Essays on Shipment Consolidation Scheduling and Decision Making in the Context of Flexible Demand

Sepideh Alavi
University of Wisconsin-Milwaukee

ESSAYS ON SHIPMENT CONSOLIDATION SCHEDULING AND DECISION MAKING IN THE CONTEXT OF FLEXIBLE DEMAND

by

Sepideh Alavi

A Dissertation Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

in Management Science

at

The University of Wisconsin-Milwaukee

May 2020

ABSTRACT

ESSAYS ON SHIPMENT CONSOLIDATION SCHEDULING AND DECISION MAKING IN THE CONTEXT OF FLEXIBLE DEMAND

by

Sepideh Alavi

The University of Wisconsin-Milwaukee, 2020
Under the Supervision of Professors Xiaohang Yue and Matthew Petering

This dissertation contains three essays related to shipment consolidation scheduling and decision making in the presence of flexible demand. The first essay is presented in Section 1. This essay introduces a new mathematical model for shipment consolidation scheduling for a two-echelon supply chain. The problem addresses shipment coordination and consolidation decisions that are made by a manufacturer who provides inventory replenishments to multiple downstream distribution centers. Unlike previous studies, the consolidation activities in this problem are not restricted to specific policies such as aggregation of shipments at regular times or consolidating when a predetermined quantity has accumulated. Rather, we consider the construction of a detailed shipment consolidation schedule over a planning horizon. The problem sheds light on the trade-off between transportation costs, manufacturer inventory holding costs, DC inventory holding costs, and DC backorder costs. We develop a mixed-integer quadratic optimization model to identify the shipment consolidation schedule that minimizes total cost. A genetic algorithm is developed to handle large problem instances.

The other two essays explore the concept of flexible demand. In Section 2, we introduce a new variant of the vehicle routing problem (VRP): the vehicle routing problem with flexible repeat visits (VRP-FRV). This problem considers a set of customers at certain locations with certain maximum inter-visit time requirements. However, they are flexible in their visit times. Unlike most other VRPs, the same customer may be visited multiple times within one route. The VRP-FRV has several real-world applications. One scenario is that of caretakers who provide service to elderly people at home. Each caretaker is assigned a number of elderly people to visit one or more times per day. Elderly people differ in their requirements and the minimum frequency at which they need to be visited every day. The VRP-FRV can also be imagined as a police patrol routing problem where the customers are various locations in the city that require frequent observations. Such locations could include known high-crime areas, high-profile residences, and/or safe houses. We develop a math model to minimize the total number of vehicles needed to cover the customer demands and determine the optimal customer visit schedules and vehicle routes. A heuristic method is developed to handle large problem instances.

In the third study, presented in Section 3, we consider a single-item cyclic coordinated order fulfillment problem with batch supplies and flexible demands. The system in this study consists of multiple suppliers who each deliver a single item to a central node from which multiple demanders are then replenished. Each supplier is capable of providing no more than a specific batch size to the central node at a rate that cannot exceed a specified frequency. Moreover, each demander must be replenished with a specific minimum amount over a regular replenishment frequency. Importantly, demand is flexible and is a control action that the decision maker applies to optimize the system. The objective is to minimize total system cost subject to several operational constraints. The decisions include the timing and sizes of batches delivered by the suppliers to the central node and the timing and amounts by which demanders are replenished. We develop an integer programing model, provide several theoretical insights related to the model, and solve the math model for different problem sizes.

This dissertation is dedicated to

my beloved parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am using this opportunity to express my gratitude towards everyone who assisted me in carrying out this dissertation. I would like to express my deepest gratitude to my advisors, Professor Xiaohang Yue and Professor Matthew Petering for their continuous guidance, care, patience, support, and providing me with an excellent atmosphere for conducting research. I believe that I was truly fortunate to have such exceptional professors as my mentors in my life and advisors in my Ph.D. studies. Their knowledge and high standards on research made it a privilege to work with them.

I sincerely appreciate my dissertation committee members, Professor Kaan Kuzu, Professor Samar Mukhopadhyay, and Professor Long Gao for their support, patience, insights and suggestions. I am also grateful for their time and efforts in evaluating this research work. Additionally, I also want to express my gratitude towards Professor Ehsan Soofi and Professor Layth Alwan for all their support in my Ph.D. studies.

Above all, I would like to thank my dear and lovely parents Ahmad Alavi and Soheila Keshavarzian who always amazingly encouraged me in continuing my education, and supported me during the whole duration of my studies from far away.

I would also like to thank my lovely sister Zahrasadat who always made all her efforts to support me in different aspects of my life. She has been my best friend in this long journey far from home. Additionally, I would like to thank my beloved brother Mehdi and my beloved sister Mahla. They were always supporting me and encouraging me with their best wishes.

Essay 1: Shipment Consolidation Scheduling and Inventory Control for a Two-Echelon Supply Chain

1.1. Abstract

This essay introduces a new mathematical model for shipment consolidation scheduling in a two-echelon supply chain. The problem addresses shipment coordination and consolidation decisions that are made by a manufacturer who provides inventory replenishments to multiple downstream distribution centers (DCs). Unlike previous studies, the consolidation activities in this problem are not restricted to specific *policies* such as aggregation of shipments at regular times or consolidating when a predetermined quantity has accumulated. Rather, we consider the construction of a detailed shipment consolidation *schedule* over a planning horizon. The problem sheds light on the trade-off between (1) transportation costs, (2) manufacturer inventory holding costs, (3) DC inventory holding costs, and (4) DC backorder costs. We develop a mixed-integer quadratic optimization model to identify the shipment consolidation schedule that minimizes total cost. A genetic algorithm is developed to handle large problem instances. Experimental results show the effectiveness of the proposed solution method.

1.2. Introduction

In nearly all production contexts, manufacturers fabricate products and distribute them to consumers or to the next customer in the downstream industrial and retail distribution channel. The business climate in which these manufacturers compete is shaped by (1) the ubiquitous role of information technologies that are thought to improve the coordination of inventory decisions, (2) the complexity of the surrounding supply chain, (3) demand volatility, and (4) the need for strategic responsiveness to a variety of unexpected disruptions. As a result, designing supply chain strategies and planning supply chain operations remain complex tasks. To ensure reliable operations whilst preserving efficiency and cost effectiveness, manufacturers and their partners must continually contend with inventory replenishment, transportation, production lot-sizing, stockout/backordering, and shipment consolidation decisions (Russell and Krajewski, 1991). Regarding shipment consolidation, decision makers must consider a number of practical issues such as when to consolidate product shipments and which customer orders will be consolidated.

In this study, we consider a shipment consolidation problem for a two-echelon supply chain in which a manufacturer replenishes inventory at multiple downstream distribution centers (DCs). Unlike previous papers, the consolidation activities in this problem are not restricted to specific *policies* such as aggregation of shipments at regular times or consolidating when a predetermined quantity has accumulated. Rather, we consider the problem of constructing a detailed shipment consolidation *schedule* that minimizes the sum of transportation, manufacturer inventory holding, DC inventory holding, and DC backorder costs over a planning horizon. We develop a mixed-integer quadratic programming model and genetic algorithm to identify the shipment consolidation schedule (i.e. the DC inventory replenishment times and amounts) that minimizes total cost.

The remainder of this study is organized as follows. Section 1.3 reviews the relevant literature on coordinated replenishment and shipment consolidation. A formal description of the problem is provided in Section 1.4. Section 1.5 presents a mathematical formulation of the problem. A genetic algorithm is described in Section 1.6. In Section 1.7, we describe the experimental setup and discuss the experimental results. We conclude in Section 1.8.

1.3. Related literature

In this study, we focus on coordinating shipments from a manufacturer to multiple downstream distribution centers when there is transportation, inventory holding, and backorder costs. Two main streams of literature relate to our work. The first stream considers joint production-inventory replenishment decisions. The second stream of literature considers shipment consolidation decisions. We now discuss these two streams in detail.

Joint production-inventory replenishment problems have been studied broadly in the literature for two-echelon supply chains. These include studies which develop deterministic and stochastic models. Stochastic integrated production-inventory models with random production yield have been studied by Huang (2004), Papachristos and Konstantaras (2006), Maddah and Jaber (2008), Kelle et al. (2009), Chen and Kang (2010), and Liu and Çetinkaya (2011). Our work assumes a deterministic setting. Therefore, we focus our attention on the deterministic models described below.

The theme of deterministic, integrated production-inventory replenishment studies is that a coordinated production and replenishment policy minimizing the joint total cost incurred by both parties is more desirable than the individual optimal policies (Banerjee and Burton 1994). The assumption is that the product is manufactured continuously or in batches at a finite rate and is depleted by the buyer's replenishment orders. The buyer's inventory is then consumed by market demand at a fixed rate. These problems aim to jointly derive the manufacturer's production lot-size and the buyer's replenishment order quantity in order to minimize total costs incurred by both vendor and buyer. The total cost usually includes the inventory holding costs at the vendor and the buyer, vendor's setup cost, vendor's unit production cost, and buyer's ordering cost. The papers in this stream build on each other in terms of policies to determine replenishment sizes. The buyer's replenishments may be equally sized (Lu 1995) or have different sizes as in Hill (1997).

Joint production-inventory replenishment has been studied for a vendor supplying product(s) to a single or multiple buyers. For a single-vendor, single-buyer system, Goyal (1988) requires the supplier's production batch to be an integer multiple of the buyer's shipment size and assumes a shipment is sent to the buyer only after the batch production is completed. Goyal (1995) generalized the results of this model by considering the sizes of successive shipments in each production cycle to increase by a rate equal to the ratio of production and demand rate. A more generalized policy for the size of shipments was developed by Hill (1997). Also, Viswanathan (1998) compared the performance of different policies studied in the aforementioned papers. Following up on the previous studies, Hill (1999) obtained the integrated optimal production and shipment policy which includes successive shipments with increasing sizes followed by a sequence of equal-sized shipments. Goyal and Nebbebe (2000) provided a note on Hill's study and consider one small-sized shipment followed by equal-sized shipments to ensure quick delivery of the first shipment. Toptal and Çetinkaya (2008) obtain analytical results for replenishment order quantities for the vendor and the buyer under different replenishment cost structures.

Several studies consider single-vendor, multiple-buyer settings. For example, Banerjee and Burton (1994) show that classical lot sizing models do not adequately describe a situation where a single vendor produces and supplies a product to multiple customers who buy in discrete lots. They suggest a common replenishment cycle-based inventory model. Lu (1995) extends the results of Goyal (1988) to release shipments before the complete batch is produced. He develops a model to solve for the buyer's optimal economic order quantity for one-vendor, one-buyer and one-vendor, multiple-buyer cases. These studies provide a foundation for joint replenishment decisions. However, they do not focus on order consolidation where the vendor can take advantage of economies of scale and reduce transportation costs. All of the above studies assume that a common order frequency must be adopted by the buyers. In this research, we loosen this assumption.

Vendor Managed Inventory (VMI) is an important concept within the integrated production-inventory replenishment literature that continues to receive the attention of researchers and practitioners. Under this strategy, the vendor assumes responsibility for managing inventories at retailers—choosing a shipment time and replenishment quantity—using advanced or online tracking systems (Çetinkaya and Lee 2000, 2002; Taleizadeh et al. 2015). The VMI arrangement allows a vendor to determine the optimal timing and quantity of replenishment to retailers without shipment constraints. Most such studies do not consider shipment consolidation as an explicit decision, so the vendor is unable to take advantage of economies of scale. Cheung and Lee (2002), however, incorporate shipment consolidation decisions into a VMI context. They study utilizing information on the retailers' inventory positions to coordinate shipments from the supplier to enjoy economies of scale in shipments, such as shipping full truckloads, but they do not consider the backorder costs of delaying shipments for retailers that are considered in the current study. Çetinkaya et al. (2006) and Mutlu and Çetinkaya (2010) develop models in a VMI context under specific consolidation policies which include a customer waiting time penalty. The above studies only incorporate inventory costs at the vendor and do not consider inventory holding costs at the buyers. The current research, on the other hand, considers a different situation in which buyers hold inventory but do so at a generally lower holding cost than the vendor.

The shipment consolidation literature is another stream of literature relevant to this study. The main motivation of shipment consolidation is to take advantage of the decreased per unit dispatch costs due to transportation economies of scale. However, shipment consolidation is usually performed at the expense of prolonged order holding at the vendor and therefore customer waiting due to delayed delivery. Thus, inventory replenishment and shipment consolidation must be optimized jointly.

The literature on shipment consolidation focuses on three types of consolidation policies: time-based policies, quantity-based policies, and time-and-quantity (TQ)-based policies as well as same-day delivery problems. We first discuss three shipment consolidation policies. A time-based policy dispatches accumulated orders every *T* periods whereas a quantity-based policy dispatches accumulated orders when a predetermined economic dispatch quantity has been reached. Under a TQ-based policy, a consolidated shipment is released either when the target load is accumulated or when the waiting time of an order exceeds a certain limit.

Time-based consolidation policies have been the focus of several studies over the last twenty years. A time-based policy was first studied by Çetinkaya and Lee (2000). They develop a renewal theoretical model in a VMI setting for the case of Poisson demands and compute the optimal replenishment quantity and dispatch frequency simultaneously. An approximate solution is derived in their study. Axsäter (2001) later provides an exact solution. Çetinkaya and Lee (2002) revisit the problem in Çetinkaya and Lee (2000). They assume a deterministic constant demand and examine a non-stationary consolidation policy under step-wise cargo costs. Later, a review on coordination of inventory and shipment consolidation decisions was studied in

Çetinkaya (2005). Moon et al. (2011) extend the results of Çetinkaya and Lee (2002) to consider multiple items and develop two joint replenishment and consolidated freight delivery policies for a third party warehouses. Howard and Marklund (2011) study a one-warehouse, multiple-retailer system with time-based shipment consolidation at the warehouse. They evaluate cost benefits of using state-dependent myopic allocation policies instead of a first-come first-serve (FCFS) policy to allocate shipments to retailers. Stenius et al. (2016) propose a model for time-based consolidation with backorder costs at retailers where allocation of orders is based on FCFS. Shipment consolidation and pricing decisions have been jointly investigated in Ülkü and Bookbinder (2012) as well as Hong and Lee (2013). Ülkü and Bookbinder (2012) propose four temporal pricing schemes and derive the corresponding optimal length of shipment consolidation cycles and the prices. Hong and Lee (2013) show that jointly deciding consolidation timing and price significantly improves the total profit compared to a time-based policy without pricing. Çetinkaya et al. (2006, 2008) consider general bulk demand processes under quantity-based policies.

TQ policies have also attracted significant research attention from researchers. Mutlu et al. (2010) is the first study to provide an exact analytical model for computing optimal parameters for the TQ-based policy (the target load q and maximum waiting time) in a pure consolidation setting without inventory costs. Multiple other studies investigate both time-based and quantity-based policies or compare the performance of all three policies. Çetinkaya and Bookbinder (2003) apply renewal theory to two quantity-based and time-based shipment consolidation policies. Chen et al. (2005) and Çetinkaya et al. (2006) present two models based on quantity-based and time-based policies for joint stock replenishment and shipment consolidation in the context of VMI. Çetinkaya et al. (2006) show that quantity-based models result in substantial cost savings compared to time-based models. They also show that TQ policies lead to better service for the retailer when service is measured by the long-run average cumulative waiting time. All of these integrated models consider private carriage which is used when shipment sizes are large enough to justify full truckloads.

Extending the previous works, Mutlu and Çetinkaya (2010) develop a model within a VMI context to find the integrated policy for inventory replenishment and outbound shipment scheduling for both time-based and quantity-based consolidation policies. They consider common carriage in which the transportation cost structure has a more complex form and charges reflect the economies of scale. Swenseth and Godfrey (2002) specifically address the practical complexities of incorporating transportation costs into inventory replenishment decisions from the carrier perspective. They motivate and show the relevance of shipping costs in the form of inverse freight rate functions. Hong and Lee (2013) show that the quantity-based policy outperforms their proposed integrated pricing and time-based policy in terms of total profit. They also show that their proposed TQ policy does not reduce service quality. Çetinkaya et al. (2014) study the performance of alternative consolidation policies based on service–based criteria and study the trade-off between expected consolidation cycle length and the average order delay. They numerically demonstrate that the TQ-based policies improve on the quantity-based policies in terms of the expected maximum waiting time. Bookbinder et al. (2011) and Cai et al. (2014) are other studies which also show that TQ-based policies improve on the quantity-based policies.

We now discuss the problem of same-day deliveries. In the same-day delivery problems, the decision maker dynamically receives order requests during the day and must decide when to dispatch a vehicle that delivers (a subset of) the accumulated orders. If a request is not fulfilled during the day, a penalty is incurred. Voccia et al. (2019) define a corresponding Markov decision model and propose a solution method that takes into account information about future requests. Arslan et al. (2016) solve a ride-sharing variant of the problem using a rolling horizon approach. Klapp et al. (2018) study a single-vehicle variant of this problem in which the customers are located on a line and route duration depends only on the customer located farthest away from the depot. These papers focus on same-day delivery and as a result, the penalty is incurred only if the order is not satisfied during the day. Our study considers penalty for backorders caused by delayed shipments at any time throughout the day. Van Heeswijk et al. (2019) study the delivery dispatching problem with time windows. This paper addresses the dispatching problem with dispatch time windows faced by an urban consolidation center. The center receives orders according to a stochastic arrival process and dispatches them in batches for the last-mile distribution. In their study, every order must be shipped within its dispatch window. Our problem considers order schedules for each distribution center characterized by the default order quantity and timing. Unlike their study, our work explores accumulating orders which will make changes to the default order schedules.

This study builds on the shipment consolidation ideas that have been explored in the literature and proposes a model which is novel in at least two ways. First, we consider the impact of consolidation on the inventory levels at both the manufacturer and downstream distribution centers. Second, we do not assume a particular kind of shipment consolidation policy. Rather, we consider the problem of constructing a detailed consolidation schedule over a (cyclic) planning horizon. To our knowledge, this is the first study to propose a scheduling approach for shipment consolidation decisions in a two-echelon supply chain.

1.4. Problem description and illustrative example

Consider a single-item two-echelon supply chain consisting of a manufacturer and D downstream distribution centers (DCs). The item (i.e. product) is infinitely divisible, and time is discretized into periods. Each DC demands from the manufacturer a fixed quantity of the item at a specific frequency. In particular, DC d demands exactly Q_d units of the product (Q_d is a real number > 0) every F_d periods (F_d is an integer ≥ 2). The item is continuously consumed at each DC at a constant, deterministic rate that equals the long-run rate at which the DC demands items from the manufacturer. In other words, the item is consumed at DC d at the constant rate of Q_d/F_d units per period. Items that have been received by a DC but not consumed are held as inventory at the DC, and the *default* inventory diagram for each DC—*if there is no consolidation of shipments and each DC is replenished individually according to its preferred timing and quantity*—is a perfect sawtooth curve whose repeating pattern consists of a vertical rise followed by negative sloping line segment down to zero.

The manufacturer continuously produces the item at a constant, deterministic rate that is just enough to keep pace with the combined consumption at the DCs. This production rate cannot be increased or decreased. Items are transported from the manufacturer to the DCs via trucks whose travel times are negligible compared to the duration of a period. The trucks are assumed to be large enough (or the items small enough) so that there is no limit on the number of items that can be transported in, or the number of DCs that can be served by, one truck. Items that have been produced but not yet sent to a DC are held as inventory at the manufacturer, and the manufacturer's *default* inventory diagram—*if there is no consolidation of shipments from the manufacturer to the DCs*—is an irregular sawtooth curve containing two types of line segments: positive sloping line segments of equal slope (corresponding to production) and vertical drops (corresponding to truck departures from the manufacturer). The operations repeat every T periods where T = LCM($F_1, F_2, \ldots, F_D$). Hence, the inventory status at the manufacturer and each DC repeats every T periods.

Four types of cost are considered: (1) transportation costs, (2) manufacturer inventory holding costs, (3) DC inventory holding costs, and (4) DC backorder costs. Regarding category 1, CT is the fixed cost in dollars of one truck departure from the manufacturer (to one or more DCs). Regarding categories 2 and 3, CHM (CH_d) is the manufacturer's (DC d's) cost in dollars of holding one unit of inventory per period. Regarding category 4, CB_d is DC d's cost in dollars of backordering one unit of inventory per period, i.e. the cost per unit of negative inventory held at DC d per period.

We make two additional assumptions. First, DC inventory cannot be replenished earlier than demanded, but it can be replenished later than demanded. This assumption is appropriate in cases of limited DC personnel and/or high transportation costs which make it logistically or economically impossible to replenish DCs more frequently (on average) than indicated by the F_d parameters. Second, the inventory at *all* DCs is *fully* replenished—to a level equivalent to that in each DC's *default inventory diagram*—whenever a truck departs from the manufacturer. In other words, the inventory status at every DC and the manufacturer reverts to the *default scenario* whenever a truck departs from the manufacturer; partial restoration of inventory stocks at one or more DCs is not allowed. For reasonable cost structures, this assumption logically follows from the (a) infinite truck capacity and (b) fixed truck cost assumptions above which make it economically foolish to partially restore inventory stocks at one or more DCs when a truck departs from the manufacturer. The second assumption above means that DC d is always resupplied in amounts that are integer multiples of Q_d. This could be an appropriate assumption—independent of (a) and (b) above—if DC d only receives items in containers of size Q_d.

The objective is to identify a shipment consolidation schedule for the T-period cycle—that is, to decide the real-valued times when trucks should depart from the manufacturer—so the total cost incurred by all parties during a cycle is minimized.

Figure 1 illustrates an instance of this problem in which $D = 2$, $Q_1 = 7$, $F_1 = 5$, $Q_2 = 9$, and $F_2 = 3$. In this scenario, DC 1 demands a *replenishment* of size 7 every five periods, and DC 2 demands a *replenishment* of size 9 every three periods. Also, $CT = 200$, $CHM = 10$, $CH_1 = 9$, $CH_2 = 10$, $CB_1 = 20$, and $CB_2 = 30$. Note that $T = \text{LCM}(F_1, F_2) = \text{LCM}(5, 3) = 15$ in this instance. We use the term *replenishment* to refer to a unique instance during the cycle in which the inventory at a particular DC is re-stocked in the *default scenario* (in which there is no consolidation of shipments). For example, there are eight replenishments in the instance shown in Figure 1—three to DC 1 (at default times 5, 10, and 15) and five to DC 2 (at default times 3, 6, 9, 12, and 15).

Figure 1 shows inventory plots for the two DCs and the manufacturer in the default scenario in which the manufacturer replenishes the DCs exactly as demanded with no shipment consolidation. In other words, the manufacturer sends a shipment of size 7 to DC 1 every five periods and a shipment of size 9 to DC 2 every three periods. Figures 1a and 1b show the resulting inventory diagrams for the DCs which are perfect sawtooth curves. Note that DC 1 (2) consumes a total of 21 (45) units of the product during the 15-period cycle. Figure 1c shows the inventory curve for the manufacturer who produces the item at the rate of 66 units every 15 periods which is just enough to keep pace with the consumption at the DCs. In other words, the manufacturer produces the item at the constant rate of 66/15 units per period which is the slope of every positive sloping line segment in its inventory diagram.

Figure 1. Default solution to illustrative instance

(x-axis measures time; y-axis measures inventory quantity)

Figure 1a: DC 1 Inventory Status

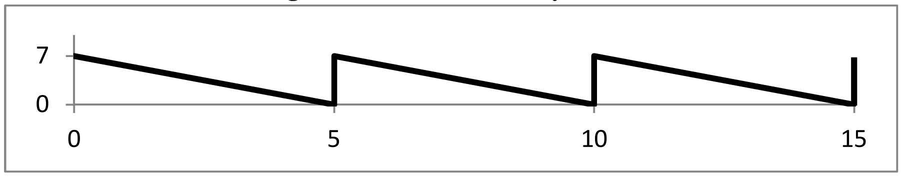


Figure 1b: DC 2 Inventory Status

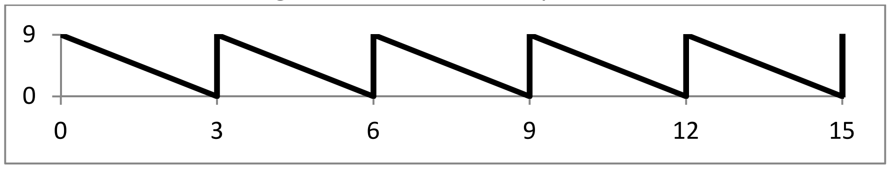
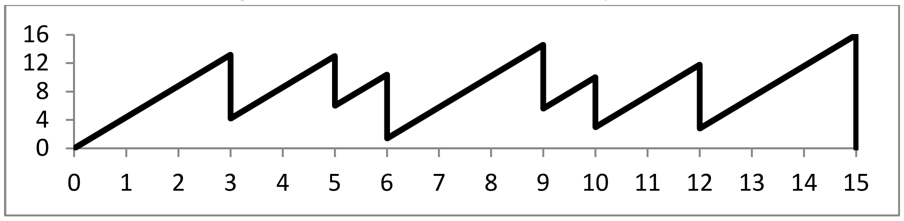


Figure 1c: Manufacturer Inventory Status



The total cost of the operation shown in Figure 1 is \$3747.5 per cycle. This consists of \$472.5 in inventory holding costs at DC 1; \$0 in backorder costs at DC 1; \$675 in inventory holding costs at DC 2; \$0 in backorder costs at DC 2; \$1200 in inventory holding costs at the manufacturer; and \$1400 in transportation costs per cycle. The \$472.5 equals \$9 times the area under DC 1's inventory diagram; \$675 equals \$10 times the area under DC 2's inventory diagram; \$1200 equals \$10 times the area under the manufacturer's inventory diagram; and \$1400 equals the cost per truck (= \$200) times the number of truck departures from the manufacturer—i.e. the number of vertical drops in the manufacturer's inventory diagram—during the cycle (= 7).

Figure 2 shows inventory plots for the two DCs and the manufacturer in an alternate scenario in which there is some shipment consolidation. Here, DC 1's first replenishment—originally scheduled at time 5—is delayed by one period so it can be consolidated with (i.e. sent in the same truck as) DC 2's second replenishment at time 6. Also, DC 2's third replenishment—originally scheduled at time 9—is delayed by one period so it can be consolidated with DC 1's second replenishment at time 10. The consolidation reduces transportation and DC inventory holding costs but increases manufacturer inventory holding and DC backorder costs. In particular, transportation costs decrease compared to Figure 1 because two fewer trucks depart the manufacturer per cycle. Also, DC inventory holding costs (i.e. the area above the x-axis and below the solid DC inventory curves) decrease compared to Figure 1. However, the manufacturer's inventory holding cost increases, and DC backorder costs (i.e. the area below the x-axis and above the solid DC inventory curves) increase. The total cost of the operation shown in Figure 2 is $3434.8 per cycle. In Figure 2, dashed lines indicate the default solution from Figure 1, and shaded regions indicate DC backorder costs.

Figure 2. Alternate solution #1 to illustrative instance

Dashed lines indicate default solution (Figure 1) and shaded regions indicate DC backorders.

Figure 2a: DC 1 Inventory Status

Figure 2b: DC 2 Inventory Status

Figure 2c: Manufacturer Inventory Status

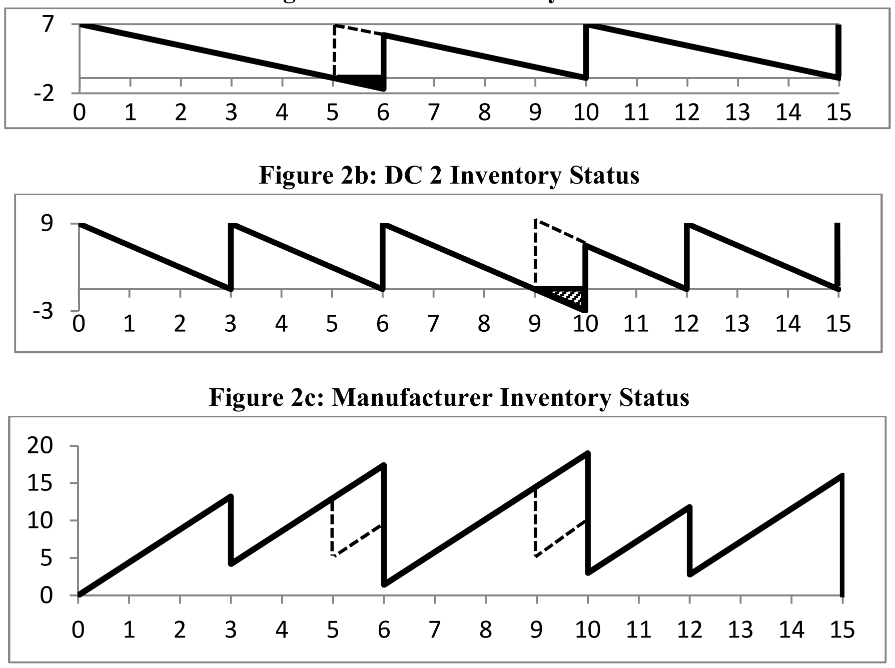


Figure 3 shows a second alternative in which there is even more shipment consolidation. Here, DC 1's first, DC 2's second, and DC 2's third replenishment—originally scheduled at time 5, 6, and 9 respectively—are delayed and consolidated with DC 1's second replenishment at time 10. Also, DC 2's fourth replenishment—originally scheduled at time 12—is delayed by three periods to be consolidated with DC 2's final replenishment at time 15. In this scenario, only three trucks depart the manufacturer per cycle. Also, DC inventory holding costs are much lower than in

Figures 1-2. However, the manufacturer's inventory holding cost and DC backorder costs are much higher than in Figures 1-2. The total cost of the operation in Figure 3 is $4990 per cycle. Overall, the scenario depicted in Figure 2—with a modest amount but not too much shipment consolidation—achieves the lowest total cost per cycle among the three alternatives. The preceding discussion indicates that the problem under investigation is not trivial and requires a sophisticated solution approach.

Figure 3. Alternate solution #2 to illustrative instance

Dashed lines indicate default solution (Figure 1) and shaded regions indicate DC backorders.

Figure 3a: DC 1 Inventory Status

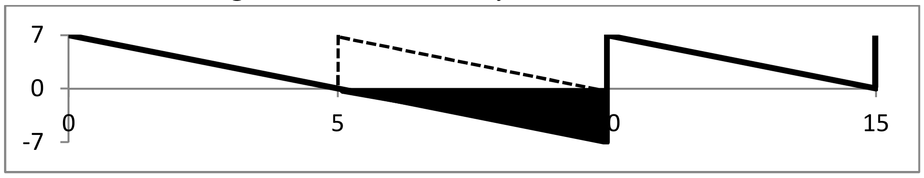


Figure 3b: DC 2 Inventory Status

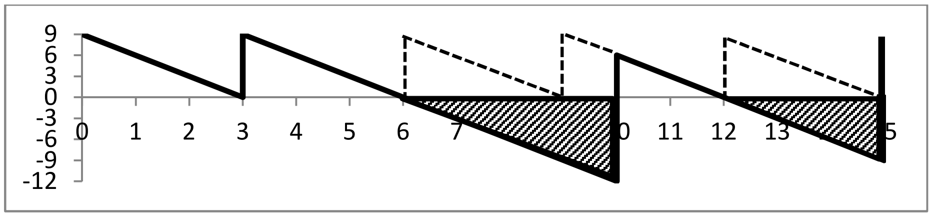
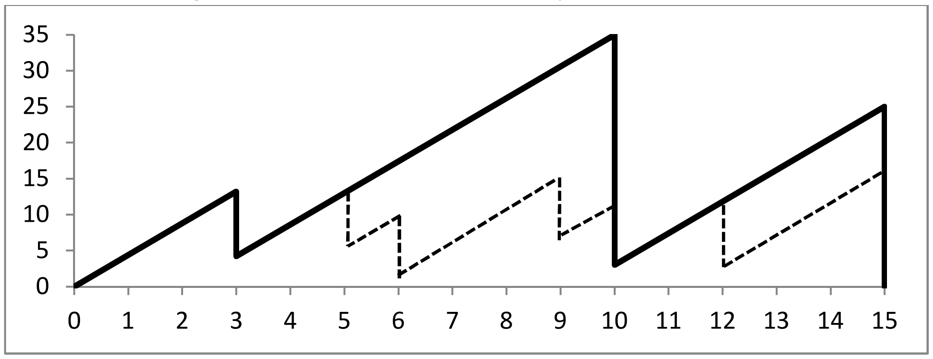


Figure 3c: Manufacturer Inventory Status



1.5. Mathematical formulation

A mixed-integer quadratic programming (MIQP) formulation of the problem (Math Model 1) is shown in Tables 1-2. The primary input parameters shown at the top of Table 1 are used to derive the values of the secondary input parameters listed in the middle of Table 1. Decision variables are shown at the bottom of Table 1. The model makes three important assumptions. First, the manufacturer's production rate is fixed and cannot be changed. Second, DC inventory cannot be replenished earlier than demanded, but it can be replenished later than demanded. Third, the inventory at all DCs is fully replenished whenever a truck departs from the manufacturer. In other words, the inventory status at every DC and the manufacturer reverts to the *default scenario* (see Section 1.4) whenever a truck departs from the manufacturer. For example, at time 10 in Figure 3, the inventory status at the manufacturer and all DCs is restored to the default scenario shown in Figure 1; partial restoration of inventory stocks at one or more DCs is not allowed.

Table 1. Indices, parameters, and decision variables in Math Model 1

Indices	
d	Distribution center (DC) (d = 1 to D)
r	Replenishment (r = 1 to R)
c	Consolidation opportunity (c = 1 to R)
Primary Input Parameters	
D	Number of DCs (integer, ≥ 2)
F_d	Default frequency at which DC d is restocked (integer, ≥ 2)
Q_d	Default quantity sent to DC d when it is restocked (real, > 0)
CT	Cost in dollars of sending one truck from the manufacturer to restock one or more DCs (real, ≥ 0)
CHM	Cost in dollars of holding one unit of inventory per period at manufacturer (real, ≥ 0)
CH_d	Cost in dollars of holding one unit of inventory per period at DC d (real, ≥ 0)
CB_d	Cost in dollars of backordering one unit of inventory per period at DC d (real, ≥ 0)
Secondary Input Parameters	
T	Cycle length for the entire operation, (T=LCM($F_1, F_2, \ldots, F_D$)) (integer, > 0)
R_d	Number of DC d replenishments in each cycle $\left(R_d = \frac{T}{F_d}\right)$ (integer, > 0)
R	Total number of individual replenishments made per cycle. Replenishments are ordered according to DC then according to time from 0 to T $\left(R = \sum_d R_d\right)$ (integer, > 0)
DT_r	Default time when replenishment r occurs if there is no consolidation (integer, > 0)
I_{rd}	= 1 if replenishment r is for DC d; = 0 otherwise (binary)
$Size_r$	Size of (i.e. quantity associated with) replenishment r $\left(Size_r = \sum_d I_{rd} Q_d\right)$ (real, > 0)
$TotalQ$	Total quantity produced in one cycle ($TotalQ = \sum_d R_d\, Q_d$) (real, > 0)
P	Production rate (per period) at manufacturer ($P = TotalQ/T$) (real, > 0)
Decision Variables	
W_c	End of time window for consolidation opportunity (CO) c (real, ≥ 0, ≤ T)
$Time_c$	Departure time of (real or fictitious) truck for CO c (real, ≥ 0, ≤ T)
$Y_{rc} = \begin{cases} 1 \\ 0 \end{cases}$	If replenishment r is assigned to CO c Otherwise (binary)
T_r	Time when replenishment r occurs (real, ≥ 0, ≤ T)
L_r	Lateness of replenishment r (due to consolidation with other replenishments) (real, ≥ 0)
$Z_c = \begin{cases} 1 \\ 0 \end{cases}$	If at least one replenishment is assigned to CO c (i.e. if CO c is real and requires a truck departure) Otherwise (binary)
G_{dc}	Greatest delay of a DC d replenishment that is assigned to CO c. This equals zero if no DC d replenishment is assigned to CO c (real, ≥ 0)
$DelayFull_{dc}$	Maximum number of full DC d inventory cycles worth of delay for a DC d replenishment that is assigned to CO c. This equals zero if no DC d replenishment is assigned to CO c (integer, ≥ 0)
$DelayFrac_{dc}$	Smallest lateness among all DC d replenishments that are assigned to CO c. This equals zero if no DC d replenishment is assigned to CO c (real, ≥ 0)
$X_{rdc} = \begin{cases} 1 \\ 0 \end{cases}$	If replenishment r is the most delayed DC d replenishment that is assigned to CO c If replenishment r is not for DC d, r is not assigned to CO c, or r is not the most delayed DC d replenishment that is assigned to CO c (binary)

1.5.1. Input parameters and decision variables

Three primary input parameters define the system design and operations. Parameter D is the number of DCs that receive inventory from the manufacturer. Parameters F_d and Q_d are the frequency and quantity associated with DC d's demand process (Figure 1). They are also the frequency and quantity with which DC d is restocked *in the default scenario* (in which there is no shipment consolidation). Four primary input parameters are cost related. Parameter CT is the cost per truck that departs from the manufacturer. Parameter CHM (CH_d) is the manufacturer's (DC d's) cost of holding one unit of inventory per period. Parameter CB_d is DC d's cost of backordering one unit of inventory per period.

The secondary input parameters are as follows. Parameter T is the operational cycle length which equals LCM($F_1, F_2, \ldots, F_D$). Parameter R_d is the number of replenishments made to DC d in each cycle. For example, (R_1, R_2) = (3, 5) in Figures 1-3. Parameter R is the total number of unique replenishments made to all DCs during the cycle. Replenishments are indexed first according to DC then according to time from 0 to T. Parameter DT_r is the default time when replenishment r occurs (if there is no consolidation). Parameter I_{rd} indicates if replenishment r is for DC d or not. Parameter $Size_r$ is the size of (i.e. quantity associated with) replenishment r. For example, in Figures 1-3, $R = 8$ and replenishments (1, 2, 3, 4, 5, 6, 7, 8) are for DC (1, 1, 1, 2, 2, 2, 2, 2) and have default times (5, 10, 15, 3, 6, 9, 12, 15) and sizes (7, 7, 7, 9, 9, 9, 9, 9) respectively. Parameter $TotalQ$ is the total quantity produced by the manufacturer—which equals the total quantity consumed at all DCs—during one cycle. This equals 66 in Figures 1-3. Finally, P is the manufacturer's constant production rate. This equals 66/15 in Figures 1-3. Note that the values of all secondary input parameters are derived from the primary input parameters.

Ten decision variables give the decision maker flexibility in deciding when the R replenishments take place. The most fundamental decision variable, T_r, is the time when replenishment r occurs. This variable is real-valued and ranges from 0 to T. In the model, we exhaustively divide the [0,T] time horizon into R discrete, non-overlapping *time intervals* such that *at most one truck departs from the manufacturer during each interval*. Real-valued decision variables W_c (c = 1 to R) determine the upper endpoints of these R intervals; 0 is the lower endpoint of the first interval; and variables W_c (c = 1 to R-1) determine the lower endpoints of intervals 2 to R. We say there is one *consolidation opportunity* (i.e. CO) per interval. In other words, we assume that there is one opportunity for a truck to depart the manufacturer in each interval. Thus, there are R consolidation opportunities during the cycle.

Attached to each CO is an instant of time—$Time_c$—when zero, one, or several of the aforementioned replenishments may simultaneously occur (i.e. when a truck may depart from the manufacturer). No other trucks may depart from the manufacturer during the time window corresponding to CO c. Binary decision variable Y_{rc} indicates which replenishment r (r = 1 to R) is assigned to which CO c (c = 1 to R). Binary variable Z_c tracks whether at least one replenishment is assigned to CO c. If this equals 1, an extra truck is dispatched at a cost of CT; otherwise no extra truck is dispatched. Variable L_r is the lateness of replenishment r. This variable equals zero if replenishment r is made at its default time DT_r.

As mentioned above, the model assumes that the inventory status at every DC and the manufacturer reverts to the default scenario whenever a truck departs from the manufacturer, i.e. whenever a CO is utilized and variable Z_c = 1. The four decision variables G_{dc}, $DelayFull_{dc}$, $DelayFrac_{dc}$, and X_{rdc} keep track of the DCs and replenishments involved in each CO; they help to evaluate the cost of consolidation in the model's objective function. Variable G_{dc} is the greatest delay of a DC d replenishment that is assigned to CO c. This equals $F_d*DelayFull_{dc}$ + $DelayFrac_{dc}$, where $DelayFull_{dc}$ ("full" meaning "full DC inventory cycle") is the integer portion of G_{dc}/F_d and $DelayFrac_{dc}$ ("frac" meaning "fraction of a DC inventory cycle") is the remainder when G_{dc} is divided by F_d. Variables G_{dc}, $DelayFull_{dc}$, and $DelayFrac_{dc}$ equal zero if no DC d replenishment is assigned to CO c. Binary variable X_{rdc} equals 1 if replenishment r is the most delayed DC d replenishment that is assigned to CO c.

Consider the solution shown in Figure 3. The values of the decision variables for this solution are as follows. The value of T_r is (10, 10, 15, 3, 10, 10, 15, 15) and L_r is (5, 0, 0, 0, 4, 1, 3, 0) for r = (1, 2, 3, 4, 5, 6, 7, 8) respectively. There are eight $Time_c$ variables and three of them—say $Time_1$, $Time_5$, and $Time_8$—equal 3, 10, and 15. The other five may take any values as long as $Time_{c-1} < Time_c$ for all c from 2 to R (see constraints 2-4 in the math model). Eight of the 64 Y_{rc} variables—namely Y_{15}, Y_{25}, Y_{38}, Y_{41}, Y_{55}, Y_{65}, Y_{78}, Y_{88}—equal 1; the others equal zero. Three Z_c variables—namely Z_1, Z_5, Z_8—equal 1; the others equal zero. Three of the 16 G_{dc} variables—namely G_{15}, G_{25}, G_{28}—equal 5, 4, and 3 respectively and the others equal zero; these variables assist in computing backorder costs. Variables ($DelayFull_{15}$, $DelayFull_{25}$, $DelayFull_{28}$) = (1, 1, 1) respectively; they are the integer portion of (5, 4, 3) divided by (5, 3, 3) respectively. In other words, the three backordering situations in Figure 3 exist for at least one but less than two full DC inventory cycles. Variables ($DelayFrac_{15}$, $DelayFrac_{25}$, $DelayFrac_{28}$) = (0, 1, 0) respectively; they are the remainder when (5, 4, 3) is divided by (5, 3, 3) respectively. Variables $DelayFull_{dc}$ and $DelayFrac_{dc}$ are used to compute the reduction in DC inventory holding costs compared to the default scenario shown in Figure 1. Five of the 128 X_{rdc} variables—namely X_{115}, X_{318}, X_{421}, X_{525}, X_{728}—equal 1; the others equal zero.

1.5.2. Objective function and constraints

The mathematical model is shown in Table 2. The five parts of the objective function consider (1) transportation, (2) DC backorder, (3) manufacturer inventory holding, and (4-5) DC inventory holding costs respectively. Part 1 is a basic multiplication of CT times the number of Z_c variables that equal 1. In part 2, variable G_{dc}—the greatest delay of a DC d replenishment that's assigned to CO c—is used to compute the backorder cost for each possible combination of d and c. Note that G_{dc} equals zero if no DC d replenishment is assigned to CO c. Note in Figures 2-3 that backorder cost equals CB_d times the area of a triangle that exists below the x-axis in DC d's inventory diagram. The base of the triangle is G_{dc}; its height is $(G_{dc})[(Q_d)/(F_d)]$; and division by two gives the area of the triangle.

Part 3 equals CHM multiplied by the area under the manufacturer's inventory diagram. This latter quantity equals (A) the area under the inventory diagram *in the default scenario* (Figure 1) plus (B) any additional area that exists due to delayed replenishments. For example, in Figure 3c, this equals (A) the area below the dashed curve plus (B) the area above the dashed curve and below the solid curve. Item A equals the sum of $Q_d*T/2$ over all d, and item B equals the sum of $I_{rd}*L_r*Q_d$ over all r and d which is the total area of all parallelograms into which the region above the dashed curve and below the solid curve can be divided. The formula for item A is obtained by recognizing that the area under the manufacturer's inventory diagram in the default scenario (Figure 1) equals the total area under all DC inventory diagrams in this scenario.

Part 4 computes total DC inventory holding cost in the default scenario; this equals CH_d times the duration of the cycle T times the average height of DC d's inventory diagram during the cycle—$Q_d/2$—summed over all d. Part 5 is the reduction in DC inventory holding costs due to consolidation. Note that the quantity in large square brackets is zero if $DelayFull_{dc}$ = $DelayFrac_{dc}$ = 0. Otherwise, it should equal the total area below the dashed curve and above the x-axis in DC *d*'s inventory diagram. For example, in Figure 3b, this equals the area of three triangles with base 3 and height 9 minus the area of one triangle with base 2 and height 6. Roughly speaking, the first term in square brackets is the total area of the larger (3 by 9) triangles, and the second term is the total area of the smaller (2 by 6) triangles. Note that parts 2 and 5 of the objective function are quadratic.

Constraint (2) establishes the W_c variables as markers for the ends of the *R* time intervals into which the *T*-period cycle is divided. Constraints (3-4) ensure there is exactly one CO during each time interval, i.e. exactly one time $Time_c$ during each time interval when a truck is allowed to depart from the manufacturer. Constraint (5) ensures that each replenishment is assigned to exactly one of the aforementioned time intervals (i.e. COs). If replenishment *r* is assigned to CO *c* (i.e. if $Y_{rc} = 1$), constraints (6-7) ensure that the time T_r when replenishment *r* occurs equals the time $Time_c$ when the truck connected to CO *c* departs the manufacturer; otherwise these constraints have no effect. Constraint (8) ensures that replenishments take place no earlier than their default times. Constraint (9) enforces the proper relationship among variables L_r and T_r. Constraint (10) requires that $Z_c = 1$ if Y_{rc} equals 1. In other words, a real truck departure is connected to CO *c* if at least one replenishment is assigned to CO *c*.

Constraints (11-16) ensure that G_{dc} is properly computed. Constraint (11) ensures that G_{dc} is at least L_r if replenishment *r* is for DC *d* and replenishment *r* is assigned to CO *c*. Constraint (12) forces at least one variable $X_{r'dc}$ (r' = 1 to *R*) to equal one if both I_{rd} and Y_{rc} are one. Note that, when both I_{rd} and Y_{rc} are one, at least one replenishment—namely *r*—is for DC *d* and is assigned to CO *c*. In this case the constraint guarantees that at least one replenishment *r'* is identified as the most delayed DC *d* replenishment that is assigned to CO *c*. Constraints (13-14) force variable X_{rdc} to be zero if either I_{rd} or Y_{rc} is zero. Constraint (15) ensures that $G_{dc} = 0$ if all variables X_{rdc} (r = 1 to *R*) are zero, i.e. if there is no DC *d* replenishment that is assigned to CO *c*. Constraint (16) states that G_{dc} is at most L_r if variable X_{rdc} is one, i.e. if replenishment *r* is identified as the most delayed DC *d* replenishment that is assigned to CO *c*. Constraints (16) and (11) together ensure that (a) G_{dc} equals L_r when *r* is the most delayed DC *d* replenishment that is assigned to CO *c* and (b) $X_{rdc} = 1$ correctly identifies the most delayed DC *d* replenishment assigned to CO *c*.

Constraints (17-19) break the delay term G_{dc} into two portions: $DelayFull_{dc}$ and $DelayFrac_{dc}$. The former is the integral number of full "default DC d inventory replenishment cycles" worth of delay that exist in resupplying DC d via CO c. The latter is the additional delay, beyond that embodied in $DelayFull_{dc}$, that exists in resupplying DC d via CO c. Constraints (17-18) use G_{dc} to compute $DelayFull_{dc}$, and constraint (19) computes $DelayFrac_{dc}$ using the formula $G_{dc} = F_d{*}DelayFull_{dc} + DelayFrac_{dc}$.

Table 2. Math Model 1 for shipment consolidation problem

Objective Function

$$Minimize\ \ CT\left(\sum_{c=1}^{R} Z_c\right) + \sum_{d=1}^{D}\sum_{c=1}^{R} CB_d\left(\frac{(G_{dc})^2}{2}\right)\frac{Q_d}{F_d} + CHM\left[\sum_{d=1}^{D}\frac{Q_d}{2}T + \sum_{d=1}^{D}\sum_{r=1}^{R} I_{rd}\,L_r Q_d\right] + \sum_{d=1}^{D} CH_d\frac{Q_d}{2}T - \sum_{d=1}^{D} CH_d\left[\sum_{c=1}^{R}\left((DelayFull_{dc}+1)\frac{Q_d F_d}{2} - \left(\frac{(F_d - DelayFrac_{dc})^2}{2}\right)\frac{Q_d}{F_d}\right)\right] \quad (1)$$

Constraints

$W_{c-1} + 0.001 \le W_c$	$\forall c: 2 \le c \le R$	(2)
$W_{c-1} \le Time_c \le W_c$	$\forall c: 2 \le c \le R$	(3)
$Time_1 \le W_1$		(4)
$\sum_{c=1}^{R} Y_{rc} = 1$	$\forall r$	(5)
$T_r \le Time_c + T(1 - Y_{rc})$	$\forall r\ \forall c$	(6)
$T_r \ge Time_c - T(1 - Y_{rc})$	$\forall r\ \forall c$	(7)
$T_r \ge DT_r$	$\forall r$	(8)
$L_r = T_r - DT_r$	$\forall r$	(9)
$Z_c \ge Y_{rc}$	$\forall r\ \forall c$	(10)
$G_{dc} \ge L_r - T(2 - I_{rd} - Y_{rc})$	$\forall r\ \forall c\ \forall d$	(11)
$\sum_{r'=1}^{R} X_{r'dc} \ge I_{rd} + Y_{rc} - 1$	$\forall r\ \forall c\ \forall d$	(12)
$X_{rdc} \le I_{rd}$	$\forall r\ \forall c\ \forall d$	(13)
$X_{rdc} \le Y_{rc}$	$\forall r\ \forall c\ \forall d$	(14)
$G_{dc} \le T * \sum_{r=1}^{R} X_{rdc}$	$\forall c\ \forall d$	(15)
$G_{dc} \le L_r + T(1 - X_{rdc})$	$\forall r\ \forall c\ \forall d$	(16)
$DelayFull_{dc} \ge \frac{G_{dc}}{F_d} - 1 + 0.001$	$\forall c\ \forall d$	(17)
$DelayFull_{dc} \le \frac{G_{dc}}{F_d}$	$\forall c\ \forall d$	(18)
$DelayFrac_{dc} = G_{dc} - F_d(DelayFull_{dc})$	$\forall c\ \forall d$	(19)

1.6. Genetic algorithm

The problem at hand—modeled as a mixed integer quadratic program—appears to be very difficult, so we expect it is necessary to develop efficient heuristic algorithms for solving large instances. Our efforts in this area led to the development of a genetic algorithm (GA). A GA is a mathematical search technique based on the principles of natural selection and genetic recombination (Goldberg and Holland, 1988; Holland, 1992). For reasons that will become clear in Section 1.7.4, our GA only searches for feasible solutions in which trucks depart the manufacturer at the default replenishment times DT_r (Table 1). The main question to be decided by the GA is whether or not a truck departs from the manufacturer at each default replenishment time DT_r. Whenever a truck departs, we assume that the inventory level at every DC and the manufacturer reverts to the default scenario, i.e. that the consolidated shipment on the truck includes all shipments which have been delayed up to that point (Section 1.5). We now describe the GA in detail.

Consider the list of default replenishment times DT_r ordered from least to greatest in which duplicate values have been deleted. Let U be the number of values in this list; this is the number of unique replenishment times DT_r in the default scenario. For example, in the instance depicted in Figures 1-3, U is 7 and the list is (3, 5, 6, 9, 10, 12, 15). In our GA, each feasible solution is represented by a *chromosome* consisting of a sequence of U *genes*—(g_1, g_2, g_3, …, g_U)—where $U < R$ and g_u is the gene for the u^{th} unique replenishment time in the default scenario, i.e. the u^{th} value in the above list. Each gene takes a binary value. A gene value of zero means that no truck departs from the manufacturer at that time. A gene value of 1 indicates that a truck departs from the manufacturer at that time. For example, the solutions shown in Figures 1, 2, and 3 are represented by the chromosomes (1, 1, 1, 1, 1, 1, 1), (1, 0, 1, 0, 1, 1, 1), and (1, 0, 0, 0, 1, 0, 1) respectively.

The GA procedure is shown in Table 3. The initial generation consists of N unique chromosomes (i.e. solutions). Each chromosome is constructed by randomly assigning 0 or 1 (with 50%/50% chance) to each of the first U-1 genes. The value 1 is always assigned to the U^{th} gene (corresponding to a truck departure at time T). The fitness (i.e. objective value) of each solution is then evaluated by (a) computing $Time_c$, Y_{rc}, T_r, L_r, and Z_c for the solution; (b) using constraints

(11-19) in the math model to quickly compute G_{dc}, $DelayFull_{dc}$, and $DelayFrac_{dc}$ for the solution; and (c) plugging these values into the math model's objective function (Tables 1-2).

Table 3. Genetic algorithm procedure

```
Let $U$ = number of unique replenishment times $DT_r$ in the default scenario. These are the possible truck departure
    times considered in the GA.
Each chromosome is a sequence of $U$ binary values (i.e. genes). A value of 1 (0) for gene $g$ means that a truck
    departs (does not depart) from the manufacturer (for one or more DCs) at the $g^{th}$ unique $DT_r$ value.
Decide value for $N$, $N_0$, $N_1$, and $N_2$ ($N = N_0+N_1+N_2$).
$N$ = Number of chromosomes per generation.
$N_0$ = Number of chromosomes created by copying.
$N_1$ = Number of chromosomes created by mutation.
$N_2$ = Number of chromosomes created by crossover.
Initial generation consists of $N$ unique chromosomes. Each chromosome is constructed by randomly assigning 0 or
    1 (with 50%/50% chance) to each of the first $U$-1 genes. The value 1 is always assigned to the $U^{th}$ gene
    (corresponding to a truck departure at time $T$).
While time elapsed < $TimeLimit$
    Rank the $N$ chromosomes in the current generation according to total cost from best to worst.
    Copy the $N_0$ best chromosomes into the next generation.
    For $i = 1, \ldots, N_1$
        Let $Unique$ = False.
        While $Unique$ = False
            Randomly select a chromosome in the current generation.
            For $g = 1, \ldots, U-1$
                If U(0,1) random variable $\leq MProb$ Then
                    Change value of gene $g$ to the opposite binary value.
                End If
            End For
            If new chromosome is not identical to any chromosome in the next generation Then
                $Unique$ = True.
                Add new chromosome to next generation.
            End If
        End While
    End For
    For $i = 1, \ldots, N_2/2$
        Let $Unique$ = False.
        While $Unique$ = False
            Select two unique parents to participate in a crossover that will generate two children.
                Parent selection probability is proportional to parent's objective value ranking in
                current generation ($N$ is best chromosome).
            Randomly select one of the $U$-2 possible crossover points which exist between the first
                $U$-1 genes.
            Generate two children by swapping the genes of the two parents after the crossover
                point.
            If two children are not identical to any chromosome in the next generation Then
                $Unique$ = True.
                Add two children chromosomes to next generation.
            End If
        End While
    End For
    Copy all chromosomes in next generation into the current generation.
End While
Report the chromosome with the minimum cost among all chromosomes that were generated.
```

The next generation of N chromosomes is formed as follows. First, the N_0 best chromosomes in the current generation are copied into the next generation. Next, N_1 mutated chromosomes are added to the next generation. Each mutated chromosome is formed by selecting a random solution in the current generation and then, for each g from 1 to U-1, setting the value of gene g to the opposite binary value with probability *MProb*. Then, parent chromosomes from the current generation are mated, and a total of N_2 ($= N - N_0 - N_1$) children are added to the next generation (N_2 must be even). In each crossover operation, two parent chromosomes ($g1_1$, $g1_2$, $g1_3$, …, $g1_U$) and ($g2_1$, $g2_2$, $g2_3$, …, $g2_U$) are mated—forming two children—by performing a crossover operation at a random position p ($1 \leq p \leq U$-2) in the parent chromosomes. Random variable p follows a discrete uniform distribution with minimum 1 and maximum U-2. The resulting children are ($g1_1$, …, $g1_p$, $g2_{p+1}$, …, $g2_U$) and ($g2_1$, …, $g2_p$, $g1_{p+1}$, …, $g1_U$). Each parent's selection probability is proportional to its fitness ranking in current generation where the chromosome in the current generation with the lowest (highest) cost has ranking N (1). Ties are broken randomly. Unless it is impossible to do so (i.e. unless fewer than N unique chromosomes exist), we require that every chromosome in a given generation be unique. When a predefined time limit is reached, the GA procedure terminates and the best chromosome that was encountered is displayed.

1.7. Experimental setup, results, and discussion

The math model from Section 1.5 and GA from Section 1.6 were coded into MS Visual C++ 2017 Professional. IBM ILOG Concert Technology was used to code the math model in C++ and call the MIQP solver IBM ILOG CPLEX 12.9 to solve instances defined in text files. All results are obtained using a desktop computer with 16 gigabytes of RAM, the Windows 10 Education 64-bit operating system, and an Intel Core i7-8700 processor with twelve, 3.2 gigahertz logical processors.

1.7.1. Generating problem instances

Table 4 shows the parameter values that define the problem instances considered in our experiments. Four problem sizes are considered: small, medium, large, and very large. The size of an instance is defined by U = the number of unique default replenishment times DT_r. Note that U-1 is the number of yes/no decisions made regarding truck departures in the GA (Section 1.6).

Thus, U is a good measure of problem instance difficulty. As shown in Table 4, the number of DCs ranges from two to five. The values of the primary input parameters fall within the ranges shown. The parameter values satisfy $CHM \geq CH_d$ for all d in all instances considered in experiment 2 (see below). Text files defining all problem instances can be found in the supplementary material accompanying this paper. Parameter (D, CT, CH_d, CHM, CB_d, F_d, Q_d, T) is shown in row (1, 2, 3, 4, 5, 6, 7, 8) of each file.

Table 4. Problem instances considered in the experiments

	Instance Size			
	Small (S)	Medium (M)	Large (L)	Very Large (V)
U = Number of unique default replenishment times DT_r	4 - 7	8 - 25	26 - 59	60 - 150
D = Number of DCs	2	2 - 3	3 - 4	3 - 5
CT	100 - 500			
CH_d	1 - 9			
CHM	10 - 20			
CB_d	10 - 30			
F_d	2 - 9			
Q_d	2 - 9			

1.7.2. GA and CPLEX settings

Table 5 shows the GA parameter settings used in the experiments. These settings were chosen based on preliminary experiments whose results are not shown here. Note that more computation time is allocated for attacking larger problems. Each generation has 100 chromosomes: 10 copied from the previous generation, 40 formed by mutation, and 50 formed by the crossover operation. The gene mutation probability, *MProb*, is 0.6. The CPLEX computation time limit is one hour for all instances.

Table 5. GA settings used in the experiments

GA Parameter	Description	Value Used
N	Number of chromosomes per generation	100
N_0	Number of chromosomes copied from current generation into next generation	10
N_1	Number of chromosomes in each generation formed by mutation	40
N_2	Number of chromosomes in each generation formed by crossover	50
MProb	Mutation probability for each gene	0.6
TimeLimit	Computation time limit (in seconds)	10 for small 60 for medium 600 for large 600 for very large instances

1.7.3. Experiment 1 setup and results

In this experiment we consider three *base instances* and two *variations* of each base instance: one in which the manufacturer's inventory holding cost is greater than or equal to the inventory holding cost at each DC ($CHM \geq CH_d$ for all d) and one in which the manufacturer's inventory holding cost is less than the inventory holding cost at one or more DCs ($CHM < CH_d$ for some d). The second variation has the same input parameter values as the first variation except that the inventory holding cost at each DC is multiplied by 2.

Table 6 shows the instances considered and results for Experiment 1. Column 1 shows the instance number. Columns 2-3 show the main parameters defining each instance. Column 4 shows how instance X.2 is obtained from instance X.1. Column 5 shows whether $CHM \geq CH_d$ for all d; this inequality holds for the first variation of each base instance ("Y" = Yes) but not the second variation of each base instance ("N" = No). Columns 6-10 show the results for CPLEX including the objective value of the best solution identified within the time limit ("Total CPLEX cost"); whether the best solution was proven to be optimal or not; number of trucks departing the manufacturer per cycle in the best solution; whether one or more truck departures occur at fractional (i.e. non-integral) times in the best solution; and the total computation time. Columns

11-12 show results for the GA including the objective value of the best solution identified within the time limit ("Total GA cost") and total computation time. It is important to note that Figures 1-3 are based on instance S2.1, and Figure 2 shows an optimal solution for instance S2.1.

Table 6. Experiment 1 results

Instance	# DCs	Number of unique default replenishment times (DT_r)	Change	$CHM \geq CH_d$ for all d?	Total CPLEX cost	Optimal?	# Trucks	Trucks depart at fractional times?	Time (sec)	Total GA cost	Time (sec)
S1.1	2	6	-	Y	3019.8	Y	3	N	1	3019.8	10
S1.2			CH_d doubled	N	3524.64	Y	3	Y	2	3529.2	10
S2.1	2	7	-	Y	3434.8	Y	5	N	8	3434.8	10
S2.2			CH_d doubled	N	4375.41	Y	5	Y	18	4450.6	10
S3.1	3	6	-	Y	6152	Y	5	N	6	6152	10
S3.2			CH_d doubled	N	8009.48	Y	5	Y	13	8057	10

1.7.4. Experiment 1 discussion

As the results demonstrate, CPLEX solves all instances to optimality within 20 seconds. Column 9 shows that the optimal replenishment times (i.e. truck departure times) are always integral when $CHM \geq CH_d$ for all d. This is reasonable. Indeed, slightly postponing a shipment (e.g. by a fraction of a period) is not beneficial in this case because the increase in the manufacturer's inventory holding cost outweighs the reduction in the DCs' inventory holding costs. On the other hand, at least one of the optimal truck departure times is fractional in instances S1.2, S2.2, and S3.2 (i.e. when $CHM < CH_d$ for some d). This result is also reasonable. Here, it is cheaper to store inventory at the manufacturer than at DC d for at least one d, so there may be a financial incentive to slightly delay a replenishment to DC d in order to reduce overall inventory holding costs for as long as possible until the DC d backorder cost—visualized as a triangle that grows over time and initially has negligible area—becomes prohibitively large. For example, trucks depart at integer times 3, 6, 10, 12, and 15 in the optimal solution to instance S2.1 (Figure 2), but they depart at fractional times 3.6, 6.457, 10, 12.6, and 15 in the optimal solution to instance S2.2.

Note that the best GA solution is identical to (different than) the optimal solution obtained by CPLEX when $CHM \geq CH_d$ for all d ($CHM < CH_d$ for some d). This is reasonable given that the GA only searches among solutions with replenishment times equal to DT_r for some r (Section 1.6). Given these findings, we shall require that $CHM \geq CH_d$ for all d in the next experiment in order to eliminate the advantage of, and therefore any possibility of, fractional replenishment times.

1.7.5. Experiment 2 setup and results

In this experiment we consider a total of 84 problem instances—twelve base instances and seven variations of each base instance. Three base instances are considered for each problem size shown in Table 4. Each base instance has a main variation and six other variations. Table 7 shows how the six other variations relate to the main variation. In general, the values of the primary input parameters in each of the six other variations are identical to those in the main variation except that one kind of cost parameter is modified. This experimental design helps us identify the impact of individual cost coefficients on the optimal consolidation schedule, which in turn helps to verify the correctness of the math model and reasonableness of the results. In all instances $CHM \geq CH_d$ for all d. Also, $CHM \leq CB_d$ for all d in all instances except those associated with variations X.4 and X.7.

Table 7. Variations considered in experiment 2

Variation	Change compared to variation X.1
X.1	None
X.2	CT is multiplied by 2
X.3	CT is divided by 2
X.4	CHM and CH_d (for all d) are multiplied by 2
X.5	CHM and CH_d (for all d) are divided by 2
X.6	CB_d is multiplied by 2 for all d
X.7	CB_d is divided by 2 for all d

Table 8 shows the instances considered and results for Experiment 2. Columns 4-6 show the CPLEX results. The value “nf” in columns 4-5 means that no feasible solution was found prior to termination, and “?” in column 5 means that CPLEX was unable to prove the optimality of the best solution found prior to termination. Columns 7-14 show the GA results. Column 7 shows the total cost of the best solution identified by the GA. The CPLEX and GA costs in columns 4 and 7 are rounded to the nearest integer. Columns 9-14 show how this cost is divided into categories. Columns (9, 10-11, 12-13, 14) show the transportation, manufacturer inventory holding, DC inventory holding, and DC backorder costs respectively. Column 10 shows the manufacturer’s *baseline* (i.e. default) inventory holding cost (in the case of no consolidation); this is part 3A of the objective function. Column 11 shows the increase in the manufacturer’s inventory holding cost due to consolidation; this is part 3B of the objective function (Table 2, Section 1.5). Column 12 shows the total baseline inventory holding cost at all DCs combined; this is part 4 of the objective function. Column 13 shows the reduction in total DC inventory holding cost due to consolidation; this is part 5 of the objective function. Column 8 shows the number of trucks departing in each cycle.

Table 8. Experiment 2 results

Instance	# DCs	Number of unique default replenishment times (DT_r)	Total CPLEX cost	Optimal?	Time (sec)	Total GA cost	# Trucks (No. trucks departing manufacturer)	Truck cost	Baseline manuf. inv. holding cost	Manuf. inv. holding cost adjustment	Baseline DC inv. holding cost	DC inv. holding cost adjustment	DC backorder costs
S1.1			2949	Y	4	2949	4	880	1050	330	720	-166.5	135
S1.2			3785	Y	4	3785	3	1320	1050	600	720	-220.5	315
S1.3			2410	Y	3	2410	5	550	1050	180	720	-135	45
S1.4	2	7	4775	Y	30	4775	5	1100	2100	360	1440	-270	45
S1.5			1982	Y	4	1982	4	880	525	165	360	-83.25	135
S1.6			3005	Y	3	3005	5	1100	1050	180	720	-135	90
S1.7			2881	Y	13	2881	4	880	1050	330	720	-166.5	67.5
S2.1			2031	Y	1	2031	3	810	450	300	255	-72	288
S2.2			2841	Y	1	2841	3	1620	450	300	255	-72	288
S2.3			1449	Y	2	1449	5	675	450	60	255	-27	36
S2.4	2	6	2862	Y	3	2862	5	1350	900	120	510	-54	36
S2.5			1565	Y	1	1565	3	810	225	150	127.5	-36	288
S2.6			2160	Y	1	2160	5	1350	450	60	255	-27	72
S2.7			1878	Y	2	1878	3	810	450	300	255	-99	162
S3.1			1300	Y	1	1300	3	570	429	91	201	-26.25	35
S3.2			1768	Y	1	1768	2	760	429	273	201	-35	140
S3.3			1010	Y	1	1010	4	380	429	0	201	0	0
S3.4	2	4	1995	Y	1	1995	3	570	858	182	402	-52.5	35
S3.5			952	Y	1	952	3	570	214.5	45.5	100.5	-13.13	35
S3.6			1335	Y	1	1335	3	570	429	91	201	-26.25	70
S3.7			1282	y	1	1282	3	570	429	91	201	-26.25	17.5
M1.1			6941	?	3600	6941	8	2480	2464	504	1652	-322	163.30
M1.2			9650	?	3600	9270	7	4340	2464	784	1652	-362	391.88
M1.3			5701	?	3600	5701	8	1240	2464	504	1652	-322	163.3
M1.4	2	14	11,239	?	3600	11,239	8	2480	4928	1008	3304	-644	163.3
M1.5			4792	?	3600	4792	8	2480	1232	252	826	-161	163.30
M1.6			7105	?	3600	7105	8	2480	2464	504	1652	-322	326.61
M1.7			6860	?	3600	6860	8	2480	2464	504	1652	-322	81.65
M2.1			3377	Y	1823	3377	4	1120	1188	418	618	-151	184.33
M2.2			4382	Y	338	4382	3	1680	1188	638	618	-131	389.33
M2.3			2712	Y	2561	2712	5	700	1188	220	618	-107.5	93.33
M2.4	3	8	5330	?	3600	5330	5	1400	2376	440	1236	-215	93.33
M2.5			2341	Y	445	2341	4	1120	594	209	309	-75.5	184.33
M2.6			3505	Y	985	3505	5	1400	1188	220	618	-107.5	186.67
M2.7			3285	?	3600	3285	4	1120	1188	418	618	-151	92.17
M3.1			11,068	?	3600	11,015	11	4070	3600	1272	1875	-304.72	502.8
M3.2			17,513	?	3600	14,358	7	5180	3600	2724	1875	-485.5	1464
M3.3			8979	?	3600	8,894	15	2775	3600	648	1875	-209.8	205.3
M3.4	3	22	18,014	?	3600	17,455	11	4070	7200	2544	3750	-646.93	537.8
M3.5			24,938	?	3600	7631	9	3330	1800	930	937.5	-149.28	782.8
M3.6			11,506	?	3600	11,506	11	4070	3600	1272	1875	-281.38	970.6
M3.7			10,888	?	3600	10,679	9	3330	3600	1872	1875	-432.5	434.5

Instance	# DCs	Number of unique default replenishment times (DT_r)	Total CPLEX cost	Optimal?	Time (sec)	Total GA cost	# Trucks (No. trucks departing manufacturer)	Truck cost	Baseline manuf. inv. holding cost	Manuf. inv. holding cost adjustment	Baseline DC inv. holding cost	DC inv. holding cost adjustment	DC backorder costs
L1.1			398,639	?	3600	9610	15	3750	2646	1057	1869	-390.07	677.8
L1.2			90,171	?	3600	12,536	10	5000	2646	2002	1869	-555.86	1575.24
L1.3			86,471	?	3600	7566	18	2250	2646	728	1869	-265.57	339.05
L1.4	3	30	35,259	?	3600	14,699	17	4250	5292	1652	3738	-677.14	443.63
L1.5			385,470	?	3600	6879	12	3000	1323	798	934.5	-197.89	1021.61
L1.6			39,602	?	3600	10,108	17	4250	2646	826	1869	-313.57	830.6
L1.7			10,649	?	3600	9146	13	3250	2646	1428	1869	-532.36	485.33
L2.1			1,953,850	?	3600	11,714	32	4480	3780	996	2467.5	-405.36	396.29
L2.2			1,963,790	?	3600	14,939	21	5880	3780	2022	2467.5	-562.79	1351.83
L2.3			741,619	?	3600	9307	35	2450	3780	708	2467.5	-330.21	232.17
L2.4	3	57	386,844	?	3600	18,383	35	4900	7560	1416	4935	-660.43	232.17
L2.5			1,664,780	?	3600	8089	23	3200	1890	888	1233.75	-251.96	1129.14
L2.6			nf	n/a	3600	11,990	35	4900	3780	708	2467.5	-330.21	464.34
L2.7			837,386	?	3600	11,280	23	3200	3780	1776	2467.5	-513.93	569.57
L3.1			411,440	?	3600	21,649	20	5800	8700	1690	5850	-911.38	519.88
L3.2			315,441	?	3600	26,777	16	9280	8700	2750	5850	-1085.71	1282.44
L3.3			691,078	?	3600	18,563	23	3335	8700	1060	5850	-691.71	309.49
L3.4	4	36	568,707	?	3600	36,764	22	6380	17,400	2400	11,700	-1532.75	416.83
L3.5			1,490,850	?	3600	13,962	19	5510	4350	1025	2925	-486.85	638.44
L3.6			890,517	?	3600	22,086	21	6090	8700	1430	5850	-808.58	824.23
L3.7			328,054	?	3600	21,369	18	5220	8700	2110	5850	-986.04	474.60
V1.1			1,107,970	?	3600	30,020	29	7540	12,096	3204	7224	-1005.02	960.64
V1.2			4,269,890	?	3600	37,336	25	13,000	12,096	4428	7224	-1164.12	1751.57
V1.3			224,287	?	3600	25,631	39	5070	12,096	1413	7224	-556.13	384.14
V1.4	3	60	2,890,450	?	3600	50,796	36	9360	24,192	3528	14448	-1293.33	561.36
V1.5			2,296,230	?	3600	19,251	29	7540	6,048	1602	3612	-481.68	930.64
V1.6			2,808,080	?	3600	30,839	33	8580	12,096	2520	7224	-857.98	1276.71
V1.7			860,196	?	3600	29,513	29	7540	12,096	3168	7224	-1045.3	530.14
V2.1			nf	n/a	3600	82,654	61	25,620	29,260	9306	19,180	-3375	2662.85
V2.2			nf	n/a	3600	104,790	44	36,960	29,260	16,951	19,180	-4273.85	6713.22
V2.3			nf	n/a	3600	68,207	74	15,540	29,260	5896	19,180	-2995.14	1326.07
V2.4	4	136	nf	n/a	3600	135,000	72	30,240	58,520	12,584	38,360	-6187.66	1483.96
V2.5			nf	n/a	3600	54,995	52	21,840	14,630	6358	9590	-1853.31	4430.67
V2.6			nf	n/a	3600	84,562	69	28,980	29,260	7194	19,180	-3285.25	3233.21
V2.7			nf	n/a	3600	81,237	57	23,940	29,260	10,758	19,180	-3818.73	1917.53
V3.1			nf	n/a	3600	38,247	37	12,210	12,480	4176	8160	-1506.59	2727.44
V3.2			nf	n/a	3600	48,987	29	19,140	12,480	6400	8160	-1589.26	4395.81
V3.3			nf	n/a	3600	31,277	48	7920	12,480	2352	8160	-956.09	1320.94
V3.4	5	88	nf	n/a	3600	61,145	42	13,860	24,960	6304	16,320	-2482.97	2183.94
V3.5			nf	n/a	3600	26,435	33	10,890	6240	2608	4080	-737.23	3353.81
V3.6			nf	n/a	3600	40,458	45	14,850	12,480	2832	8160	-1126.09	3261.88
V3.7			nf	n/a	3600	36,671	32	10,560	12,480	5072	8160	-1531.59	1930.72

1.7.6. Experiment 2 discussion

We first discuss the reasonableness of the results and the performance of the solution methods. Then we turn to managerial insights. Overall, the results appear reasonable on at least three counts. First, the total cost of the best solutions identified by the GA for variation X.3, X.5 and X.7 are always less than or equal to that for variation X.1 (see Table 7 for the variation definitions). Also, the total cost of the best solutions identified by the GA for variations X.2, X.4, and X.6 are always greater than or equal to that for variation X.1. The same trends hold for all but one solution identified by CPLEX for the small and medium-sized instances (but not the larger instances). Second, in all twelve base instances the number of truck departures in the best solution found by the GA in variation (X.2, X.3, X.4, X.5, X.6,X.7) is ($\leq$, $\geq$, $\geq$, $\leq$, $\geq$, $\leq$,) the number of truck departures in variation X.1. These results are reasonable; they reflect the fundamental economic consequences of changing the cost structure according to Table 7. Third, the breakdown of the "Total GA cost" into categories is reasonable. In particular, the costs in columns 11, 13, and 14 are always (never) zero when the total number of truck departures equals (does not equal) the number of unique default replenishment times DT_r. Also, in all instances, backorder costs are usually not significant, and transportation and inventory holding costs include the majority of the total GA cost. All of the above observations help to verify the correctness of the math model and coding of the proposed GA.

We now turn our attention to the performance of the solution methods. For the small problem instances, CPLEX finds optimal solutions in less than 30 seconds and the GA finds the same solutions in 10 seconds. On the other hand, the GA outperforms CPLEX on the medium-sized instances. Indeed, the GA only uses 60 seconds of computation time and finds a better solution than (the same solution as) CPLEX on 7 (14) of the 21 medium-sized instances. On the large instances, the GA significantly outperforms CPLEX. Indeed, in 600 seconds the GA usually finds solutions whose cost is 1-2 orders of magnitude less than the cost identified by CPLEX within an hour. Note that, for these instances, the costs identified by CPLEX do not agree with intuition. For example, the total cost of the best solutions identified by CPLEX for variations X.2, X.4, X.6 are sometimes less than that for variation X.1. This indicates that CPLEX is overwhelmed by the large instances and does not make meaningful progress toward identifying an optimal solution prior to termination. Finally, CPLEX either fails to identify feasible solutions or identifies very poor solutions for the very large instances; the GA, however, identifies reasonable solutions within 600 seconds. These results highlight the effectiveness of the proposed GA.

The results in Table 8 show that the best shipment consolidation plan is highly sensitive to the cost structure. For example, the best shipment consolidation schedule identified by the GA has anywhere from (10-18, 21-35, 16-23) truck departures for instance group (L1, L2, L3) respectively, and it has anywhere from (25-39, 44-74, 29-48) truck departures for instance group (V1, V2, V3) respectively. We also note that, in every instance, the best number of truck departures identified by the GA is neither very high (e.g. approaching the value in column 3) nor very low (e.g. approaching 1). These results indicate that managers of real-world enterprises should carefully develop their shipment consolidation plans. Managers may wish to develop detailed shipment consolidation *schedules* rather than simply following a consolidation *policy*. More generally, the results show the importance of considering a variety of factors—transportation costs, manufacturer inventory holding costs, DC inventory holding costs, and backorder costs—in one integrated model. This kind of multi-faceted analysis may assist third party logistics (3PL) providers in deciding about stocking and shipping options in a distribution network in which the manufacturer and DCs are active collaborators. In this case, collaboration is a win-win situation for both the manufacturer and the DCs and has the potential to reduce total system cost and/or increase the total supply chain profit.

1.8. Conclusion

In this essay, we introduced a novel operational problem and a new model for shipment consolidation scheduling in a two-echelon supply chain. The problem addresses shipment consolidation decisions made by a manufacturer who provides inventory replenishments to multiple downstream distribution centers (DCs). Unlike previous work, the consolidation activities in this problem are not restricted to specific rules such as aggregation of shipments at regular times or consolidating when a predetermined quantity has accumulated. The problem sheds light on the trade-off between transportation costs, manufacturer inventory holding costs, DC inventory holding costs, and DC backorder costs.

We developed a mixed-integer quadratic programming (MIQP) model and genetic algorithm (GA) to determine the shipment consolidation schedule that minimizes total cost. Two experiments were conducted. In Experiment 1, we showed that the optimal replenishment times may be highly irregular—taking seemingly arbitrary, fractional values—if the manufacturer's inventory holding cost, *CHM*, is less than the inventory holding cost CH_d at one or more DCs *d*. In Experiment 2 the effectiveness of two solution methods—(1) CPLEX's default MIQP solver with a one-hour time limit and (2) the GA with a shorter time limit—was evaluated on instances that avoided the possibility of irregular replenishments. Experimental results showed the superiority of the GA over CPLEX.

The novelty of the problem under investigation can be viewed as strength or a shortcoming. The main shortcoming is that the problem is most relevant to cases with (1) limited DC personnel and/or high transportation costs which make it logistically or economically impossible to replenish DCs frequently, (2) items that are small compared to the size of a truck, and/or (3) replenishment of DCs via containers of particular sizes. It could be argued that such cases are rare in industry. On the other hand, the problem's novelty may make this study a starting point for future research that explores whole new approaches to supply chain modeling.

Future research should address some limitations of this work. One major limitation is that the DC locations and truck routes and capacities are ignored. A future extension of this work might therefore consider a geographically diverse set of DCs and develop feasible routes for finite-capacity trucks in which DCs in close proximity are grouped together in order to take advantage of economies of scale. Future work might also consider more realistic attributes such as stochastic and/or seasonal demands, a multi-product distribution network, and different types of carriers. Entirely different approaches may be required for considering these aspects. Moreover, it is necessary to further evaluate the effectiveness of the heuristic algorithm in solving large/very large problem sizes in future work. This could be performed by finding bounds for the objective function for different problem instances. Also, other heuristic procedures such as Ant Colony and Simulated Annealing will be developed to compare the results with the results of the GA heuristic in this essay.

References

Arslan, A. M., Agatz, N., Kroon, L., Zuidwijk, R. (2019). Crowdsourced delivery-A dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science*, *53*(1), 222-235.

Axsäter, S., (2001). A note on stock replenishment and shipment scheduling for vendor-managed inventory systems. *Management Science*, *47*(9), 1306–1310.

Banerjee, A. Burton J. S. (1994). Coordinated vs. independent inventory replenishment policies for a vendor and multiple buyers. *International Journal of Production Economics, 35*(1-3), 215-222.

Bookbinder, J. H., Cai, Q., He, Q. M. (2011). Shipment consolidation by private carrier: The discrete time and discrete quantity case. *Stochastic Models*, *27*(4), 664–686.

Cai, Q., He, Q. M., Bookbinder, J. H. (2014). A tree-structured Markovian model of the shipment consolidation process. *Stochastic Models*, *30*(4), 521–553.

Çetinkaya, S., Lee, C. Y. (2000). Stock replenishment and shipment scheduling for vendor-managed inventory systems. *Management Science*, *46*(2), 217-232.

Çetinkaya, S., Lee, C. Y. (2002). Optimal outbound dispatch policies: Modeling inventory and cargo capacity. *Naval Research Logistics*, *49*(6), 531-556.

Çetinkaya, S., Bookbinder, J. H. (2003). Stochastic models for the dispatch of consolidated shipments. *Transportation Research Part B: Methodological*, *37*(8), 747-768.

Çetinkaya, S. (2005). Coordination of inventory and shipment consolidation decisions: A review of premises, models, and justification. *Applications of Supply Chain Management and E-Commerce Research*, 3–51

Çetinkaya, S., Mutlu, F., Lee, C. Y. (2006). A comparison of outbound dispatch policies for integrated inventory and transportation decisions. *European Journal of Operational Research*, *171*(3), 1094-1112.

Çetinkaya, S., Tekin, E., Lee, C. Y. (2008). A stochastic model for joint inventory and outbound shipment decisions. *IIE Transactions*, *40*(3), 324-340.

Çetinkaya, S., Mutlu, F., Wei, B. (2014). On the service performance of alternative shipment consolidation policies. *Operations Research Letters*, *42*(1), 41-47.

Chen, F. Y., Wang, T., Xu, T. Z. (2005). Integrated inventory replenishment and temporal shipment consolidation: A comparison of quantity-based and time-based models. *Annals of Operations Research*, *135*(1), 197-210.

Chen, L. H., Kang, F. S. (2010). Coordination between vendor and buyer considering trade credit and items of imperfect quality. *International Journal of Production Economics*, *123*(1), 52-61.

Cheung, K. L., Lee, H. L. (2002). The inventory benefit of shipment coordination and stock rebalancing in a supply chain. *Management Science*, *48*(2), 300-306.

Goldberg, D. E., Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning*, *3*, 95-99.

Goyal, S. K. (1988). A joint economic-lot-size model for purchaser and vendor: A comment. *Decision Sciences*, *19*(1), 236-241.

Goyal, S. K. (1995). A one-vendor multi-buyer integrated inventory model: A comment. *European Journal of Operational Research*, *82*(1), 209-210.

Goyal, S. K., Nebebe, F. (2000). Determination of economic production–shipment policy for a single-vendor–single-buyer system. *European Journal of Operational Research*, *121*(1), 175-178.

Hill, R. M. (1997). The single-vendor single-buyer integrated production-inventory model with a generalized policy. *European Journal of Operational Research*, *97*(3), 493-499.

Hill, R.M., (1999). The optimal production and shipment policy for single-vendor single-buyer integrated production-inventory problem. *International Journal of Production Research, 37* (11), 2463–2475

Holland, J. H. (1992). Genetic algorithms. *Scientific American*, *267*(1), 66-73.

Hong, K. S., Lee, C. (2013). Optimal time-based consolidation policy with price sensitive demand. *International Journal of Production Economics*, *143*(2), 275-284.

Howard, C., Marklund, J. (2011). Evaluation of stock allocation policies in a divergent inventory system with shipment consolidation. *European Journal of Operational Research*, *211*(2), 298-309.

Huang, C. K. (2004). An optimal policy for a single-vendor single-buyer integrated production–inventory problem with process unreliability consideration. *International Journal of Production Economics*, *91*(1), 91-98.

Kelle, P., Transchel, S., Minner, S. (2009). Buyer–supplier cooperation and negotiation support with random yield consideration. *International Journal of Production Economics*, *118*(1), 152-159.

Klapp, M. A., Erera, A. L., Toriello, A. (2018). The one-dimensional dynamic dispatch waves problem. *Transportation Science*, *52*(2), 402-415.

Liu, X., Çetinkaya, S. (2011). The supplier–buyer integrated production-inventory model with random yield. *International Journal of Production Research*, *49*(13), 4043-4061.

Lu, L. (1995). A one-vendor multi-buyer integrated inventory model. *European Journal of Operational Research*, *81*(2), 312-323.

Maddah, B., Jaber, M. Y. (2008). Economic order quantity for items with imperfect quality: revisited. *International Journal of Production Economics*, *112*(2), 808-815.

Moon, I. K., Cha, B. C., Lee, C. U. (2011). The joint replenishment and freight consolidation of a warehouse in a supply chain. *International Journal of Production Economics*, *133*, 344–350.

Mutlu, F., Çetinkaya, S. (2010). An integrated model for stock replenishment and shipment scheduling under common carrier dispatch costs. *Transportation Research Part E: Logistics and Transportation Review*, *46*(6), 844-854.

Mutlu, F., Çetinkaya, S. I. L., Bookbinder, J. H. (2010). An analytical model for computing the optimal time-and-quantity-based policy for consolidated shipments. *IIE Transactions*, *42*(5), 367-377.

Papachristos, S., Konstantaras, I. (2006). Economic ordering quantity models for items with imperfect quality. *International Journal of Production Economics*, *100*(1), 148-154.

Russell, R. M., Krajewski, L. J. (1991). Optimal purchase and transportation cost lot sizing for a single item. *Decision Sciences*, *22*(4), 940-952.

Stenius, O., Karaarslan, A. G., Marklund, J., De Kok, A. G. (2016). Exact analysis of divergent inventory systems with time-based shipment consolidation and compound Poisson demand. *Operations Research*, *64*(4), 906-921.

Swenseth, S. R., Godfrey, M. R. (2002). Incorporating transportation costs into inventory replenishment decisions. *International Journal of Production Economics*, *77*(2), 113-130.

Taleizadeh, A. A., Noori-daryan, M., Cárdenas-Barrón, L. E. (2015). Joint optimization of price, replenishment frequency, replenishment cycle and production rate in vendor managed inventory system with deteriorating items. *International Journal of Production Economics*, *159*, 285-295.

Toptal, A., Çetinkaya, S. (2008). Quantifying the value of buyer–vendor coordination: Analytical and numerical results under different replenishment cost structures. *European Journal of Operational Research*, *187*(3), 785-805.

Ülkü, M. A., Bookbinder, J. H. (2012). Optimal quoting of delivery time by a third party logistics provider: The impact of shipment consolidation and temporal pricing schemes. *European Journal of Operational Research*, *221*(1), 110-117.

Viswanathan, S. (1998). Optimal strategy for the integrated vendor-buyer inventory model. *European Journal of Operational Research*, *105*(1), 38-42.

Van Heeswijk, W. J., Mes, M. R., Schutten, J. M. (2019). The delivery dispatching problem with time windows for urban consolidation centers. *Transportation science*, *53*(1), 203-221.

Voccia, S. A., Campbell, A. M., Thomas, B. W. (2019). The same-day delivery problem for online purchases. *Transportation Science*, *53*(1), 167–184.

Essay 2: The Vehicle Routing Problem with Flexible Repeat Visits

2.1. Abstract

This essay introduces a new variant of the vehicle routing problem called the vehicle routing problem with flexible repeat visits (VRP-FRV). The problem considers a set of customers with certain locations and a set of available vehicles for serving customers. Time is discretized into periods and the planning horizon is cyclic. Each customer must be repeatedly visited under a maximum inter-visit time requirement but customers are otherwise indifferent to the exact times when they are visited. In this problem, each vehicle may make repeated visits to the same customer within the same route. The goal is to identify customer visit times which satisfy their maximum inter-visit time requirements and create vehicle routes that minimize the number of vehicles used. This problem is modeled as a binary integer programming (BIP) model which is solved by IBM ILOG CPLEX. A constructive heuristic with multiple local search improvement procedures is developed to handle large problem instances. We then compare the performance of CPLEX and our proposed heuristic on a set of benchmark instances.

2.2. Introduction

Logistics service providers have experienced significant changes in recent decades and have shifted towards being more customer-oriented. Their competition continues to be focused on the ability to satisfy customer needs with respect to time and cost. Today, with the increasing demand in the market, there is a lot of pressure on carriers and this requires more cost-effective operations while still making the infrastructural investments needed to remain competitive (Soriano et al. 2018). Consequently, transportation companies are continuously seeking ways to improve the efficiency of their distribution system's operations. Such efforts include reduction in the number of vehicles used and designing more efficient vehicle routes which have huge impacts on reducing the company's transportation costs. As a result, better customer service can be achieved through shorter distribution times that can play an important role in the competitiveness of the company. These benefits are often achievable through the use of mathematical models of various vehicle routing problems (VRPs). Such models combine the theoretical realm of mathematics with the practical realities of day-to-day business operations.

Vehicle routing problems mainly focus on the operational planning of vehicle routes. These problems involve the challenge of optimizing the size and composition of vehicles needed and designing optimal vehicle routes from a depot to a set of destinations. Nowadays, many situations can be found where a high volume of flexible requests have to be serviced on a daily basis between distant areas. In this research, we introduce the vehicle routing problem with flexible repeat visits (VRP-FRV). The problem considers a set of customers at certain locations who each want to be visited under a different maximum inter-visit time requirement. However, they are flexible in their visit times and do not have exact requirements regarding when they are visited. Time is discretized into periods, and the planning horizon consists of a single T-period cycle. The goal is to develop cyclic vehicle routs that satisfy the customer inter-visit time requirement while minimizing the number of vehicles needed to cover customer demand. This problem applies for cases where the cost of using each vehicle is so high and the travel cost within a small city is negligible compared with the cost of using vehicles. Thus, we focus on minimizing the total number of vehicles needed to cover customer demand.

This research contributes to the existing VRP studies in two aspects. First, each vehicle is not restricted to visit each customer only once within a given route. In fact, the same customer may be visited multiple times within one route. Second, the customer requirements are of different types compared with the existing VRP studies with time windows for deliveries. In particular, each customer requires to be visited at least once during *every* time window of a specified length *that could possibly be formed* within the cycle instead of being visited once during one time window among a given subset of pre-specified time windows.

The VRP-FRV has several real-world applications. One scenario is that of caretakers who provide service to elderly people at home. Each caretaker is assigned a number of elderly people to visit one or more times per day. Customers (elderly people) differ in their requirements and the minimum frequency at which they need to be visited every day. For example, one person may need to be checked on at least once within every time window of 4 hours while another person is needier and must be visited at least once every 2 hours. The VRP-FRV can also be imagined as a police patrol routing problem where the customers are various locations in the city that require frequent observations. Such locations could include known high-crime areas, high-profile residences, and/or safe houses. A third application is the routing of preventive maintenance workers for a major utility company or manufacturer who need to check key system components on a regular basis.

The remainder of this study is organized as follows. Section 2.3 reviews the related literature. A formal description of the problem and an illustrative example are provided in Section 2.4. Section 2.5 presents a mathematical formulation of the problem. A heuristic algorithm is described in Section 2.6. In Section 2.7, we describe the experimental setup and discuss the experimental results. We then conclude in Section 2.8.

2.3. Related literature

Several streams of literature relate to our problem. These include the literature on variations of the traveling salesman problem (TSP), major types of vehicle routing problems, and patrol operations planning. Different vehicle routing problems considered include the capacitated vehicle routing problem (CVRP), vehicle routing problem with time windows (VRP-TW), dial-a-ride problem (DARP), and consistent vehicle routing problem (ConVRP). We now discuss these streams of literature in detail.

2.3.1. Traveling salesman problems

The traveling salesman problem (TSP) is a classic optimization problem whose objective is to determine the shortest possible route for a salesman who visits each city only once and returns to the starting city (Fischetti et al. 1997). Some real-world problems including personnel scheduling (Masmoudi and Mellouli 2014), patrol planning (An et al. 2013), and goods distribution (Liu and Zhang 2014) need to be modeled with a generalization of the traveling salesman problem. Such problems use the multiple traveling salesman problem (MTSP) which has been specifically designed to consider a situation with two or more salesmen. The goal of the MTSP is to minimize the total distance traveled across a set of cities where two/more than two salesmen make a route visiting each city only once and finally returning to the origin city. Heuristic approaches can be utilized to solve the MTSP such as ant colony optimization (ACO) (Singh and Mehta, 2014), particle swarm optimization (PSO) (Yan et al. 2012), and two-phase heuristics including the *k*-means algorithm and genetic algorithm (GA) (Avin et al. 2012, Yuan et al. 2013, Xu et al. 2018).

One category of the MTSP is the pickup and delivery traveling salesman problem (PDTSP) where a single vehicle with a certain capacity performs both pick-ups and deliveries at a set of customer locations. The problem focuses on finding the minimum cost tour (Hamilton cycle) such that the pickup vertex of a given request is visited before the corresponding delivery vertex of that request. This problem has been studied widely (Carrabs et al. 2007, Peterson et al. 2009, Toulouse 2010, Cordeau et al. 2010, Bonomo et al. 2011, Alba Martinez et al. 2013).

Another variant of the MTSP is the multiple traveling salesman problem with time windows (MTSPTW). This problem considers salesmen who depart from a depot, visit a number of cities within predetermined time windows, and then return to the depot (Dumas et al. 1995). MTSPTW has been studied broadly by researchers. They approach the problem in different ways including using exact algorithms (Dumas et al. 1995, Kara and Derya 2015), discretization methods (Wang and Regan 2009), branch-and-cut algorithms (Dash et al. 2012), column generation and dynamic programming algorithms (Baldacci et al. 2012), and greedy methods with variable neighborhood search (Da Silva et al. 2010, Karabulut and Tasgetiren 2014). A special case of the MTSPTW is when travel times are considered to be time dependent to deal with traffic issues during rush hours. This problem has been addressed in several studies including Montero et al. (2017), Arigliano et al. (2019), Vu et al. (2018), Sun et al. (2018), Vu et al. (2019), and Cacchiani et al. (2020). Multiple traveling salesman problems address major applications; however, they only assume one visit to each customer during each salesman's route. In other words, the salesman is not allowed to visit a customer at multiple times. This essay takes a new perspective into account for the salesman's route by including repeat visits to the same customer within the same route.

2.3.2. Vehicle routing problems

The vehicle routing problem (VRP) covers a variety of problems in which a set of routes for a fleet of vehicles based at one or several depots must be determined for a set of geographically dispersed cities or customers. The general goal in the VRP is to find minimum-cost routes through which vehicles visit a set of customers with known demands such that vehicles start at and return to a depot and each vehicle visits each customer only once. There is a major difference between the vehicle routing problems and the traveling salesman problems. The difference is that in the vehicle routing problems, the depot where the vehicles start and end their route is a separate node other than the customer locations, whereas in the traveling salesman problems, the salesman's tour starts from one customer location. Consequently, there is a single depot for all vehicles in the vehicle routing problem while each salesman has a different starting point in the multiple traveling salesman problems. The objective in the VRP could be different depending on the application including minimizing the transportation costs, minimizing number of vehicles needed, minimizing the vehicle load, etc. One of the most important and well-known problems in vehicle routing is the capacitated vehicle routing problem (CVRP). The goal of this problem is to find the lowest cost vehicle routes from an origin to the customers and back to the origin so that each customer is visited exactly once by exactly one vehicle while respecting each vehicle's capacity (Golden et al. 2008). The first mathematical formulation and algorithm for the solution of the CVRP was proposed by Dantzig and Ramser (1959). Later, Clarke and Wright (1964) proposed the first heuristic for this problem. To date, many variants and solution procedures for the CVRP have been published by scores of authors including Toth and Vigo (2002), Lysgaard et al. (2004), Fukasawa et al. (2006), Lee et al. (2010), Kara (2010), Baldacci (2010), Jepsen (2013), Laporte et al. (2014), Foulds et al. (2015), Wang et al. (2017), and Zhang et al. (2017). General surveys can be found in Laporte (2007, 2009), Toth and Vigo (2014) and Ritzinger et al. (2016).

The vehicle routing problem with time windows (VRP-TW) is another VRP variant that has received a lot of consideration in the literature. The VRP-TW involves finding a set of vehicle routes starting and ending at a depot to satisfy demands of a set of customers within pre-determined time windows. In other words, each vehicle must arrive at/depart from a customer within a specific time window. The solution to the VRP-TW consists of a set of routes that minimizes the total distance traveled. A large two-part survey of the VRP-TW has been authored by Bräysy and Gendreau (2005a, 2005b). Different studies analyze various heuristic methods for the VRP-TW including Ombuki et al. (2006), Alvarenga et al. (2007), Kallehauge (2008), Kritikos and Ioannou (2010), Yu and Yang (2011), Kumar and Panneerselvam (2012), Banos et al. (2013), Toth and Vigo (2014), and Koch et al. (2018). The VRP-FRV studied in this essay is different from the vehicle routing problem with time windows in that the VRP-TW does not allow repeat visits to the same customer within the same route.

The dial-a-ride problem (DARP) generalizes vehicle routing problems with time windows and pickup and delivery problems. The DARP models are applicable in demand responsive transportation service such as patient transportation. It involves providing specific services to elderly or handicapped people including transportation of patients to medical centers or dropping them off home. Such transportation requests have to be completed with respect to patients' convenience considerations within certain time windows. One evident characteristic of the DARP is that in contrast to other vehicle routing problems, in the DARP humans are transported rather than goods. Therefore, the overall goal is to make a trade-off between the users' convenience and operating transportation costs. This is done by introducing additional constraints that limit the maximum user ride time or by specifying tight time windows. Variants of the DARP have been considered by Cordeau and Laporte (2003), Ropke et al. (2007), Parragh et al. (2010), Schilde et al. (2011), Parragh and Schmid (2013), Ritzinger et al. (2016), Detti et al. (2017), and Riedler (2018). A comprehensive overview of the DARP variants and existing solution methods can be found in Cordeau and Laporte (2007), Ritzinger et al. (2016), and Ho et al. (2018).

Another important VRP variant is the consistent vehicle routing Problem (ConVRP) which was introduced by Groer et al. (2009). Service consistency is applicable to a number of service industries where there are repeated deliveries to customers throughout a planning horizon.

Examples include small package delivery and home health care (Tarantilis et al. 2012, Lian et al., 2016). A review of the ConVRP is provided in Kovacs et al. (2014). Two main elements of service consistency are driver consistency and time consistency. Some companies want their drivers to develop relationships with customers on a route and have the same drivers visit the same customers at roughly the same time on each day that the customers need service. These service requirements, together with traditional constraints on vehicle capacity are addressed in the ConVRP (Kovacs et al. 2015). The ConVRP typically assumes a customer visit frequency of once a day or once a week with consistencies on visit times. However, if customers must be visited repeatedly each day and they are flexible regarding their visit times, a new variant of the problem will emerge. This motivates our introduction of the vehicle routing problem with flexible repeat visits (VRP-FRV).

2.3.3. Patrol routing problems

We now discuss another set of studies that relate to our problem. The real-world problem of scheduling and routing a fleet of assets to provide patrol coverage to geographically dispersed locations has received attentions from researchers. Different contexts include maritime and land surveillance, security beat planning, motor vehicle accident prevention, crime deterrence, and infrastructure protection. Considering these specific contexts of the patrol routing problems, it is evident that frequent observations are required to be made at the same locations during a day in these problems while this is not the focus in any vehicle routing or traveling salesman problems. A comprehensive survey on police patrol routing problems has recently been authored by Dewinter et al. (2020).

One example of a patrol routing problem is the bi-objective grid patrol routing problem (BGPRP) studied by Hsieh et al. (2015). The problem is defined for residents in certain regions of Taiwan who travel abroad during Chinese festival celebrations. Those residents may have already applied for police patrolling service requesting the local police patrol their neighborhoods repeatedly during their traveling time. The paper provides a modeling approach for planning patrol routes and assumes a network in the form of a grid structure. The authors consider two objectives, one is to minimize the total distance travelled and the second is to maximize the coverage of the patrol lanes.

Another related problem is the maximum covering and patrol routing problem (MCPRP) which was first studied by Keskin et al. (2012). This problem considers finding patrol routes for a set of police cars to maximize the coverage of all the hotspots during time intervals with high risks of accidents. Each patrol car begins its route at a station and returns to the same station at the end of the coverage. The problem is modeled as a mixed integer programming formulation and solved with heuristic techniques. Çapar et al. (2015) show that the math model of Keskin et al. (2012) can be improved in terms of reducing the number of variables using the information on the structure of candidate routes in an optimal solution. The research by Dewil et al. (2015) has demonstrated that the MCPRP can be modelled as a minimum cost network flow problem (MCNFP). They provide a time-space network formulation of the problem to ensure that all accident hotspots are covered. However, each police route does not include repeated visits to the same location within a shift.

Other emerged problems address applications where randomness in patrol routes and schedules are evident. Such randomized patrol planning is important in contexts where the planned patrols could not be successful since an enemy or attacker may be able to predict a defender's patrol movements. Combating criminal activities and successful execution of different security operations are among such applications. Examples of randomized patrol mission planning which have considered Markov decision process methods include Jain et al. (2010) and Erdogan et al. (2010). Jain et al. (2010) create randomized schedules of canine patrols, to determine security checkpoint locations at Los Angeles International Airport (LAX). This work was used to establish software for daily patrol and security planning at LAX. Erdogan et al. (2010) study the problem of scheduling paramedic crews and ambulances to stations throughout a shift to maximize the expected coverage of an area. Other studies in this stream include Shieh et al. (2013) and Fang et al. (2015).

In another related study, Lau and Gunawan (2016) consider the problem of scheduling security teams to patrol a mass rapid transit rail network of a large urban city. They model the deployment of security teams to stations at varying time periods with rostering as well as security-related constraints. They also discuss different scenarios by varying the number of visits required for each station according to their reported vulnerability. However, each team can only visit a particular station at most once per day (i.e. once in each route).

Our study has a more delicate look at customer visit requirements based on home healthcare and police patrol routing applications. In such applications, the customer requires repeated visits at different times during a day which could be made by the same vehicle/police within the same route. We introduce the vehicle routing problem with flexible repeat visits (VRP-FRV). The problem considers a set of customers at certain locations who need to be visited repeatedly with certain frequencies. Each customer has a maximum inter-visit time requirement but is flexible regarding exact visit times. In particular, the two unique contributions in this essay relate to the customer visit requirements. First, this study addresses both temporal and spatial aspects of visits to customers by the vehicles while allowing each vehicle to make repeated visits to the same location at different times within the same route. Second, these repeat visits are flexible and our way of modeling this flexibility is unique. The customer requirements are of different types compared with the existing VRP studies with time windows for deliveries. In our study, each customer requires to be visited at least once during every time window of a predetermined length that could possibly be formed within the cycle instead of being visited once during one time window among a given subset of pre-specified time windows.

2.4. Problem description and illustrative example

We now fully define the VRP-FRV. Consider a setting in which C customers need to be visited by a set of vehicles. Time is discretized into periods. Customer *c* needs to be visited at least once every CT_c time periods by a vehicle. We assume that the time spent by a vehicle to visit a customer is negligible compared to the duration of one period and that all customer visits take place at the beginning of a time period. The vehicles are identical, and customer *c* does not have any preference regarding which vehicle(s) serve(s) it. Furthermore, D_{cd} is the travel time between customers *c* and *d* which is integral and ≥ 1. The operations are cyclic, repeating every T periods where $T \geq \max_c\{CT_c\}$. The goal is to feasibly satisfy the customers' maximum inter-visit time requirements (CT_c) while minimizing the total number of vehicles used.

Tables 9-13 and Figure 4 illustrate an instance of this problem in which $C = 10$ and $T = 24$. The first row of Table 9 shows the customer numbers (*c*). The second row shows the CT_c values and the next two rows show imaginary (*x,y*) coordinates for the locations of the ten customers. Table 10 shows the symmetric travel time matrix (D_{cd}) which is derived from these (*x,y*) coordinates.

Each value in the matrix (D_{cd}) equals the straight-line Euclidean distance between customers c and d rounded up to the nearest integer.

Table 9. Parameters defining the illustrative instance part 1: CT_c and (*x,y*) coordinates for customers

Customer	1	2	3	4	5	6	7	8	9	10
CT_c	14	10	5	15	8	13	11	14	12	5
x coordinate	2.17	1.86	2.79	2.24	2.98	3.71	4.59	4.45	4.66	0.34
y coordinate	4.98	1.56	1.05	0.37	0.28	0.77	4.55	2.07	1.66	2.88

Table 10. Parameters defining the illustrative instance part 2: D_{cd}

Customer #	1	2	3	4	5	6	7	8	9	10
1	0	4	4	5	5	5	3	4	5	3
2	4	0	2	2	2	3	5	3	3	3
3	4	2	0	1	1	1	4	2	2	4
4	5	2	1	0	1	2	5	3	3	4
5	5	2	1	1	0	1	5	3	3	4
6	5	3	1	2	1	0	4	2	2	4
7	3	5	4	5	5	4	0	3	3	5
8	4	3	2	3	3	2	3	0	1	5
9	5	3	2	3	3	2	3	1	0	5
10	3	3	4	4	4	4	5	5	5	0

A feasible solution is characterized by (i) a *visit schedule* for each customer c that specifies when the customer is visited, and (ii) a set of vehicle routes that visit the customers at the times specified in (i). Regarding item (i), a feasible visit schedule for each customer c consists of visit times within the cycle T which satisfy the maximum inter-visit time requirement CT_c. In other words, the time interval between each two consecutive visit times for a given customer c must not be greater than the inter-visit time requirement CT_c. Table 11 shows a set of feasible visit times for the ten customers during the 24-period cycle. This is a feasible visit schedule since the inter-visit time requirement for each customer is satisfied. For example, for customer 1, $CT_1 = 14$ and this customer is visited at times (1, 2, 12, 24). Note that the time that elapses between consecutive visits is (1, 10, 12 and 1) periods respectively, and all values in this list are less than or equal to CT_1=14. Thus, the visit schedule (1, 2, 12, 24) satisfies customer 1's visit requirement.

Table 11. Feasible customer visit times

Customer 1	1	2	12	24								
Customer 2	4	11	17	23								
Customer 3	1	6	9	12	15	19	23					
Customer 4	2	14										
Customer 5	8	13	21	22	24							
Customer 6	7	20										
Customer 7	3	4	15	16								
Customer 8	8	10	19									
Customer 9	7	9	20									
Customer 10	2	3	4	5	6	7	8	9	14	15	20	21

Three vehicles can be used to make the required repeated visits to the ten customers. Tables 12 and 13 show a set of feasible vehicle routes that visit the customers at the times specified in Table 11. Table 12 shows the time when each vehicle visits a customer, and Table 13 shows which customer is visited at each time specified in Table 12. For example, vehicle 2 makes visits at the beginning of periods (1, 2, 4, 6, 7) to customers (3, 4, 2, 3, 6), respectively.

Table 12. Vehicle visit times in the feasible solution

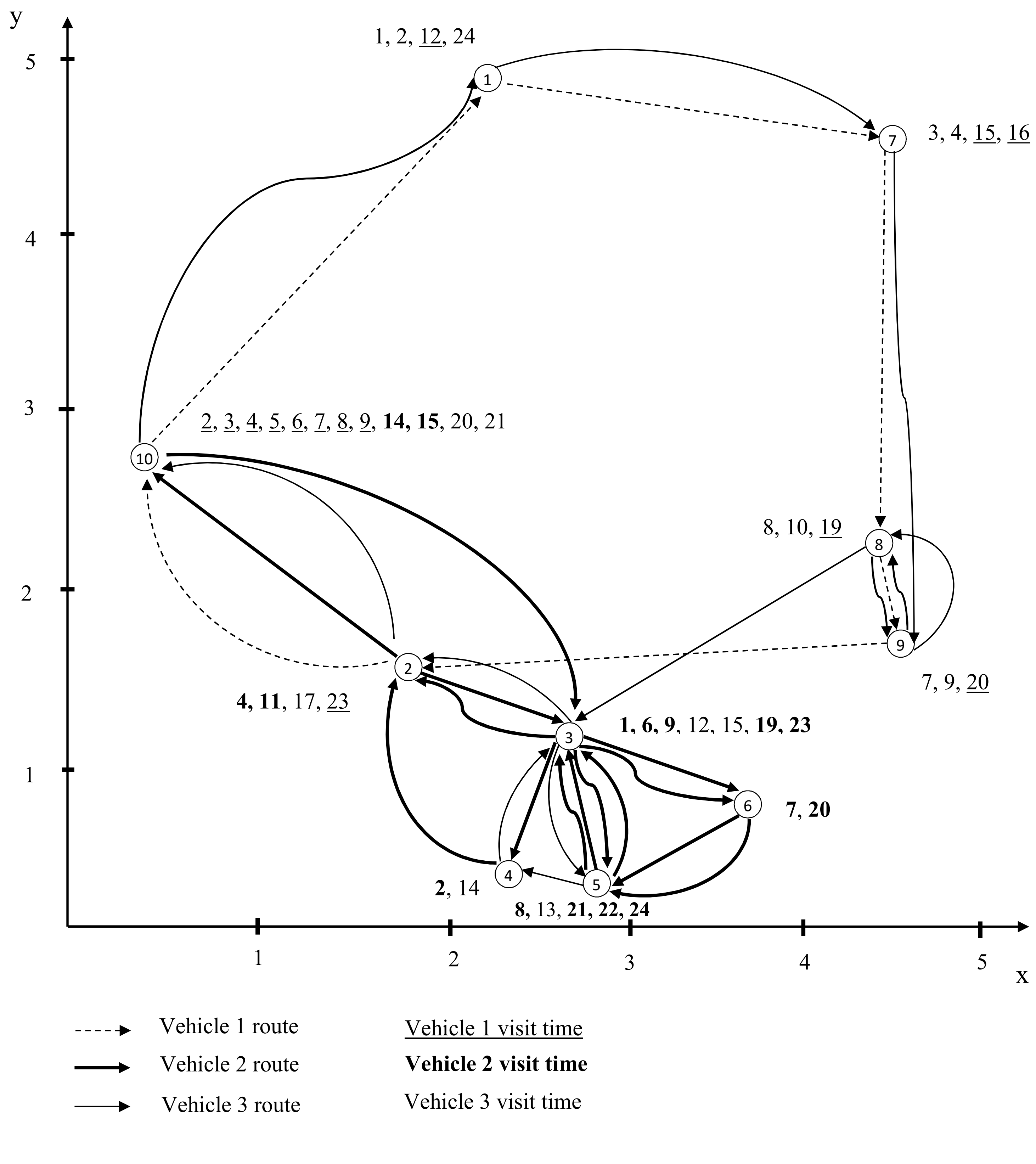
Vehicle 1	2	3	4	5	6	7	8	9	12	15	16	19	20	23		
Vehicle 2	1	2	4	6	7	8	9	11	14	15	19	20	21	22	23	24
Vehicle 3	1	2	3	4	7	8	9	10	12	13	14	15	17	20	21	24

Table 13. List of customers visited in the feasible solution

Vehicle 1	10	10	10	10	10	10	10	10	1	7	7	8	9	2		
Vehicle 2	3	4	2	3	6	5	3	2	10	10	3	6	5	5	3	5
Vehicle 3	1	1	7	7	9	8	9	8	3	5	4	3	2	10	10	1

Figure 4 illustrates the vehicle routes specified in Tables 12-13. Numbered circles illustrate the customers. Vehicle 1's travel path is indicated by dashed arrows, and the times of the visits that are made by this vehicle are shown by underlined values. Vehicle 2's travel path is represented by bold arrows, and the times of the visits that are made by this vehicle are shown by bold values. Vehicle 3's travel path is indicated by regular arrows, and the times of the visits that are made by this vehicle are shown in regular font.

Figure 4. Vehicle routes and visit times for the illustrative example



If we track the visit times for each vehicle to different customers in the same route, we can observe that the gap in visit times from one customer to the next customer that are visited by the same vehicle is not greater than the travel time between the two customers based on the distance matrix. This makes it a feasible solution. For example, consider vehicle 1's route. This vehicle is at customer 10 at time 2 and stays at the same customer for the next 7 periods until time 9 when it leaves customer 10 to travel to customer 1. Customer 1 must be visited by this vehicle at time 12. This is possible since the travel time between customers 10 and 1 ($D_{1,10}$) is 3 time periods which is equal to the visit time gap of 12 – 9 = 3. At time 12, vehicle 1 continues its route and leaves customer 1 to travel to customer 7, arriving at time 15. This journey takes 15 – 12 = 3 periods. The minimum travel time from customer 1 to 7 ($D_{1,7}$) is 3 time periods which makes this visit possible. Vehicle 1 stays at customer 7 for another period until time 16 and must be at customer 8 at time 19. The vehicle is able to catch up with that visit considering the travel time of 3 between customers 7 and 8. The next stop for vehicle 1 is customer 9 at time 20. Then, vehicle 1 leaves customer 9 at time 20 to travel to customer 2 and must arrive at time 23 which is possible based on the travel time of 3 between these two customers. The next visit for vehicle 1 is customer 10 at time 2 which completes the cycle.

We now turn our attention to vehicle 2's route. Vehicle 2 starts its route at customer 3 at time 1 and must make a visit to customer 4 at time 2. The travel time of only 1 period between these two customers makes this visit possible. It will then travel to customer 2 to make a visit at time 4 which is feasible because $D_{1,10} = 2$. It then travels to customer 3 to arrive at time 6 and from there it travels to customer 6 to make a visit at time 7. The feasibility of the rest of vehicle 2's route and vehicle 3's route can be verified in the same manner.

2.5. Mathematical formulation

A binary integer program named VRP-FRV was developed to model this problem. Table 14 presents the indices, parameters, and decision variables in Math Model VRP-FRV.

Table 14. Indices, parameters, and decision variables in Math Model VRP-FRV

Indices	
c,d	customers ($c, d = 1$ to C)
t, u	time period ($t, u = 1$ to T)
v	vehicles ($v = 1$ to V)
Input parameters	
C	Number of customers
V	Number of vehicles
T	Cycle length for the system (integer > 0)
CT_c	Maximum inter-visit time that can be tolerated by customer *c* (integer, ≥ 2)
D_{cd}	Travel time (in time periods) from customer *c* to *d* for each vehicle (integer, ≥ 1)
Decision variables	
$X_{vct} \begin{cases} 1 \\ 0 \end{cases}$	If vehicle *v* visits customer *c* at the start of time interval t ($v = 1$ to V, $c = 1$ to C, $t = 1$ to T) Otherwise (binary)
$Y_v \begin{cases} 1 \\ 0 \end{cases}$	If vehicle *v* is used at all ($v = 1$ to V) Otherwise (binary)

Math model VRP-FRV is shown in Table 15. The objective of the model is to minimize the number of vehicles that are needed in order to satisfy the maximum customer inter-visit times. Constraint (21) indicates that in order for a vehicle to visit a customer, it must be selected as one of the vehicles that are used. Constraint (22) satisfies the inter-visit time requirement for each customer *c*. In other words, it ensures that customer *c* is visited at least once during every time window of CT_c periods. Constraints (23) and (24) indicate that travel times between customers (D_{cd}) need to be respected. Constraint (23) specifies that a vehicle can only visit customer *c* at time *t* and customer d at time u ($u > t$) if $u - t$ is greater than or equal to the travel time from customer *c* to customer d (D_{cd}). Constraint (24) complements constraint (23) and considers the cyclic aspect of the problem. In particular, constraint (24) specifies that a vehicle can only visit customer d at time *u* and customer *c* at time *t* ($u > t$) if ($t + T$) – u is greater than or equal to the travel time from customer d to customer *c* (D_{dc}). Constraint (25) indicates that each vehicle *v* can visit no more than one customer at each time *t*. Constraint (26) ensures that lower index vehicles are used. In other words, if vehicle *v* is not used, vehicle *v* + 1 is also not used. This eliminates symmetries and redundant solutions, allowing a commercial solver to find optimal solutions more quickly.

Table 15. Math Model VRP-FRV

$$Min \sum_{v=1}^{V} Y_v \tag{20}$$

Subject to:

$$X_{vct} \leq Y_v \qquad \forall v, \forall c, \forall t \tag{21}$$

$$\sum_{v=1}^{V} \sum_{u=t}^{t+CT_c-1} X_{v,c,((u-1)\, mod\, T)+1} \geq 1 \qquad \forall c, \forall t \tag{22}$$

$$(X_{vct} + X_{vdu} - 1)\, D_{cd} \leq u - t \qquad \forall v, \forall c, \forall t, \forall d, \forall u > t \tag{23}$$

$$(X_{vct} + X_{vdu} - 1)\, D_{dc} \leq (t + T) - u \qquad \forall v, \forall c, \forall t, \forall d, \forall u > t \tag{24}$$

$$\sum_{c=1}^{C} X_{vct} \leq 1 \qquad \forall v, \forall t \tag{25}$$

$$Y_v \geq Y_{v+1} \qquad \forall v = 1\ to\ v - 1 \tag{26}$$

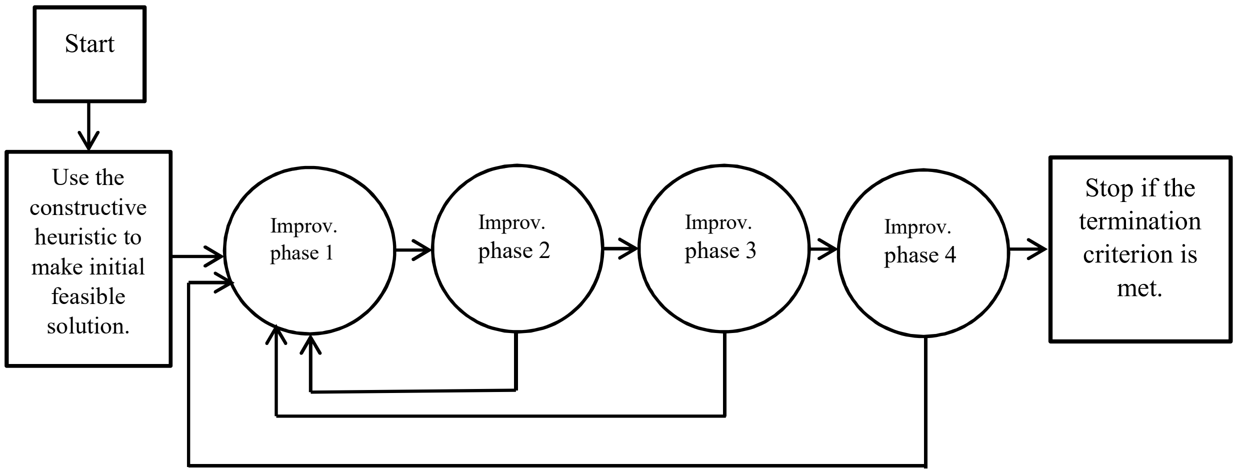
2.6. Heuristic algorithm

This problem appears to be NP-hard, so it is necessary to develop an efficient heuristic algorithm for solving large instances. In this section, we provide a high-level overview, medium-level summary, and detailed description of the heuristic.

The overall heuristic procedure that we developed begins with a constructive heuristic that creates an initial feasible solution. This is followed by several local search improvement procedures. After a termination criterion is met, the process restarts with a new initial feasible solution. This is repeated for thousands of restarts while time has not expired. Figure 5 shows the general procedure for each restart.

We now describe each part of the heuristic in a moderate level of detail. The construction of the initial feasible solution is a two-step process. First, we create a visit schedule for each customer *c* that satisfies CT_c. Next, we create naïve vehicle routes in which all vehicles do not move and stay at the same customer. This is done by assigning each customer's visits to one separate vehicle. There are four local search improvement procedures, i.e. improvement phases. All improvement phases form neighboring solutions by transferring individual customer visits between vehicles. During improvement phase 1, we transfer visits from vehicles that make fewer visits to vehicles that make more visits. Once all such possibilities are exhausted, the process moves to phase 2. During phase 2, in order to diversify the search, we transfer visits between vehicles with the same number of customer visits. Once all such transfers are made, the process moves to phase 3. In phase 3, we allow transfers from vehicles with more visits to vehicles which make fewer visits, with specific conditions. After all possible transfer are made, we move to phase 4 where transfers are allowed from vehicles which make more visits to vehicles which make fewer visits, with no restrictions. Once a certain number of attempts have been made, the procedure terminates and begins with a new initial solution characterized by entirely new customer visit schedules.

Figure 5. Summary of the procedure for each restart in the heuristic



Start
Use the constructive heuristic to make initial feasible solution.
Improv. phase 1
Improv. phase 2
Improv. phase 3
Improv. phase 4
Stop if the termination criterion is met.

We now provide all details of each part of the heuristic. To start the heuristic, we create an initial feasible solution. We now present a method for automatically generating an initial feasible solution. As described in Section 2.4, a feasible solution is characterized by (i) a visit schedule for each customer c, and (ii) a set of feasible vehicle routes that visit the customers at the times specified in (i). A feasible visit schedule for customer c is specified by two elements: 1) *visit start point*, 2) *inter-visit times.* A *visit start point* is the first visit time for customer c which is a random time from 1 to T. *Inter-visit times* represent the time periods between consecutive visits for customer c beginning with the *visit start point*. In what follows, we describe how to create random inter-visit times for customer c. First, the minimum number of visits that has to be made to customer c is computed which is $MinNumVisits_c = \left\lceil \frac{T}{CT_c} \right\rceil$. Then we set maximum number of visits that is possible to make for customer c as $MaxNumVisits_c$ = Min $\{MinNumVisits_c+2, T\}$ in order to allow flexibility in number of visits to that customer. A random number of visits ($RandNumVisits_c$) is then selected randomly in the range [$MinNumVisits_c$, $MaxNumVisits_c$]. Next, $RandNumVisits_c$ inter-visit times are created. This is done by initially setting each inter-visit time to CT_c and then repeatedly reducing by one the value of a randomly selected inter-visit time until the sum is T. Then, the visit schedule is made using the visit start point and inter-visit times.

Now, we describe phase 1 of the heuristic procedure. In phase 1, we explore all possible transfers of visits from vehicles which make fewer visits to vehicles which make more visits. To do this, we first make a sorted list of vehicles according to the increasing number of customer visits that they make. In such a sorted list, a lower index vehicle makes fewer visits than a higher index vehicle. Suppose that there are n vehicles. Let us denote the vehicles in the sorted list as (v_1, v_2, ...,v_n). We refer to the vehicle from which we attempt to transfer a visit, as the giving vehicle. The vehicle to which we try to move a visit is also referred to as the accepting vehicle. The first giving and accepting vehicles are the first low index vehicle (v_1) and the first high index vehicle (v_n) in the sorted list of vehicles, respectively. If no transfer is possible between those two vehicles, we keep the giving vehicle the same vehicle (v_1) and change the accepting vehicle to the next high index vehicle in the list (v_{n-1}). We keep doing this until a transfer is made or no transfer is made from v_1 to all other higher index vehicles (v_2, ...,v_n). If this happens, we then consider the second low index vehicle in the list (v_2) as the giving vehicle and attempt transferring a visit from that vehicle to a higher index vehicle starting from v_n, in the same manner that was discussed earlier. In the following paragraph, we describe in details how to approach transferring a visit from a giving vehicle to an accepting vehicle.

Consider a giving vehicle. Suppose that this vehicle currently makes J visits. Let us denote the j^{th} visit by cv_j. So, the list of customer visits for the giving vehicle is (cv_1, cv_2 ,..., cv_J). We first attempt to transfer the first customer visit (cv_1) from that vehicle to the accepting vehicle. If such a transfer is not possible, we move to the next customer visit (cv_2) for the giving vehicle and consider transferring that to the accepting vehicle. We try until we can transfer a visit from the giving vehicle to the accepting vehicle or we have tried all customer visits (cv_1, cv_2 ,…, cv_J) of the giving vehicle and no transfer is possible. We now describe how to investigate if a visit transfer is possible between two vehicles in the next paragraph.

Consider a customer visit time as t which belongs to customer c is to be transferred from vehicle v_1 to vehicle v_2. Suppose that vehicle v_2 currently makes J visits. Let us denote vehicle v_2's customer visit times as (t'_1, t'_2 ,…, t'_J). If t is equal to any of v_2's customer visits, this indicates that v_2 already makes a visit to a customer at that time and it is not able to make a visit to another customer at that time. Therefore, such a transfer is not possible. Otherwise, we need to evaluate if this transfer is possible considering the travel time between customers for the accepting vehicle v_2. Suppose that t falls between customer visit times t'_3 and t'_4. Also, suppose that visit time t'_3 relates to customer d and visit time t'_4 relates to customer u. This means that vehicle v_2 must be able to make a visit to customer d at time t'_3, then leaves customer d to customer c to arrive at time t and then leaves customer c to customer u to arrive at time t'_4. We need to check two criteria: 1) compute t - t'_3 . If this is greater than or equal to the travel time D_{dc}, this ensures that vehicle v_2 is able to leave customer d at time t'_3 and arrive at customer c at time t. 2) compute t'_4 - t. If this is greater than or equal to the travel time D_{cu}, this ensures that vehicle v_2 is able to leave customer c at time t and arrive at customer u at time t'_4. Both these conditions need to be satisfied for such a transfer to be possible.

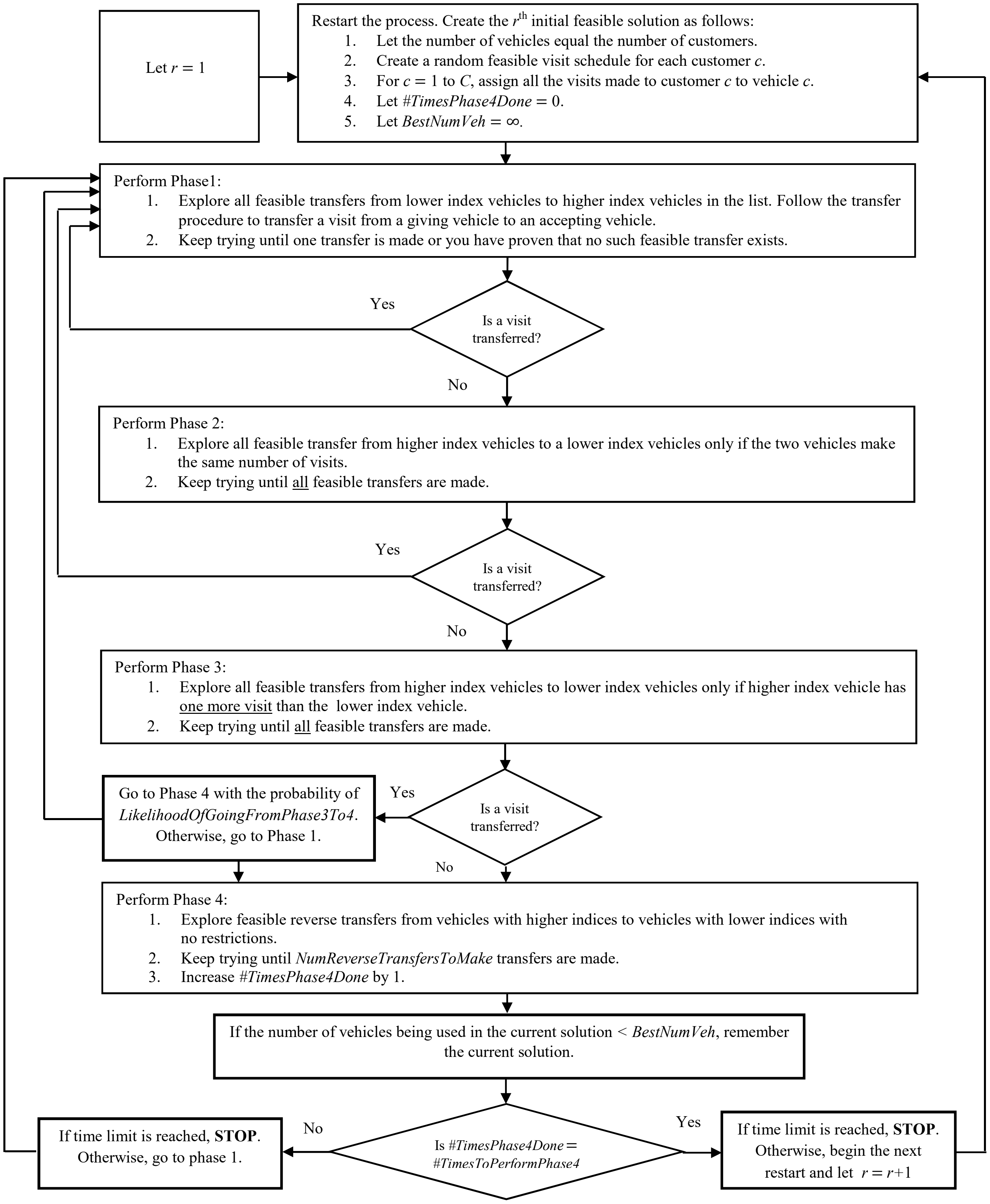
Note that when an attempt to transfer a visit is not successful, we re-evaluate the case to find out if changing that visit time makes the transfer possible. To do this, (i) we make a list of all feasible times for that visit, and (ii) try to transfer a particular visit from the giving vehicle v_1 to the accepting vehicle v_2 using visit times in the list until we find a possible transfer or we have tried all visit times in the list and no such transfer is possible. Part (i) is done with respect to satisfying the inter-visit requirement CT_c for customer c for which transferring the visit is being evaluated.

In phase 2, we explore all possible transfers from higher index vehicles to lower index vehicles that have the same number of visits to diversify the search. First, we create a sorted list of vehicles in the order of increasing in number of visits made. Suppose that there are *n* vehicles. Let us denote the vehicles in the sorted list as (v_1, v_2, ..., v_n). We start evaluating from the beginning of the list to see if each two consecutive vehicles have the same number of visits. In other words, we first check if v_1 and v_2 have the same number of visits. If so, we explore transferring a visit from v_2 to v_1. If such a transfer is not possible or these two do not make the same number of visits, we move forward in the list and evaluate if v_2 and v_3 have the same number of visits. We continue in the same manner until a transfer is made or we have covered the entire list and no such transfer is possible. Note that throughout phase 2, we need to re-sort the list of vehicles after each transfer is made.

Phase 3 is designed to make neutral moves which help to avoid being stuck in local solutions and add variety to feasible solutions. In other words, in phase 3, there is no progress in reducing number of vehicles and there is no regression. We explore all possible visit transfers from higher index vehicles to lower index vehicles only if the higher index vehicle has one more visit than the lower index vehicle. Note that, we do not re-sort the list of vehicles after each transfer is made to prevent repeating the opposite transfers and being stuck in a loop. If there is no possible transfer by the end of phase 3, we certainly move to phase 4. If there has been at least one successful transfer in phase 3, we perform phase 4 with a pre-determined probability (*LikelihoodOfGoingFromPhase3To4*) or repeat phase 1 with the probability of (1-*LikelihoodOfGoingFromPhase3To4*).

To start phase 4, we need to re-sort the list of vehicles if any success was made in phase 3. Phase 4 is considered an uphill move to create variety in the solutions. In this phase, we make a certain number of reverse transfers (*NumReverseTransfersInPhase4*) from higher index vehicles to lower index vehicles with no restrictions. We re-sort the list of vehicle once a transfer is made each time. Once we are done with phase 4, we evaluate how many times this phase has been done. If Phase 4 has been performed for a certain number of times (*#TimesToPerformPhase4*), the best solution so far is stored, the current restart is finished, and a new restarts begins. Each restart begins with a different initial feasible solution. A flowchart of the proposed heuristic is also presented in Figure 6. Table 16 provides brief descriptions for each of the four phases.

Figure 6. Flowchart of the heuristic algorithm



Let $r = 1$

Restart the process. Create the r^{th} initial feasible solution as follows:

1. Let the number of vehicles equal the number of customers.
2. Create a random feasible visit schedule for each customer c.
3. For $c = 1$ to C, assign all the visits made to customer c to vehicle c.
4. Let *#TimesPhase4Done* = 0.
5. Let *BestNumVeh* = ∞.

Perform Phase1:

1. Explore all feasible transfers from lower index vehicles to higher index vehicles in the list. Follow the transfer procedure to transfer a visit from a giving vehicle to an accepting vehicle.
2. Keep trying until one transfer is made or you have proven that no such feasible transfer exists.

Perform Phase 2:

1. Explore all feasible transfer from higher index vehicles to a lower index vehicles only if the two vehicles make the same number of visits.
2. Keep trying until all feasible transfers are made.

Perform Phase 3:

1. Explore all feasible transfers from higher index vehicles to lower index vehicles only if higher index vehicle has one more visit than the lower index vehicle.
2. Keep trying until all feasible transfers are made.

Go to Phase 4 with the probability of *LikelihoodOfGoingFromPhase3To4*. Otherwise, go to Phase 1.

Perform Phase 4:

1. Explore feasible reverse transfers from vehicles with higher indices to vehicles with lower indices with no restrictions.
2. Keep trying until *NumReverseTransfersToMake* transfers are made.
3. Increase *#TimesPhase4Done* by 1.

If the number of vehicles being used in the current solution < *BestNumVeh*, remember the current solution.

If time limit is reached, **STOP**. Otherwise, go to phase 1.

If time limit is reached, **STOP**. Otherwise, begin the next restart and let $r = r+1$

Table 16. Summary of the different solution improvement phases in the heuristic

Phase	Action
1	Explore *all* feasible visit transfers from lower index vehicles to higher index vehicles.
2	Explore *all* feasible transfers from higher index vehicles to lower index vehicles that make the same number of visits.
3	Explore *all* feasible transfers from higher index vehicles to lower index vehicles only if higher index vehicle has one more visit than the lower index vehicle.
4	Make *no more than NumReverseTransfersToMake* transfers from higher index vehicles to lower index vehicles with no restrictions.

We now explain the heuristic procedure using the instance defined in Tables 9-10 in Section 2.4. The minimum number of visits to customer 1 is $MinNumVisits_1 = \left\lceil \frac{24}{14} \right\rceil = 2$ and the maximum number of visits is $MinNumVisits_1 + 2 = 4$. Assume that $RandNumVisits_1$ is 3. In this case, the three inter-visit times would each be initially set to $CT_1 = 14$ and then repeatedly reduced until they sum to $T = 24$. For example, the inter-visit times for customer 1 could be: (10, 7, 7). A random visit start point from 1 to T is then selected. If the visit start point were 7, customer 1 would be visited at times (7, 17, 24). We randomly create visit times for each customer in the same manner.

Note that ten vehicles are used initially to make the required repeated visits to the ten customers. An initial feasible set of vehicle routes is then made by assigning all customer 1's visits to vehicle 1, customer 2's visits to vehicle 2 and so on until customer 10's visits are assigned to vehicle 10. Tables 17-18 show a set of feasible routes that visit the customers at the times specified based on the discussion in the previous paragraph. Table 17 shows the initial *VisitTime* matrix which represents the time when each vehicle visits a customer. Table 18 shows the initial *WhoIsVisited* matrix that represents which customer is visited at each time specified in Table 17. For example, vehicle 2 visits customer 2 at times (5, 9, 14, 17, 21).

Table 17. Initial *VisitTime* matrix

Vehicle 1	7	17	24				
Vehicle 2	5	9	14	17	21		
Vehicle 3	5	10	15	19	24		
Vehicle 4	3	15					
Vehicle 5	7	10	18	24			
Vehicle 6	2	14					
Vehicle 7	7	13	15	18	22		
Vehicle 8	4	16					
Vehicle 9	6	18					
Vehicle 10	2	7	8	13	15	19	21

Table 18. Initial *WhoIsVisited* matrix

Vehicle 1	1	1	1				
Vehicle 2	2	2	2	2	2		
Vehicle 3	3	3	3	3	3		
Vehicle 4	4	4					
Vehicle 5	5	5	5	5			
Vehicle 6	6	6					
Vehicle 7	7	7	7	7	7		
Vehicle 8	8	8					
Vehicle 9	9	9					
Vehicle 10	10	10	10	10	10	10	10

The initial feasible solution is stored in the best solution found so far (*BestSol)* and best number of vehicles so far*, BestNumVeh* is 10. Prior to each phase, vehicles are sorted according to the increasing total number of visits that they make.

To start phase 1 for this instance, the list of number of total visits for each vehicle *c* =1,…, 10 is (3, 5, 5, 2, 4, 2, 5, 2, 2, 7). Therefore, the sorted list of vehicles is (4, 6, 8, 9, 1, 5, 2, 3, 7, 10). In phase 1, we explore a feasible transfer from lower index vehicles to higher index vehicles. The first giving and accepting vehicles are the first low index vehicle and the first high index vehicle (4,10). If no transfer is possible from vehicle 4 to vehicle 10, we consider the accepting vehicle as the second high index vehicle in the list (vehicle 7). Again, if we find no possible transfer, the next high index vehicle in the list will be the candidate for the accepting vehicle (vehicle 3) and so on. In other words, the order of selected pairs of giving and accepting vehicles based on the list until we find a successful transfer from vehicle 4 is (4,10), (4,7), (4,3), (4,2), (4,5), (4,1), (4,9), (4,8), and (4,6). If no transfer is possible from vehicle 4 to other higher index vehicles, we move to the next low index vehicle as for the giving vehicle (vehicle 6). Again, the order of selected pairs of giving and accepting vehicles based on the list until we find a successful transfer from vehicle 6 is (6,10), (6,7), (6,3), (6,2), (6,5), (6,1), (6,9), and (6,8). For each selected pair, we attempt to move the first visit from the giving vehicle to the accepting vehicle. If that was not possible, we try to move the second visit from the giving vehicle to the accepting vehicle. We continue until a transfer is made or we have tried all visit times for the giving vehicle and none are possible.

The first pair of giving and accepting vehicles based on the sorted list of vehicles is (4, 10). In other words, we try to make a transfer from vehicle 4 to vehicle 10. The visit times for vehicle 4 in the $VisitTime$ matrix are (3, 15). We then try to transfer the first visit for vehicle 4 which belongs to customer 4 at time 3. The customers that vehicle 10 currently visits based on the $WhoIsVisited$ matrix is (10, 10, 10, 10, 10, 10, 10) and the current visit schedule for vehicle 10 is to visit customer 10 at times (2, 7, 8, 13, 15, 19, 21). If customer 4's visit at time 3 is transferred to vehicle 10, customer 4 will be part of the list of customers that vehicle 10 serves in the $WhoIsVisited$ matrix: (10, **4**, 10, 10, 10, 10, 10, 10). We need to check to find out if the visit at time 3 can fit into vehicle 10's schedule between time periods 2 and 7. In this case, the potential visit schedule would be (2, **3**, 7, 8, 13, 15, 19, 21). We go through the transfer procedure and check to see whether (i) there is enough time for vehicle 10 to travel from the customer 10 at time 2 and arrive at customer 4 at time 3 and (ii) there is enough time for vehicle 10 to travel from the customer 4 at time 3 and be at customer 10 at time 7. These are evaluated using the travel time between customers 4 and 10 ($D_{4,10}$). Vehicle 10 is visiting customer 10 at time 2 and it should next arrive at customer 4 at time 3. Therefore the visit gap is 1 time period between the two customers. This is not possible since $D_{4,10} =$ 4 time periods which is greater than the visit gap of 1 time period.

We then investigate further if we can change the visit time of 3 for customer 4 to make it transferable. We find all possible visit times that customer 4 can have instead of time 3.

Customer 4's inter-visit requirement is 15 time periods ($CT_4 = 15$). It is currently visited at times 3 and 15. The resulting inter-visit times are (12, 12). Each of these values is less than $CT_4 = 15$. This means that the visit at time 3 is flexible to move up to 3 periods earlier or be pushed up to 3 periods later. Therefore, the list of all feasible visit times for customer 4 with respect to time 3 is (24, 1, 2, 4, 5, 6). We then scramble this list and start with the first element in the list to figure out if that is possible to transfer. If that is possible, a transfer is made and we repeat phase 1 starting with making a sorted list of vehicles in terms of busyness. Otherwise, if a transfer is not possible, we will test the second value in the list. We will continue to test the visit times in the list until we find a possible visit transfer or we have tried all visit times in the list and none of them is possible. For this list, there is no visit time that is possible considering the travel time between customers 10 and 4.

Since moving the first visit of vehicle 4 at time 3 is not possible, we investigate moving the second visit time of vehicle 4 which belongs to customer 4 at time 15. The accepting vehicle 10 already has a visit at time 14 to the other customer 10. Therefore, this transfer is not possible. So far, we evaluated transferring all visit times from the giving vehicle 4 to the accepting vehicle 10 and no transfer is possible. Therefore, we keep the same giving vehicle 4 and change the accepting vehicle to the next high index vehicle in the list which is vehicle 7 and evaluate the possibility of transferring a visit in the same way that was explained earlier. In summary, we keep trying phase 1 until a transfer is made or we have tried all cases but no such possible transfer exists. If a transfer was successfully made, we repeat phase 1 starting with sorting vehicles based on their updated number of visits. Otherwise, if no transfer is made, we move to phase 2.

In phase 2, we explore all possible transfers from higher index vehicles to lower index vehicles that have the same number of visits. Suppose that for this instance, no transfer was made in phase 1 and therefore the number of visits for each vehicle (3, 5, 5, 2, 4, 2, 5, 2, 2, 7) and the sorted list of vehicles (4, 6, 8, 9, 1, 5, 2, 3, 7, 10) are not updated. In this case, the first two candidates as giving and accepting vehicles are vehicles 6 and 4, respectively, since both serve 2 visits. We attempt to transfer all visits (both visit times 2 and 14) from vehicle 6 to 4. If no transfer is possible from vehicle 6 to 4, we move through the rest of the sorted list of vehicles and find that vehicles 6 and 8 also make equal number of visits (2 visits). Therefore, we explore possible transfers from vehicle 8 to vehicle 6 and so on. Note that throughout this process, once a transfer is made, the sorted list of vehicles needs to be updated. Phase 2 ends when all possible transfers are made based on the sorted list of vehicles or all transfers have been evaluated and no such possible transfer exists. If we do not gain any success in phase 2, we start phase 3. In what follows, we explain phase 3.

In phase 3, we explore all possible visit transfers from higher index vehicles to lower index vehicles only if the higher index vehicle has one more visit than the lower index vehicle. Phase 3 is started if no transfer is made in phases 1 and 2. Suppose that no success was made in phase 1 and phase 2. In that case, the number of visits for each vehicle (3, 5, 5, 2, 4, 2, 5, 2, 2, 7) and the sorted list of vehicles (4, 6, 8, 9, 1, 5, 2, 3, 7, 10) are not updated. For this case, we first consider transferring a visit from vehicle 6 to 4. These two are not eligible since both make same number of visits. Next, we consider transferring a visit from vehicle 8 to 6. These two are also not eligible since both make same number of visits. We first two vehicles eligible in phase 3 are vehicles 1 and 9. Vehicle 1 makes 3 visits and vehicle 9 makes 2 visits. So, we may start investigating if a transfer can be made from vehicle 1 to vehicle 9. Note that we do not re-sort the list of vehicles after each transfer is made to prevent repeating the opposite transfers and being stuck in a loop. We continue in this manner and if there is no possible transfer by the end of phase 3, we certainly move to phase 4. If there has been at least one successful transfer in phase 3, we perform phase 4 with a probability of *LikelihoodOfGoingFromPhase3To4* or repeat phase 1 with the probability of (1- *LikelihoodOfGoingFromPhase3To4*).

To start phase 4, we need to re-sort the list of vehicles if any success was made in phase 3. In this phase, we make a certain number of reverse transfers (*NumReverseTransfersInPhase4*) from higher index vehicles to lower index vehicles with no restrictions. One possibility of getting to phase 4 is the case where no transfer was made in phases 1, 2 and 3. So, let us suppose that number of visits for each vehicle (3, 5, 5, 2, 4, 2, 5, 2, 2, 7) and the sorted list of vehicles (4, 6, 8, 9, 1, 5, 2, 3, 7, 10) are not updated. We first attempt to make a transfer from vehicle 6 to vehicle 4, then from vehicle 8 to vehicle 6, then from vehicle 9 to vehicle 8 and so on until *NumReverseTransfersInPhase4* transfers are made. Note that, we re-sort the list of vehicle once a transfer is made each time. Once we are done with phase 4, we evaluate how many times this phase has been done. If phase 4 has been performed for *#TimesToPerformPhase4* times*,* the best solution so far is stored in *BestSol* and the current restart ends.

2.7. Experimental setup, results, and discussion

We performed two sets of experiments which consider two methods for solving the math model: (1) integer programming using IBM ILOG CPLEX 12.9; (2) the heuristic algorithm described in Section 2.6.

2.7.1. Generating problem instances

Tables 19-20 show the parameter values that define the problem instances considered in the experiments. We use the discrete uniform (DU) distribution to randomly generate the inter-visit time requirement CT_c for each customer. The travel times between customers are computed by a three-step process. First, we randomly generate imaginary (x, y) coordinates for each customer. These either follow the U(0,5) or U(0,10) distribution. Second, we compute the distance between each pair of customers using the Pythagorean Theorem. Based on this distance formula, the travel time between two points with coordinates (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Then, the resulting non-integer value is rounded up to the nearest integer.

Table 19. Parameter values used in experiment 1

Parameter	Possible values
C	10, 30, 50
T	24, 50, 100
CT_c	DU(5,15)
x, y coordinates	U(0,5)
D_{cd}	Integers from 1 to 8

Table 20. Parameter values used in experiment 2

Parameter	Possible values
C	10, 30, 50
T	24, 50, 100
CT_c	DU(5,15)
x, y coordinates	U(0,10)
D_{cd}	Integers from 1 to 15

We consider nine problem sizes in experiments 1 and 2: small, medium and large. These problem sizes correspond to all combinations of three possible numbers of customers−10, 30 and 50 − and three values for the cycle time (T) − 24, 50 and 100. Each problem size is displayed as a seven-digit code "CccTttt". "cc" represents the number of customers and "ttt" indicates the cycle length. For each problem size, we consider ten different instances. Therefore, 90 different instances were considered in total for each experiment. Experiment 1 considers the case where the (x,y) coordinates of each customer's location are generated using a uniform distribution from 0 to 5. Experiment 2 considers the case where the (x,y) coordinates of each customer are generated using a uniform distribution from 0 to 10.

2.7.2. Hardware settings, CPLEX settings, heuristic settings, and termination criteria

The math model from Section 2.5 and the heuristic algorithm from Section 2.6 were coded into MS Visual C++ 2017 Professional. IBM ILOG Concert Technology was used to code the math model in C++ and call IBM ILOG CPLEX 12.9 to solve instances defined in text files. All results are obtained using a desktop computer with 16 gigabytes of RAM, the Windows 10 Education 64-bit operating system, and an Intel Core i7-8700 processor with 3.2 gigahertz processors. The CPLEX-based method terminates after 3600 seconds has elapsed. To reduce CPLEX runtime, V is set to a value much less than C when the math model is created and solved by CPLEX. In particular, V is set to (10, 15, 20) when C equals (10, 30, 50), respectively. All other CPLEX parameters are set to their default values. The termination criteria and settings for the heuristic algorithm are reported in Table 21.

Table 21. Parameter values used for the heuristic method

Parameter	Value
NumTimesToPerformPhase4	5
NumReverseTransfersInPhase4	3
LikelihoodOfGoingFromPhase3To4	0.5
Run time for small problems with c = 10	120 sec
Run time for medium-sized problems with c = 30	300 sec
Run time for large problems with c = 50	600 sec

2.7.3. Results and discussion

Table 22 shows the CPLEX results for experiment 1. For each of the 90 instances considered, this table shows the number of vehicles used in the best feasible solution found by CPLEX within the time limit and the time (in seconds) used by CPLEX. The average number of vehicles used for each problem size is also shown. The bold highlighted values show optimal solutions identified by CPLEX. The value "N/A" in the table means that CPLEX ran out of memory due to the complexity of the problem and no feasible solution was found prior to termination. Therefore, the run time for instances where CPLEX ran out of memory is not reported and is presented as "*". Table 23 shows the results for the heuristic method by experiment 1. The structure of this table is similar to Table 22 except that the runtime information is not provided because runtimes are predetermined. We next discuss the results for different problem sizes and the performance of solution methods.

For instances with 10 customers, CPLEX was successful in finding optimal solutions for all 10 instances of problem size C10T024 within less than an hour. For problem size C10T050, CPLEX was partially successful in solving problems to optimality within an hour. It identified an optimal solution for 5 out of 10 such instances in less than an hour. This is expected since for the same number of customers, as the cycle time increases, the problem becomes more complex as it is more likely to involve more visits to customers. For the C10T100 problem size, CPLEX only reported one optimal solution for the first instance in 396 seconds, and it was able to find feasible solutions for other 9 instances as displayed in Table 22. The CPLEX results for small problem instances indicate that the problem at hand is very complex since CPLEX is not able to find optimal solutions for many instances. Results also indicate that the average number of required vehicles increases across the three discussed problems. In particular, that increases from 3 vehicles for problem size C10T024 to 3.3 vehicles for problem size C10T050 and finally to 3.7 vehicles as for problem size C10T100. Either more vehicles are needed to satisfy more repeating visits during a longer cycle time or the higher complexity of the problem reduces CPLEX's effectiveness.

The heuristic results in Table 23 show that the heuristic method finds solutions of the same quality as CPLEX for all instances of size C10T024 and C10T050. Moreover, the heuristic method is able to find a lower average number of vehicles (3.4) within 120 seconds for the

C10T100 problem size than CPLEX with a 1-hour limit. This indicates that the heuristic method is outperforming CPLEX for the small problem instances in experiment 1.

Table 22. CPLEX results for experiment 1

Problem Size		Instance										Average
		1	2	3	4	5	6	7	8	9	10	
C10T024	#vehicles	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	3
	run time(s)	**15**	**457**	**339**	**13**	**17**	**712**	**9**	**21**	**74**	**57**	
C10T050	#vehicles	3	**3**	4	4	**3**	**3**	**3**	**3**	4	3	3.3
	run time(s)	3602	**126**	3609	3606	**156**	**1632**	**335**	**1995**	3602	3602	
C10T100	#vehicles	**3**	4	4	3	4	4	3	4	4	4	3.7
	run time(s)	**396**	3608	3607	3608	3608	3609	3608	3608	3609	3609	
C30T024	#vehicles	8	9	15	8	7	8	8	8	8	9	8.8
	run time(s)	3606	3607	3608	3606	3607	3606	3607	3607	3606	3607	
C30T050	#vehicles	9	8	14	8	8	8	9	8	9	9	9
	run time(s)	3717	3651	3712	3671	3659	3659	3653	3732	3789	3658	
C30T100	#vehicles	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	run time(s)	*	*	*	*	*	*	*	*	*	*	
C50T024	#vehicles	13	14	12	11	12	12	14	13	11	12	12.4
	run time(s)	3621	3616	3618	3618	3620	3625	3618	3619	3626	3618	
C50T050	#vehicles	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	run time(s)	*	*	*	*	*	*	*	*	*	*	
C50T100	#vehicles	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	run time(s)	*	*	*	*	*	*	*	*	*	*	

Bold: Optimal solution   *: Out-of-memory

Table 23. Heuristic results for experiment 1

Problem Size		Instance										Average
		1	2	3	4	5	6	7	8	9	10	
C10T024	#vehicles	3	3	3	3	3	3	3	3	3	3	3
C10T050	#vehicles	3	3	4	4	3	3	3	3	4	3	3.3
C10T100	#vehicles	3	4	4	3	3	4	3	3	4	3	3.4
C30T024	#vehicles	8	9	9	8	7	9	7	8	8	9	8.2
C30T050	#vehicles	9	8	9	7	8	8	8	8	8	9	8.2
C30T100	#vehicles	9	9	9	8	8	8	9	8	9	8	8.5
C50T024	#vehicles	11	13	13	11	12	13	12	12	11	11	11.9
C50T050	#vehicles	11	13	12	11	14	13	13	12	12	13	12.4
C50T100	#vehicles	12	12	14	12	12	12	13	12	14	14	12.7

For instances with 30 customers, CPLEX reported no optimal solutions, and it reported feasible solutions after an hour only for the instances where $T = 24$ and $T = 50$. This agrees with intuition. As the problem size increases, we expect to see more difficulty for CPLEX in solving the problem. Also, as we move from C30T024 to C30T050 problem size, the average number of vehicles found in the best solution increases from 8.8 to 9. When the cycle time $T = 100$, CPLEX runs out of memory in all cases and is not able to find any feasible solutions prior to termination. The heuristic results in Table 23 show that the heuristic method outperforms CPLEX for these 30-customer instances. Indeed, the heuristic finds solutions that use fewer vehicles on average than CPLEX for each problem size: 8.2, 8.2, and 8.5 vehicles on average for problem sizes C30T024, C30T050, and C30T100 compared to 8.8, 9, and N/A for CPLEX respectively.

For large problem instances with 50 customers, CPLEX was not able to find optimal solutions for any instances, it ran out of memory and did not report any feasible solutions for all instances of size C50T050 and C50T100. The heuristic method significantly outperforms CPLEX for these instances. The heuristic method finds solutions that use an average of 11.9 vehicles for problem size C50T024 which is lower than the CPLEX result of 12.4. Also, the heuristic is able to find solutions that use an average of 12.4 and 12.7 vehicles for problem sizes C50T050 and C50T100 respectively, whereas CPLEX did not find any feasible solutions for these two problem sizes.

In summary, the proposed heuristic outperforms CPLEX in experiment 1. Indeed, the heuristic reports a lower average number of vehicles used for 7 out of 9 problem sizes, and it finds the same average number of vehicles as CPLEX for the other two problem sizes C10T024 and C10T050 within only 120 seconds. This indicates that the heuristic method is efficient in handling small, medium, and large problem instances within a short amount of time.

Table 24 shows the CPLEX results for experiment 2. This table has the same structure as Table 22. Table 25 shows the results for the heuristic method by experiment 2. The structure of this table is similar to Table 24 except that the runtime information is not provided because runtimes are predetermined. We discuss the performance of solution methods across the different problem sizes.

Table 24. CPLEX results for experiment 2

Problem Size		Instance										Average
		1	2	3	4	5	6	7	8	9	10	
C10T024	#vehicles	4	5	5	6	**4**	**4**	5	4	**4**	4	4.5
	run time(s)	3602	3615	3602	3601	**760**	**2001**	3602	3631	**3192**	3601	
C10T050	#vehicles	4	6	6	7	4	5	5	4	5	4	5
	run time(s)	3602	3602	3602	3602	3602	3602	3602	3602	3602	3602	
C10T100	#vehicles	5	6	7	7	5	6	6	9	5	6	6.2
	run time(s)	3609	3609	3607	3607	3606	3607	3607	3606	3607	3607	
C30T024	#vehicles	13	13	15	12	12	12	14	15	14	15	13.5
	run time(s)	3619	3625	3621	3615	3621	3615	3745	3621	3618	3650	
C30T050	#vehicles	14	15	15	13	15	13	15	15	14	14	14.3
	run time(s)	3645	3750	3392	3650	3760	3791	3364	3733	3818	3693	
C30T100	#vehicles	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	run time(s)	*	*	*	*	*	*	*	*	*	*	
C50T024	#vehicles	17	17	18	17	19	16	17	19	20	18	17.8
	run time(s)	3779	3727	3738	3817	3736	3764	3713	3698	3724	3734	
C50T050	#vehicles	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	run time(s)	*	*	*	*	*	*	*	*	*	*	
C50T100	#vehicles	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	run time(s)	*	*	*	*	*	*	*	*	*	*	

Bold: Optimal solution *: Out-of-memory

Table 25. Heuristic results for experiment 2

Problem Size		Instance										Average
		1	2	3	4	5	6	7	8	9	10	
C10T024	#vehicles	4	5	5	6	4	5	5	4	4	4	4.6
C10T050	#vehicles	5	6	6	6	4	5	5	4	5	4	5
C10T100	#vehicles	5	5	6	7	4	5	6	5	5	4	5.2
C30T024	#vehicles	12	12	13	12	13	12	12	14	14	15	12.9
C30T050	#vehicles	13	13	12	14	13	13	14	13	13	13	13.1
C30T100	#vehicles	13	14	13	14	13	15	14	14	14	14	13.8
C50T024	#vehicles	16	17	16	18	17	16	16	18	16	17	16.7
C50T050	#vehicles	17	18	18	17	17	18	17	16	17	16	17.1
C50T100	#vehicles	18	17	17	18	18	17	18	18	17	18	17.6

CPLEX was able to solve only three instances of problem size C10T024 to optimality. CPLEX finds only feasible solutions for 7 instances of problem size C10T024 and all 10 instances for the other two problem sizes C10T050 and C10T100. As we move across these three problem sizes with 10 customers; from C10T024 to C10T050 and C10T100, the average numbers of vehicles increases from 4.5 to 5 and to 6.2.

We now explain about the results for instances with 30 customers. For the first two problem sizes C30T024 and C30T050, CPLEX finds feasible solutions for all 10 instances within an hour. However, CPLEX runs out of memory for all C30T100 instances and is not able to find any feasible solutions. The heuristic method outperforms CPLEX for instances with 30 customers. It reports an average of 12.9 vehicles for problem size C30T024 which is lower than an average of 13.5 vehicles found by CPLEX. Also, the heuristic method finds an average of 13.1 vehicles for problem size C30T050 which is lower than the average of 14.3 vehicles reported by CPLEX for the same problem size. The heuristic also significantly outperforms CPLEX for problem size C30T100 as CPLEX does not find any feasible solution for 10 instances. We observe that the same pattern holds for instances with 10 customers. As the cycle time increases from 24 to 50 and from 50 to 100, the average number of vehicles needed increases. Either more vehicles are needed to satisfy more repeating visits during a longer cycle time, or the higher complexity of the problem reduces the CPLEX/heuristic effectiveness.

For instances with 50 customers, CPLEX was not able to find optimal solutions for any instances of problem size C50T024. Moreover, for all instances in the other two problem sizes C50T050 and C50T100, CPLEX ran out of memory and did not report any feasible solutions. The heuristic significantly outperforms CPLEX for instances with 50 customers. The heuristic finds on average 16.7 vehicles for problem size C50T024 which is lower than the CPLEX result of 17.8 vehicles on average. Also, the heuristic is able to find on average 17.1 and 17.6 vehicles for problem sizes C50T050 and C50T100 respectively, whereas CPLEX did not find any feasible solutions for these two problem sizes.

In summary for experiment 2, the proposed heuristic method outperforms CPLEX in 7 out of 9 problem sizes and it performs equally well as CPLEX in one problem size C10T050. This indicates that the heuristic method proves to be effective in solving all small, medium and large problem sizes within only a short run time.

2.8. Conclusion

We studied a new variant of the vehicle routing problem: the vehicle routing problem with flexible repeat visits (VRP-FRV). The VRP-FRV has several real-world applications. One scenario is that of caretakers who provide service to elderly people at home. Each caretaker is assigned a number of elderly people to visit one or more times per day. Elderly people differ in their requirements and the minimum frequency at which they need to be visited every day. The VRP-FRV is also important in the context of a police patrol routing problem where the customers are various locations in the city that require frequent observations. Such locations could include known high-crime areas, high-profile residences, and/or safe houses. Moreover, the VRP-FRV applies to the routing of preventive maintenance workers for a major utility company or manufacturer who need to check key system components on a regular basis.

The VRP-FRV problem considers a set of customers with certain locations and a set of available vehicles for serving customers. The novelty of this problem relates to the customer visit requirements which are of different types compared with the existing VRP studies with time windows for deliveries. In particular, each customer must be visited at least once during every time window of a certain length that can be formed during a cycle. Therefore, each vehicle may make repeated visits to the same customer within the same route. The operations are cyclic, repeating over a cyclic planning horizon. We developed a binary integer programing model to find the customer visit schedules and vehicle routes which minimize the total number of vehicles that are needed to meet the customers' requirements. We used CPLEX to solve the proposed math model and developed a heuristic algorithm to handle large problem instances. Two sets of experiments were performed. Overall, the heuristic outperformed CPLEX in both experiments which shows the heuristic's efficiency in solving the VRP-FRV.

Future research will focus on developing a bi-objective model which considers both minimizing the number of vehicles and minimizing the total distance traveled by these vehicles. Another possible extension is to include a depot in the model and require each vehicle start and end its route at the depot. Future work will also include investigating further the effectiveness/robustness of the heuristic algorithm by finding bounds for the objective function for different problem instances as well as comparing the results of the developed heuristic with other heuristic procedures such as Simulated Annealing and Tabu Search.

References

Alba Martínez, M. A., Cordeau, J. F., Dell'Amico, M., Iori, M. (2013). A branch-and-cut algorithm for the double traveling salesman problem with multiple stacks. *INFORMS Journal on Computing*, *25*(1), 41-55.

Alvarenga, G.B., Mateus, G.R, de Tomi, G. (2007). A Genetic and Set Partitioning Two-phase Approach for the Vehicle Routing Problem with Time Windows. *Computers and Operations Research, 34*, 1561– 1584.

An, B., Ordóñez, F., Tambe, M., Shieh, E., Yang, R., Baldwin, C., DiRenzo III, J., Moretti, K., Maule, B., Meyer, G. (2013). A deployed quantal response-based patrol planning system for the US Coast Guard. *Interfaces*, *43*(5), 400–420.

Arigliano, A., Ghiani, G., Grieco, A., Guerriero, E., Plana, I. (2019). Time-dependent asymmetric traveling salesman problem with time windows: Properties and an exact algorithm. *Discrete Applied Mathematics*, *261*, 28-39.

Avin, R., Agin, D., Adnan, M. (2012). Solving TSP using genetic algorithms—case of Kosovo. *Advances in Computer Science*, 256–260

Baldacci, R., Toth, P., Vigo, D. (2010). Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research*, *175*(1), 213-245.

Baldacci, R., Mingozzi, A., Roberti, R. (2012). New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS Journal on Computing*, *24*(3), 356-371.

BañOs, R., Ortega, J., Gil, C., FernáNdez, A., De Toro, F. (2013). A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. *Expert Systems with Applications*, *40*(5), 1696-1707.

Benavent, E., Martínez, A. (2013). Multi-depot multiple TSP: a polyhedral study and computational results. *Annals of Operations Research*, *207*(1), 7-25.

Bonomo, F., Mattia, S., Oriolo, G. (2011). Bounded coloring of comparability graphs and the pickup and delivery tour combination problem. *Theoretical Computer Science*, *412*(45), 6216–6268.

Bräysy, O., Gendreau, M. (2005a). Vehicle routing problem with time windows, part I: route construction and local search algorithms. *Transportation Science 39*(1):104–118

Bräysy, O., Gendreau, M. (2005b). Vehicle routing problem with time windows, part II: metaheuristics. *Transportation Science 39*(1):119–139

Carrabs, F., Cordeau, J. F., Laporte G. (2007). Variable neighbourhood search for the pickup and delivery traveling salesman problem with LIFO loading. *INFORMS Journal on Computing*. *19*, 618–632.

Cacchiani, V., Contreras-Bolton, C., Toth, P. (2020). Models and algorithms for the Traveling Salesman Problem with Time-dependent Service times. *European Journal of Operational Research*, *283*(3), 825-843.

Çapar, İ., Keskin, B. B., Rubin, P. A. (2015). An improved formulation for the maximum coverage patrol routing problem. *Computers and Operations Research*, *59*, 1-10.

Clarke, G., Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research, 12*(4), 568581.

Cordeau, J. F., Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, *37*(6), 579–594.

Cordeau, J. F., Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of Operations Research, 153*, 29–46.

Cordeau, J. F., Iori, M., Laporte G., Salazar-Gonzalez, J. J. (2010). A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. *Networks*, *55*, 46–59.

Dantzig, G. B., Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6*(1), 80-91.

Da Silva, R. F., Urrutia, S. (2010). A General VNS heuristic for the traveling salesman problem with time windows. *Discrete Optimization*, *7*(4), 203-211.

Dash, S., Günlük, O., Lodi, A., Tramontani, A. (2012). A time bucket formulation for the traveling salesman problem with time windows. *INFORMS Journal on Computing*, *24*(1), 132-147.

Detti, P., Papalini, F., de Lara, G. Z. M. (2017). A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. *Omega, 70*, 1–14

Dewinter, M., Vandeviver, C., Beken, T. V., Witlox, F. (2020). Analysing the Police Patrol Routing Problem: A Review. *ISPRS International Journal of Geo-Information*, *9*(3), 157.

Dewil, R., Vansteenwegen, P., Cattrysse, D., Van Oudheusden, D. (2015). A minimum cost network flow model for the maximum covering and patrol routing problem. *European Journal of Operational Research*, *247*(1), 27-36.

Dumas, Y., Desrosiers, J., Gelinas, E., Solomon, M. M. (1995). An optimal algorithm for the traveling salesman problem with time windows. *Operations research*, *43*(2), 367-371.

Erdoğan, G., Erkut, E., Ingolfsson, A., Laporte, G. (2010). Scheduling ambulance crews for maximum coverage. *Journal of the Operational Research Society*, *61*(4), 543-550.

Fang, F., Stone, P., Tambe, M. (2015). When security games go green: Designing defender strategies to prevent poaching and illegal fishing. *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Fischetti, M., Salazar González, J. J., Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, *45*(3), 378-394.

Foulds, L., Longo, H., Martins, J. (2015). A compact transformation of arc routing problems into node routing problems. *Annals of Operations Research*, *226*(1), 177-200.

Fukasawa, R., Longo, H., Lysgaard, J., De Aragão, M. P., Reis, M., Uchoa, E., Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, *106*(3), 491-511.

Golden, B. L., Raghavan, S., Wasil, E. A. (2008). The vehicle routing problem: latest advances and new challenges. *Springer Science and Business Media*, *43*.

Groër, C., Golden, B., Wasil, E. A. (2009). The consistent vehicle routing problem. *Manufacturing and service operations management, 11*(4), 630-643.

Ho, S. C., Szeto, W. Y., Kuo, Y. H., Leung, J. M., Petering, M., Tou, T. W. (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, *111*, 395-421.

Hsieh, Y. C., You, P. S., Lee, P. J., Lee, Y. C. (2015). A novel encoding scheme based evolutionary approach for the bi-objective grid patrol routing problem with multiple vehicles. *Scientia Iranica*, *22*(4), 1576-1585.

Jain, M., Tsai J., Pita J., Kiekintveld C., Rathi S., Tambe M., Ordóñez F. (2010). Software assistants for randomized patrol planning for the LAX Airport police and the federal air marshal service. *Interfaces*, *40*(4): 267–290.

Jepsen, M., Spoorendonk, S., Ropke, S. (2013). A branch-and-cut algorithm for the symmetric two-echelon capacitated vehicle routing problem. *Transportation Science*, *47*(1), 23-37.

Kallehauge, B. (2008). Formulations and Exact Algorithms For the Vehicle Routing Problem With Time Windows, *Computers and Operations Research 35*, 2307 – 2330.

Kara, I. (2010). Tightening Bounding Constraints of the Miller-Tucker-Zemlin Based Formulation of the Capacitated Vehicle Routing Problems and Some Extensions. *Proceeding of the 2nd International Conference on Manufacturing Engineering*, *Quality and Production Systems*, 137-142.

Kara, I., Derya, T. (2015). Formulations for minimizing tour duration of the traveling salesman problem with time windows. *Procedia Economics and Finance*, *26*, 1026-1034.

Karabulut, K., Tasgetiren, M. F. (2014). A variable iterated greedy algorithm for the traveling salesman problem with time windows. *Information Sciences*, *279*, 383-395.

Keskin, B. B., Li, S. R., Steil, D., Spiller, S. (2012). Analysis of an integrated maximum covering and patrol routing problem. *Transportation Research Part E: Logistics and Transportation Review*, *48*(1), 215-232.

Koch, H., Bortfeldt, A., Wäscher, G. (2018). A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints. *OR Spectrum*, *40*(4), 1029-1075.

Kovacs, A. A., Golden, B. L., Hartl, R. F., Parragh, S. N. (2014). Vehicle routing problems in which consistency considerations are important: A survey. *Networks, 64*(3), 192-213.

Kovacs, A. A., Parragh, S. N., Hartl, R. F. (2015). The multi-objective generalized consistent vehicle routing problem. *European Journal of Operational Research, 247*(2), 441-458.

Kritikos, M.N., Ioannou, G. (2010). The balanced cargo vehicle routing problem with time windows, *International Journal of Production Economics*, *123*(1), 42-51.

Kumar, S. N., Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants. *Scientific Research (online).*

Laporte, G., Ropke, S., Vidal, T. (2014). Heuristics for the vehicle routing problem. *Vehicle routing: Problems, Methods and Applications*, *Second Edition.* 87-116.

Laporte, G. (2007). What you should know about the vehicle routing problem. *Naval Research Logistics*, *54*, 811–819.

Laporte, G. (2009). Fifty Years of Vehicle Routing. *Transportation Science*, *43*(4), 408-416.

Lau, H. C., Yuan, Z., Gunawan, A. (2016). Patrol scheduling in urban rail network. *Annals of Operations Research*, *239*(1), 317-342.

Lee, C. Y., Lee, Z. J., Lin, S. W., Ying, K. C. (2010). An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem. *Applied Intelligence*, *32*(1), 88-95.

Lian, K., Milburn, A. B., Rardin, R. L. (2016). An improved multi-directional local search algorithm for the multi-objective consistent vehicle routing problem. *IIE Transactions*, *48*(10), 975-992.

Liu M., Zhang P. Y. (2014). New hybrid genetic algorithm for solving the multiple traveling saleman problem: an example of distribution of emergence materials. *Journal of Systems Management*, *23*(02), 247–254

Lysgaard, J., Letchford, A. N., Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, *100*(2), 423-445.

Masmoudi M, Mellouli, R. (2014). MILP for synchronized-MTSPTW: application to home healthcare scheduling. *International Conference on Control, Decision and Information Technologies,* 297-302.

Montero, A., Méndez-Díaz, I., & Miranda-Bront, J. J. (2017). An integer programming approach for the time-dependent traveling salesman problem with time windows. *Computers and Operations Research*, *88*, 280-289.

Ombuki, B, Ross, B.J and Hanshar, F. (2006). Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows, *Applied Intelligence*, *24*, 17–30.

Parragh, S. N., Doerner, K. F., Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers and Operations Research*, *37*, 1129–1138.

Parragh, S. N., Schmid, V. (2013). Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers and Operations Research*, *40*(1), 490–497.

Petersen, H. L., O. B. G. Madsen (2009). The double travelling salesman problem with multiple stacks. *European Journal of Operational Research*. *198,* 139–147.

Riedler, M., Raidl, G. (2018). Solving a selective dial-a-ride problem with logic-based Benders decomposition. *Computers and Operations Research*, *96*, 30-54.

Ritzinger, U., Puchinger, J., Hartl, R. F. (2016). A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, *54*(1), 215-231.

Ritzinger, U., Puchinger J., Hartl, R. F. (2016). Dynamic programming based metaheuristics for the dial-a-ride problem. *Annals of Operations Research 236.2*, 341-358.

Ropke, S., Cordeau, J. F., Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks, 49(4), 258–272.

Schilde, M., Doerner, K. F., Hartl, R. F. (2011). Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers & Operations Research*, *38*(12), 1719–1730.

Shieh, E. A., Jain, M., Jiang, A. X., Tambe, M. (2013). Efficiently solving joint activity based security games. *Twenty-Third International Joint Conference on Artificial Intelligence*.

Singh, G., Mehta, R. (2014). Implementation of travelling salesman problem using ant colony optimization. *Journal of Engineering Research and Applications*, *6*(3), 385–389

Soriano, A., Gansterer, M., Hartl, R. F. (2018). The two-region multi-depot pickup and delivery problem. *OR Spectrum 40*(4), 1077-1108.

Sun, P., Veelenturf, L. P., Dabia, S., Van Woensel, T. (2018). The time-dependent capacitated profitable tour problem with time windows and precedence constraints. *European Journal of Operational Research*, *264*(3), 1058-1073.

Tarantilis, C. D., Stavropoulou, F., Repoussis, P. P. (2012). A template-based tabu search algorithm for the consistent vehicle routing problem. *Expert Systems with Applications*, *39*(4), 4233-4239.

Toth, P., Vigo, D. (2002). Models, relaxations and exact approaches for capacitated vehicle routing problem. *Discrete Applied Mathematics*, *123*: 487–512.

Toth, P., Vigo, D. (2014). Vehicle routing: problems, methods, and applications, *SIAM ,18*.

Toulouse, S. (2010). Approximability of the multiple stack TSP. *Electronic Notes Discrete Math*. *36*, 813–820.

Vu, D. M., Hewitt, M., Boland, N., Savelsbergh, M. (2018). Solving time dependent traveling salesman problems with time windows. *Optimization online*, *6640*.

Vu, D. M., Hewitt, M., Boland, N., Savelsbergh, M. (2019). Dynamic discretization discovery for solving the time-dependent traveling salesman problem with time windows. *Transportation Science*.

Wang, X., & Regan, A. C. (2009). On the convergence of a new time window discretization method for the traveling salesman problem with time window constraints. *Computers and Industrial Engineering*, *56*(1), 161-164.

Wang K., Shulin L., Zhao Y. (2017) A genetic-algorithm-based approach to the two-echelon capacitated vehicle routing problem with stochastic demands in logistics service. *Journal of the Operational Research Society*, *68*.11, 1409-1421.

Xu, X., Yuan, H., Liptrott, M., Trovati, M. (2018). Two phase heuristic algorithm for the multiple-travelling salesman problem. *Soft Computing*, *22*(19), 6567-6581.

Yan, X. S., Zhang, C., Luo, W., (2012) Solve traveling salesman problem using particle swarm optimization algorithm. *International Journal of Computer Science*, *9*(6), 264–271

Yu. B, Yang, Z. Z. (2011). An Ant Colony Optimization Model: The Period Vehicle Routing Problem With Time Windows, Transportation Research Part E 47, 166–181.

Yuan, S., Skinner, B., Huang, S., Liu, D. (2013). A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. *European Journal of Operational Research*, *228*(1), 72–82

Zhang, S., Yuvraj, G., Appadoo, S. S. (2017). A meta-heuristic for capacitated green vehicle routing problem. *Annals of Operations Research*, 1-19.

Essay 3: Coordinated Order Fulfillment Planning for a Three-Echelon Supply Chain with Batch Supplies and Flexible Demands

3.1. Abstract

We study a single-item cyclic coordinated order fulfillment problem with batch supplies and flexible demands. This problem relates to both three-echelon supply chains and multi-stage production systems. The system in this study consists of multiple suppliers who each deliver a single item to a central node or buffer from which multiple demanders are then replenished. Each supplier is capable of providing no more than a specific batch size to the central node at a rate that cannot exceed a specified frequency. Moreover, each demander must be replenished with a specific minimum amount over a regular replenishment frequency. Importantly, demand is flexible; it is a control action that the decision maker applies to optimize the system. The objective is to minimize total cost subject to several operational constraints. The decisions include the timing and sizes of batches delivered by the suppliers to the central node and the timing and amounts by which demanders are replenished from the central node. This problem is modeled as an integer linear program. Results when solving several problem instances using IBM ILOG CPLEX are presented and discussed.

3.2. Introduction

Manufacturing firms are searching for new ways to strengthen their position in the global marketplace. In an environment characterized by rapidly advancing technology and increasingly demanding customers, companies within the same supply chain have to cooperate to satisfy customer requirements better than their competitors. As a result, competition no longer occurs merely between isolated entities, but rather between groups of interlinked companies or even whole supply chains.

To fully realize the benefits of cooperation, it is necessary to formulate supply chain strategies and implement channel-wide coordination mechanisms to guarantee that all members of the supply chain work towards a common goal (Glock 2012). From a strategic point of view, supply chain management integrates a variety of interdependent processes and may create a strategic fit between different value-creating activities in a supply chain, contributing to superior customer value and sustainable competitive advantage (Porter 1996).

Inventory is an inevitable aspect of supply chain management which is caused by a mismatch between supply and demand. This mismatch could be in terms of misalignment between the times that supply occurs and the times that demand is realized. The mismatch could also be due to different amounts supplied and demanded at one time. Processing or transit times, batch production, production capacity and uncertainty in demand are reasons for mismatches which make the process of minimizing inventory complicated. Although substantial synergies may be realized when coordinating inventories across a supply chain, inventory management has long been treated as an isolated function solely focused on individual entities. However, the consequences of coordinated inventory replenishment decisions have been analyzed in inventory models that focus on the total costs of the system in question, which are referred to as joint economic lot size (JELS) models.

In this research, we consider a JELS problem where heterogeneous suppliers supply a single product to multiple independent, heterogeneous demanders. All suppliers deliver the product to, and all demanders are replenished from, a single node that sits in the middle of the system. This scenario may be applicable to both macroscopic and microscopic settings. For example, the central node may be a city, business campus, factory, or storage rack. In the macroscopic case, this problem may apply to supply chain settings where supply and demand processes link a central storage facility to supplier and buyer facilities. In the microscopic case, the supply and demand processes would represent stages of manufacturing with parallel machines and the central node would be a buffer for storing work-in-process (WIP). The goal of this problem is to find a cyclic schedule for the incoming supplies and outgoing demand replenishments that minimizes total system-wide cost.

This research contributes to existing JELS models in two respects. First, we focus on the concept of flexible demand and study a joint supply and replenishment planning problem in the case where demand is flexible. Flexible demand (FD) is a term used to describe cases when a demander is generally flexible regarding how often it receives items and how much it receives when it is replenished. In other words, each demander needs to receive a minimum total quantity during every time span of a given length. Besides these requirements, the demanders leave it to the vendor to decide exactly how and when they are replenished while meeting the minimum requirement that they not be starved. Second, unlike the only other JELS focused article (Petering et al. 2019) that considers FD, in this study, demanders are capable of storing items on their premises and do not need to receive their required orders all at once in large batches.

This study is organized as follows. Section 3.3 reviews the relevant literature. Section 3.4 describes the problem and presents an illustrative example of it. In Section 3.5, a mathematical formulation of the problem is presented. Section 3.6 presents several theoretical insights regarding this problem. In Section 3.7, we present a method for automatically creating feasible solutions whenever a problem instance is feasible. Section 3.8 describes the experimental setup, presents the results, and discusses their significance. Finally, we conclude in Section 3.9.

3.3. Literature review

Several streams of literature relate to our problem. These include the literature on the capacitated lot-sizing problem (CLSP), joint economic lot sizing (JELS), and vendor-managed inventory (VMI). We now sequentially discuss these streams of literature.

One stream of literature is the capacitated lot-sizing problem (CLSP), which determines the levels and timing of production along a certain planning horizon. There has been a variety of studies in CLSP literature which use different CLSP formulations and develop solution algorithms; Kuhn and Quadt (2008) provides a comprehensive review of literature on CLSP studies. Other papers in this stream include Eppen and Martin (1987), Trigeiro et al. (1989), Katok et al. (1998), Karimi et al. (2003), Kuhn and Quadt (2008), Sahling et al. (2009), de Araujo et al. (2015), Fragkos et al. (2016), Fiorotto et al. 2017, Bayley et. Al (2018) and Taş, Duygu, et al. (2019). Our problem is different from many CLSP studies in that our model does not include job scheduling. In fact, our problem can be categorized under CLSP big bucket models that allow the production of multiple units in one period and do not consider job scheduling decisions.

The system that we study is related to a three-tier supply chain with coordinated inventory replenishment decisions. Coordinated inventory replenishment decisions have been analyzed in inventory models that focus on the total costs of the system in question, which are referred to as joint economic lot size (JELS) models. JELS models are especially useful as planning tools in situations where companies have established long-term relationships with their suppliers or customers, which is common in the automotive industry, for example. In such a case, the members of the supply chain have an incentive to work together towards a reduction of total system costs, since cooperation gains that emerge as a result of investments or changes in the order and production policies of the companies can be distributed among the members of the supply chain (Glock, 2012). Vendor-managed inventory (VMI), for example, is a common practice among such implementations.

The literature on JELS two-stage models treats coordinated inventory management between supplier(s) and buyer(s). One of the first lot-size models dealing with buyer–vendor coordination was proposed by Goyal (1976), who analyzed a system consisting of a single vendor and a single buyer. He assumed that the vendor provides an infinite replenishment rate, and that the production lot is transferred to the buyer in equal-sized shipments. An integrated inventory model which generalizes Goyal (1976) model was proposed by Banerjee (1986), who considered the same system and assumed that the order quantity of the buyer equals the production quantity of the vendor. Goyal (1988) later developed a single-vendor, single-buyer model where the lot size of the vendor equals an integer multiple of the buyer's order quantity to reduce total costs. Braglia and Zavanella (2003) extended this model and assumed that buyer and vendor have implemented a consignment stock (CS) agreement.

Gumus et al. (2008) differentiated between CS and VMI and analyzed their effect on the total costs of a JELS model. In contrast to Braglia and Zavanella (2003), the authors assumed that in the case of CS, the customer maintains control on the timing and quantity of replenishments, but leaves the ownership of the inventory at the vendor until the goods are used. However, if VMI is implemented, the vendor initiates orders on behalf of the customer as well. The authors showed that the system can benefit from such agreements, especially if the inventory holding costs at the buyer exceed those at the vendor, or if the order-processing costs of the vendor are lower than those at the buyer. Different variants of batch shipments including equal and unequal-sized shipments are studied by Viswanathan (1998), Hill (1997), Goyal and Nebebe (2000), Hill and Omar (2006), Hoque (2009) and Sajadieh and Jokar (2009a), Kim and Glock (2013).

The single-vendor, single-buyer integrated inventory model is generally considered as the basic building block of a JELS model. For more real-world practical relevance of such models, researchers have extended their analysis to include more aspects in the model and developed mechanisms to deal with supply chains of greater complexity.

There have been studies in the JELS literature with parallel supplier workstations and/or multiple buyers. One of the first models to analyze single vendor and multiple identical buyers in an integrated inventory model is the one of Joglekar and Tharthare (1990). We denote such models as SVMB models in the following. The authors assumed that orders are evenly distributed throughout the planning horizon and adopted an integer ratio-policy. A variation of Joglekar and Tharthare's model was offered by Banerjee and Burton (1994), who analyzed a SVMB system as well, but focused on heterogeneous buyers rather than identical buyers. Banerjee and Burton's model was extended by Siajadi et al. (2006), who assumed that the order cycle time for each buyer is equal to the production cycle time, and that the vendor can deliver a different number of equal-sized shipments to each of the buyers.

Zavanella and Zanoni (2009) analyzed a SVMB system where the vendor is responsible for managing the warehouses at the buyers. Similar to Siajadi et al. (2006), they assumed that the vendor delivers equal-sized batch shipments to the buyers. Chen et al. (2010) studied a similar system where a VMI-agreement has been implemented, and assumed that demand at the buyers is sensitive to price and inventory level. The results of the paper demonstrate that VMI results into a lower retail price and higher inventory levels. Darwish and Odah (2010) modeled a SVMB supply chain under a VMI agreement where the retailers do not incur any order costs. Cunha et al. (2017) considered the multi-item economic lot-sizing problem with remanufacturing and incapacitated production. Roy et al. (2018) considered a two-level supply chain, including one manufacturer and two competing retailers with sales price dependent demand and random arrival of customers. They found the order quantity of a single product for stochastic demand pattern that minimized the expected cost.

Our problem considers multiple suppliers who provide replenishments to multiple downstream demanders through a central node. Therefore, it is worth discussing the other stream of literature on JELS models with multiple vendors and multiple buyers. Ben-Daya et al. (2008) presented a comprehensive review of such JELS problem and also provided some extensions of this important problem. Pineyro and Viera (2010) investigated the economic lot-sizing problem with product returns and one-way substitution. Transchel and Minner (2011) studied a problem of the dynamic quantity competition and economic lot-sizing with two competing retailers who each are faced different replenishment cost structures. Feng et al. (2011) addressed a single product economic lot-sizing problem with constant capacity, non-increasing setup cost and convex inventory cost function. Das et al. (2011) developed an economic production lot-size (EPLS) model for an item with imperfect quality by considering random machine failure. Sari et al. (2012) developed a mathematical model of JELS with temporary price discounts (JELPTPD). Glock (2012) reviewed joint economic lot-size models which were focused on coordinated inventory replenishment decisions between buyers. Rezaei and Davoodi (2012) formulated a case where the buyer's goal is to achieve the optimal selling price and lot size of multiple products across different suppliers.

Bouslah et al. (2013) solved the problem of joint determination of the optimal lot-sizing and production-inventory control policy for unreliable and imperfect manufacturing. Archetti et al. (2014) presented two scenarios of the problems with cost discounts. In the first scenario, the cost function was a modified function and in the second scenario, the cost function was an incremental discount function. Piñeyro and Viera (2014) improved Pineyro and Viera (2010) model. They studied a lot-sizing problem with different demand channels for new and remanufactured products. Karimi-Nasab et al. (2015) presented the math model of the joint lot-sizing problem and formulated a realistic production scheduling problem. Sifaleras et al. (2015) suggested a variable neighborhood search (VNS) metaheuristic algorithm for the economic lot-sizing problem with product returns and recovery.

This stream extends to other studies. Helmrich et. al. (2015) considered a lot-sizing problem with a global emission constraint. Önal et al. (2015) introduced the economic lot-sizing problem for handling perishable items. Sarakhsi et al. (2016) introduced a hybrid algorithm of Nelder–Mead and scatter search algorithms called SSNM to solve JELS problem with multiple vendors and multiple buyers. Pishchulov and Richter (2016) studied the classical JELS model as an adverse selection problem with asymmetric cost information. Marchi et al. (2016) presented a JELS model with financing the investments cooperatively among the members of the supply chain. Glock and Kim (2016) developed a supply chain model with returnable transport items (RTIs). Cunha and Melo (2016) studied the multi-item economic lot-sizing problem with remanufacturing and incapacitated production. Telha and Van Vyve (2016) considered a continuous-time variant of the classical economic lot-sizing problem.

Our study considers a three-level supply chain where suppliers supply inventory to a central node from which multiple demanders are replenished. This brings in inventory considerations at the central node. JELS models for multi-level supply chains are more complex and involve more alignment among different stages. We now discuss the papers which study multi-level supply chains. Abdelsalam and Elassal (2014) considered the JELS for a supply chain with multiple retailers and multiple manufacturers and a supplier. Modak et al. (2016) analyzed a single-item model for a supply chain including the manufacturer, distributors, and retailers. They derived the optimal pricing strategies under a two-stage tariff and bargaining process. Önal (2016) considered the economic lot-sizing problem for a multi-level supply chain to address the joint procurement and distribution decisions for a perishable item. Ferretti et al. (2017) designed JELS model in a three-level supply chain. They analyzed different supply chain configurations related to outsourced manufacturing operations. Sargut and Işık (2017) considered a dynamic economic lot-sizing problem. Ou (2017) studied a classical single-item economic lot-sizing problem, where production cost functions were assumed to be piecewise linear. Salas Navarro, Chedid, Caruso, and Sana (2018) made a comparison between the join economic lot-size model for three-level and two-level supply chains using two schemes, collaborative and non-collaborative. Optimization of the integrated lot-sizing decisions in a four-level supply chain is studied in several papers: Gharaei and Pasandideh (2016), Gharaei et al. (2016), Gharaei and Pasandideh (2017a, 2017b), Gharaei et al. (2017), Gharaei et al. (2017) and Gharaei et al. (2019).

One of the most relevant papers to this research is Petering et al. (2019) which adopts a JIT-oriented objective with parallel workstations at two stages of a manufacturing system with batch supplies and flexible demand processes. The authors integrate the features of parallel workstations at the suppliers and multiple buyers, and consider batch size constraints for suppliers. However, the objective in their study is to minimize inventory at the storage facility in a JIT supply chain which differs from the objectives within most of the JELS literature of minimizing total system cost.

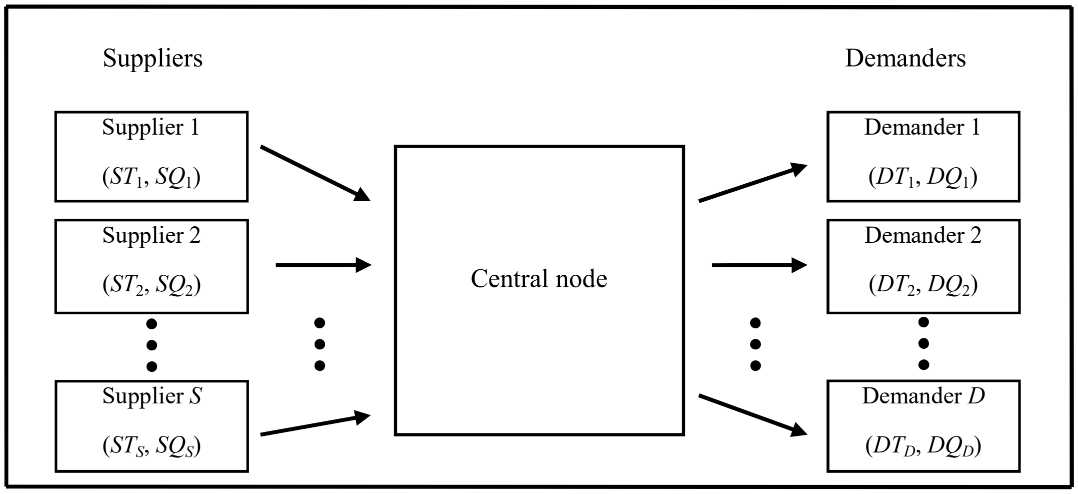
We study a cyclic coordinated supply and inventory replenishment problem with batch supplies, flexible demands and a central node that holds inventory received from the suppliers and is used to satisfy the demanders. Each supplier is capable of providing no more than a specific batch size to the central node at a rate that cannot exceed a specified frequency. Moreover, each demander must be replenished with a certain minimum amount during time windows of a given duration for any possible time window start time. The decisions include the timing and sizes of batches delivered by the suppliers to the central node and the timing and amounts by which demanders are replenished.

This study fills the gap in the discussed literature in three major respects. First, this is only the third study to consider the concept of flexible demand, the other two being Petering et al. (2019) and Section 2 of this dissertation. Flexible demand (FD) is a term used for when a demander is flexible in generally how much/ how often they want to receive their minimum requirement over every certain number of consecutive periods. In other words, the demanders leave it to the vendor to decide exactly how and when they are replenished while meeting the minimum requirement. Second, unlike the literature's only other study to consider flexible demand, in this study, demanders are capable of storing items on their premises and do not need to receive their items in large batches. Third, the focus of our problem is not only minimizing inventory as in JIT models. Rather, we seek to minimize the total cost related to production, holding inventory at the central node, and transportation to deliver items to the demanders.

3.4. Problem description

In this section, we formally explain the problem under study. Consider an infinite-capacity *central node*, *buffer*, or *facility* that serves as a temporary storage area for a single type of discrete product (i.e. item, SKU, part). The stock in the central node is consumed by D *demanders* and replenished by S *suppliers*. Time is discretized into periods. Two parameters define the requirements of each demander d: a *demand time requirement* DT_d and a *(minimum) accumulated demand quantity* DQ_d. Demander d is indifferent to when it receives product and how much it receives at any given time, but it must receive at least DQ_d units of the product during *each* DT_d-period-long time interval that exists over the planning horizon. Two parameters define the capabilities of each supplier s: a *minimum inter-supply time* ST_s and a (*maximum) supply quantity* SQ_s. Supplier s is capable of delivering a batch of at most SQ_s units to the central node every ST_s periods or less often. Figure 7 represents this system.

Figure 7. Supply chain under consideration



We assume that supplies are received at the beginning of a period and are followed immediately by demands. The amount left over after the demand is satisfied is held as inventory for that period. The operations are cyclic, repeating every T periods where $T \geq max_d\{DT_d\}$ and $T \geq max_s\{ST_s\}$. This means that every supply and demand process repeats after a T-period cycle, and period 1 immediately follows period T. In other words, the inventory level during period 1 equals the inventory level during period T, plus the total amount supplied to the central node at the beginning of period 1, minus the total amount demanded from the central node at the beginning of period 1.

In this problem, the demand timing and amounts for each demander as well as the supply timing and amounts for each supplier are to be decided with respect to the discussed requirements and capabilities. The goal is to feasibly satisfy the demands with the available supplies (i.e. to avoid stock-out in each period at the central node) while minimizing the total cost incurred during the cycle. Total cost is defined by a per-unit inventory holding cost per time period at the central node (H_1), a per-unit inventory holding cost during the period of maximum inventory at the central node (H_2), a fixed cost of using and contracting with selected suppliers (FC_s), fixed and variable production costs at suppliers (FCP_s and VC_s), and a fixed cost of transporting replenishments from the central node to demanders (FCD_d).

3.4.1. Problem applicability to industry

This problem represents different microscopic or macroscopic industrial activities. From a microscopic perspective, the problem could relate to a buffer that links two stages of a manufacturing process in a facility. In this system, there are parallel machines at the first stage which are characterized by heterogeneous capabilities (processing times and batch sizes). The first-stage machines produce the same part that feeds the machines at the second stage. Each second-stage machine uses the part supplied and produces a different product which indicates that each machine has a different cycle time DT_d and/or minimum quantity requirement DQ_d.

On a macroscopic level, the problem could represent a three-stage supply chain with multiple suppliers supplying a single product to a retailer that has multiple retail stores, each facing a different demand. The suppliers first deliver their products to a distribution center (i.e. the central node); the products are then distributed to the various retail stores. A single truck with limited capacity SQ_s and minimum round-trip travel time ST_s, transports items from supplier s to the distribution center for each s.

The requirement that operations repeat every T periods is imposed by managers who are interested in system reliability and predictability. Without this assumption, the system state could evolve in a non-repetitive manner from time 0 to infinity. In a manufacturing setting, T could be measured in hours and a value of T that agrees with the length of one or more work shifts might be appropriate. On the other hand, in a supply chain setting, T could be measured in days and it could be a multiple of seven.

3.4.2. Illustrative example

Tables 26-27 illustrate the problem for a case in which $D = S = 3$. Table 26 shows the input parameters for this instance. Table 27 shows a feasible solution for this problem instance. In this solution, demander 1 is given 4, 2, 3, 2, 3, 4 and 1 units at the beginning of each of the periods T1, T2, T4, T5, T7, T8 and T9; demander 2 is given 1 unit at the beginning of periods T3, T4, T6, T8, T9 and T10; and demander 3 is given 1, 3, 2 and 2 units at the beginning of periods T2, T3, T7 and T9. It is important to note that the amounts delivered to each demander accumulate to a size of greater than or equal to the values of DQ_1, DQ_2, and DQ_3 respectively. For example, demander 1 is given at least 5 units during *every* 3-period-long time interval. Also, demander 2 is provided with at least 2 units during *every* 4-period-long time interval and demander 3 is given at least 4 units during *every* 6-period-long time interval.

In the solution, supplier 1 replenishes the central node with 2, 3, 3, 4 and 1 units at the beginning of periods T1, T3, T5, T7, and T9; supplier 2 replenishes the central node with 1 and 3 units at the beginning of periods T2 and T7; and supplier 3 replenishes the central node with 6, 6, and 4 units at the beginning of periods T1, T4, and T7. Note that the amount delivered by the suppliers never exceeds the values of SQ_1, SQ_2, and SQ_3—4, 3, and 6—for suppliers 1, 2, and 3 respectively. Also, the time that elapses between consecutive supply deliveries is never less than the values of ST_1, ST_2, ST_3—2, 5, and 3 periods—for suppliers 1, 2, and 3 respectively. A zero in Table 27 means that no demand is made or nothing is supplied at the beginning of a period.

Table 26. Input parameters for the illustrative instance

Number of demanders: 3		Number of suppliers: 3	
DT_1: 3	DQ_1: 5	ST_1: 2	SQ_1: 4
DT_2: 4	DQ_2: 2	ST_2: 5	SQ_2: 3
DT_3: 6	DQ_3: 4	ST_3: 3	SQ_3: 6
T = 10			

Table 27. Feasible solution for the illustrative instance

	Period									
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
Demander 1	4	2	0	3	2	0	3	4	1	0
Demander 2	0	0	1	1	0	1	0	1	1	1
Demander 3	0	1	3	0	0	0	2	0	2	0
Sum up (DI)	4	3	4	4	2	1	5	5	4	1
Supplier 1	2	0	3	0	3	0	4	0	1	0
Supplier 2	0	1	0	0	0	0	3	0	0	0
Supplier 3	6	0	0	6	0	0	4	0	0	0
Sum up (SI)	8	1	3	6	3	0	11	0	1	0
SI-DI	4	-2	-1	2	1	-1	6	-5	-3	-1
Inventory held	4	2	1	3	4	3	9	4	1	0

The central node's inventory level during each period is shown in the row at the bottom of Table 27. The sum of these values—31—is the total inventory held during the cycle and the maximum inventory held in any period is 9. The displayed solution is not optimal and is only one of thousands of feasible solutions to this problem instance.

3.5. Mathematical formulation

Table 28 shows the indices, input parameters, and decision variables in our mathematical model of this problem named Math Model 3. The inputs to the math model include: the cycle length T; number of demanders D; demand time requirement for demander d (DT_d); minimum accumulated demand quantity for demander d (DQ_d); number of suppliers S; minimum inter-supply time for supplier s (ST_s); maximum batch size (i.e. supply quantity) delivered by supplier s (SQ_s); and six cost-related parameters. Parameter H_1 is the per-unit inventory holding cost per time period at the central node. Parameter H_2 is the per-unit inventory holding cost during the period of maximum inventory at the central node. Parameter FC_s indicates the fixed cost of using and contracting with selected suppliers. Parameters FCP_s and VC_s are fixed and variable production costs at suppliers. Parameter FCD_d indicates the fixed cost of transporting replenishments from the central node to demanders.

The decision variables are displayed at the bottom of Table 28. Our problem can be modeled as an integer linear program (ILP) which includes both binary and integer variables. There are three binary decision variables in the model: variable DYN_{dt} takes a value 1 if demander d receives a replenishment of any size from the central node at the beginning of period t, variable SYN_{st} indicates if supplier s provides a batch to the central node at the beginning of period t or not, variable X_s indicates if supplier s is used or not. Moreover, four integer decision variables capture the number of units delivered from the central node to demander d at the beginning of period t ($DAmt_{dt}$): the number of units transferred to the central node by supplier s at the beginning of period t ($SAmt_{st}$): the inventory level during period t (I_t); and finally the maximum inventory level achieved during the T-period cycle *(Imax)*.

Table 28. Indices, parameters, and decision variables in Math Model 3

Indices	
d	demander ($d = 1$ to D)
s	supplier ($s = 1$ to S)
t, u	time period ($t, u = 1$ to T)
Input parameters	
T	Cycle length for the inventory system. (integer, ≥ 2)
D	Number of demanders (integer, ≥ 2)
DT_d	Demand time requirement for demander d (integer, ≥ 2)
DQ_d	Minimum accumulated demand quantity for demander d (integer, ≥ 1)
S	Number of suppliers (integer, ≥ 1)
ST_s	Minimum inter-supply time for supplier s (integer, ≥ 2)
SQ_s	Maximum batch size provided by supplier s (integer, ≥ 1)
H_1	Inventory holding cost at central node per unit per time unit (real, ≥ 0)
H_2	Cost per unit of maximum inventory held at central node (real, ≥ 0)
FC_s	Fixed cost of using supplier s in general (real, ≥ 0)
FCP_s	Fixed cost of production at supplier s (real, ≥ 0)
VC_s	Variable production cost per unit at supplier s (real, ≥ 0)
FCD_d	Fixed cost of transportation from the central node to demander d (real, ≥ 0)
Decision variables	
$DYN_{dt} \begin{cases} 1 \\ 0 \end{cases}$	If demander d is given a replenishment of any size at the beginning of period t. Otherwise (binary)
$DAmt_{dt}$	Quantity received by demander d at the beginning of period t. (integer, ≥ 0)
$SYN_{st} \begin{cases} 1 \\ 0 \end{cases}$	If supplier s supplies a batch of any size at the beginning of period t. Otherwise (binary)
$SAmt_{st}$	Amount supplied by supplier s at the beginning of period t. (integer, ≥ 0)
$X_s \begin{cases} 1 \\ 0 \end{cases}$	If supplier s is used at all. Otherwise (binary)
I_t	Inventory level at the central node during period t (integer, ≥ 0)
I_{Max}	Maximum inventory observed at the central node during the cycle (integer, ≥ 0)

Table 29. Math Model 3 for coordinated order fulfillment planning with FD

Math Model 3
Minimize

$$(H_2)(I_{Max}) + \sum_{t=1}^{T}(H_1)(I_t) + \sum_{s=1}^{S}(FC_s)(X_s) + \sum_{t=1}^{T}\sum_{s=1}^{S}(FCP_s)(SYN_{st}) + \sum_{t=1}^{T}\sum_{s=1}^{S}(VC_s)(SAmt_{st}) + \sum_{t=1}^{T}\sum_{d=1}^{D}(FCD_d)(DYN_{dt}) \quad (27)$$

Subject to:

$$\sum_{u=t}^{t+ST_s-1} SYN_{s,((u-1)\bmod T)+1} \le 1 \qquad \forall s,t \qquad (28)$$

$$SAmt_{st} \le SQ_s \cdot SYN_{st} \qquad \forall s,t \qquad (29)$$

$$X_s \ge SYN_{st} \qquad \forall s,t \qquad (30)$$

$$\sum_{u=t}^{t+DT_d-1} DAmt_{d,((u-1)\bmod T)+1} \ge DQ_d \qquad \forall d,t \qquad (31)$$

$$DAmt_{dt} \le (DYN_{dt})(DQ_d) \qquad \forall d,t \qquad (32)$$

$$I_1 = I_T + \sum_{s=1}^{S} SAmt_{s1} - \sum_{d=1}^{D} DAmt_{d1} \qquad (33)$$

$$I_t = I_{t-1} + \sum_{s=1}^{S} SAmt_{st} - \sum_{d=1}^{D} DAmt_{dt} \qquad \forall t\colon\ 2 \le t \le T \qquad (34)$$

$$I_t \le I_{Max} \qquad \forall t \qquad (35)$$

$$I_T = 0 \qquad (36)$$

Table 29 shows the math model. The objective of the model is to minimize the sum of the (a) cost incurred for the maximum inventory level observed at the central node during the cycle, (b) inventory holding cost at the central node, (c) fixed cost of using and contracting with selected suppliers, (d) fixed, (e) variable production cost at suppliers, and (f) fixed cost of transporting items to the demanders. Constraint (28) ensures that no more than one batch is supplied by supplier *s* during any ST_s consecutive time periods. Constraint (29) specifies that the amount supplied by supplier *s* does not exceed SQ_s in any period. Based on this constraint, if $SYN_{st} = 0$, supplier *s* does not provide a batch to the central node at the beginning of period *t*. Constraint (30) indicates that in order for a supplier to supply a batch, it must be selected as one of the suppliers that is used. Constraint (31) ensures that the total amount transferred to demander *d* over any consecutive DT_d periods must add up to at least DQ_d. Constraint (32) indicates that the amount given to demander d does not exceed DQ_d when the demander is replenished in that period ($DYN_{dt} = 1$) and it will be 0 if the demander is not replenished in that period ($DYN_{dt} = 0$). Constraints (33–34) ensure that the inventory on hand during each period is computed correctly. Constraint (35) indicates that the inventory level in any period does not exceed the maximum inventory level. Constraint (36) forces a zero inventory level in the central node in the final period. This constraint eliminates symmetries and redundant solutions. With non-negativity restrictions on variable I_t, the inventory level never falls below zero and stock-out never occurs.

3.6. Theoretical insights

Several theoretical insights can be made regarding the problem at hand. In order to derive these insights, several terms need to be defined.

Definition 1: *A* ***replenishment schedule*** *for demander d is a set of values $DAmt_{d1}$, $DAmt_{d2}$,…, $DAmt_{dT}$ where $DAmt_{dt} \geq 0$ for all t.*

Definition 2: *A* ***feasible replenishment schedule*** *for demander d is a replenishment schedule for demander d that also satisfies constraint (31) in the math model.*

Definition 3: *A* ***reduced replenishment schedule*** *for demander d is a feasible replenishment schedule for demander d ($DAmt_{d1}$, $DAmt_{d2}$, … , $DAmt_{dT}$) in which the reduction of $DAmt_{dt}$ for any t violates constraint (31) in the math model.*

Discussion: We explain the reduced replenishment schedule using the illustrative instance shown in Tables 26-27. Consider demander 1 with $DT_1 = 3$ and $DQ_1 = 5$. The replenishment schedule for this demander during the ten-period cycle is: (4, 2, 0, 3, 2, 0, 3, 4, 1, 0). Consider any time window of length $DT_1 = 3$. The total amount replenished in any time window of length $DT_1 = 3$ sum to at least $DQ_1 = 5$. If the amount provided to this demander at the beginning of period 1 is reduced to 3, the resulting replenishment schedule would violate constraint (31) because fewer than 5 units would be provided to the demander during periods 9, 10, and 1 combined. Similar observations can be made regarding all other time periods when this demander receives items. Overall, in this replenishment schedule, we cannot reduce any replenishment amount for any period t without violating the demander's requirements of

$DQ_1 = 5$ and $DT_1 = 3$. Therefore, the displayed feasible replenishment schedule for demander 1 is also a reduced replenishment schedule.

Theorem 1: *There always exists an optimal solution to the math model in which every demander is satisfied via a reduced replenishment schedule.*

Proof: This is true because the schedule for any demander who is replenished with a feasible non-reduced replenishment schedule can be changed to a reduced replenishment schedule without increasing the objective value. Consider any feasible solution *Z* in which there is a non-reduced replenishment schedule for at least one demander *d**. In other words, there are one or more "extra units of demand" in this feasible solution *Z*. We can make changes to this solution by reducing demand amounts $DAmt_{d^*,t}$ and supply amounts $SAmt_{st}$ to generate another feasible solution *Z′* which has an objective value which is at least as good or even better than the objective value for *Z*. To do this, consider each "extra unit of demand" one at a time in solution *Z*. Delete each such extra unit of demand, and then delete one unit of supply for any supplier occurring in the same period or the latest period that comes before this period in the cyclic sense. The resulting solution *Z′* (a) has an objective no higher than that of *Z* and (b) is feasible owing to the feasibility of *Z* and the non-negativity of the inventory values in *Z*. ■

Theorem 2: *The minimum total amount that can be demanded by demander d in a reduced replenishment schedule is*

$$MinAmtD_d = \left\lceil (T)\left(\frac{DQ_d}{DT_d}\right)\right\rceil \tag{37}$$

Proof: Demander *d* needs to be replenished with a minimum total of DQ_d units every DT_d periods. This indicates that the replenishment rate per unit time must be at least $\frac{DQ_d}{DT_d}$ rounded up to the nearest integer. Therefore, the total replenishment units that must be given to demander *d* during a cycle *T* is $(T)\left(\frac{DQ_d}{DT_d}\right)$ rounded up to the nearest integer which is $\left\lceil (T)\left(\frac{DQ_d}{DT_d}\right)\right\rceil$. ■

Definition 4: *A **minimal replenishment schedule** for demander d is a reduced replenishment schedule for demander d in which exactly $MinAmtD_d$ is provided.*

Theorem 3: *The maximum total amount that can be demanded by demander d in a reduced replenishment schedule is:*

$$MaxAmtD_d = (DQ_d)(NumFullBatchesRepl_d) \tag{38}$$

where

$$NumFullBatchesRepl_d = \begin{cases} \left\lfloor \dfrac{T}{\left\lceil \frac{DT_d}{2} \right\rceil + \left\lceil \frac{DT_d + 1}{2} \right\rceil} \right\rfloor * 2 & if\, T \bmod \left(\left\lceil \frac{DT_d}{2} \right\rceil + \left\lceil \frac{DT_d + 1}{2} \right\rceil \right) \leq DT_d - \left\lceil \frac{DT_d}{2} \right\rceil \\ \left\lfloor \dfrac{T}{\left\lceil \frac{DT_d}{2} \right\rceil + \left\lceil \frac{DT_d + 1}{2} \right\rceil} \right\rfloor * 2 + 1 & otherwise \end{cases} \tag{39}$$

Proof: One way to achieve the maximum total amount that can be demanded by demander *d* is to either provide demander *d* full batches of size DQ_d or nothing during every time period. Let us call such a replenishment schedule a "full-batch-only replenishment schedule." An example of a full-batch-only replenishment schedule for demander 1 in Table 26 would be (5, 0, 0, 5, 0, 0, 5, 0, 0, 5). Our proof focuses on the number of full-batch replenishments provided to demander *d* which we call ($NumFullBatchesRepl_d$). Our goal is to provide full-batch replenishments to demander *d* as often as possible while still needing each of those replenishments in a way that none could be removed. For demander 1 in Table 26, this is accomplished in the replenishment schedule (5, 0, 5, 0, 5, 0, 5, 0, 5, 0). In this replenishment schedule, five full batches are provided to demander 1, each spread just far enough apart.

We prove this theorem in two steps. We first consider some examples, and then we will generalize our observations to prove the theorem. Suppose that T = 100, DT_d = 8, and DQ_d = 5 for demander *d*. In order to provide full-batch replenishments to demander *d* as often as possible while still needing each of those replenishments, we must alternative between 4 and 5 periods as inter-replenishment times over the cycle. The only exception is that the final inter-replenishment time may differ because the inter-replenishment times must sum to 100. This means that in theory the *initial inter-replenishment times* are (4, 5, 4, 5, …, 4, 5, 4, 5, 1). However, the last inter-replenishment time of 1 is not high enough to make a reduced replenishment schedule. This 1 time period needs to be absorbed by other inter-replenishment times and the *final inter-replenishment times* would be (4, 5, 4, 5, …, 4, 5, 4, 6). The resulting full-batch-only replenishment schedule is (5, 0, 0, 0, 5, 0, 0, 0, 0, 5, 0, 0, 0, 5, 0, 0, 0, 0, 5,…, 5, 0, 0, 0, 5, 0, 0, 0, 0, 0). In such a replenishment schedule, none of the replenishments can be removed; if one batch is removed, there would be an inter-replenishment time of 9 or more which violates the DT_d requirement of 8. Let us consider two consecutive inter-replenishment times of 4 and 5 or 4 and 6 as one *cluster*. The number of clusters in a cycle is $\frac{100}{4+5}$ rounded down to the nearest integer which is 11. This means that we have at least 11*2 = 22 full-batch replenishments. We cannot fit another replenishment at the end of the cycle. The last period when a full-batch replenishment is provided is period 95. There are only five time periods − 96, 97, 98, 99, 100 − between this period and period 1, and providing a full-batch replenishment during any of these periods will make this full-batch-only replenishment schedule a non-reduced replenishment schedule. Thus, the "busiest" full-batch-only replenishment schedule has exactly 22 full batch replenishments. In other words, $NumFullBatchesRepl_d$ = 22 and $MaxAmtD_d$ = $(DQ_d)(NumFullBatchesRepl_d)$ = 110. This is the top part of equation (39).

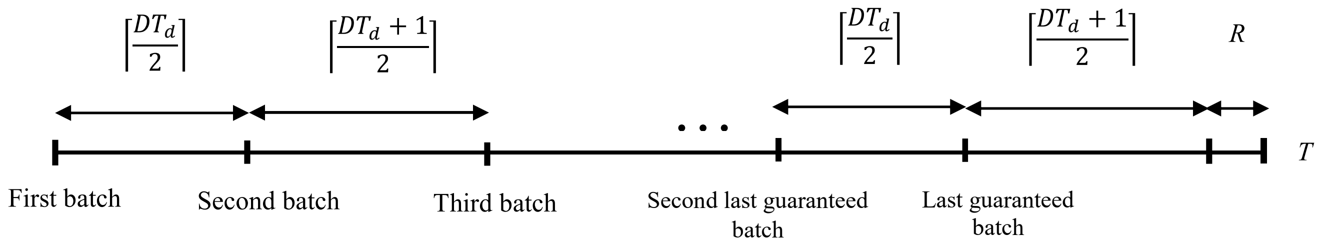
Now, let us consider the case where T = 103, DT_d = 8, and DQ_d = 5 for demander d. In order to provide full-batch replenishments to demander d as often as possible while still needing each of those replenishments, we must alternative between 4 and 5 time periods as inter-replenishment times over the cycle. In theory, the initial inter-replenishment times are (4, 5, 4, 5, …, 4, 5, 4, 5, 4). However, the last inter-replenishment time of 4 is not high enough to make a reduced replenishment schedule, so these 4 periods need to be absorbed by other inter-replenishment times and the final inter-replenishment times would be (4, 5, 4, 5, …, 4, 5, 5, 8). The resulting full-batch-only replenishment schedule is (5, 0, 0, 0, 5, 0, 0, 0, 0, 5, 0, 0, 0, 5, 0, 0, 0, 0, 5,…, 5, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0). In this case, none of the replenishment can be removed; if one batch is removed, there would be an inter-replenishment time of 9 or more which violates the DT_d requirement of 8. In this replenishment schedule, we have at least 22 ($\frac{103}{4+5}$ rounded down to the nearest integer times 2) full-batch replenishments. The last period when a full-batch replenishment is provided is period 93. There are seven time periods − 94, 95, 96, 97, 98, 99, 100 – between this period and period 1, and providing another full-batch replenishment during any of these periods will make this full-batch-only replenishment schedule a non-reduced replenishment schedule. Thus, in the busiest full-batch-only replenishment schedule,

$NumFullBatchesRepl_d$ = 22 and $MaxAmtD_d$ = $(DQ_d)(NumFullBatchesRepl_d)$ = 110. This is the top part of equation (39).

If we increase the cycle time by 1, we have T = 104, DT_d = 8, and DQ_d = 5 for demander d. The inter-replenishment times are (4, 5, 4, 5, …, 4, 5, 4, 5, 5). The last inter-replenishment time 5 cannot be absorbed by inter-replenishment times by either side of it $-$ left or right by wrapping around the end of horizon $-$ because by doing so, the inter-replenishment time would become 10 or 9 which exceeds the DT_d requirement of maximum 8. This increases the number of full-batch replenishments from 22 ($\frac{104}{4+5}$ rounded down to the nearest integer times 2) to 23. Thus, $NumFullBatchesRepl_d$ = 23 and $MaxAmtD_d$ = $(DQ_d)(NumFullBatchesRepl_d)$ = 115. This is the bottom part of equation (39).

To generalize these observations, we can compute the two alternating inter-replenishment times as $\frac{DT_d}{2}$ rounded up to the nearest integer ($\left\lceil\frac{DT_d}{2}\right\rceil$) and $\frac{DT_d+1}{2}$ rounded up to the nearest integer ($\left\lceil\frac{DT_d+1}{2}\right\rceil$). In theory, the initial inter-replenishment times would be ($\left\lceil\frac{DT_d}{2}\right\rceil$, $\left\lceil\frac{DT_d+1}{2}\right\rceil$, $\left\lceil\frac{DT_d}{2}\right\rceil$, $\left\lceil\frac{DT_d+1}{2}\right\rceil$, …, $\left\lceil\frac{DT_d}{2}\right\rceil$, $\left\lceil\frac{DT_d+1}{2}\right\rceil$, $T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right)$). Let us consider two consecutive inter-replenishment times of $\left\lceil\frac{DT_d}{2}\right\rceil$ and $\left\lceil\frac{DT_d+1}{2}\right\rceil$ as one cluster. We can find out how many clusters we have in a cycle length. That is $\frac{T}{\left\lceil\frac{DT_d}{2}\right\rceil+\left\lceil\frac{DT_d+1}{2}\right\rceil}$ rounded down to the nearest integer which is $\left\lfloor\frac{T}{\left\lceil\frac{DT_d}{2}\right\rceil+\left\lceil\frac{DT_d+1}{2}\right\rceil}\right\rfloor$. Since we have two full-batch replenishments within each cluster, this indicates we have at least $\left\lfloor\frac{T}{\left\lceil\frac{DT_d}{2}\right\rceil+\left\lceil\frac{DT_d+1}{2}\right\rceil}\right\rfloor * 2$ number of full-batch replenishments during the cycle. We must investigate further if we can fit another replenishment at the end of the cycle any time after the last full-batch replenishment is provided. For this, we first define the time periods elapsed between the last full-batch replenishment time and the end of the cycle T as R which is $T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right)$. The question is that, is R high enough to accommodate one more batch or not. A schematic representation of a replenishment schedule is illustrated in Figure 8.

Figure 8. Representation of a replenishment schedule for Theorem 3


$\left\lceil\frac{DT_d}{2}\right\rceil$
$\left\lceil\frac{DT_d+1}{2}\right\rceil$
$\left\lceil\frac{DT_d}{2}\right\rceil$
$\left\lceil\frac{DT_d+1}{2}\right\rceil$
R
T
First batch
Second batch
Third batch
Second last guaranteed batch
Last guaranteed batch

Based on our observations, if R is less than or equal to $\left\lceil\frac{DT_d}{2}\right\rceil$, adding one more full batch replenishment at any time period within R creates a non-reduced replenishment schedule. Thus, if $R \leq \left\lceil\frac{DT_d}{2}\right\rceil$, number of full-batch replenishments is $\left\lfloor\frac{T}{\left\lceil\frac{DT_d}{2}\right\rceil+\left\lceil\frac{DT_d+1}{2}\right\rceil}\right\rfloor * 2$. However, if $R \geq \left\lceil\frac{DT_d}{2}\right\rceil + 1$, number of full-batch replenishments is increased to $\left\lfloor\frac{T}{\left\lceil\frac{DT_d}{2}\right\rceil+\left\lceil\frac{DT_d+1}{2}\right\rceil}\right\rfloor * 2 + 1$ because a full-batch replenishment must be provided at any time within R to create a reduced full-batch-only replenishment schedule.

We now evaluate two scenarios where DT_d is even or odd. If this is even, $R = T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right)$ must be at most $\frac{DT_d}{2}$ to not allow fitting one more replenishment in a reduced sense at the end of the cycle. If DT_d is odd, $R = T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right)$ must be at most $\frac{DT_d-1}{2}$ to not allow fitting another replenishment in a reduced sense at the end of the cycle. We arrange these two scenarios below:

For even DT_d:

$R \leq \frac{DT_d}{2}$, it means that no additional batch can fit. This is equivalent to:

$T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right) \leq \frac{DT_d}{2}$. This is equivalent to:

$T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right) \leq DT_d - \frac{DT_d}{2}$, and this can be written as follows:

$$T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right) \leq DT_d - \left\lceil\frac{DT_d}{2}\right\rceil.$$

For odd DT_d:

$R \leq \frac{DT_d-1}{2}$, it means that no additional batch can fit. This is equivalent to:

$T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right) \leq \frac{DT_d-1}{2}$. This is equivalent to:

$T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right) \leq DT_d - \left(\frac{DT_d+1}{2}\right)$, and this can be written as below:

$$T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right) \leq DT_d - \left\lceil\frac{DT_d}{2}\right\rceil.$$

So, regardless of whether DT_d is even or odd, we get the same formula above. We conclude that if $T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right) \leq DT_d - \left\lceil\frac{DT_d}{2}\right\rceil$, $NumRepl_d = \left\lfloor\frac{T}{\left\lceil\frac{DT_d}{2}\right\rceil+\left\lceil\frac{DT_d+1}{2}\right\rceil}\right\rfloor * 2$. Otherwise, if $T \bmod \left(\left\lceil\frac{DT_d}{2}\right\rceil + \left\lceil\frac{DT_d+1}{2}\right\rceil\right) \geq DT_d - \left\lceil\frac{DT_d}{2}\right\rceil + 1$, $NumRepl_d = \left\lfloor\frac{T}{\left\lceil\frac{DT_d}{2}\right\rceil+\left\lceil\frac{DT_d+1}{2}\right\rceil}\right\rfloor * 2 + 1$. This proves Theorem 3. ■

Definition 5: *The minimum and maximum total amounts that can possibly be demanded by all demanders during a cycle are as follows:*

$$MinTotalD = \sum_{d=1}^{D} MinAmtD_d \tag{40}$$

$$MaxTotalD = \sum_{d=1}^{D} MaxAmtD_d \tag{41}$$

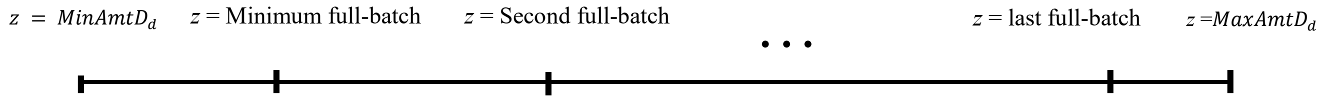
Conjecture 1: *For every integer z such that $MinAmtD_d \leq z \leq MaxAmtD_d$, there exists a reduced replenishment schedule for demander d in which a total of exactly z units is provided.*

Discussion: We experimentally analyzed the continuum of all possible total replenishment amounts *z* in the range $[MinAmtD_d, MaxAmtD_d]$ for every possible scenario in which T, DT_d, and DQ_d satisfy $10 \leq T \leq 30$, $2 \leq DT_d \leq 9$, and $1 \leq DQ_d \leq 9$. Using a smaller version of the math model that has only one demander and no suppliers, it was experimentally verified that a reduced replenishment schedule can be created for demander d in all scenarios. This gives us confidence that Conjecture 1 is true.

Figure 9 illustrates a continuum of total replenishment amounts z in the range $[MinAmtD_d$, $MaxAmtD_d]$. A formal proof of this conjecture would likely follow a line of reasoning related to an idea called the "trio operation." The trio operation applies to any reduced replenishment schedule with the total replenishment amount z and $k * DQ_d \leq z \leq MaxAmtD_d$ where k is the smallest integer that $k * DQ_d \geq MinAmtD_d$. We refer to the reduced replenishment schedule that has a total replenishment of $k * DQ_d$ as the minimum full-batch schedule. Therefore, if there are n full-batch schedules within the range $[MinAmtD_d, MaxAmtD_d]$, $(k + 1) * DQ_d$ would be the total replenishment for the second full-batch schedule and so on until $(k + n - 1) * DQ_d$ which is the total replenishment for the last full-batch schedule, $(k + n - 1) * DQ_d \leq MaxAmtD_d$.

To elaborate, any total replenishment of $z \geq k * DQ_d$ falls within a range of total replenishment of one full-batch schedule and that of the next full-batch schedule. It is found that within the range of one full-batch schedule to the next full-batch schedule, a new reduced replenishment schedule can be created by using a trio-operation on the current reduced replenishment schedule which adds 1 to the total units replenished. Let us refer to the total replenishment for the starting full-batch schedule of this range as z_1 and the total replenishment for the ending full-batch schedule of this range as z_2. Consider any reduced replenishment schedule in this range as R_1. A trio operation can always be performed on R_1 to increase the total replenishment by 1 and create a new reduced replenishment schedule (R_2). In other words, in order to create R_2 which has in total n units ($n <= z_2 - z_1$) more than z_1, we must perform the trio operation on the starting full-batch schedule of the range, n times.

Figure 9. Continuum of total replenishment amounts z



The trio operation is to select three locations (time periods) for which the replenishment values are changed in a way that (a) the resulting schedule is still feasible and (b) the reduced nature of the replenishment schedule is still preserved. The three replenishment values are increased by 1, decreased by 1 and increased by 1, respectively so that the net amount of increase in the total replenished units is 1. Suppose that a reduced replenishment schedule for demander d is to be created using T, DT_d, DQ_d, and z. Step 1 is to have a starting full batch schedule that has the largest total replenishment less than or equal to z. In step 1, we create a list of interval lengths that satisfies (i) the list has v values (# of full batches), (ii) values sum to T, (iii) each value is at least 1 and less than or equal to T, (iv) each two consecutive values must sum to at least DT_d+1, and (v) there should be three consecutive values that sum to less than or equal to $2*DT_d$. These values identify the trio location. Creating the starting full batch schedule in this way ensures that the first three time periods which have non-zero replenishment amounts are the appropriate periods to perform step 2 of trio operation which is to increase by 1, decrease by 1 and increase by 1 the three replenishment amounts.

Definition 6: *A* ***feasible supply schedule*** *for supplier s is a set of values* $SAmt_{s1}$, $SAmt_{s2}$, ... , $SAmt_{sT}$ *satisfying the supplier constraints (28-29) in the math model.*

Theorem 4: *The maximum total amount that can be provided by supplier s in a feasible supply schedule is:*

$$MaxAmtS_s = (MaxNumS_s)(SQ_s) \tag{42}$$

where $$MaxNumS_s = \left\lfloor \frac{T}{ST_s} \right\rfloor \tag{43}$$

Proof: Supplier s is able to supply one batch every ST_s or less often. Therefore, during a cycle length of T, supplier s can supply at most $MaxNumS_s$ times which is $\frac{T}{ST_s}$ rounded down to the nearest integer. The maximum total amount that supplier s is able to supply results from providing full batches (SQ_s) whenever it delivers products to the central node. This indicates that the maximum total amount that supplier s is able to supply equals $(MaxNumS_s)(SQ_s)$ where $MaxNumS_s = \left\lfloor \frac{T}{ST_s} \right\rfloor$. ■

Theorem 5: *For every integer z such that* $0 \leq z \leq MaxAmtS_s$, *there exists a feasible supply schedule for supplier s in which a total of exactly z units is supplied.*

Proof: As discussed in Theorem 4, the maximum total amount that supplier *s* is able to supply can be derived by having a feasible schedule with $MaxNumS_s$ full batches of size SQ_s. Based on definition 6, since $SAmt_{st} = 0$ or $SAmt_{st} = SQ_s$ for each *t* satisfy the supplier constraints (28-29) in the math model, any supply amounts $SAmt_{s,t}$ (for any *t*) less than the full batch size SQ_s also satisfy the supplier constraints (28-29). As part of the problem's assumptions, supplier *s* is able to supply one batch with a size of no more than SQ_s every ST_s or less often. In other words, it is allowed to reduce the size of the batches. Therefore, there is a feasible schedule for any total amount *z* that is less than or equal to the maximum total supply $MaxAmtS_s$. ■

Definition 7: *The maximum total amount that can possibly be supplied by all suppliers during a cycle is as follows:*

$$MaxTotalS = \sum_{s=1}^{S} MaxAmtS_s \tag{44}$$

Theorem 6: *A problem instance is feasible if and only if MaxTotalS ≥ MinTotalD.*

Proof: We first show that the problem is infeasible if *MaxTotalS* < *MinTotalD*. If we sum up constraint (33) and all constraints of type (34) in the math model, we obtain the following:

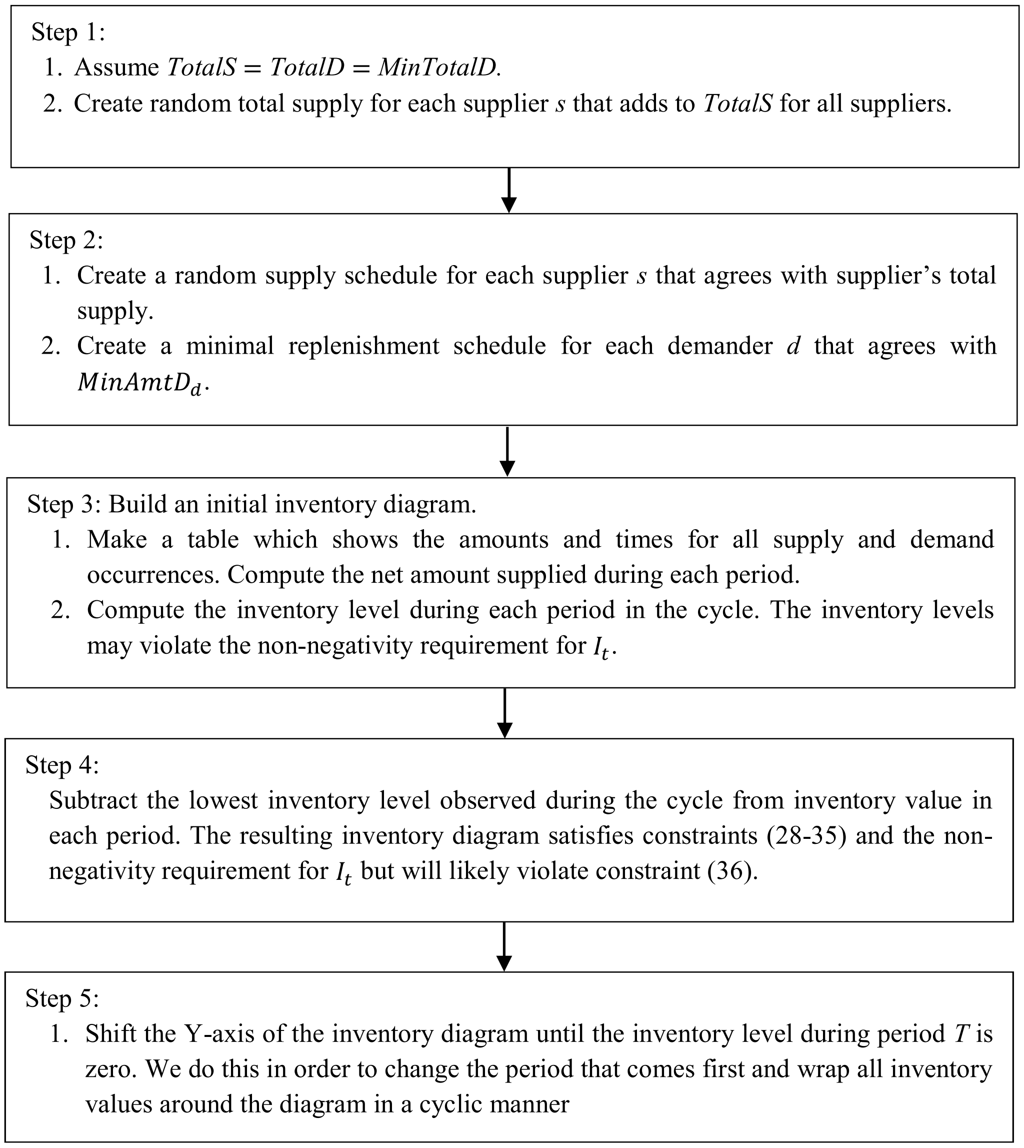
$$\sum_{t=1}^{T}\sum_{s=1}^{S} SAmt_{st} = \sum_{t=1}^{T}\sum_{d=1}^{D} DAmt_{dt} \tag{45}$$

In other words, the total amount supplied during the entire cycle should equal the total amount demanded. If *MaxTotalS<MinTotalD*, the supplies fall short of the demands, the above requirement cannot be met, and the problem is infeasible. Section 3.7 shows how to construct a feasible solution whenever *MaxTotalS ≥ MinTotalD*. ■

3.7. Method for automatically creating a feasible solution

We now present a method for automatically generating a feasible solution whenever *MaxTotalS* ≥ *MinTotalD*. We show how to create a feasible solution when *MaxTotalS* = *MinTotalD*. An overall five-step process for creating such an initial feasible solution is shown in Figure 10.

Figure 10. Method for automatically creating a feasible solution



Step 1:

1. Assume *TotalS* = *TotalD* = *MinTotalD*.
2. Create random total supply for each supplier *s* that adds to *TotalS* for all suppliers.

Step 2:

1. Create a random supply schedule for each supplier *s* that agrees with supplier's total supply.
2. Create a minimal replenishment schedule for each demander *d* that agrees with $MinAmtD_d$.

Step 3: Build an initial inventory diagram.

1. Make a table which shows the amounts and times for all supply and demand occurrences. Compute the net amount supplied during each period.
2. Compute the inventory level during each period in the cycle. The inventory levels may violate the non-negativity requirement for I_t.

Step 4:

Subtract the lowest inventory level observed during the cycle from inventory value in each period. The resulting inventory diagram satisfies constraints (28-35) and the non-negativity requirement for I_t but will likely violate constraint (36).

Step 5:

1. Shift the Y-axis of the inventory diagram until the inventory level during period *T* is zero. We do this in order to change the period that comes first and wrap all inventory values around the diagram in a cyclic manner

In the feasible solution that we create, the total amount provided by all suppliers and to all demanders equals *MinTotalD*. This requires each demander d to be provided with the bare minimum – a total of $MinAmtD_d$ – units during the cycle. In other words, each demander is satisfied via a minimal replenishment schedule.

We show that a minimal replenishment schedule for demander d can be created using two types of subschedules or *strings*: (1) a *repeating string* that repeats at the start of the replenishment schedule, and (2) a *final string* that completes the replenishment schedule. The repeating string has DT_d periods and DQ_d units are provided during this string. There are $\left\lfloor \frac{T}{DT_d} \right\rfloor$ such strings. The final string, if it exists, has $FinalT_d$ periods where $FinalT_d = T \bmod DT_d$ and the total amount provided during this string is $FinalQ_d = MinAmtD_d - \left\lfloor \frac{T}{DT_d} \right\rfloor * DQ_d$. Figure 11 illustrates a minimal replenishment schedule. R represents a repeating string and F represents a final string.

Figure 11. Representation of minimal replenishment schedule

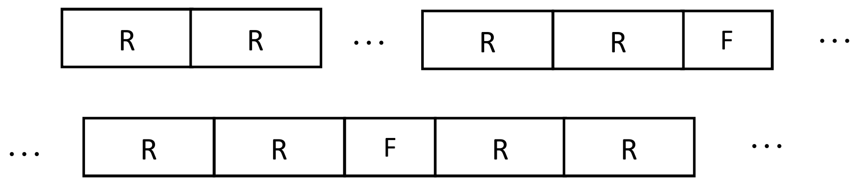


Figure 12 shows a minimal replenishment schedule for a demander with $T = 12$, $DT_d = 5$, and $DQ_d = 6$. In this scenario, $MinAmtD_d = 15$. There are $\left\lfloor \frac{T}{DT_d} \right\rfloor = \left\lfloor \frac{12}{5} \right\rfloor = 2$ repeating schedules, each repeating schedule has $DT_d = 5$ periods and within each string, a total of $DQ_d = 6$ is provided. The repeating string is (2, 1, 1, 1, 1). The final string has $T \bmod DT_d = 2$ periods and the total amount provided in this string is $MinAmtD_d - \left\lfloor \frac{T}{DT_d} \right\rfloor * DQ_d = 15 - \left\lfloor \frac{12}{5} \right\rfloor * 6 = 3$. The final string is (2, 1).

Figure 12. An example of minimal replenishment schedule

2, 1, 1, 1, 1	2, 1, 1, 1, 1	2, 1

The repeating string is formed to satisfy several conditions as follows:

(1) The string has DT_d values.

(2) The values sum to DQ_d.

(3) The average value in the string is $\frac{DQ_d}{DT_d}$.

(4) The amount provided in each period is either $\left\lceil\frac{DQ_d}{DT_d}\right\rceil$ or $\left\lceil\frac{DQ_d}{DT_d}\right\rceil - 1$.

(5) For all p from 1 to DT_d, the average of the first p values in the string should be greater than or equal to $\frac{DQ_d}{DT_d}$.

(6) For all p from 1 to DT_d, the value in the string's p^{th} position should be as low as possible while still keeping the average of the first p values greater than or equal to $\frac{DQ_d}{DT_d}$.

The final string is formed to satisfy the following conditions:

(0) The final string exists if $(T \bmod DT_d) > 0$.

(1) The final string has $(T \bmod DT_d)$ values.

(2) The values sum to $MinAmtD_d - \left\lfloor\frac{T}{DT_d}\right\rfloor * DQ_d$.

(3) The average value in the string is $\frac{MinAmtD_d - \left\lfloor\frac{T}{DT_d}\right\rfloor * DQ_d}{T \bmod DT_d}$

(4) The amount provided in each period is either $\left\lceil\frac{MinAmtD_d - \left\lfloor\frac{T}{DT_d}\right\rfloor * DQ_d}{T \bmod DT_d}\right\rceil$ or $\left\lceil\frac{MinAmtD_d - \left\lfloor\frac{T}{DT_d}\right\rfloor * DQ_d}{T \bmod DT_d}\right\rceil - 1$.

(5) For all p from 1 to $(T \bmod DT_d)$, the average of the first p values in the string should be greater than or equal to $\frac{MinAmtD_d - \left\lfloor\frac{T}{DT_d}\right\rfloor * DQ_d}{T \bmod DT_d}$.

(6) For all p from 1 to $(T \bmod DT_d)$, the value in the string's p^{th} position should be as low as possible while still keeping the average of the first p values greater than or equal to $\frac{MinAmtD_d - \left\lfloor\frac{T}{DT_d}\right\rfloor * DQ_d}{T \bmod DT_d}$.

We first discuss how to create replenishment schedules by following the procedure above using three examples. We then prove that all replenishment schedules created in the above manner are feasible and minimal.

Consider $T = 15$, $DT_d = 9$ and $DQ_d = 4$. In this scenario, $MinAmtD_d = \left\lceil T\frac{DQ_d}{DT_d}\right\rceil = \left\lceil (15)\frac{4}{9}\right\rceil = 7$. The repeating string has length 9 and 4 units must be provided during this string. In other words, each repeating string has 9 values that sum to 4. The amount provided in each period in such a string is either $\left\lceil\frac{DQ_d}{DT_d}\right\rceil - 1 = \left\lceil\frac{4}{9}\right\rceil - 1 = 0$ or $\left\lceil\frac{DQ_d}{DT_d}\right\rceil = \left\lceil\frac{4}{9}\right\rceil = 1$. As we move forward in the string from period 1 to period DT_d, the amount provided during period *p* should be as low as possible while still keeping the average of the string's first *p* values greater than or equal to $\frac{DQ_d}{DT_d} = \frac{4}{9}$. If we start with a batch of size 0, the average of values passed so far is 0 which is less than $\frac{4}{9}$ and this violates condition (6) in creating the repeating string. So, we must start with an amount of 1. So far, the list of amounts is (1). For the next batch size, the lowest possible value to select is 0. If we select 0, the average of the two values passed so far would be $\frac{1+0}{2} = \frac{1}{2}$ which is greater than $\frac{4}{9}$ and this satisfies condition (6). The list of batch sizes is updated to (1, 0). For the next batch, if we select 0, the average of the three values passed so far would be $\frac{1+0+0}{3} = \frac{1}{3}$ which is less than $\frac{4}{9}$ and this violates condition (6), whereas if we select 1, the average of the three values passed so far would be $\frac{1+0+1}{3} = \frac{2}{3}$ which is greater than $\frac{4}{9}$ and this satisfies condition (6). The list of batch sizes is updated to (1, 0, 1). In the same way, we continue and select the correct batch sizes (either 0 or 1) that agree with condition (6). The resulting repeating string is (1, 0, 1, 0, 1, 0, 1, 0, 0). This repeating string repeats for only $\left\lfloor\frac{T}{DT_d}\right\rfloor = \left\lfloor\frac{15}{9}\right\rfloor = 1$ time. We now create the final string. This string has $T \bmod DT_d = 15 \bmod 9 = 6$ values which must sum to $MinAmtD_d - \left\lfloor\frac{T}{DT_d}\right\rfloor * DQ_d = 7 - 1 * 4 = 3$. Possible batch sizes are $\left\lceil\frac{MinAmtD_d - \left\lfloor\frac{T}{DT_d}\right\rfloor * DQ_d}{T \bmod DT_d}\right\rceil - 1 = \left\lceil\frac{7 - \left\lfloor\frac{15}{9}\right\rfloor * 4}{15 \bmod 9}\right\rceil - 1 = 0$ and $\left\lceil\frac{MinAmtD_d - \left\lfloor\frac{T}{DT_d}\right\rfloor * DQ_d}{T \bmod DT_d}\right\rceil = \left\lceil\frac{7 - \left\lfloor\frac{15}{9}\right\rfloor * 4}{15 \bmod 9}\right\rceil = 1$. In order to satisfy condition (6) for the final string, each batch size is selected as low as possible while still keeping the average of values passed so far is greater than or equal to $\frac{MinAmtD_d - \left\lfloor\frac{T}{DT_d}\right\rfloor * DQ_d}{T \bmod DT_d} = \frac{7 - 1 * 4}{15 \bmod 9} = \frac{1}{2}$. For the first batch size, if we select 0, the average of the only value passed so far is 0 which is less than $\frac{1}{2}$. So, we must start with a batch size 1. The list of final string so far is (1). For the next batch size, if we select the lowest possible value 0, the average of the two values passed so far is $\frac{1+0}{2} = \frac{1}{2}$ which is equal to $\frac{1}{2}$ and satisfies condition (6) in creating the final string. The updated list of final string so far is (1, 0). We continue in the same manner and select batch sizes of 1 or 0 to satisfy condition (6). The resulting final string would be (1, 0, 1, 0, 1, 0). Thus, the entire reduced replenishment schedule is (1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0).

A second example is the scenario where $T = 10$, $DT_d = 3$ and $DQ_d = 8$. In this scenario, $MinAmtD_d = \left\lceil T\frac{DQ_d}{DT_d}\right\rceil = \left\lceil (10)\frac{8}{3}\right\rceil = 27$. The repeating string has length 3 and 8 units must be provided during this string. That means each repeating string has 3 values that sum to 8. The batch sizes in such a string are $\left\lceil\frac{DQ_d}{DT_d}\right\rceil - 1 = \left\lceil\frac{8}{3}\right\rceil - 1 = 2$ and $\left\lceil\frac{DQ_d}{DT_d}\right\rceil = \left\lceil\frac{8}{3}\right\rceil = 3$. As we move forward in the string, each batch size is selected as low as possible while still keeping the average of values passed so far as greater than or equal to $\frac{DQ_d}{DT_d} = \frac{8}{3}$. If we start with a batch of size 2, the average of values passed so far is 2 which is less than $\frac{8}{3}$ and this violates condition (6) in creating the repeating string. So, we must start with a batch of 3. For the next batch size, the lowest possible value to select is 2. If we select 2, the average of the two values passed so far would be $\frac{3+2}{2} = \frac{5}{2}$ which is less than $\frac{8}{3}$ and this violates condition (6). However, if we select 3, the average of the two values passed so far would be $\frac{3+3}{2} = 3$ which is greater than $\frac{8}{3}$ and this satisfies condition (6). For the third and the last batch size, if we start with the lowest possible value which is 2, the average of the three values passed so far would be $\frac{3+3+2}{3} = \frac{8}{3}$ which is equal to $\frac{8}{3}$ and this satisfies condition (6). Therefore, the resulting repeating string is (3, 3, 2) which repeats $\left\lfloor\frac{T}{DT_d}\right\rfloor = \left\lfloor\frac{10}{3}\right\rfloor = 3$ times. We now create the final string. This string has $T - DT_d * \left\lfloor\frac{T}{DT_d}\right\rfloor = 10 - 3 * 3 = 1$ value which must sum to $MinAmtD_d - \left\lfloor\frac{T}{DT_d}\right\rfloor * DQ_d = 27 - 3 * 8 = 3$.

The two possible batch sizes are $\left\lceil \frac{MinAmtD_d - \left\lfloor \frac{T}{DT_d} \right\rfloor * DQ_d}{T \bmod DT_d} \right\rceil - 1 = \left\lceil \frac{27 - \left\lfloor \frac{10}{3} \right\rfloor * 8}{10 \bmod 3} \right\rceil - 1 = 2$ and $\left\lceil \frac{MinAmtD_d - \left\lfloor \frac{T}{DT_d} \right\rfloor * DQ_d}{T \bmod DT_d} \right\rceil = \left\lceil \frac{27 - \left\lfloor \frac{10}{3} \right\rfloor * 8}{10 \bmod 3} \right\rceil = 3$. The only way to have 1 batch which sum to 3 is to have a batch of size 3. Thus, the entire replenishment schedule is (3, 3, 2, 3, 3, 2, 3, 3, 2, 3).

The third example is related to the case where $T = 12$, $DT_d = 5$ and $DQ_d = 6$. In this scenario, $MinAmtD_d = \left\lceil T \frac{DQ_d}{DT_d} \right\rceil = \left\lceil (12) \frac{6}{5} \right\rceil = 15$. Following the procedure for creating the repeating string, it will be (2, 1, 1, 1, 1). Also, following the procedure for making the final string, this string would be (2, 1). The overall replenishment schedule made in this manner would be (2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1).

Theorem 7: A replenishment schedule created in the above fashion is minimal.

Proof: A minimal replenishment schedule has two features:

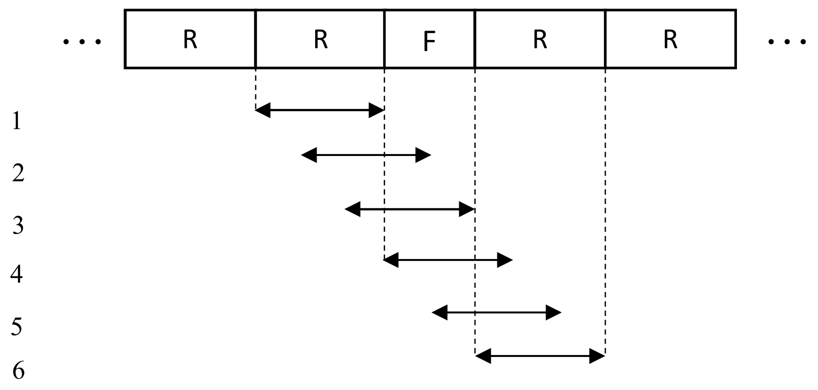
(1) The minimum total amount provided to each demander *d* during a cycle is $MinAmtD_d$.
(2) It is feasible.

By design, a replenishment schedule made in the above manner provides a total of $MinAmtD_d$ to demander *d*, because the total amount provided to demander *d* in all repeating strings is $DQ_d * \left\lfloor \frac{T}{DT_d} \right\rfloor$ and the total amount provided to demander *d* in the final string is $MinAmtD_d - DQ_d * \left\lfloor \frac{T}{DT_d} \right\rfloor$. This shows that (1) is true. We now show that such a replenishment schedule is feasible, i.e. that it satisfies constraint (31) in the math model. In other words, we show that during every time window of length DT_d, a minimum amount of DQ_d is provided to demander *d*.

Using the conditions to create the two types of strings, we are confident that replenishment schedules in each repeating string is feasible. We must investigate if feasibility is maintained as the time window of length DT_d slides from a particular position to the left/right in a cyclic manner so that it starts overlapping at least partially with the final string. Let us refer to sliding such a time window as a *Sliding Process*. We now discuss three existing Sliding Processes based on Figure 13.

The Sliding Process 1 occurs as we move from interval 1 to interval 3 in Figure 13. In particular, the Sliding Process 1 starts by sliding a time window of length DT_d to the right from the initial position where it covers the entire repeating string. This process ends when the right edge of the time window coincides with the right edge of the final string. The Sliding Process 2 occurs as we move from interval 3 to 4 in Figure 13. It starts by sliding a time window of length DT_d to the right from the initial position where the right edge of the time window coincides with the right edge of the final string and ends when the left edge of the time window coincides with the left edge of the final string. The Sliding Process 3 occurs as we move from interval 6 to 4 in Figure 13. It starts by sliding a time window of length DT_d to the left from the initial position where it covers the entire repeating string and ends when the left edge of the time window coincides with the left edge of the final string.

Figure 13. Schematic presentation of sliding a time window of length DT_d in a cyclic manner



We have three statements on how the values of repeating and final strings must compare in order to ensure that feasibility is maintained during each Sliding Process. These three statements are as follows:

1) For all p from 1 to $FinalT_d$, the average of the first p values in the final string is greater than or equal to the average of the first p values in the repeating string.

2) For all p from 1 to $DT_d - FinalT_d$, the average of the first p values in the repeating string is greater than or equal to the average of the values in positions $FinalT_d + 1$ to $FinalT_d + p$ inclusive in the repeating string.
3) For all p from 1 to $FinalT_d$, the average of the last p values in the final string is greater than or equal to the average of the last p values in the repeating string.

If statement 1 is true, it means that at least DQ_d units are provided in every time window of length DT_d encountered during Sliding Process 1. If statement 2 is true, it indicates that at least DQ_d items are provided in every time window of length DT_d encountered during Sliding Process 2 and if statement 3 is true, it shows that at least DQ_d items are provided in every time window of length DT_d encountered during Sliding Process 3.

To demonstrate the truth of these statements, we first show that (i) the average value in the final string is always greater than or equal to that of the repeating string, (ii) *the larger possible value* for the final string is greater than or equal to *the larger possible value* in the repeating string, and (iii) *the smaller possible value* for the final string is greater than or equal to *the smaller possible value* in the repeating string.

Regarding item (i), if the average value in the final string is always greater than or equal to that of the repeating string, we have:

$$\frac{MinAmtD_d - \left\lfloor \frac{T}{DT_d} \right\rfloor * DQ_d}{T \bmod DT_d} \geq \frac{DQ_d}{DT_d}, \text{ and this is equivalent to:} \tag{46}$$

$$\frac{MinAmtD_d - \left\lfloor \frac{T}{DT_d} \right\rfloor * DQ_d}{T - \left\lfloor \frac{T}{DT_d} \right\rfloor * DT_d} \geq \frac{DQ_d}{DT_d}$$

Therefore: $\left(MinAmtD_d * DT_d - \left\lfloor \frac{T}{DT_d} \right\rfloor * DQ_d * DT_d\right) \geq \left(DQ_d * T - DQ_d * DT_d * \left\lfloor \frac{T}{DT_d} \right\rfloor\right)$ and by doing mathematical manipulations, we get:

$MinAmtD_d \geq T\,\frac{DQ_d}{DT_d}$ which is true because $MinAmtD_d = \left\lceil T\,\frac{DQ_d}{DT_d} \right\rceil$ and $\left\lceil T\,\frac{DQ_d}{DT_d} \right\rceil \geq T\,\frac{DQ_d}{DT_d}$.

From inequality (46), we can also conclude that $\left\lceil \frac{MinAmtD_d - \left\lfloor \frac{T}{DT_d} \right\rfloor * DQ_d}{T \bmod DT_d} \right\rceil \geq \left\lceil \frac{DQ_d}{DT_d} \right\rceil$ as well as $\left\lceil \frac{MinAmtD_d - \left\lfloor \frac{T}{DT_d} \right\rfloor * DQ_d}{T \bmod DT_d} \right\rceil - 1 \geq \left\lceil \frac{DQ_d}{DT_d} \right\rceil - 1$. The former indicates that *the larger possible value* for the final string is greater than or equal to *the larger possible value* in the repeating string which shows (ii) is true. The latter indicates that *the smaller possible value* for the final string is greater than or equal to *the smaller possible value* in the repeating string showing (iii) is true.

We now argue that statement (1) is true. Consider a time window of length DT_d which only covers the entire repeating string to the left of the final string and the total amount provided within this time window is at least DQ_d. Now, we slide this time window one period to the right so that it starts with the second period in the repeating string and ends with the first period of the final string. In other words, the first value of the repeating string is replaced with the first value of the final string with this move. This time window has at least DQ_d units because both repeating and final strings are always started with generally the larger possible value to maintain the minimum average value required and the larger possible value for the final string is always greater than or equal to the larger possible value in the repeating string (item ii). Therefore, by moving 1 period to the right, we gain the first value of the final string which is as large as the first value of the repeating string that we lost and this ensures to maintain a minimum of DQ_d units within this recently formed window of length DT_d.

As we slide the time window of length DT_d one other period to the right, it starts with the third period in the repeating string and ends with the second period of the final string. In other words, the first and second values of the repeating string are replaced with the first and second values of the final string. This time window certainly has at least DQ_d units because the average of the first two values (which we gain by sliding the time window to the right) in the final string is greater than or equal to the average of the first two values (which we lose by sliding the time window to the right) in the repeating string. This is true based on our observations. The second value in final string is generally the smaller possible value for this string that would suffice to maintain the required average so far. Also, the second value in repeating string is generally the smaller possible value that would suffice to keep the required average. Considering item (iii), the second value of the final string is always greater than or equal to that of the repeating string. This together with the first value for the final string being greater than or equal to the first value for the repeating string indicates that the average of the first two values in the final string is always greater than or equal to that in the repeating string. The same pattern holds as we slide the time window for p time periods ($p \leq FinalT_d$). We now show that this pattern exists in the first example that was discussed earlier.

In the first example with the repeating string of (1, 0, 1, 0, 1, 0, 1, 0, 0) and final string (1, 0, 1, 0, 1, 0), we observe that for any p from 1 to $FinalT_d$ = 6, the average of the first p values in the final string is at least the average of the first p values in the repeating string. The average of the first value in the final string is 1 which is equal to the average of the first value of repeating string. Therefore, as we slide a time window of length $DT_d = 9$ one period to the right, the first period of the repeating string is replaced with the first period of the final string and the total amount provided sum to at least $DQ_d = 4$. The average of the first two value in the final string is $\frac{1}{2}$ which is equal to the average of the first two value of repeating string ($\frac{1}{2}$). Therefore, as we slide a time window of length 9 one more period to the right, the first and second periods of the repeating string are replaced with the first and second periods of the final string and the total amount provided sum to at least $DQ_d = 4$. For p = 3, the average of the first three values in the final string is $\frac{2}{3}$ which is equal to the average of the first three values in the repeating string. If we continue in the same way and slide the time window of length 9 for p times ($p \leq 6$), such a shift always maintains a minimum of 4 units within that time window.

We argue that statement 2 is true. This statement ensures that feasibility is maintained in any window of length DT_d that covers the entire final string. Such a time window also includes part of the repeating string (at most $DT_d - FinalT_d$ periods) to the left of the final string or part of the repeating string (at most $DT_d - FinalT_d$ periods) to the right of the final string considering the cyclic nature of the problem. Consider the window of length DT_d that includes the entire final string and $DT_d - FinalT_d$ periods of the repeating string to the left of the final string. As we slide this time window one period to the right, we warp around the cycle and the first period of the repeating string to the right of final string replaces the period in position $FinalT_d$ of the repeating string to the left of the final string. This time window of length DT_d is ensured to cover a minimum of DQ_d units because each repeating string starts with the *larger* possible value and by sliding the time window one period to the right, the *smaller* or *larger* possible value towards the end (at position $FinalT_d$) of the repeating string is certainly replaced with the *larger* possible value at the start of the repeating string to the right of the final string. This maintains a minimum of DQ_d during such a time window. As we slide the time window one more period forward in a cyclic way, the same pattern holds because each repeating string includes generally higher values towards the beginning and lower values towards the end of the string. Therefore, by shifting to the right for $p \leq DT_d - FinalT_d$ periods, lower values towards the end (at positions $FinalT_d + 1$ to $FinalT_d + p$) of the repeating string to the left of the final string are replaced with equal or higher p values at the beginning of the repeating string to the right of the final string. We now show this in the first example.

In the first example, the repeating schedule is (1, 0, 1, 0, 1, 0, 1, 0, 0) and we observe that for all p from 1 to 9 – 6 = 3 , the average of the first p values in the repeating string is greater than or equal to the average of the values in positions 7 to $6 + p$ in the repeating string. For p = 1, the average of first value in the repeating string is 1 which is equal to the average of the value at position 7. For p = 2, the average of the first two values in the string is $\frac{1}{2}$ which is equal to the average of the values at positions 7 to 8. For p = 3, the average of the first three values in the string is $\frac{2}{3}$ which is greater than the average of the values at positions 7 to 9 in the string, $\frac{1}{3}$.

Statement 3 ensures that feasibility is maintained in any time window of length DT_d that can be formed by wrapping around the cycle and includes at least one time period of the final string and at least $DT_d - 1$ periods of the first repeating string. We argue that this is true based on our observations. The values towards the end of the repeating cycle tend to generally include more of the *smaller possible value* for this string. The values towards the end of final string could be either the larger or the smaller possible value for this string which are both greater than or equal to the *smaller possible value* for the final string (items ii and iii). In the first example, the final string is (1, 0, 1, 0, 1, 0) and the repeating string is (1, 0, 1, 0, 1, 0, 1, 0, 0). all p from 1 to $FinalT_d = 6$, average of last p values in the final string is greater than or equal to average of last p values in the repeating string. For p = 1, the average of the last value in final string (0) is equal to the average of the last value in repeating string (0). For p = 2, the average of last two values for final string which is $\frac{1}{2}$ is greater than the average of last two values in repeating string (0). For p = 3, the average of last three values in final string $-\frac{1}{3}-$ is equal to the average of last three values of repeating string ($\frac{1}{3}$). For p = 4, the average of last four values in final string $-\frac{2}{4}-$ is greater than the average of last four values of repeating string ($\frac{1}{4}$). For p = 5, the average of last five values in final string $-\frac{2}{5}-$ is equal to the average of last five values of repeating string ($\frac{2}{5}$). Finally, the average of the last six values in final string $-\frac{3}{6}-$ is greater than the average of last six values of repeating string ($\frac{2}{6}$). This proves Theorem 7. ■

3.8. Experiment setup, results, and discussion

In this section, we test the performance of the integer programing Solver IBM ILOG CPLEX 12.9 on a set of benchmark problem instances.

3.8.1. Generating problem instances

Table 30 shows the parameter values that define the problem instances considered in the experiments. We use the discrete uniform (DU) distribution to randomly generate demand time requirements (DT_d), minimum inter-supply times (ST_s), and quantities (DQ_d and SQ_s). In each instance, the cost parameters H_1, H_2, FC_s, FCP_s, VC_s, and FCD_d are also randomly generated from the DU distribution as displayed in Table 30. We designed two types of instances: easy instances and hard instances. Easy instances only satisfy the requirement that they be feasible. For hard instances, the condition *MaxTotalS* – *MinTotalD* ≤ 10 must be satisfied in addition to feasibility. A 10-digit code "dDDsSStTTT" indicates the size of a problem instance. "DD" represents the number of demanders, and "SS" and "TTT" indicate the number of suppliers and the cycle length, respectively. As an example, "d02s02t010" is a problem instance with two demanders, two suppliers, and ten periods. Importantly, the values of the operational parameters $-$ D, S, T, DT_d, DQ_d, ST_s, SQ_s $-$ in all instances are identical to those considered by Petering et al. (2019). This allows us to directly compare our results to those in Petering et al. (2019).

Table 30. Parameter values used in the experiments

Parameter	Possible values
D, S	2, 6, 20, 60
T	10, 30, 100
DT_d	DU(2,9)
DQ_d	DU(1,9)
ST_s	DU(2,9)
SQ_s	DU(1,9)
H_1	DU(1,3)
H_2	DU(1,3)
FC_s	DU(20,50)
FCP_s	DU(4,10)
VC_s	DU(1,3)
FCD_d	DU(4,10)

3.8.2. Hardware settings, CPLEX settings, and termination criteria

The pure integer programming (IP) model shown in Table 29 was coded in Microsoft Visual C++ 2017 Professional. IBM ILOG Concert Technology was used within C++ to call IBM ILOG CPLEX 12.9 to solve instances defined in text files. The experiments were executed on a desktop computer with 16 gigabytes of RAM, the Windows 10 Education 64-bit operating system, and an Intel Core i7-8700 processor with 3.2 gigahertz processors. The CPLEX-based method terminates after 60 seconds has elapsed. The termination criteria and settings for the heuristic algorithm will be decided at a later time.

3.8.3. Results for easy problem instances

In the first set of experiments, we consider twelve problem sizes corresponding to all possible combinations of four values for *D* and S — 2, 6, 20, or 60 — and three cycle lengths — 10, 30, or 100 periods. Ten instances are considered for each problem size. Each instance is considered in the context of three objective functions: (A) a complex objective function in which all cost coefficients listed in Table 30 are non-zero; (B) a simple objective function in which $H_1 = 1$ and all other cost coefficients equal 0; and (C) a simple objective function in which $H_2 = 1$ and all other cost coefficients equal 0. These are referred to as objectives A-C, respectively. The purpose of considering objectives B and C is to allow a comparison with the results in Petering et al. (2019) which considers similar objectives.

Table 31 shows the detailed results when CPLEX is used to solve each of the 120 easy problem instances when objective A is considered. For each problem size, we run CPLEX on 10 different instances as indicated by the numbers # 0-9 which run across the top row of the table. The first column shows the problem size. The next ten columns show the objective value of the best solution identified by CPLEX for each instance of that problem size. Highlighted cells show optimal solutions, whereas unhighlighted ones show only feasible solutions identified within 60 seconds run time.

As expected, Table 31 shows that the instances generally get harder as *T* increases. CPLEX finds optimal solutions to all instances with $T = 10$, regardless of the values of *D* or *S*, but finds progressively fewer optimal solutions as *T* increases. For problem sizes with $T = 30$, CPLEX finds optimal solutions for all instances in problem size d02s02t030. As *D* and *S* increase and instances generally get harder, CPLEX finds fewer optimal solutions. CPLEX finds optimal solutions for only two instances of problem size d06s06t030 and finds no optimal solution for problem sizes d20s20t030 and d60s60t030. For problem sizes with $T = 100$, CPLEX finds optimal solutions for only two instances of problem size d02s02t100. As *D* and *S* increase, CPLEX has no success in finding any optimal solutions for problem sizes d06s06t100, d20s20t100, and d60s60t100.

Table 31. CPLEX results for the easy instances with objective A

Problem Size	Instance									
	0	1	2	3	4	5	6	7	8	9
d02s02t010	**149**	**186**	**189**	**246**	**154**	**187**	**219**	**210**	**243**	**115**
d02s02t030	**299**	**277**	**325**	**647**	**629**	**630**	**354**	**335**	**690**	**448**
d02s02t100	**1047**	648	904	2454	1429	1219	**2081**	1563	881	1483
d06s06t010	**482**	**483**	**546**	**577**	**571**	**571**	**524**	**700**	**471**	**535**
d06s06t030	**1181**	**788**	1311	1067	1141	1046	1299	1246	1034	1358
d06s06t100	2652	2918	4900	3617	3933	4031	3785	4973	3955	3887
d20s20t010	**2144**	**1787**	**1887**	**2151**	**1533**	**1994**	**1767**	**1546**	**1767**	**2012**
d20s20t030	5085	4726	4260	4662	4058	4040	3616	3759	3299	5175
d20s20t100	13768	13301	14617	54612	24472	15479	13495	13256	13353	15778
d60s60t010	**4830**	**6365**	**5902**	**5475**	**6415**	**6144**	**5692**	**5625**	**5794**	**4933**
d60s60t030	14755	14170	13336	15460	11885	13975	12685	13617	13800	12891
d60s60t100	65711	66224	82862	38797	80001	55918	40322	86571	77141	49802

Bold: Optimal solution

Tables 32-33 show the results when objectives B and C are considered. Our problem is a relaxation of the model considered in Petering et al. (2019). Therefore, by setting all costs in this model equal to zero except $H_1 = 1$ (Table 32) and $H_2 = 1$ (Table 33), we seek to verify that the optimal objective values for this relaxed model are lower bounds for the previous model in Petering et al. (2019). The results in Tables 32-33 verify this.

Table 32. CPLEX results for the easy instances with objective B

Problem Size	Instance									
	0	1	2	3	4	5	6	7	8	9
d02s02t010	**2**	**0**	**0**	**3**	**0**	**0**	**1**	**0**	**2**	**0**
d02s02t030	**0**	**0**	**0**	**0**	**30**	**0**	**0**	**0**	**24**	**9**
d02s02t100	**0**	**0**	**0**	187	**0**	**5**	**150**	**132**	**0**	16
d06s06t010	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d06s06t030	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d06s06t100	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d20s20t010	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d20s20t030	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d20s20t100	547	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d60s60t010	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d60s60t030	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d60s60t100	**0**	**0**	N/A	**0**	**0**	N/A	**0**	N/A	**0**	**0**

N/A: Can't find any feasible solution within 60 seconds.
Bold: Optimal solution

Table 33. CPLEX results for the easy instances with objective C

Problem Size	Instance									
	0	1	2	3	4	5	6	7	8	9
d02s02t010	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**0**
d02s02t030	**0**	**0**	**0**	**0**	**2**	**0**	**0**	**0**	**2**	**2**
d02s02t100	**0**	**0**	**0**	**4**	**0**	**1**	**5**	**3**	**0**	**1**
d06s06t010	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d06s06t030	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d06s06t100	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d20s20t010	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d20s20t030	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d20s20t100	17	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d60s60t010	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d60s60t030	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d60s60t100	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Bold: Optimal solution

3.8.4. Results for hard problem instances

The results from Section 3.9.3 show that CPLEX performs well on most instances but does not perform well on the largest instances. In this section, we analyze the performance of the solution method on more difficult problem instances. In fact, we found that when *MaxTotalS* is close to *MinTotalD*, instances become harder to solve to optimality. Thus, we randomly generated five new sets of feasible problem instances with *MaxTotalS* – *MinTotalD* $\leq$ 10. The sizes of these instances are different than the sizes of the easy problem instances. Ten instances are considered for each set. Thus, a total of 50 hard problem instances are considered.

As before, each instance is considered in the context of three objective functions: (A) a complex objective function in which all cost coefficients listed in Table 30 are non-zero; (B) a simple objective function in which $H_1 = 1$ and all other cost coefficients equal 0; and (C) a simple objective function in which $H_2 = 1$ and all other cost coefficients equal 0.

Table 34 shows the results when CPLEX is used to solve each hard problem instance when objective A is considered. CPLEX finds optimal solutions in only 5 of the 50 total hard instances. Therefore, the success rate in finding optimal solutions is much smaller for these hard instances (10 percent) than the easy instances (45 percent). Moreover, comparing the results for the easy and hard instances for objective A, it appears that the best objective values obtained for the hard instances are much higher than for the easy instances. The results in Table 34 also indicate that the value of *T* has a greater impact on problem difficulty than the values of *D* and *S*.

Table 34. CPLEX results for the hard instances with objective A

Problem Size	Instance									
	0	1	2	3	4	5	6	7	8	9
d06s06t030	1765	1527	1523	1103	1405	1234	1553	1762	1667	1490
d06s06t100	6652	4688	4618	3639	4705	5298	4174	4145	4220	4690
d10s10t010	1077	**1251**	763	833	**853**	**996**	740	813	**785**	**811**
d10s10t030	2313	2292	2577	1901	2372	1953	2549	2176	2780	3273
d10s10t100	8932	8889	8472	7599	8759	10254	8231	7721	7472	9357

Bold: Optimal solution

Tables 35-36 show the results when objectives B and C are considered. Similar to the analysis for easy instances, by setting all costs in this model equal to zero except $H_1 = 1$ (Table 35) and $H_2 = 1$ (Table 36), we seek to verify that the optimal objective values for this relaxed model are lower bounds for the results of the previous model in Petering et al. (2019) for hard instances. The results in Tables 35 and 36 verify this. Once again, these results show that the value of T affects problem difficulty more than the values of D and S.

Table 35. CPLEX results for the hard instances with objective B

Problem Size	Instance									
	0	1	2	3	4	5	6	7	8	9
d06s06t030	**0**	1	**22**	1	**0**	**0**	**1**	**35**	**0**	**9**
d06s06t100	180	83	151	46	70	83	94	26	115	107
d10s10t010	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d10s10t030	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d10s10t100	162	235	160	89	94	65	21	61	51	97

Bold: Optimal solution

Table 36. CPLEX results for the hard instances with objective C

Problem Size	Instance									
	0	1	2	3	4	5	6	7	8	9
d06s06t030	**0**	1	**3**	**1**	**0**	**0**	**1**	**6**	**0**	**1**
d06s06t100	**4**	5	5	2	3	4	5	2	6	6
d10s10t010	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d10s10t030	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d10s10t100	5	1	5	3	4	4	5	N/A	**0**	6

N/A: Can't find any feasible solution within 60 seconds.
Bold: Optimal solution

3.9. Conclusion

In this essay, we studied a single-item cyclic coordinated supply and inventory replenishment problem with batch supplies and flexible demands. The system in this study consists of multiple suppliers who each deliver a single item to a central node from which multiple demanders are then replenished. Each supplier is capable of providing no more than a specific batch size to the central node at a rate that cannot exceed a specified frequency. Moreover, each demander must be replenished with a specific minimum amount over every existing time window of a given length. The objective is to minimize total system cost subject to several operational constraints. The decisions include the timing and sizes of batches delivered by the suppliers to the central node and the timing and amounts by which demanders are replenished. This problem is modeled as an integer linear program and results show CPLEX is a good method for solving small problem instances. Future work includes proposing a heuristic algorithm for quickly finding good solutions to the math model. Such a heuristic algorithm will be composed of a three-tier hierarchy which draws upon ideas of genetic algorithms, construction heuristics, and improvement heuristics. The highest level in the hierarchy will be managed by a genetic algorithm. We will show that the proposed heuristic method outperforms CPLEX on medium-sized and large problem instances.

References

Abdelsalam, H. M., Elassal, M. M. (2014). Joint economic lot sizing problem for a three-layer supply chain with stochastic demand. *International Journal of Production Economics*, *155*, 272–283.

Archetti, C., Bertazzi, L., Speranza, M. G. (2014). Polynomial cases of the economic lot-sizing problemwith cost discounts. *European Journal of Operational Research*, *237*(2), 519–527.

Banerjee, A., (1986). A joint economic-lot-size model for purchaser and vendor. *Decision Sciences, 17*, 292–311.

Banerjee, A., Burton, J.S., (1994). A coordinated order-up-to inventory control policy for a single supplier and multiple buyers using electronic data interchange. *International Journal of Production Economics*, *35*, 85–91.

Ben-Daya, M., Darwish, M., Ertogral, K. (2008). The joint economic lot-sizing problem: Review and extensions. *European Journal of Operational Research*, *185*(2), 726–742.

Bouslah, B., Gharbi, A., Pellerin, R. (2013). Joint optimal lot-sizing and production control policy in an unreliable and imperfect manufacturing system. *International Journal of Production Economics*, *144*(1), 143–156.

Braglia, M., Zavanella, L., (2003). Modelling an industrial strategy for inventory management in supply chains: the 'consignment stock' case. *International Journal of Production Research*, *41*, 3793–3808.

Chen, J. M., Lin, I. C., Cheng, H. L. (2010). Channel coordination under consignment and vendor-managed inventory in a distribution system. *Transportation Research Part E: Logistics and Transportation Review*, *46*(6), 831-843.

Cunha, J. O., Melo, R. A. (2016). A computational comparison of formulations for the economic lot-sizing with remanufacturing. *Computers & Industrial Engineering*, *92*, 72–81.

Cunha, J. O., Konstantaras, I., Melo, R. A., Sifaleras, A. (2017). On multi-item economic lot-sizing with remanufacturing and uncapacitated production. *Applied Mathematical Modelling*, *50*, 772–780.

Darwish, M.A., Odah, O.M. (2010). Vendor managed inventory model for single- vendor multi-retailer supply chains. *European Journal of Operational Research*, *204*, 473–484.

Das, D., Roy, A., Kar, S. (2011). A volume flexible economic production lot-sizing problem with imperfect quality and random machine failure in fuzzy-stochastic environment. *Computers & Mathematics with Applications*, *61*(9), 2388–2400.

De Araujo, S. A., De Reyck, B., Degraeve, Z., Fragkos, I., Jans, R. (2015). Period decompositions for the capacitated lot sizing problem with setup times. *INFORMS Journal on Computing*, *27*(3), 431-448.

Eppen, G., Martin, R. (1987). Solving multi-item capacitated lot-sizing problems using variable redefinition. *Operations Research*, *35* (6), 832–848.

Feng, Y., Chen, S., Kumar, A., Lin, B. (2011). Solving single product economic lot-sizing problem with non-increasing setup cost, constant capacity and convex inventory cost in O (NlogN) time. *Computers & Operations Research*, *38*(4), 717–722.

Ferretti, I., Mazzoldi, L., Zanoni, S., Zavanella, L. E. (2017). A joint economic lot size model with third-party processing. *Computers & Industrial Engineering*, *106*, 222–235.

Fiorotto, D. J., Jans, R., De Araujo, S. A. (2017). An analysis of formulations for the capacitated lot sizing problem with setup crossover. *Computers & Industrial Engineering*, *106*, 338-350.

Bayley, T., Süral, H., Bookbinder, J. H. (2018). A hybrid Benders approach for coordinated capacitated lot-sizing of multiple product families with set-up times. *International Journal of Production Research*, *56*(3), 1326-1344.

Fragkos, I., Degraeve, Z., De Reyck, B. (2016). A horizon decomposition approach for the capacitated lot-sizing problem with setup times. *INFORMS Journal on Computing*, *28*(3), 465-482.

Gharaei,A., Pasandideh, S. H. R. (2016). Modelling and optimization the four-level integrated supply chain: Sequential quadratic programming. *International Journal of Computer Science and Information Security*, *14*, 650–669.

Gharaei,A., Pasandideh, S. H. R. (2017a). Four-Echelon integrated supply chain model with stochastic constraints under Shortage condition. *Industrial Engineering & Management Systems*, *16*(3), 316–329.

Gharaei, A., Pasandideh, S.H. R. (2017b). Modeling and optimization of four-level integrated supply chain with the aim of determining the optimum stockpile and period length: Sequential quadratic programming. *Journal of Industrial and Production Engineering*, *34*(7), 529–541.

Gharaei, A., Pasandideh, S. H. R., Akhavan Niaki, S. T. (2017). An optimal integrated lot-sizing policy of inventory in a bi-objective multi-level supply chain with stochastic constraints and imperfect products. *Journal of Industrial and Production Engineering*, *35*(1), 6–20.

Gharaei, A., Pasandideh, S. H. R., Arshadi Khamseh, A. (2017). Inventory model in a four-echelon integrated supply chain: Modeling and optimization. *Journal of Modelling in Management, 12*(4), 739–762.

Gharaei, A., Karimi, M., Hoseini Shekarabi, S. A. (2019). Joint economic lot-sizing in multi-product multi-level integrated supply chains: Generalized benders decomposition. *International Journal of Systems Science: Operations & Logistics*, 1-17.

Glock, Christoph H. (2012). The joint economic lot size problem: A review. *International Journal of Production Economics*, *135.2*: 671-686.

Glock, C. H., Kim, T. (2016). Safety measures in the joint economic lot size model with returnable transport items. *International Journal of Production Economics*, *181*, 24–33.

Goldberg, D. E., Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning*, *3*, 95-99.

Goyal, S.K. (1976). An integrated inventory model for a single supplier-single customer problem. *International Journal of Production Research*, *14*, 107–111.

Goyal, S.K. (1988). A joint economic-lot-size model for purchaser and vendor: a comment. *Decision Sciences*, *19*, 236–241.

Goyal, S.K., Nebebe, F. (2000). Determination of economic production-shipment policy for a single-vendor–single-buyer system. *European Journal of Operational Research, 121*, 175–178.

Gumus, M., Jewkes, E.M., Bookbinder,J.H., (2008). Impact of consignment inventory and vendor-managed inventory for a two-party supply chain. *International Journal of Production Economics*, *113*, 502–517.

Helmrich, M. J. R., Jans, R., van den Heuvel,W., Wagelmans, A. P. (2015). The economic lot-sizing problem with an emission capacity constraint. *European Journal of Operational Research*, *241*(1), 50–62.

Hill, R.M. (1997). The single-vendor single-buyer integrated production-inventory model with a generalized policy. *European Journal of Operational Research*, *97*, 493–499.

Hill, R.M., Omar, M. (2006). Another look at the single-vendor single-buyer integrated inventory production-inventory problem. *International Journal of Production Research*, *44*, 791–800.

Hoque, M.A., (2009). An alternative optimal solution technique for a single-vendor single-buyer integrated production inventory model. *International Journal of Production Research*, *47*, 4063–4076.

Holland, J. H. (1992). Genetic algorithms. *Scientific American*, *267*(1), 66-73.

Joglekar, P.N., Tharthare, S., (1990). The individually responsible and rational decision approach to economic lot sizes for one vendor and many purchasers. *Decision Sciences*, *21*, 492–506.

Karimi, B., Ghomi, S.M.T.F., Wilson, J.M., (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega*, *31*, 365–378.

Karimi-Nasab,M.,Modarres,M., Seyedhoseini, S.M. (2015). A self-adaptive PSO for joint lot-sizing and job shop scheduling with compressible process times. *Applied Soft Computing*, *27*, 137–147.

Katok, E., Lewis, H., Harrison, T., (1998). Lot sizing in general assembly systems with setup costs, setup times, and multiple constrained resources. *Management Science*, *44* (6), 859–877.

Kim, T., Glock, C. H. (2013). A multi-stage joint economic lot size model with lead time penalty costs. *Computers & Industrial Engineering*, *66*(1), 133-146.

Marchi, B., Ries, J. M., Zanoni, S., Glock, C. H. (2016). A joint economic lot size model with financial collaboration and uncertain investment opportunity. *International Journal of Production Economics*, *176*, 170–182.

Modak, N. M., Panda, S., Sana, S. S. (2016). Pricing policy and coordination for a distribution channel with manufacturer suggested retail price. *International Journal of Systems Science: Operations & Logistics*, *3*(2), 92–101.

Onal, M., Romeijn, H. E., Sapra, A., Van den Heuvel, W. (2015). The economic lot-sizing problem with perishable items and consumption order preference. *European Journal of Operational Research*, *244*(3), 881–891.

Onal, M. (2016). The two-level economic lot-sizing problem with perishable items. *Operations Research Letters*, *44*(3), 403–408.

Ou, J. (2017). Improved exact algorithms to economic lot-sizing with piecewise linear production costs. *European Journal of Operational Research*, *256*(3), 777–784.

Petering, Matthew EH, Xi Chen, and Wen-Huan Hsieh. (2019). Inventory Control with Flexible Demands: Cyclic Case with Multiple Batch Supply and Demand Processes. *International Journal of Production Economics*, *212*, 60-77.

Pineyro, P., Viera, O. (2010). The economic lot-sizing problem with remanufacturing and one way substitution. *International Journal of Production Economics*, *124*(2), 482–488.

Pineyro, P., Viera, O. (2014). Note on the economic lot sizing problem with remanufacturing and one-way substitution. *International Journal of Production Economics*, *156*, 167–168.

Pishchulov,G., Richter, K. (2016). Optimal contract design in the joint economic lot size problem with multi-dimensional asymmetric information. *European Journal of Operational Research*, *253*(3), 711–733.

Porter, Michael E. (1996). What is strategy? *Harvard business review*, *74*(6), 61-78.

Quadt, D., Kuhn, H. (2008). Capacitated lot-sizing with extensions: a review. *4OR, 6* (1), 61–83.

Sahling, F., Buschkühlb, L., Tempelmeierb, H., Helbera, S., 2009. Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. *Computers and Operations Research*, *36* (9), 2546–2553.

Rezaei, J., Davoodi,M. (2012). A joint pricing, lot-sizing, and supplier selection model. *International Journal of Production Research*, *50*(16), 4524–4542.

Roy, A., Sana, S. S., Chaudhuri, K. (2018). Optimal pricing of competing retailers under uncertain demand-a two-layer supply chain model. *Annals of Operations Research*, *260*(1–2), 481–500.

Salas Navarro, K., Chedid, J. A., Caruso, N. M., Sana, S.S. (2018). An inventory model of three-layer supply chain of wood and furniture industry in the Caribbean region of Colombia. *International Journal of Systems Science: Operations & Logistics*, *5*(1), 69–86.

Siajadi, H., Ibrahim, R. N., Lochert, P. B. (2006). Joint economic lot size in distribution system with multiple shipment policy. *International Journal of Production Economics*, *102*(2), 302-316.

Sajadieh, M.S., Jokar, M.R.A. (2009). Optimizing shipment, ordering and pricing policies in a two-stage supply chain with price-sensitive demand. *Transportation Research Part E*, *45*, 564–571.

Sarakhsi,M. K., Ghomi, S. F., Karimi, B. (2016). A new hybrid algorithm of scatter search and Nelder–Mead algorithms to optimize joint economic lot-sizing problem. *Journal of Computational and Applied Mathematics*, *292*, 387–401.

Sari, D. P., Rusdiansyah, A., Huang, L. (2012). Models of joint economic lot-sizing problem with time-based temporary price discounts. *International Journal of Production Economics*, *139*(1), 145–154.

Sargut, F. Z., Işık, G. (2017). Dynamic economic lot size model with perishable inventory and capacity constraints. *Applied Mathematical Modelling*, *48*, 806–820.

Siajadi, H., Ibrahim, R.N., Lochert, P.B., 2006. Joint economic lot size in distribution system with multiple shipment policy. *International Journal of Production Economics*, *102*, 302–316.

Sifaleras, A., Konstantaras, I., Mladenović, N. (2015). Variable neighborhood search for the economic lot-sizing problem with product returns and recovery. *International Journal of Production Economics*, *160*, 133–143.

Taş, D., Gendreau, M., Jabali, O., Jans, R. (2019). A capacitated lot sizing problem with stochastic setup times and overtime. *European Journal of Operational Research*, *273*(1), 146-159.

Telha, C., Van Vyve, M. (2016). Efficient approximation schemes for economic lot-sizing in continuous time. *Discrete Optimization*, *20*, 23–39.

Transchel, S., Minner, S. (2011). Economic lot-sizing and dynamic quantity competition. *International Journal of Production Economics*, *133*(1), 416–422.

Trigeiro, W., Thomas, L., McClain, J. (1989). Capacitated lot sizing with setup times. *Management Science*, *35* (3), 353–366.

Viswanathan, S. (1998). Optimal strategy for the integrated vendor–buyer inventory model. *European Journal of Operational Research*, *105*, 38–42.

Zavanella, L., Zanoni, S. (2009). A one-vendor multi-buyer integrated production- inventory model: the 'consignment stock' case. *International Journal of Production Economics*, *118*, 225–232.

CURRICULUM VITAE

Sepideh Alavi

Education

- PhD in Supply Chain and Operations Management, Lubar School of Business, University of Wisconsin-Milwaukee (2014-2020)
- PhD Minor in Management Information Systems, Lubar School of Business, University of Wisconsin-Milwaukee (2014-2016)
- M.Sc. in Industrial Engineering - Logistics and Supply Chain Engineering, Amirkabir University of Technology (2009- 2011)
- B.Sc. in Industrial Engineering - Systems Planning and Analysis, Alzahra University (2005-2009)

Publications

- Sepideh Alavi, Nader Azad, Mojtaba Heydar, Hamid Davoudpour "Integrating Production, Inventory and Location-Allocation Decisions in Designing Supply Chain Networks" *International Journal of Information Systems and Supply Chain Management*, 2016, 9 (4), pp 22-42
- Sepideh Alavi, Mostafa Pazoki, Kaveh Fahimi "A New Framework to Incorporate Competitive Considerations into Supply Chain Network Design" Proceedings of the 41st *International Conference on Computers & Industrial Engineering*, 2011, pp. 19-24

Conference Presentations

- Sepideh Alavi, Matthew Petering, Xiaohang Yue "Joint Supply and Order Fulfillment Planning for a Supply Chain with Flexible Demand" INFORMS Conference, Oct. 2019, Seattle.
- Sepideh Alavi, Matthew Petering, Xiaohang Yue "Decision Making in the Context of Flexible Demand" DSI Conference, Nov. 2019, New Orleans.
- Sepideh Alavi, Matthew Petering "An Order Consolidation and Inventory Control Model for a Two-Echelon Supply Chain" DSI Conference, Nov. 2018, Chicago.

- Sepideh Alavi, Matthew Petering "A Novel Mixed-Integer Quadratic Shipment Consolidation Model for a Two-Echelon Supply Chain" INFORMS Conference, Oct. 2017, Houston.

- Sepideh Alavi, Anthony D. Ross "A Data-Driven Predictive Model for Inventory Control of Products with Irregular Demand" INFORMS Annual Conference Oct. 2016, Nashville.

- Sepideh Alavi, Mariam Zahedi "Manufacturer-Salespersons Relationships in Global Markets: Inventory Policies and Cultural Effects" INFORMS Conference Oct. 2016, Nashville.

Teaching Experience

- Instructor
  - **Logistics and Transportation Management** (undergraduate), Lubar School of Business (Spring 2020)
    - Delivery method: **blended**

  - **Supply Chain Analytics** (undergraduate), Lubar School of Business (Fall 2018, Spring 2019)
    - Average Evaluation Score: **4.81/5**
    - Fall 2018 Delivery method: **blended**
    - Spring 2019 Delivery method: **face-to-face**

  - **Operations Planning and Control** (undergraduate), Lubar School of Business (Spring 2020, Fall 2019, Spring 2018, Fall 2017, Spring 2017 and Fall 2016)
    - Average Evaluation Score: **4.57/5**
    - Delivery method: **face-to-face**

  - **Purchasing and Supply Management** (undergraduate), Lubar School of Business (Summer 2018)
    - Evaluation Score: **4.82/5**
    - Delivery method: **face-to-face**

  - **Introduction to Supply Chain Management** (undergraduate), Lubar School of Business (Winterim 2017)
    - Evaluation Score: **4.2/5**
    - Delivery method: **face-to-face**

- Teaching Assistant
    - **Introduction to Management Statistics** (undergraduate), Lubar School of Business (Fall 2018, Summer 2017, Summer 2016)
        - Average Evaluation Score: **4.7/5**
        - Delivery method: **face-to-face**

    - **Predictive Analytics for Managers**, MBA and EMBA, Lubar School of Business (Spring 2018, Fall 2017)
        - Evaluations were not conducted for TAs for online courses.
        - Delivery method: **face-to-face discussion sessions, online lectures**

<u>Other Academic Experience</u>

- **AACSB Accreditation Project Assistant**, Lubar School of Business (June 2015- June 2017)

<u>Honors and Awards</u>

- Outstanding Doctoral Student Teaching Award ($250) (Feb. 2019)
- Roger L. Fitzsimonds Doctoral Scholarship ($5000) (May 2018)
- Sheldon B. Lubar Doctoral Scholarship ($5000) (May 2016, May 2017)
- PhD Chancellors' Award- Lubar School of Business - UWM (August 2014- August 2017)
- Ranked 2nd among 12 graduates of Industrial Engineering Master's program at Amirkabir University of Technology, 2011
- Ranked 66th among more than 10000 participants in National Matriculation exam for entering Master's program, 2009

<u>Computer Skills</u>

- Data Analytics: Minitab, R, WinBUGS, JMP, SAS
- Optimization: MATLAB, C++, GAMS
- Project Management: MS-Project, Primavera
- Simulation: Arena, ED

<u>PhD Coursework</u>

Statistical Analysis, Multivariate Methods, Applied Stochastic Processes, Advanced Bayesian Analytics, Advanced Operations Research Models, Doctoral Seminar: Applied Game Theory in Supply Chain Management, Doctoral Seminar: Logistics Management, Doctoral Seminar: Purchasing and Sourcing Theory, Process and Flow Management, Data and Information Management, Doctoral Seminar in Web-based Information Systems.

<u>Service and Affiliations</u>

Session Chair:

- DSI Annual Conference, Chicago (November 2018)

Journal Referee Activities:

- Journal of Computer and Industrial Engineering (CIE)

<u>Memberships</u>

- The Institute of Operations Research and Management Science (INFORMS)
- Decision Sciences Institute (DSI)
- Production and Operations Management Society (POMS)