



# CIRRELT

Centre interuniversitaire de recherche  
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre  
on Enterprise Networks, Logistics and Transportation

---

## Dynamic and Stochastic Inventory-Routing

Leandro C. Coelho  
Jean-François Cordeau  
Gilbert Laporte

July 2012

CIRRELT-2012-37

**Bureaux de Montréal :**

Université de Montréal  
C.P. 6128, succ. Centre-ville  
Montréal (Québec)  
Canada H3C 3J7  
Téléphone : 514 343-7575  
Télécopie : 514 343-7121

**Bureaux de Québec :**

Université Laval  
2325, de la Terrasse, bureau 2642  
Québec (Québec)  
Canada G1V 0A6  
Téléphone : 418 656-2073  
Télécopie : 418 656-2624

[www.cirrelt.ca](http://www.cirrelt.ca)

# Dynamic and Stochastic Inventory-Routing

Leandro C. Coelho<sup>1,2,\*</sup>, Jean-François Cordeau<sup>1,2</sup>, Gilbert Laporte<sup>1,3</sup>

<sup>1</sup> Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

<sup>2</sup> Department of Logistics and Operations Management, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

<sup>3</sup> Department of Management Sciences, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

**Abstract.** The combination of inventory management and vehicle routing decisions yields a difficult combinatorial optimization problem called the Inventory-Routing Problem (IRP). This problem arises when both types of decisions must be taken jointly, which is the case in vendor-managed inventory systems. The IRP has received significant attention in recent years. Several heuristic and exact algorithms are available for its static and deterministic versions. In the dynamic version of the IRP, customer demands are gradually revealed over time and planning must be made at the beginning of each of several periods. In this context, one can sometimes take advantage of stochastic information on demand through the use of forecasts, for example. We propose different policies to handle the dynamic and stochastic version of the IRP. We perform an extensive computational analysis on randomly generated instances in order to compare several solution policies. Amongst other conclusions we show that it is possible to take advantage of stochastic information to generate better solutions albeit at the expense of more computing time. We also show that the use of a longer rolling horizon step does not help improve solutions. Inventory holding costs have a positive correlation with solution cost. Our experiments also demonstrate that higher safety stocks lower the solution costs since customers are covered against demand variations and require fewer visits. Finally, we show that ensuring consistent solutions over time increases the cost of the solutions much more under a dynamic environment than in a static setting.

**Keywords.** Dynamic demand, stochastic demand, inventory-routing, rolling horizon algorithm, heuristic.

**Acknowledgements.** This work was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 227837-09 and 39682-10. This support is gratefully acknowledged. The authors thank the Réseau québécois de calcul de haute performance (RQCHP) for its assistance in providing computing facilities.

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

---

\* Corresponding author: Leandro.Coelho@cirrelt.ca

## 1 Introduction

In order to derive a competitive advantage, suppliers can sometimes reduce the overall costs of their operations by combining their routing, inventory and delivery decisions instead of optimizing them separately. These decisions can be centralized through the implementation of a vendor-managed inventory (VMI) strategy, which combines the replenishment and the distribution processes, leading to an overall reduction of logistics costs [33].

From an operational perspective, the VMI strategy is based on the solution of a difficult combinatorial optimization problem called the Inventory-Routing Problem (IRP), which integrates inventory management and vehicle routing decisions over several periods. The IRP has received increased attention in recent years. Several heuristics [9, 5, 14] as well as exact algorithms [3, 45, 13] have been proposed for the single vehicle version of the problem. The multi-vehicle case (MIRP) has also been solved heuristically [15] and exactly [1, 13]. In addition, an extended version of the MIRP incorporating several consistency features has been solved heuristically and exactly by Coelho et al. [15] and Coelho and Laporte [13], respectively. However, the studies described in these papers deal with a static and deterministic version of the problem in which all information is available when decisions are made. Literature reviews on the IRP can be found in Campbell et al. [12], Cordeau et al. [16] and Andersson et al. [2].

Dynamic problems are frequently encountered in practice. They reflect real-life situations in which one has to make decisions without full knowledge of future events. Examples of such problems arise in the context of the Dynamic Vehicle Routing Problem in which customer demands are gradually revealed over time [8, 46, 40]. In Dynamic and Stochastic Inventory-Routing Problems (DSIRP), customer demand is known in a probabilistic sense, thus yielding a dynamic and stochastic problem. In the IRP literature, dynamic problems have been studied by Kleywegt et al. [30, 31] who applied dynamic programming, and by Hvattum and Løkketangen [25] and Hvattum et al. [26] who used scenario trees and a progressive hedging algorithm. Recently, Bertazzi et al. [10] have formulated the stochastic IRP as a dynamic program and have solved it by means of a hybrid rolling horizon algorithm. This algorithm estimates unknown demands on the basis of their past average, and then solves a deterministic instance.

Solving a dynamic problem consists of proposing a *solution policy* as opposed to computing a static output [8]. A possible policy is to optimize a static instance whenever new information becomes available. The drawback of such a method is that it is often very time consuming to solve a large number of instances. A more common policy is to apply the static algorithm only once and then reoptimize the problem through a heuristic whenever new information is made available. A third policy, which can be combined with either of the first two, is to take advantage of the probabilistic knowledge of future information and make use of forecasts. In this paper we use forecasts in combination with the first policy. For more information on the solution of dynamic problems, see Psaraftis

[41], Ghiani et al. [21] and Berbeglia et al. [8].

The deterministic algorithms developed by Coelho et al. [14] allow the solution of DSIRPs within a rolling horizon framework, where one uses demand forecasts as an approximation of the future unknown demand. As noted by Özer [37], the use of past information can become an important aspect of the inventory management process provided it is properly used. Demand forecasts are typically needed for practical inventory control systems, the most common approach being the extrapolation of historical data based on the statistical analysis of time series [6].

Our aim is to describe and compare several solution policies for the DSIRP in which the objective is to minimize the total inventory, distribution and shortage costs. There are key differences between our approach and previous ones, in particular that of Bertazzi et al. [10]. One of these lies in the fact that we develop and compare several policies to solve the same problem, instead of only one. In particular, we are able to evaluate the performance of our method on inventory policies that are more general than the (hard constraint) assumption made in that study. Moreover, we propose a method that can make use of historical data in order to take into account future unknown demands, thus being able to efficiently solve instances in which the demand presents a trend or seasonalities, which was not previously the case. We also consider a dynamic environment in which some information arrives over time and is used in a rolling horizon framework. In addition, as in Coelho et al. [14], we allow the use of lateral transshipments between customers as a means to avoid stockouts when demand is high. Finally, we evaluate the impact of imposing some consistency features to the solutions of dynamic and stochastic instances of the IRP, thus extending the scope of the study of Coelho et al. [15]. In addition to proposing an efficient and flexible solution methodology for the DSIRP, one of our main scientific contributions is to evaluate the value of demand forecasts and transshipments.

The remainder of the paper is organized as follows. In Section 2 we formally define the DSIRP and we describe in Section 3 the strategies we have developed to solve it. Implementation details are provided in Section 4. This is followed by the results of extensive computational experiments in Section 5, and by conclusions in Section 6.

## 2 Problem description

We now formally introduce the DSIRP. The problem is defined on a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ , where  $\mathcal{V} = \{0, \dots, n\}$  is the vertex set and  $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$  is the arc set. Vertex 0 is a depot at which the supplier is located and the vertices of  $\mathcal{V}' = \mathcal{V} \setminus \{0\}$  represent customers. The problem is defined over an horizon of length  $p$  and at each time period  $t \in \mathcal{T} = \{1, \dots, p\}$  the demand  $d_i^t$  of customer  $i$  is a random variable  $D_i^t$ . In practice, the demand is not known by the decision maker who has to estimate it on the basis of historical data. We assume the decision maker can use any kind of forecast and input this information into the algorithmic framework we provide. The decision maker realizes the actual

values of  $d_i^t$  at the end of each period  $t$ . A unit inventory holding cost  $h_i$  is incurred by customer  $i$  and by the supplier at each period, and customer  $i$  has an inventory holding capacity  $C_i$ . We assume the supplier has enough inventory to meet all the demand during the planning horizon. If the demand of customer  $i$  is higher than its inventory level, it is then lost and a unit shortage penalty  $p_i$  is incurred. At the beginning of the planning horizon the decision maker knows the inventory level  $I_0^0$  and  $I_i^0$  of the supplier and customer  $i$ , respectively.

As is common in the IRP literature, we assume that a single vehicle of capacity  $Q$  is available [9, 3, 5, 9, 10, 14]. The vehicle is able to perform one route per time period, from the supplier to a subset of customers. A routing cost  $c_{ij}$  is associated with arc  $(i, j) \in \mathcal{A}$ . We also consider that the supplier uses an order-up-to inventory policy. This policy has been widely used in IRPs and related problems [9, 3, 5, 1, 14] and ensures that whenever a customer is visited, the quantity delivered is that needed to fill its inventory capacity. To ensure the feasibility of such a policy, given that there is only one capacitated vehicle available, we assume direct deliveries can take place from the supplier to any customer, by subcontracting to a carrier, to allow for planned deliveries to meet the OU requirements. In addition, after the demand is realized, if a customer faces a shortage it can arrange a lateral emergency transshipment from another customer if this is feasible. Both types of outsourced deliveries (direct deliveries and lateral emergency transshipments) are only made by direct shipping and the unit cost associated with direct deliveries or transshipments from  $i$  to  $j$  is  $\beta c_{ij}$ , where  $\beta > 0$ . As is standard in vehicle routing, travel costs are distance-dependent and are unrelated to the vehicle load. However, direct delivery and transshipment costs are distance- and volume-dependent because this is often how outsourced carriers define the terms of their contracts.

Regarding temporal issues, we consider that the decision maker first decides which customers to replenish in each period as well as the associated vehicle route and the direct shipments, if any. After demand is revealed, lateral transshipments may be arranged if any customer faces a shortage.

The variables and constraints of the model are as follows. Let  $I_i^t$  be the inventory level at customer  $i$  at the end of period  $t$ ,  $q_i^t$  the quantity delivered to customer  $i$  in period  $t$  using the supplier's vehicle,  $w_{ij}^t$  the quantity carried by the outsourced carrier from customer  $i$  to customer  $j$  in period  $t$ , and  $l_i^t$  the lost demand at customer  $i$  in period  $t$  due to insufficient inventory. The inventory level at the end of period  $t$  at customer  $i$  is then

$$I_i^t = I_i^{t-1} + q_i^t + \sum_{j \in \mathcal{V}} w_{ji}^t - \sum_{j \in \mathcal{V}'} w_{ij}^t - d_i^t + l_i^t \quad i \in \mathcal{V}' \quad t \in \mathcal{T}'. \quad (1)$$

The objective is to minimize the total inventory, shortage, routing and transshipment costs over the planning horizon, that is

$$\text{minimize } \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{V}} h_i I_i^t + \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{V}'} p_i l_i^t + \beta c_{ij} \sum_{t \in \mathcal{T}} \sum_{i, j \in \mathcal{V}'} w_{ij}^t + c_{r_t}, \quad (2)$$

where  $c_{r_t}$  represents the cost of the route performed in period  $t \in \mathcal{T}$ , which can

be obtained by solving a Traveling Salesman Problem over all the customers visited in period  $t$ .

### 3 Solution policies

The problem can be solved under a *proactive* policy or under a *reactive* policy, depending on whether demand forecasts are made or not. For each of these two policies emergency lateral transshipments can be allowed or not. This yields a total of four policies, which are all implemented in a rolling horizon fashion.

#### 3.1 Reactive policies

Under reactive policies, which are sometimes called “wait and see”, one observes the state of the system in order to make the next decision regarding routing and delivery. Formally, a reactive policy is defined as an  $(s, S)$  replenishment system under which whenever the inventory reaches the reorder point  $s$ , it triggers a replenishment order to bring the inventory position up to level  $S$ . The reorder point  $s$  should consider the delivery lead time and the stockout risk resulting from the stochasticity of the demand.

##### 3.1.1 Routing only

Under this policy, deliveries are performed by the supplier’s vehicle and no emergency lateral transshipment takes place when a customer runs out of inventory. Routing decisions are based solely on a customer-dependent threshold  $s_i$  and on its inventory level. If the inventory level at customer  $i$  is below  $s_i$  when the actual demand is realized at the end of period  $t$ , then customer  $i$  is selected to be served in period  $t + 1$ . The threshold can be updated after each period. The replenishment level  $S_i$  usually depends on ordering and holding costs and is set to bring the inventory level up to a target value. This inventory policy has been widely studied and used in other IRP studies [9, 10, 5, 15, 14]. As noted by [10] and [15], the OU policy is also relevant from a practical point of view and simplifies the decision making process while ensuring the stability and consistency of the replenishments. As in these studies, we also assume that the target level meets the customer inventory capacity. As mentioned, in order to ensure that this rule is always met and to avoid infeasibilities due to insufficient vehicle capacity, direct deliveries are allowed to take place from the depot. This ensures that all customers  $i$  whose inventory level is below the threshold  $s_i$  will have their inventories filled to their capacity in the next period.

##### 3.1.2 Routing and transshipment

This policy allows lateral transshipments between customers as an emergency measure against stockouts. The decision regarding whether or not to visit cus-

customer  $i$  is dependent on the threshold  $s_i$  as before. The inventory policy applied still follows an OU policy in which direct deliveries are allowed to take place from the supplier. After these decisions have been made, demand is revealed. If a customer runs out of inventory when its demand is realized, lateral transshipments can take place whenever they are possible and economically interesting. Lateral transshipments are allowed only as an emergency measure, i.e. they cannot be used to move inventory to a location having a lower holding cost. This policy is in line with the description of emergency transshipments provided by [36] and [38].

### 3.2 Proactive policies

A proactive policy not only observes the state of the system but also attempts to anticipate its future state by forecasting the demand and by using this information in the planning process.

#### 3.2.1 Routing only

This policy makes use of forecasts as a means of taking into account future demand but does not allow lateral transshipments. Once forecasts are obtained, the problem can be solved as a deterministic IRP. Direct deliveries from the supplier to the customers are allowed to ensure the feasibility of the OU policy. Under this policy, we first compute an  $f$ -period forecast for each of the customers, on the basis of their historical demands. A prediction interval that makes use of probabilistic information is computed for each customer. Forecasts are then used as a proxy for the unknown demands and initial inventory levels are set equal to the last known inventory level of each customer. The problem can then be solved heuristically as a deterministic IRP. The algorithm provides an  $f$ -period plan, of which only the first-period solution is implemented. Demands are then realized, new forecasts are computed and the process is reiterated.

#### 3.2.2 Routing and transshipment

As an extension of the previous policy, in this case lateral transshipments are allowed to take place after the demand is realized.

## 4 Algorithms

In this section we describe the four algorithms resulting from the solution policies described in Section 3.

## 4.1 Reactive policies

We first describe the two algorithms proposed to implement the reactive policies, with or without the use of lateral transshipments.

### 4.1.1 Routing only

The first decision made under this policy regards the level of the inventory at which the reorder point  $s_i$  of customer  $i$  is set. It is equal to an estimate of the expected demand during the lead time  $L$ , plus a safety stock dependent on demand variability, lead time and target service level. We denote the estimate of the expected demand  $\mu_i$  of customer  $i$  per period by  $\hat{\mu}_i$  and that of its standard deviation  $\sigma_i$  by  $\hat{\sigma}_i$ . These values as well as the resulting threshold can be updated at every period. Following classical inventory management practices [20], and assuming independent and normally distributed demands,  $s_i$  can be computed as

$$s_i = L\hat{\mu}_i + z_\alpha \sqrt{\hat{\sigma}_i^2 L}, \quad (3)$$

where  $\alpha$  is the probability of a stockout and  $z_\alpha$  is the  $\alpha$ -order quantile of the demand distribution. The quantity  $1 - \alpha$  is usually referred to as the *service level*.

The selection of customers to serve with the supplier's vehicles and through direct deliveries, as well as the quantities delivered by each option yields an NP-hard problem. However, since these decisions should be taken only once for every period, and given the size of the instances considered in this study, we have decided to solve this problem exactly by means of a mixed-integer linear program (MILP). The problem is defined as follows.

If the inventory level of customer  $i$  is below its threshold  $s_i$ , the total quantity that must be delivered is then the one needed to fill its capacity (i.e. an OU policy applies); otherwise no delivery is made. This quantity defines the parameter  $d'_i$ . We then solve the following MILP, called Routing-Direct (RD), in order to decide which customers are visited by the supplier's vehicle, which ones are visited through direct deliveries (and combinations of these two options), and the quantities delivered by each mode. When the routing cost matrix  $c_{ij}$  is symmetric, as is the case in our computational experiments, we work with an undirected formulation in order to reduce the number of variables. Thus, the routing variables  $x_{ij}$  ( $i < j$ ) are equal to the number of times edge  $(i, j)$  is traversed. We also introduce binary variables  $y_i$ , equal to one if and only if vertex  $i$  (the supplier or a customer) is visited by the supplier's vehicle. We denote by  $q_i$  the quantity delivered by the supplier's vehicle and by  $w_i$  the quantity delivered through direct deliveries to customer  $i$ . The problem can then be formulated as follows:

$$\text{(RD) minimize } \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}, i < j} c_{ij} x_{ij} + \beta \sum_{i \in \mathcal{V}} w_i c_{0i}, \quad (4)$$

subject to

$$q_i + w_i = d'_i \quad i \in \mathcal{V}' \quad (5)$$

$$\sum_{i \in \mathcal{V}'} q_i \leq Q \quad (6)$$

$$q_i \leq Q y_i \quad i \in \mathcal{V}' \quad (7)$$

$$\sum_{j \in \mathcal{V}, i < j} x_{ij} + \sum_{j \in \mathcal{V}, j < i} x_{ji} = 2y_i \quad i \in \mathcal{V} \quad (8)$$

$$\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}, i < j} x_{ij} \leq \sum_{i \in \mathcal{S}} y_i - y_m \quad \mathcal{S} \subseteq \mathcal{V}', \text{ for some } m \in \mathcal{S} \quad (9)$$

$$q_i, w_i \geq 0 \quad i \in \mathcal{V}' \quad (10)$$

$$x_{i0} \in \{0, 1, 2\} \quad i \in \mathcal{V}' \quad (11)$$

$$x_{ij} \in \{0, 1\} \quad i, j \in \mathcal{V}' \quad (12)$$

$$y_i \in \{0, 1\} \quad i \in \mathcal{V}. \quad (13)$$

The objective function (4) defines the minimization of routing and direct delivery costs. Constraints (5) state that the total delivered quantity  $d'_i$  is equal to the quantity  $q_i$  delivered by the supplier's vehicle, plus the quantity  $w_i$  supplied by means of a direct delivery. Constraints (6) ensure that the vehicle capacity is not exceeded, while constraints (7) guarantee that only customers assigned a visit can have quantities delivered by the supplier vehicle. Constraints (8) and (9) are degree constraints and subtour elimination constraints, respectively. Constraints (10)–(13) enforce the integrality and non-negativity conditions on the variables.

The RD model can be simplified by preprocessing all customers with zero  $d'_i$  and removing the corresponding variables.

#### 4.1.2 Routing and transshipment

The implementation of this policy is like the previous one except that after the solution has been computed, demands are revealed and lateral transshipments are allowed to take place. These are computed by means of the following min-cost network flow problem. This model, called Transshipment Origins-Destinations (TOD), optimizes the quantities as well as origins and destinations of the lateral transshipments. Note that in this model the parameter  $I_i^0$  represents the initial inventory of customer  $i$  at the beginning of each time slice of the rolling horizon, unlike the initial inventory of the instance being solved as it was defined in Section 2. It is solved once per period, after demands are realized. The problem is defined as follows:

$$\text{(TOD) minimize } \beta \sum_{i \in \mathcal{V}'} \sum_{j \in \mathcal{V}'} c_{ij} w_{ij} + \sum_{i \in \mathcal{V}'} p_i l_i + \sum_{i \in \mathcal{V}'} I_i h_i \quad (14)$$

subject to

$$I_i = I_i^0 + \sum_{j \in \mathcal{V}'} w_{ji} - \sum_{j \in \mathcal{V}'} w_{ij} + l_i \quad i \in \mathcal{V}' \quad (15)$$

$$0 \leq I_i \leq C_i \quad i \in \mathcal{V}' \quad (16)$$

$$0 \leq l_i \leq -\min\{0, I_i^0\} \quad i \in \mathcal{V}' \quad (17)$$

$$0 \leq w_{ij} \leq \min\{\max\{0, I_i^0\}, -\min\{0, I_j^0\}\} \quad i, j \in \mathcal{V}'. \quad (18)$$

The objective function (14) minimizes the total lateral transshipment, lost demand and inventory costs. Constraints (15) ensure flow conservation by stating that the final inventory of customer  $i$  is the sum of its initial inventory, plus all quantities transshipped to  $i$ , minus all quantities transshipped from  $i$  to other customers, plus the lost demand. Constraints (16) set bounds on the final inventory. Constraints (17) define bounds on the lost demand of customer  $i$ : if its initial inventory is non-negative, then no demand can be lost, and both bounds are zero; otherwise, a minimum of zero and a maximum of  $I_i^0$  units can be lost. Likewise, constraints (18) impose bounds on the flows of transshipment arcs. There are four possible combinations of inventory levels for  $i$  and  $j$ , all of which can be handled by these constraints:

1.  $I_i^0 \geq 0$  and  $I_j^0 \geq 0$ : the inner  $\min\{0, I_j^0\}$  is zero, setting the right-hand side of the constraint to zero. No transshipment should occur only to relocate inventory, since  $j$  does not need an emergency transshipment;
2.  $I_i^0 \geq 0$  and  $I_j^0 < 0$ : the inner  $\min\{0, I_j^0\}$  is  $I_j^0$  since this quantity is negative; the outer  $\min\{I_i^0, I_j^0\}$  is then the minimum between the availability  $I_i^0$  and the requirement  $-I_j^0$ . This is then the upper bound on the arc of the emergency transshipment from  $i$  to  $j$ ;
3.  $I_i^0 < 0$  and  $I_j^0 \geq 0$ : both inner functions return zero; the upper bound is then also zero, since  $j$  does not need an emergency transshipment and  $i$  does not have a surplus;
4.  $I_i^0 < 0$  and  $I_j^0 < 0$ : the max function returns zero and the inner min function returns  $-I_j^0$ ; the outer function then becomes  $\min\{0, I_j^0\}$  which returns zero as the upper bound flow on the arc flow, since  $i$  does not have enough inventory to supply to  $j$ .

We depict in Figure 1 a simple example of this network flow problem. The flow on the small dashed arcs equals the initial inventory level at customer  $i$ . Note that this number represents the surplus available at vertex  $i$ . If  $I_i^0$  is negative, it will enable customer  $i$  to have a lost demand. Then the flow over the large dashed arcs lies in the interval  $[0, -\min\{0, I_i^0\}]$  and represents the lost demand of customer  $i$ . Note that if  $I_i^0$  is positive, the flow on this arc is set to zero; if  $I_i^0$  is negative, it represents the lost demand and lies between zero and  $-I_i^0$ , i.e. this is the case in which all the excess demand is lost. The costs of these arcs are equal to  $p_i$ . The solid arcs represent the inventory carried at

customer  $i$  at the end of the period. The flows on these arcs are bounded by  $[0, C_i]$  and their associated costs are  $h_i$ . Finally, the dotted arcs in the middle represent transshipments. They are defined between any pair of vertices  $(i, j)$ , in both directions, and their cost is  $\beta c_{ij}$ . The flows on these arcs lie in the interval  $[0, \min\{\max\{0, I_i^0\}, -\min\{0, I_j^0\}\}]$ .

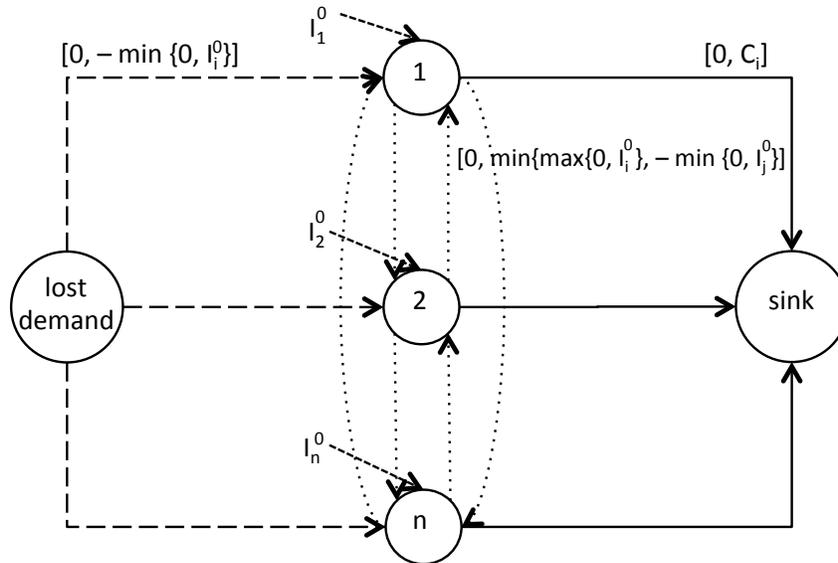


Figure 1: Example of the network flow problem solved to decide of transshipment quantities, origins and destinations.

## 4.2 Proactive policies

We now describe the two algorithms used to implement the proactive policies.

### 4.2.1 Routing only

This policy makes use of forecasts on future demand to help make current decisions. The first decision relates to the choice of a forecasting method. There exist several methods for forecasting future demand based on time series analysis. For an overview, see [34]. In this paper we apply the exponential smoothing technique which assigns exponentially smaller weights to past observations. This is a simple yet powerful method capable of identifying changes in the mean, trend or seasonalities in time series. It provides a point forecast, i.e. a single value representing the expected future demand, or a prediction interval, i.e. a point forecast and an estimated variance (see Hyndman et al. [28]).

The second decision regards the length  $f$  of the forecasting and rolling horizon. A compromise must be made between a short horizon which yields faster computations but lower solution quality, and a longer horizon which considers more information but requires more extensive computations. In Section 5.3.5 we examine the impact of  $f$  on the solution process.

Finally, the third decision is how to incorporate future demand forecasts in an IRP heuristic. We have adapted the work of [14] which uses an adaptive large neighborhood search (ALNS) matheuristic and provides very good results on benchmark static instances. This heuristic is described in Section 4.3 and can handle both the OU policy or the more general maximum level (ML) inventory policy, which does not force the deliveries to fill the customer capacity. Once forecasts are available, the dynamic problem reduces to a static one.

#### 4.2.2 Routing and transshipments

This policy works much like the previous one, except that after vehicle routes are created for all periods of the rolling horizon and the first of them is implemented, demands are revealed and lateral transshipments are allowed as an emergency measure against shortages. The way these transshipments are computed follows the same min-cost network flow problem, as in Section 4.1.2.

### 4.3 ALNS matheuristic

The algorithm proposed by [14] is an implementation of the ALNS algorithm originally proposed by [43] for the Vehicle Routing Problem and already successfully applied to a number of other contexts [39, 7, 24, 32]. In this implementation, some subproblems are solved exactly as min-cost network flow problems. It can therefore be described as a *matheuristic* [35], i.e. as a hybridization of a heuristic and of a mathematical programming algorithm. This algorithm is highly suitable for the problem at hand because of its generality and flexibility. It provides a highly diversified search through the multiplicity of its operators and through the use of a random mechanism for their selection. This implementation uses a subset of the operators used in [14] and runs for fewer iterations in order to make it faster. Because of the dynamic nature of our problem, we must indeed be able to run it several times for a single instance. The impact of this implementation choice is analyzed in Section ??.

In summary, the algorithm of [14] creates different vehicles routes at each ALNS iteration by removing and reinserting customers into vehicle routes. This is done by selecting one of several simple operators to explore different neighborhoods of the incumbent solution. Such operators include random insertions or removals, best insertions or removals, cluster insertions or removals, emptying routes, swapping routes and moving customers assignments. After vehicle routes have been created, the remaining problem is that of determining delivery quantities and transshipment origins, destinations and quantities, while minimizing the total inventory-distribution cost. This problem is solvable efficiently

and exactly using a min-cost network flow algorithm and can easily handle both the ML and OU policies. This approach was shown in Coelho et al. [14] to generate IRP solutions with value lying within 0.50% of optimality.

Each operator  $i$  is assigned a weight  $\omega_i$  whose value depends on its past performance and on its score. Given  $h$  operators with weights  $\omega_i$ , operator  $j$  will be selected with probability  $\omega_j / \sum_{i=1}^h \omega_i$ . Initially, all weights are equal to one and all scores are equal to zero. Operators are rewarded according to their past performance: they receive a high reward  $\sigma_1$  if they yield a new best solution, a medium reward  $\sigma_2$  if their solution is better than the incumbent one, or a low reward  $\sigma_3$  if the solution they provide is worse but still accepted. Initially, all operators have the same probability of being selected. After  $\varphi$  iterations, scores are computed taking into account the rewards accumulated as follows. Let  $\pi_i$  and  $o_{ij}$  be, respectively, the score of operator  $i$  and the number of times it has been used in the last segment  $j$ . The updated weights are then

$$\omega_i := \begin{cases} \omega_i & \text{if } o_{ij} = 0 \\ (1 - \eta)\omega_i + \eta\pi_i/o_{ij} & \text{if } o_{ij} \neq 0, \end{cases} \quad (19)$$

where  $\eta \in [0, 1]$  is called the reaction factor, controlling how quickly the weight adjustment reacts to changes in the movement performance. All scores are reset to zero.

New solutions are accepted or rejected according to a simulated annealing criterion: given a solution  $s$ , a neighbor solution  $s'$  is accepted if  $z(s') < z(s)$ , and with probability  $e^{-(z(s')-z(s))/\tau}$  otherwise, where  $z(s)$  is the solution cost and  $\tau > 0$  is the current temperature. The temperature is initialized at  $\tau_{start}$  and is decreased by a cooling rate factor  $\phi$  at each iteration, where  $0 < \phi < 1$ .

We have used the following destroy and repair operators. In what follows, all insertions are performed following the cheapest insertion rule and  $\rho$  is an integer randomly drawn from the interval  $[1, n]$  using a semi-triangular distribution with a negative slope.

- Destroy operators
  - **Randomly remove  $\rho$** : This operator randomly selects one period and removes one randomly selected customer from it. It is repeated  $\rho$  times.
  - **Shaw removal**: Following the ideas developed by [43] and [44], this operator removes customers that are relatively close to each other. Specifically, it randomly selects one period and one customer served in this period, it computes the distance  $dist_{min}$  to the closest customer also being served by the same route, and it removes all customers within  $2dist_{min}$  units from the selected route.
  - **Empty one period**: This operator selects one random period and removes all customers assigned to it.

- **Remove one customer:** This operator randomly selects one customer and removes all its assignments to any periods.
- Repair operators
  - **Randomly insert  $\rho$ :** This operator randomly inserts  $\rho$  customers into the current solution. Specifically, it selects one random customer and one random period, and inserts it into the route in that period if it is not already present. This operator is applied  $\rho$  times.
  - **Shaw insertions:** This operator is similar to the Shaw removal operator in the sense that it selects similar customers to be inserted together. It selects one period and one customer not served in that period. The operator then computes  $dist_{min}$  and all customers within a  $2dist_{min}$  distance are inserted in the same route.
  - **Swap  $\rho$  customers:** This operator selects two customers from two different periods and swaps their assignments. It is also applied  $\rho$  times.
  - **Insert one customer several periods:** This operator selects one customer and randomly assigns it to several periods of the planning horizon.

The operators just described generate the selection of visited customers as well as their sequence in the vehicle route. The remaining problem is that of determining delivery quantities and transshipment origins, destinations and quantities, which can be solved very efficiently by means of a min-cost network flow algorithm.

Given that the ALNS algorithm is invoked several times in a rolling horizon fashion, it had to be tuned to be extremely streamlined and fast. This drove us to the following settings for the operators and parameters after a tuning phase. The starting temperature  $\tau_{start}$  is set to 20,000 and the cooling rate  $\phi$  is 0.9993. The stopping criterion is satisfied when the temperature reaches 0.01, that is, when approximately 20,000 iterations have been performed. In our implementation, the segment length  $\varphi$  was set to 200 iterations and the reaction factor  $\eta$  was set to 0.7, that is, new weights will be composed by 70% of the performance on the last segment and 30% by the last weight value. Scores are updated with  $\sigma_1 = 10$ ,  $\sigma_2 = 5$  and  $\sigma_3 = 2$ .

At the end of each segment we also perform a 2-opt periodic postoptimization. Algorithm 1 presents a simplified pseudocode for this heuristic. For algorithmic and implementation details, the reader is referred to [14].

## 5 Computational experiments

In this section we provide some implementation specifications, we describe the generation procedure for the test instances and we present results of extensive computational experiments. These are described in Section 5.3.1 for the base case and in Sections 5.3.2 to 5.3.7 for several alternative configurations.

---

**Algorithm 1** ALNS heuristic - simplified pseudocode
 

---

```

1: Initialize: set all weights equal to 1 and all scores equal to 0.
2:  $s_{best} \leftarrow s \leftarrow$  initial solution.
3:  $\tau \leftarrow \tau_{start}$ .
4: while  $\tau > 0.01$  do
5:    $s' \leftarrow s$ .
6:   Select a destroy and a repair operator and apply them to  $s'$ .
7:   Fix routing decisions, solve the remaining network flow problem.
8:   if  $z(s') < z(s)$  then
9:      $s \leftarrow s'$ ;
10:    if  $z(s) < z(s_{best})$  then
11:       $s_{best} \leftarrow s$ ;
12:    else if  $s'$  is accepted by the simulated annealing criterion then
13:       $s \leftarrow s'$ ;
14:    end if
15:  end if
16:  if the iteration count is a multiple of  $\varphi$  then
17:    update the weights and reset the scores of all operators.
18:    perform an intra-route 2-opt.
19:  end if
20:   $\tau \leftarrow \phi\tau$ ;
21: end while
22: return  $s_{best}$ ;

```

---

## 5.1 Implementation specifications

All computations were performed on a grid with 630 nodes available and running the Scientific Linux 6.1 operating system. Each vertex is equipped with two Intel Westmere-EP X5650 hexa-core processors running at 2.67 GHz and with 24 GB or 48 GB of RAM memory.

Our algorithm was coded in C++ and makes use of only one processor. The min-cost network flow problem was implemented using the *LEMON* graph template library [18] running the network simplex algorithm for its internal computations. Forecasts were carried out using the *forecast* package [27, 29] available for R Language and Environment for Statistical Computing [42] and embeded within our C++ code using the *RInside* classes [19]. We allowed the software to run in its default settings, searching through all the 30 variants of the exponential smoothing models described in [28]. We made use of the 50 past periods immediately before the current period as historical data for the chosen forecasting method.

Given that the lead time is equal to one (all deliveries are performed in the next period) and its standard deviation is zero, the value of  $s_i$  used in equation (3) is then

$$s_i = \hat{\mu}_i + z_\alpha \hat{\sigma}_i. \quad (20)$$

Using the last known demand as an expectation of future demand is equivalent to a naïve forecast method in which the next period forecast is equal to the last known value, this being the simplest adaptive forecasting method [22].

## 5.2 Instance generation

We have generated instances following some of the standards used for the instances generated for the IRP by [3, 4], namely the mean customer demand, initial inventories, vehicle capacity and geographical location of the vertices are the same as in their tests. Instances were generated with 50 past periods of demand information before the future  $p$  periods such that it can be used as historical data. Our set is generated according to the following data:

- number of customers  $n$ :  $5k$  where  $k = 1, 2, 3, 5, 10, 15, 20, 25, 30, 40$ ;
- horizon  $p$ : equal to 5, 10 or 20 periods;
- demand distributions: mean demand  $\mu_i$  is generated as an integer random number following a discrete uniform distribution in the interval [10, 100], and standard deviation  $\sigma_i$  as an integer random number following a discrete uniform distribution in the interval [2, 10]. The demands are generated following a normal distribution with these parameters. If a negative demand value is generated, it is substituted by zero;
- product availability at the supplier: mean production  $\bar{r}$  is generated as an integer random number following a discrete uniform distribution in the

interval  $[100n, 140n]$ , and  $\sigma_0$  as an integer random number following a discrete uniform distribution in the interval  $[2, 10]$ . The production is generated following a normal distribution with these parameters. They are used only to account for inventory costs at the supplier, as in Archetti et al. [3];

- maximum inventory level  $C_i$ :  $\mu_i g_i$ , where  $g_i$  is randomly selected from the set  $\{2, 3, 4\}$ ;
- starting inventory level  $I_0^0$ :  $\sum_{i \in \mathcal{V}'} C_i$ ;
- starting inventory level  $I_i^0$ :  $C_i - \mu_i$ ;
- inventory holding cost  $h_0$ : 0.01;
- inventory holding cost  $h_i$  ( $i > 0$ ): randomly generated from a continuous uniform distribution in the interval  $[0.02, 0.10]$ ;
- shortage penalty:  $p_i = 200h_i$ ;
- vehicle capacity  $Q$ :  $\frac{3}{2} \sum_{i \in \mathcal{V}'} \mu_i$ ;
- distance/cost  $c_{ij}$ :  $\lfloor \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} + 0.5 \rfloor$ , where the points  $(X_i, Y_i)$  are the coordinates of vertex  $i$  and are obtained randomly from a discrete uniform distribution in the interval  $[0, 500]$ .

This set of instances will be called the *stationary* data set since the mean of the demand distribution is stationary. We have also generated other sets of instances in order to evaluate the dynamics of real-life demand, which often presents some seasonality. Indeed, [11] show that the key challenges faced in practical supply chains are related to, among others, non-stationary demand and inventory imbalances. To this end, we have generated the following two extra sets of instances, called *seasonal* and *correlated*.

In the *seasonal* data set each customer presents an independent seasonal pattern every five periods. This simulates the weekly variations of orders that are likely to occur. In its lower state, the demand is allowed to be as low as 40% of the usual demand, and as high as 200% at its peak. Seasonalities are independent so that, on average, they should cancel each other and the supplier should not face an overall high or low demand on any given day. In the *correlated* data set, on the other hand, all customers present the same seasonality pattern, that is, all have their lower and higher demands in the same period. This way the supplier faces a bottleneck of its vehicle capacity when the demand is high and has spare capacity when the demand is low. All else is kept unchanged from the *standard* stationary data set. We should mention that computing the reorder point with equation (20) assumes that demands of consecutive periods are independent, which is no longer the case in the presence of seasonality. However, we believe that computing the reorder point in this approximate way does not have a major impact on the results.

For each of the three data sets, and each of the 30 combinations of  $n$  and  $p$ , we have generated five instances, yielding 150 instances in each set, for a total of 450 instances. Their nomenclature follows the rule *dirp-n-p-1* through *dirp-n-p-5*. In Section 5.3 we provide summaries aggregating instances by their size: those with less than 50 customers are labeled *small*, those containing between 50 and 100 customers are called *medium*, and those with more than 100 customers are called *large* instances. These sets of instances as well as the solutions presented in the next sections are available at the URL <http://www.leandro-coelho.com/instances/>.

### 5.3 Computational results

We now report the results of our extensive computational experiments. The OU policy is first used to allow fair comparisons; the ML policy will be analyzed later. The transshipment cost  $\beta$  was set to 0.01 as in [14] and 95% prediction intervals were used, as in [23]. We first present results for the base case, and later we provide analyzes for a number of variations of the problem and of the algorithm.

#### 5.3.1 Results for the base case

We first provide results for the *base case* defined with the standard data set in Table 1 for the cases without and with lateral transshipments. For each method we present the solution cost, the average running time and the average lost demand per customer per period. Some conclusions can be drawn from Table 1. First, regarding the use of forecasts one can see that, as expected, the value added by forecasting a stationary time series is no better than using the naïve method employed by the reactive policy [17]. Nevertheless, on average the solution cost is lower and there is significantly less lost demand. More interestingly, allowing transshipments has a twofold effect: first this helps satisfy the demand by decreasing the average lost demand under both the reactive and proactive policies; second, by decreasing the lost demand, it also lowers the average solution cost. Finally, the computational cost of forecasting and solving the ALNS heuristic for many customers and several periods is not negligible.

In Table 2 we provide results for the base case defined with the seasonal data set. Our first observation is that the average running time is higher than in the standard data set. This reflects the difficulty of solving these instances. The value of lateral transshipments is corroborated, as in the standard case: allowing transshipments reduces the average lost demand per customer per period while significantly decreasing the solution cost. Finally, comparing policies in Table 2 shows that a more streamlined policy helps prevent stockouts. However, in both cases the average cost of the proactive policy is slightly higher than under the reactive policy.

Finally, we provide the results for the base case defined with the correlated data set in Table 3. When demands are correlated and peaks occur simul-

Table 1: Summary of computational results for the Dynamic and Stochastic Inventory-Routing Problem on the standard data set

Transshipment	Instance size	Reactive policy			Proactive policy		
		Solution	Time (s)	Avg. lost	Solution	Time (s)	Avg. lost
No	small ( $n < 50$ )	14974.17	0.0	0.62	14224.84	47.2	0.10
	medium ( $50 \leq n \leq 100$ )	39546.01	4.3	0.41	32774.85	453.3	0.00
	large ( $n > 100$ )	64854.75	408.5	0.46	64784.85	3781.0	0.00
Average		39791.64	137.6	0.50	37261.51	1427.2	0.03
Yes	small ( $n < 50$ )	14382.67	0.0	0.00	8586.53	46.9	0.05
	medium ( $50 \leq n \leq 100$ )	37720.58	4.4	0.00	27743.95	452.7	0.00
	large ( $n > 100$ )	61455.93	498.4	0.00	56506.38	3934.4	0.00
Average		37853.06	167.6	0.00	30945.62	1478.0	0.02

Table 2: Summary of computational results for the Dynamic and Stochastic Inventory-Routing Problem on the seasonal data set

Transshipment	Instance size	Reactive policy			Proactive policy		
		Solution	Time (s)	Avg. lost	Solution	Time (s)	Avg. lost
No	small ( $n < 50$ )	15994.92	0.1	0.41	14510.22	48.4	0.00
	medium ( $50 \leq n \leq 100$ )	40953.04	5.5	0.38	41071.74	499.7	0.00
	large ( $n > 100$ )	70442.24	758.3	0.41	73252.94	4734.7	0.00
Average		42463.40	254.6	0.40	42944.97	1760.9	0.00
Yes	small ( $n < 50$ )	15515.02	0.1	0.00	14160.04	48.1	0.00
	medium ( $50 \leq n \leq 100$ )	39164.04	5.8	0.00	40433.81	501.1	0.00
	large ( $n > 100$ )	66093.51	751.5	0.00	68918.08	4739.1	0.00
Average		40257.52	252.5	0.00	41170.64	1762.8	0.00

taneously, emergency transshipments are still a powerful tool to mitigate lost demand, relocating inventory and making the system more robust, yet decreasing the average solution values. The use of forecasts helps reduce routing costs and stockouts.

Table 3: Summary of computational results for the Dynamic and Stochastic Inventory-Routing Problem on the correlated data set

Transshipment	Instance size	Reactive policy			Proactive policy		
		Solution	Time (s)	Avg. lost	Solution	Time (s)	Avg. lost
No	small ( $n < 50$ )	15546.15	0.1	0.43	16466.03	48.3	0.00
	medium ( $50 \leq n \leq 100$ )	42940.79	14.4	0.48	40867.58	503.8	0.00
	large ( $n > 100$ )	75067.20	1506.5	0.47	71152.13	4727.4	0.00
Average		44518.05	507.0	0.46	42828.58	1759.8	0.00
Yes	small ( $n < 50$ )	15132.41	0.2	0.00	14822.28	48.2	0.00
	medium ( $50 \leq n \leq 100$ )	40526.86	15.4	0.00	40224.01	502.9	0.00
	large ( $n > 100$ )	70536.52	1749.1	0.00	68844.68	4713.8	0.00
Average		42065.26	588.2	0.00	41296.99	1755.0	0.00

Tables 1–3 show the solution values produced by the proactive policies are sometimes worse than those generated by the reactive policies, especially on large instances. A possible explanation is that the algorithm developed for the reactive policies solves the routing problem exactly, whereas the one proposed for the proactive policies relies on the ALNS matheuristic to sequence the customers. Even though this heuristic has been shown in earlier studies to provide good solutions [15, 14], this time the number of customers is much larger. In particular, the *large* instances push the algorithm to its limit, and the ALNS implementation is streamlined to be executed several times in a rolling horizon fashion, which could explain the decrease in the solution quality. We further analyze the effect of running the ALNS algorithm in Section 5.3.2.

In addition to the analyses presented so far, we have investigated a number of other scenarios using the best of the proposed policies, i.e. the one described in Section 3.2.2.

### 5.3.2 Increasing the number of ALNS iterations

We first analyze the quality of the solutions obtained for the problem solved at each period when the ALNS heuristic is allowed to perform twice the original number of iterations, thus also roughly doubling the execution time. We now allow the ALNS to iterate 40,000 times. Average solutions for the proactive policy without and with transshipments are shown in Table 4. We see that allowing more computing time improves the average solution cost. For the case without transshipments, improvements are on average larger than 5%. This shows that these policies perform well if the algorithm used to solve the problem at each period is able to identify high quality solutions.

Table 4: Summary of solutions when applying the OU inventory policy for the Dynamic and Stochastic Inventory-Routing Problem on the standard data set with longer ALNS iterations

Instance	Without transshipments				With transshipments			
	Solution	Time (s)	Avg. lost	Increase (%)	Solution	Time (s)	Avg. lost	Increase (%)
small ( $n < 50$ )	9131.45	67.1	0.10	-7.20	8355.69	67.4	0.05	-2.16
medium ( $50 \leq n \leq 100$ )	30137.81	888.3	0.00	-4.39	26891.26	880.5	0.00	-4.23
large ( $n > 100$ )	60051.36	9248.9	0.00	-3.76	55530.30	9196.0	0.00	-2.08
Average	33106.87	3401.4	0.03	-5.12	30259.08	3381.3	0.02	-2.82

### 5.3.3 Applying an ML inventory policy

We have also implemented an ML inventory policy which relaxes the OU rule. Under this policy, the ALNS heuristic optimizes the quantities delivered while respecting the vehicle and the customer capacities. A summary of results on the standard data set is provided in Table 5. Specifically, we compute the average cost savings with respect to the OU policy when such a policy is applied, as well as the average lost demand (per customer per period) both without and with transshipments. Applying the ML policy yields reductions in solution costs and in lost demands.

Table 5: Summary of cost savings when applying the ML inventory policy for the Dynamic and Stochastic Inventory-Routing Problem on the standard data set

Instance	Without transshipments				With transshipments			
	Solution	Time (s)	Avg. lost	% increase over OU	Solution	Time (s)	Avg. lost	% increase over OU
small ( $n < 50$ )	10225.93	46.3	0.24	-0.78	7926.71	44.6	0.16	-9.96
medium ( $50 \leq n \leq 100$ )	30360.66	452.7	0.01	-1.50	26527.05	444.1	0.01	-3.78
large ( $n > 100$ )	61250.17	3860.1	0.01	-0.49	54292.38	4100.1	0.00	-4.19
Average	33945.59	1453.0	0.09	-0.92	29582.05	1529.6	0.06	-5.97

### 5.3.4 Varying the inventory holding costs

The inventory holding cost parameters play an important role in changing the balance between making more frequent deliveries or holding higher average inventories. To this end, we have analyzed two different scenarios: one in which inventory holding costs are doubled, and another in which they are halved. We present in Table 6 the results of these experiments. For all situations tested the variations occurred as expected, exhibiting a positive correlation between the inventory holding cost and the solution cost. Moreover, multiplying or dividing the inventory cost by two does not change the conclusion that the proactive policy still performs better than the reactive one.

Table 6: Summary of cost savings when varying the inventory holding costs for the Dynamic and Stochastic Inventory-Routing Problem on the standard data set

Inventory cost	Instance	Reactive policy				Proactive policy			
		Solution	Time (s)	Avg. lost	% increase	Solution	Time (s)	Avg. lost	% increase
Halved	small ( $n < 50$ )	14036.64	0.1	0.00	-2.52	8011.02	47.3	0.08	-8.29
	medium ( $50 \leq n \leq 100$ )	39703.27	18.9	0.00	-4.86	30366.73	721.7	0.00	-6.72
Average		26869.95	9.5	0.00	-3.69	19188.87	384.5	0.04	-7.51
Doubled	small ( $n < 50$ )	15074.69	0.1	0.00	4.66	9291.39	47.4	0.05	8.26
	medium ( $50 \leq n \leq 100$ )	45346.88	19.3	0.00	8.45	36087.193	707.3	0.00	10.62
Average		30210.79	9.7	0.00	6.55	22689.29	377.4	0.02	9.44

### 5.3.5 Increasing the length $f$ of the planning horizon

We now evaluate the impact on the final solution cost of using a larger planning horizon. To this end, we have doubled to six the length  $f$  of the horizon used in the forecasts and in the ALNS, and we have solved a subset of instances from the standard data set. The fact that the ALNS matheuristic has to make twice as many decisions should be taken into account. In other words, keeping the number of ALNS iterations fixed, solution quality degradation is most likely to occur when doubling the length of the horizon. As a result, it makes sense to apply the idea used in Section 5.3.2 which consists of running the ALNS over a longer number of iterations. The average cost increases (or savings, when negative) are shown in Table 7. As expected, solution quality deteriorates with a longer horizon and computation times approximately double. Horizons of less than three periods make little sense since the main advantage of the proactive policy is to plan ahead and avoid visits to the same geographical area over consecutive periods, which is unlikely when  $f$  is very small.

Table 7: Summary of the impact on cost when time slices  $f$  are doubled (to six periods) under an OU inventory policy for the Dynamic and Stochastic Inventory-Routing Problem on the standard data set

Instance	Without transshipments				With transshipments			
	Solution	Time (s)	Avg. lost	% increase over OU	Solution	Time (s)	Avg. lost	% increase over OU
small ( $n < 50$ )	15553.77	62.8	0.04	0.25	10322.57	63.1	0.02	0.09
medium ( $50 \leq n \leq 100$ )	45963.58	1337.8	0.00	0.22	38864.04	1341.1	0.00	0.16
Average	30758.67	700.3	0.02	0.24	24593.31	702.1	0.01	0.12

### 5.3.6 Varying the service level

The percentage of the unknown demand covered against stockouts also plays an important role in the decision making process. We have varied the service

level parameter, which directly affects the safety stock level of the proactive policy. We have run the algorithm on a subset of instances with a service level equal to 90% and to 99%, and we have summarized the results in Table 8. As the table shows, a lower service level means that customers are more likely to face a stockout, translating into increased transshipment costs; on the other hand, a higher service level protects customers against demand variations and emergency transshipments are then no longer needed as often, thus decreasing the total solution cost.

Table 8: Summary of cost savings when varying the service level for the Dynamic and Stochastic Inventory-Routing Problem on the standard data set

Instance	Low service level ( $1 - \alpha = 90\%$ )				High service level ( $1 - \alpha = 99\%$ )			
	Solution	Time (s)	Avg. lost	% increase	Solution	Time (s)	Avg. lost	% increase
small ( $n < 50$ )	8774.46	47.0	0.07	3.57	8145.81	47.4	0.03	-10.77
medium ( $50 \leq n \leq 100$ )	32257.77	702.2	0.00	-0.48	33566.23	738.6	0.00	2.55
Average	20516.12	374.6	0.03	1.54	20856.02	393.0	0.01	-4.11

### 5.3.7 Implementing consistency features in a dynamic environment

From a business and practical perspective, the decision making process is not only driven by costs but by quality of customer service. Our analysis has so far focused on cost minimization, disregarding other factors which may affect quality of service. Some of these factors were studied by [15] who have analyzed the effect of incorporating different consistency features into IRP solutions. For example, it may be undesirable to dispatch an almost empty vehicle, or one would not like to frequently deliver small amounts to the same customer since this is time consuming for both parties. To this end, we have run a subset of instances subject to two consistency features next described.

We first apply the *vehicle filling rate* consistency feature ensuring that the vehicle is only used if it is at least  $\gamma\%$  full, under the policy described in Section 3.2.2. We have tested the ML inventory policy with  $\gamma$  equal to 30, 50 and 70. Table 9 provides the average cost increase and the average lost demand (per customer per period) with respect to the base case. Running times are highly stable and, in general, as the requirement for the vehicle load increases, so does the solution cost. Adding this requirement to a deterministic environment [15] did not produce an increase of this magnitude.

Second, we apply a *quantity consistency* feature requiring that a customer can be visited only if the quantity delivered to it is at least twice its average demand. Results provided in Table 10 show that this policy yields a significant average cost increase with respect to the base case, and with respect to the average lost demand (per customer per period). Once more, ensuring consistent solutions over time turns out to be very costly in a dynamic environment, even though computational times are practically unchanged. Moreover, a slight in-

Table 9: Summary of the analysis for the Dynamic and Stochastic Inventory-Routing Problem with the vehicle filling rate consistency on the standard data set

Instance	$\gamma = 30$				$\gamma = 50$				$\gamma = 70$			
	Solution	Time (s)	Avg. lost	% increase	Solution	Time (s)	Avg. lost	% increase	Solution	Time (s)	Avg. lost	% increase
medium ( $50 \leq n \leq 100$ )	41233.80	707.8	0.00	31.06	41680.54	731.4	0.00	31.39	42474.15	744.6	0.00	34.22

crease in the average lost demand is observed when the quantities delivered to the customers are somewhat restricted.

Table 10: Summary of the analysis for the Dynamic and Stochastic Inventory-Routing Problem with the quantity consistency feature on the standard data set

Instance set	Under quantity consistency			
	Solution	Time (s)	Avg. lost	% increase in cost
medium ( $50 \leq n \leq 100$ )	48892.78	702.0	0.26	53.38

## 5.4 Final remarks

The two main features analyzed in these paper are now summarized. First, the use of demand forecasts has proved a powerful asset for the solution of the DSIRP. However, it requires the use of an optimization algorithm that can sometimes take very long to run if high quality solutions are expected. Nevertheless, our implementation of the ALNS as a means of solving each periodic problem has proved to be very efficient and flexible in the sense that we have solved the problem under two inventory policies and with two consistency features.

The second option considered in this paper concerns the use of lateral transshipments. Even if there are relatively few stockouts when transshipments are not considered, allowing them further reduces stockouts as well as the total cost. From an algorithmic point of view, enabling transshipments does not make the problem more difficult to solve since these can easily be integrated within the min-cost network flow problem which is used to compute the delivery quantities.

We have also analyzed the cost breakdown into its routing, inventory, direct deliveries and transshipments, and penalty components. Corroborating our preliminary findings from Sections 5.3.1 and 5.3.2, we found that routing costs are significantly reduced under proactive policies. This is due to the fact that when forecasts are used, the algorithm can avoid consecutive and costly visits to the same geographical area, yielding a better equilibrium between routing and inventory costs, in addition to reducing the use of emergency deliveries.

Finally, it is important to note that thanks to our choice of policies and to the algorithm design, the solution quality does not deteriorate when instances with very long horizons are solved. If a 20-period instance were to be solved

by dynamic or stochastic programming, it is likely that it would be intractable, which is not the case for the rolling horizon algorithms we have developed.

## 6 Conclusions

We have successfully solved the dynamic and stochastic version of the IRP under different policies. The first one uses a reactive framework, in which future visiting decisions are based only on the current state of the inventory of the customers. We have also implemented a more involved policy under which demand forecasts are used to support future decisions. In both cases, we have solved the problem without and with lateral transshipments as a means of reducing lost demand and diminishing total costs. We have implemented these policies in a rolling horizon fashion. We have shown through extensive computational experiments that the algorithms proposed perform very well and allow the proactive policies to take advantage of stochastic information in the form of demand forecasts. We have shown that increasing the length of the rolling horizon does not have a positive impact on the overall solution quality. In contrast, increasing the computation time of the subproblem associated with each period significantly improves solution quality. We have analyzed the impact of different inventory holding costs and service levels. Our experiments have shown that solution costs are correlated with the inventory holding cost for all policies. Imposing a high service level ensures that customers are protected against demand variations, which avoids unnecessary emergency transshipments and reduces lost demand. Decreasing the service level even slightly negatively impacts the solution cost. Moreover, we have considered the inclusion of consistency features in the solutions of the DSIRP. Our experiments show that ensuring consistent solutions over time under a dynamic and stochastic environment is much more expensive than under a deterministic setting.

## References

- [1] Y. Adulyasak, J.-F. Cordeau, and R. Jans. Formulations and branch-and-cut algorithms for multi-vehicle production and inventory routing problems. Technical Report G-2012-14, GERAD, Montreal, Canada, 2012.
- [2] H. Andersson, A. Hoff, M. Christiansen, G. Hasle, and A. Løkketangen. Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research*, 37(9):1515–1536, 2010.
- [3] C. Archetti, L. Bertazzi, G. Laporte, and M. G. Speranza. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science*, 41(3):382–391, 2007.
- [4] C. Archetti, L. Bertazzi, G. Paletta, and M. G. Speranza. Analysis of the maximum level policy in a production-distribution system. *Computers & Operations Research*, 12(38):1731–1746, 2011.

- [5] C. Archetti, L. Bertazzi, A. Hertz, and M. G. Speranza. A hybrid heuristic for an inventory routing problem. *INFORMS Journal on Computing*, 24(1):101–116, 2012.
- [6] S. Axsäter. Forecasting. In *Inventory control*, volume 90 of *International Series in Operations Research & Management Science*, pages 7–42. Springer, New York, 2006.
- [7] P. Bartodziej, U. Derigs, D. Malcherek, and U. Vogel. Models and algorithms for solving combined vehicle and crew scheduling problems with rest constraints : an application to road feeder service planning in air cargo transportation. *OR Spectrum*, 31(2):405–429, 2009.
- [8] G. Berbeglia, J.-F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010.
- [9] L. Bertazzi, G. Paletta, and M. G. Speranza. Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science*, 36(1):119–132, 2002.
- [10] L. Bertazzi, A. Bosco, F. Guerriero, and D. Laganà. A stochastic inventory routing problem with stock-out. *Transportation Research Part C*, 2012. doi: 10.1016/j.trc.2011.06.003.
- [11] R. Bhatnagar and C.-C. Teo. Role of logistics in enhancing competitive advantage: A value chain framework for global supply chains. *International Journal of Physical Distribution & Logistics Management*, 39(3):202–226, 2009.
- [12] A. M. Campbell, L. Clarke, A. J. Kleywegt, and M. W. P. Savelsbergh. The inventory routing problem. In T. G. Crainig and G. Laporte, editors, *Fleet management and logistics*, pages 95–113. Springer, 1998.
- [13] L. C. Coelho and G. Laporte. The exact solution of several classes of inventory-routing problems. Technical Report CIRRELT-2012-22, Montreal, Canada, 2012.
- [14] L. C. Coelho, J.-F. Cordeau, and G. Laporte. The inventory-routing problem with transshipment. *Computers & Operations Research*, 39(11):2537–2548, 2012.
- [15] L. C. Coelho, J.-F. Cordeau, and G. Laporte. Consistency in multi-vehicle inventory-routing. *Transportation Research Part C*, 24(1):270–287, 2012.
- [16] J.-F. Cordeau, G. Laporte, M. W. P. Savelsbergh, and D. Vigo. Vehicle routing. In C. Barnhart and G. Laporte, editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, pages 367–428. North-Holland, Amsterdam, 2007.

- [17] A. F. Darrat and M. Zhong. On testing the random-walk hypothesis: A model-comparison approach. *Financial Review*, 35(3):105–124, 2000.
- [18] B. Dezső, A. Jüttner, and P. Kovács. LEMON - an open source C++ graph template library. *Electronic Notes in Theoretical Computer Science*, 264(5):23–45, 2011.
- [19] D. Eddelbuettel and R. François. *RInside: C++ classes to embed R in C++ applications*, 2012. URL <http://CRAN.R-project.org/package=RInside>. R package version 0.2.6.
- [20] G. D. Eppen and R. K. Martin. Determining safety stock in the presence of stochastic lead time and demand. *Management Science*, 34(11):1380–1390, 1988.
- [21] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno. Real-time vehicle routing: Solutions concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 151(1):1–11, 2003.
- [22] M. Goetschalckx. *Supply Chain Engineering*, volume 161 of *International Series in Operations Research & Management Science*. Springer, New York, 2011.
- [23] P. Goodwin, D. Önköl, and M. Thomson. Do forecasts expressed as prediction intervals improve production planning decisions? *European Journal of Operational Research*, 205(1):195–201, 2010.
- [24] M. Hewitt, G. L. Nemhauser, and M. W. P. Savelsbergh. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing*, 22(2):314–325, 2010.
- [25] L. M. Hvattum and A. Løkketangen. Using scenario trees and progressive hedging for stochastic inventory routing problems. *Journal of Heuristics*, 15(6):527–557, 2009.
- [26] L. M. Hvattum, A. Løkketangen, and G. Laporte. Scenario tree-based heuristics for stochastic inventory-routing problems. *INFORMS Journal on Computing*, 21(2):268–285, 2009.
- [27] R. J. Hyndman and Y. Khandakar. Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, 27(3):1–22, 2008.
- [28] R. J. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder. *Forecasting with Exponential Smoothing: the State Space Approach*. Springer-Verlag, Berlin, 2008.
- [29] R. J. Hyndman, S. Razbash, and D. Schmidt. *forecast: Forecasting functions for time series and linear models*, 2012. URL <http://CRAN.R-project.org/package=forecast>. R package version 3.19.

- [30] A. J. Kleywegt, V. S. Nori, and M. W. P. Savelsbergh. The stochastic inventory routing problem with direct deliveries. *Transportation Science*, 36(1):94–118, 2002.
- [31] A. J. Kleywegt, V. S. Nori, and M. W. P. Savelsbergh. Dynamic programming approximations for a stochastic inventory routing problem. *Transportation Science*, 38(1):42–70, 2004.
- [32] G. Laporte, R. Musmanno, and F. Vocaturro. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44(1):125–135, 2010.
- [33] H. L. Lee and W. Seungjin. The whose, where and how of inventory control design. *Supply Chain Management Review*, 12(8):22–29, 2008.
- [34] S. G. Makridakis, S. C. Wheelwright, and R. J. Hyndman. *Forecasting: Methods and Applications*. Wiley, New York, 1998.
- [35] V. Maniezzo, T. Stützle, and S. Voß. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. Springer, New York, 2009.
- [36] L. M. Nonås and K. Jörnsten. Optimal solution in the multi-location inventory system with transshipments. *Journal of Mathematical Modelling and Algorithms*, 6(1):47–75, 2007.
- [37] Ö. Özer. Inventory management: Information, coordination, and rationality. In K. G. Kempf, P. Keskinocak, and R. Uzsoy, editors, *Planning Production and Inventories in the Extended Enterprise*, volume 151 of *International Series in Operations Research & Management Science*, pages 321–365. Springer, New York, 2011.
- [38] C. Paterson, G. Kiesmüller, R. Teunter, and K. Glazebrook. Inventory models with lateral transshipments: A review. *European Journal of Operational Research*, 210(2):125–136, 2011.
- [39] A.-S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman. A comparison of five heuristics for the multiple depot vehicle scheduling problem. *Journal of Scheduling*, 12(1):17–30, 2009.
- [40] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia. A review of dynamic vehicle routing problems. Technical Report CIRRELT-2011-62, Montreal, Canada, 2011.
- [41] H. N. Psaraftis. Dynamic vehicle routing problems. In B. L. Golden and A. A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 223–248. North-Holland, Amsterdam, 1998.
- [42] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. URL <http://www.R-project.org>.

- [43] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.
- [44] P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, University of Strathclyde, Glasgow, 1997.
- [45] O. Solyali and H. Süral. A branch-and-cut algorithm using a strong formulation and an a priori tour based heuristic for an inventory-routing problem. *Transportation Science*, 45(3):335–345, 2011.
- [46] M. Wen, J.-F. Cordeau, G. Laporte, and J. Larsen. The dynamic multi-period vehicle routing problem. *Computers & Operations Research*, 37(9): 1615–1623, 2010.