “allocation of stores” as “selection of stores for fulfilment”.
-page 16: “allocation of items to retailers” seems also inaccurate, as before it is mentioned that there is a single retailer.

Thanks for the comment. We have reworded “allocation of stores” as “selection of stores for fulfilment”.

-page 11 and 16: the cost $c_r$ is introduced on page 11 as the cost of dispatching the final consolidating order from $r$. On page 16, it is suggested that an approximate cost $c_r$ is used, to avoid the solution of (SC) to choose stores that are far from each other and from customer. It is necessary to clarify whether $c_r$ is an input or is it decided such that the heuristic behaves in a certain way.

Thanks for the comment. We are describing an approximate model here. For model (P), which is exact, $c_r$ is an input. For model (SC), which is approximate, $c_r$ is a parameter. We now state this as “We underline that $c_r$ is an input for the exact model (P), but it is used in the approximate model (SC) as a user-set parameter.”

-page 16, last paragraph: “Moreover, .... . This is motivated...” It is not clear to what does “this” refer to. The sentence starting with “This...” should also be revised.

Thanks for the comment. We agree with the reviewer that the meaning of the sentence was not clear. We have deleted the sentence, and did not find any other sentences starting with “This ...” within the paragraph.
M. Hakan Akyüz: Data curation, Methodology, Software, Visualization, Roles/Writing - original draft, Writing - review & editing.

İbrahim Muter: Conceptualization, Formal analysis, Methodology, Software, Roles/Writing - original draft, Writing - review & editing.

Güneş Erdoğan: Formal analysis, Methodology, Roles/Writing - original draft, Writing - review & editing.

Gilbert Laporte: Supervision, Validation, Roles/Writing - original draft, Writing - review & editing.
Click here to access/download
**LaTeX Source Files**
E-Commerce Logistics_COR.zip