Computers & Operations Research 52 (2014) 55–67

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Heuristics for dynamic and stochastic inventory-routing

Leandro C. Coelho [a,b,*], Jean-François Cordeau [a,c], Gilbert Laporte [a,c]

[a] Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Canada
[b] Faculté des sciences de l'administration, Université Laval, 2325 de la Terrasse, Québec, Canada G1V 0A6
[c] HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

ARTICLE INFO

Available online 9 July 2014

Keywords:
Dynamic demand
Stochastic demand
Inventory-routing
Rolling horizon algorithm
Heuristic

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 made jointly, which is the case in vendor-managed inventory systems. The IRP has received significant attention in recent years, with several heuristic and exact algorithms 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. We propose different heuristic 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 find that the use of a longer rolling horizon step does not help improve solutions. 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.

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 [40].

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 [6,13,23] as well as exact algorithms [5,20,54] have been proposed for the single-vehicle version of the problem. In addition, an extended version of the multi-vehicle IRP (MIRP) incorporating several consistency features has been solved heuristically [24] and exactly [20,22]. 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 have been written on applications of the IRP by Andersson et al. [4] and on its methodological aspects by Coelho et al. [25].

An important variant of the problem is the stochastic IRP (SIRP) that arises when future demand is not known with certainty but only in a probabilistic sense. Several algorithms have been proposed for the infinite-horizon case in which one assumes that demand distributions are stationary and the objective is to optimize the discounted value of delivery decisions at each time step (see, e.g., [1,18,32,37,38]). Jaillet et al. [36] have also considered stationary demands but have solved the problem in a rolling horizon framework by considering constant replenishment intervals. Recently, Bertazzi et al. [15] have formulated the SIRP as a dynamic program and have solved it by means of a heuristic rollout algorithm. This algorithm samples unknown demands and solves a series of deterministic instances to choose the best action at each step in the planning horizon.

When demand is not assumed to be stationary but has a distribution that changes over time, one obtains a dynamic and stochastic problem (DSIRP). Solving a dynamic problem consists of

* Corresponding author at: Faculté des sciences de l'administration, Université Laval, Québec, Canada.
Tel.: +1 418 656 2131x5042; fax: +1 418 656 2624.
E-mail addresses: leandro.coelho@cirrelt.ca (L.C. Coelho), jean-francois.cordeau@cirrelt.ca (J.-F. Cordeau), gilbert.laporte@cirrelt.ca (G. Laporte).

http://dx.doi.org/10.1016/j.cor.2014.07.001
0305-0548/© 2014 Elsevier Ltd. All rights reserved.

proposing a *solution policy* as opposed to computing a static output [12]. A possible policy is to optimize a static instance whenever new information becomes available. The drawback of such a procedure 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 [48], Ghiani et al. [28] and Berbeglia et al. [12].

The algorithms developed by Coelho et al. [23] for deterministic IRPs 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 [44], 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 [9].

This paper makes five main scientific contributions. First, we describe and compare several solution policies for the DSIRP in which the objective is to minimize the total inventory, distribution and shortage costs. Second, we propose an algorithm that uses historical data in order to take into account unknown future demands, thus being able to efficiently solve instances in which the demand presents a trend or a seasonality. Third, as in Coelho et al. [23], we allow the use of lateral transshipments between customers as a means of avoiding stockouts when demand is high. This is in line with traditional inventory sharing policies and some recent studies [43,45]. Fourth, 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 deterministic study of Coelho et al. [24]. We investigate whether the effect of enforcing consistent solutions is similar to what is observed in a deterministic environment. Fifth, in addition to proposing an efficient and flexible solution algorithm for the DSIRP, we evaluate the value of demand forecasts and transshipments in the context of the dynamic IRP.

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 $\mathcal{V}'=\mathcal{V}\,\backslash\{0\}$ 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. Considering that the supplier's customers are often retailers which are themselves facing an external demand from the end customers, this value may be interpreted as the total orders received by a retailer in a given time period. In practice, the demand is not known by the decision maker (usually the supplier) who has to estimate it on the basis of historical data. We assume that the decision maker can use any kind of forecast and input this information into the algorithmic framework we provide. The decision maker becomes aware of 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 further assume the supplier has enough inventory to meet al.l 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 z_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 of each customer i, respectively.

As is common in the IRP literature, we assume that a single vehicle of capacity Q is available [5,6,13,15,23]. 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 assume that these costs satisfy the triangle inequality. We also consider two inventory policies. The first one, called maximum level (ML) policy, allows the supplier to freely choose the quantity to deliver to the customers, limited only by the inventory capacity at the customers. The second assumes that the supplier uses an order-up-to (OU) inventory policy. This policy has been widely used in IRPs and related problems [3,5,6,13,23] 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 that 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 βc_{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

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

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

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

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.

From a business and practical perspective, the decision making process is not only driven by costs but by quality of customer service. Traditional IRP analysis has focused on cost minimization, disregarding other factors which may affect quality of service. Some of these factors were studied by Coelho et al. [24] who have analyzed the effect of incorporating different consistency features in IRP solutions. For example, it may be undesirable to dispatch an almost empty vehicle, or one may not like to frequently deliver small amounts to the same customer since this is time consuming for both parties. Some of the consistency features proposed for the deterministic IRP are also meaningful in a dynamic and stochastic environment. Here we extend two of the consistency features to the DSIRP. The first one is the *vehicle filling rate* consistency, which ensures that the vehicle is only used if it is filled at least to a given percentage. The second is the *quantity consistency* feature, which requires that a customer be visited only if the quantity delivered to it lies within a given interval. These two features help increase the quality of the service, but their impact in terms of cost has not yet been addressed in dynamic and stochastic IRPs.

This problem can also be formulated as a dynamic program (DP). The states of the DP at time t are represented by a vector of inventory levels for the supplier and all customers. The set of feasible controls that can be applied to any state represent which customers to be visited in the next period and the quantity of products to be delivered to each of them. Transshipments are computed later once the demand is revealed. Since our aim is to propose a simpler heuristic capable of handling large-scale instances, we do not adopt this methodology. For more information on dynamic programming, the reader is referred to Bertsekas [16] and Powell [47].

3. Solution policies

The DSIRP can be solved under a *proactive* policy or under a *reactive* policy, depending on whether demand forecasts are used 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. We provide details for each one of them in the following sections.

3.1. Reactive policies

Under reactive policies one observes the state of the system in order to make the next decision regarding routing and delivery. Formally, a reactive policy triggers a replenishment order to bring the inventory position up to level S whenever the inventory reaches the reorder point s. The reorder point s should consider the delivery lead time and the stockout risk resulting from the stochasticity of the demand. In systems in which demand can be measured by the unit, with very small time intervals, this is formally defined as an (s, S) replenishment system [7,19]. The situation we describe is better represented as an (R, s, S) system with a reorder interval $R = 1$ due to inventory measurements being made once per period (see, e.g., [53]).

We now present the two reactive policy strategies, without and with the use of lateral transshipments.

3.1.1. Reactive: 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 [6,13,15,23,24]. 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 [15,24]. As in these studies, we also assume that the target level meets the customer inventory capacity. As already 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. Reactive: routing and transshipment

This policy allows lateral transshipments between customers. The decision to visit or not visit 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 interesting from a cost point of view. Note that lateral transshipments are allowed only as an emergency measure, i.e., they cannot be used just to move inventory to a location having a lower holding cost.

3.2. Proactive policies

In a proactive policy one 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. With more information available, the decision maker also gains more flexibility in terms of the amplitude of his decisions. One of these is related to the replenishment policy used, which is no longer necessarily the hard OU policy. A relaxation of this policy yields the more general ML replenishment policy, which is possible under these proactive strategies. Such policies constitute a form of lookahead policies, used extensively in DP and specifically in several instances for the IRP. One recent combination of lookahead with approximate DP for the IRP and some generalizations is presented in Adelman and Klabjan [2]. Lookahead policies have also been used within a mixed-integer programming framework for the IRP in Toriello et al [55]. However, neither of these two papers considers stochastic demand.

We describe next the two proactive policies we propose, without and with the use of emergency lateral transshipments.

3.2.1. Proactive: 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 and the proper comparison with the reactive policies. Nevertheless, the more general ML inventory policy is equally possible.

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 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. Proactive: routing and transshipment

As an extension of the previous policy, in this case lateral transshipments can take place after the demand is realized. Again, transshipments are only allowed as a recourse measure against stockouts.

4. Algorithms

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

4.1. Algorithms for the 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. Algorithm for the reactive policy: 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. Because replenishment decisions cannot be taken at any time, but only at the end of each period, after demand has occurred, we employ an (R, s, S) system with $R=1$. Let L denote the delivery lead time and let R_i be the interval at which orders can be placed for customer i. We estimate by $\hat{\mu}_i$ the expected demand per period of customer i, and by $\hat{\sigma}_i$ its standard deviation. These values as well as the resulting threshold can be updated at every period. Following classical inventory management policies [53] and assuming independent and normally distributed demands, s_i can be computed as

$$s_i = \hat{\mu}_i (R_i + L_i) + z_\alpha \hat{\sigma}_i \sqrt{R_i + L_i}, \tag{3}$$

where α is the probability of a stockout and z_α is the α-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 $\mathcal{NP}$-hard problem since it includes the Traveling Salesman Problem as a special case. 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, assuming an OU policy applies.

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; 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 (or 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:

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

subject to

$$q_i + w_i = d_i' \quad i \in \mathcal{V}' \tag{5}$$

$$\sum_{i \in \mathcal{V}'} q_i \le Q \tag{6}$$

$$q_i \le Q y_i \quad i \in \mathcal{V}' \tag{7}$$

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

$$\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}, i<j} x_{ij} \le \sum_{i \in \mathcal{S}} y_i - y_m \quad \mathcal{S} \subseteq \mathcal{V}', m \in \mathcal{S} \tag{9}$$

$$q_i, w_i \ge 0 \quad i \in \mathcal{V}' \tag{10}$$

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

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

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

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's vehicle. Constraints (8) and (9) are degree constraints and subtour elimination constraints, respectively. Constraints (10)–(13) enforce the non-negativity and integrality conditions on the variables. The RD model can be simplified by preprocessing all customers with zero d_i' and removing the corresponding variables. Algorithm 1 provides a pseudocode of this policy implementation.

Algorithm 1. Pseudocode: routing only.

```
1: for $t=0$ to $p-1$ do
2:   for $i=1$ to $n$ do
3:     Compute $s_i$ on the basis of the past demand.
4:     if $I_i^t < s_i$ then
5:       Include $i$ in the set of customers that follow an OU
         policy in period $t+1$.
6:     end if
7:   end for
8:   Solve RD to define direct deliveries destinations and
     quantities.
9: end for
```

4.1.2. Algorithm for the reactive policy: 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. Here 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. The model is solved once per period, after demands are realized and can handle all customers at once. The problem is defined as follows:

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

subject to

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

$$0 \le I_i \le C_i \quad i \in \mathcal{V}' \tag{16}$$

$$0 \le l_i \le -\min\{0, I_i^0\} \quad i \in \mathcal{V}' \tag{17}$$

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

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. $I_i^0 \ge 0$ and $I_j^0 \ge 0$: 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. $I_i^0 \ge 0$ 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 $I_j^0 \ge 0$: 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 Fig. 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 z_i. The

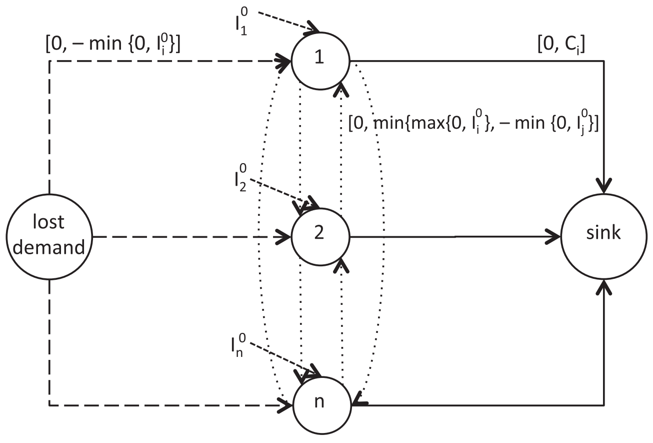

Fig. 1. Example of the network flow problem solved to decide of transshipment quantities, origins and destinations.

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 βc_{ij}. The flows on these arcs lie in the interval $[0, \min\{\max\{0, I_i^0\}, -\min\{0, I_j^0\}\}]$.

The pseudocode corresponding to this policy is described in Algorithm 2. Note that lines $1-8$ are identical to Algorithm 1.

Algorithm 2. Pseudocode: routing and transshipment.

```
1:  for $t=0$ to $p-1$ do
2:    for $i=1$ to $n$ do
3:      Compute $s_i$ on the basis of the past demand.
4:      if $I_i^t < s_i$ then
5:        Include $i$ in the set of customers that follow an OU
          policy in period $t+1$.
6:      end if
7:    end for
8:  Solve RD to define direct delivery destinations and
    quantities.
9:  Reveal demands of period $t+1$.
10: for $i=1$ to $n$ do
11:   if $I_i^t + q_i^{t+1} - d_i^{t+1} < 0$ then
12:     Allow transshipments to customer $i$.
13:   end if
14:   end for
15:   Solve TOD to define transshipment origins, destinations
    and quantities.
16: end for
```

4.2. Algorithms for the proactive policies

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

4.2.1. Algorithm for the proactive policy: routing only

This policy makes use of forecasts of future demand to help make current decisions. There are three main decisions affecting the performance of the overall algorithm. We discuss them here and perform computational experiments on them in Section 5.3.

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 [41]. 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 [34] for details).

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 opted for an adaptive large neighborhood search (ALNS) matheuristic, which was proved to provide very good results on benchmark static instances [23]. This heuristic is described in Section 4.3 and can handle both the ML and OU inventory policies. Once forecasts are available, the dynamic problem reduces to a static one.

Algorithm 3 provides the pseudocode of this policy.

Algorithm 3. Pseudocode: routing only.

```
1: for $t=0$ to $p-1$ do
2:   for $i=1$ to $n$ do
3:     Compute an $f$-period forecast for $i$ based on past demand observations.
4:   end for
5:   Apply the ALNS-based heuristic to the reduced $f$-period problem.
6:   Implement the route obtained for the first period.
7: end for
```

4.2.2. *Algorithm for the proactive policy: routing and transshipments*

This policy works much like the previous one, except that after vehicle routes have been created for all periods of the rolling horizon and the first of them has been 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.

The pseudocode of this policy is presented in Algorithm 4.

Algorithm 4. Pseudocode: routing and transshipments.

```
1:  for $t=0$ to $p-1$ do
2:    for $i=1$ to $n$ do
3:      Compute an $f$-period forecast for $i$ based on past demand observations.
4:    end for
5:    Apply the ALNS-based heuristic to the reduced $f$-period problem.
6:    Implement the route obtained for the first period.
7:    Reveal demands of period $t+1$.
8:    for $i=1$ to $n$ do
9:      if $I_i^t+q_i^{t+1}-d_i^{t+1}<0$ then
10:       Allow transshipments to customer $i$.
11:     end if
12:   end for
13:   Solve TOD to define transshipments origins, destinations and quantities.
14: end for
```

4.3. *ALNS matheuristic*

The algorithm proposed in [23] is an implementation of the ALNS algorithm originally proposed in [51] for the Vehicle Routing Problem and already successfully applied to a number of other contexts [11,31,39,46]. In this implementation, some subproblems are solved exactly as min-cost network flow problems. It can therefore be described as a *matheuristic* [42], 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 works with a subset of the operators used in [23] and runs for fewer iterations in order to make it faster, because it has to be used several times in the rolling horizon mechanism. 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 5.3.2.

In summary, the algorithm of [23] 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. [23] to generate IRP solution values lying within 0.50% of optimality.

Each operator i is assigned a weight ω_i whose value depends on its past performance and on its score. Given h operators with weights ω_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 the their past performance: they receive a high reward σ_1 if they yield a new best solution, a medium reward σ_2 if their solution is better than the incumbent one, or a low reward σ_3 if the solution they provide is worse but still accepted. Initially, all operators have the same probability of being selected. After φ iterations, scores are computed taking into account the rewards accumulated as follows. Let π_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

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

where $\eta \in [0,1]$ is called the reaction factor, controlling how quickly the weight adjustment reacts to changes in the movement performance. Also after φ iterations, 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 τ_{start} and is decreased by a cooling rate factor ϕ 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 ρ is an integer randomly drawn from the interval [1, n] using a semi-triangular distribution with a negative slope. This ensures frequent small values of ρ, which are desirable to fine tune the solution at hand, and occasional large values of ρ, which help make significant changes to the incumbent solution.

- Destroy operators
  - *Randomly remove* ρ: This operator randomly selects one period and removes one randomly selected customer from it. It is repeated ρ times.
  - *Shaw removal*: Following the ideas developed in [51] and [52], 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* ρ: This operator randomly inserts ρ 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 ρ 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 ρ customers*: This operator selects two customers from two different periods and swaps their assignments. It is also applied ρ 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 in order to be reasonably fast. This drove us to the following settings for the operators and parameters after a tuning phase. The starting temperature τ_{start} is set to 20,000 and the cooling rate ϕ 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 φ was set to 200 iterations and the reaction factor η 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 5 presents a simplified pseudocode for this heuristic. For algorithmic and implementation details, the reader is referred to [23].

Algorithm 5. ALNS heuristic – simplified pseudocode.

```
1:  Initialize: set al.l 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. 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 5.3.2–5.3.7 for several alternative configurations.

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 node is equipped with two Intel Westmere-EP X5650 hexa-core processors running at 2.67 GHz and with up to 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 [26] running the network simplex algorithm for its internal computations. Forecasts were carried out using the *forecast* package [33,35] available for *R* Language and Environment for Statistical Computing [49] and embedded within our C++ code using the *RInside* classes [27]. We allowed the software to run under its default settings, searching through all the 30 variants of the exponential smoothing models described in Hyndman et al. [34]. We made use of the 50 past periods immediately before the current period as historical data for the chosen forecasting method. An alternative would have been to forecast the aggregate demand of the supplier. However, this method would require a disaggregation process to obtain the demand forecast of each customer.

Given that the lead time is equal to zero (there is no demand taking place between the moment one decides to deliver and the time of the delivery, i.e., the delivery taking place in period t can be used to satisfy the demand of period t), and that the order interval R_i is equal to one (deliveries can take place every period), the value of s_i used in Eq. (3) is then

$$s_i = \hat{\mu}_i + z_\alpha \hat{\sigma}_i. \tag{20}$$

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 [29].

5.2. Instance generation

We have generated instances following some of the standards used for the instances generated for the IRP with datasets used by more than a dozen papers [3,5,6,8,10,13–15,20–24,54], 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 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 μ_i is generated as an integer random number following a discrete uniform distribution in the

interval [10, 100], and standard deviation σ_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: the mean production $\bar{r}$ is generated as an integer random number following a discrete uniform distribution in the interval $[100n, 140n]$, and σ_0 is generated 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. These are used only to account for inventory costs at the supplier, as in Archetti et al. [5];
- maximum inventory level C_i $(i>0)$: $\mu_i g_i$, 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 $(i>0)$: $C_i - \mu_i$;
- 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: $z_i = 200h_i$;
- 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 exhibits some seasonalities. Indeed, Bhatnagar and Teo [17] 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 when the same product is offered in different markets. 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 Eq. (20) assumes that demands of consecutive periods are independent, which is no longer the case in the presence of seasonality. However, since this baseline policy is used to first compare the solution costs with the more elaborate proactive policy, we tolerate computing the reorder point in this approximate way.

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 between the two policies, due to the nature of the design of the reactive policies, which employ an OU repleshment strategy. The proactive policies can handle both the OU and the ML policies, and we provide a comparative analysis later. The transshipment cost β was set to 0.01 as in [23] and 95% prediction intervals were used, as in [30]. We first present results for the base case, and later we provide analyses for a number of variations of the problem and of the algorithm which provide important insights. These analyses are supported by one-tailed Wilcoxon signed-rank tests [50] which show that most observed differences are highly significant $(p<0.0001)$.

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 case 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 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 (in units) under both the reactive and proactive policies; second, by decreasing the lost demand, it also lowers the average solution cost. Finally, the computational effort of forecasting and

Table 1
Summary of computational results for the dynamic and stochastic inventory-routing problem on the standard data set (150 instances).

Transshipment	Instance	Reactive policy			Proactive policy			p-value
	size	Cost	Time (s)	Avg. lost	Cost	Time (s)	Avg. lost	
No	Small $(5 \le n \le 25)$	14,974.17	0.0	0.62	14,224.84	47.2	0.10	
	Medium $(50 \le n \le 100)$	39,546.01	4.3	0.41	32,774.85	453.3	0.00	
	Large $(125 \le n \le 200)$	64,854.75	408.5	0.46	64,784.85	3781.0	0.00	
Average		39,791.64	137.6	0.50	37,261.51	1427.2	0.03	<0.0001
Yes	Small $(5 \le n \le 25)$	14,382.67	0.0	0.00	8586.53	46.9	0.05	
	Medium $(50 \le n \le 100)$	37,720.58	4.4	0.00	27,743.95	452.7	0.00	
	Large $(125 \le n \le 200)$	61,455.93	498.4	0.00	56,506.38	3934.4	0.00	
Average		37,853.06	167.6	0.00	30,945.62	1478.0	0.02	<0.0001
p-value		<0.0001			<0.0001			

Table 2
Summary of computational results for the dynamic and stochastic inventory-routing problem on the seasonal data set (150 instances).

Transshipment	Instance	Reactive policy			Proactive policy			p-value
	size	Cost	Time (s)	Avg. lost	Cost	Time (s)	Avg. lost	
No	Small ($5 \leq n \leq 25$)	15,994.92	0.1	0.41	14,510.22	48.4	0.00	
	Medium ($50 \leq n \leq 100$)	40,953.04	5.5	0.38	41,071.74	499.7	0.00	
	Large ($125 \leq n \leq 200$)	70,442.24	758.3	0.41	73,252.94	4734.7	0.00	
Average		42,463.40	254.6	0.40	42,944.97	1760.9	0.00	0.0735
Yes	Small ($5 \leq n \leq 25$)	15,515.02	0.1	0.00	14,160.04	48.1	0.00	
	Medium ($50 \leq n \leq 100$)	39,164.04	5.8	0.00	40,433.81	501.1	0.00	
	Large ($125 \leq n \leq 200$)	66,093.51	751.5	0.00	68,918.08	4739.1	0.00	
Average		40,257.52	252.5	0.00	41,170.64	1762.8	0.00	< 0.0001
p-value		< 0.0001			0.5823			

Table 3
Summary of computational results for the dynamic and stochastic inventory-routing problem on the correlated data set (150 instances).

Transshipment	Instance	Reactive policy			Proactive policy			p-value
	size	Cost	Time (s)	Avg. lost	Cost	Time (s)	Avg. lost	
No	Small ($5 \leq n \leq 25$)	15,546.15	0.1	0.43	16,466.03	48.3	0.00	
	Medium ($50 \leq n \leq 100$)	42,940.79	14.4	0.48	40,867.58	503.8	0.00	
	Large ($125 \leq n \leq 200$)	75,067.20	1506.5	0.47	71,152.13	4727.4	0.00	
Average		44,518.05	507.0	0.46	42,828.58	1759.8	0.00	< 0.0001
Yes	Small ($5 \leq n \leq 25$)	15,132.41	0.2	0.00	14,822.28	48.2	0.00	
	Medium ($50 \leq n \leq 100$)	40,526.86	15.4	0.00	40,224.01	502.9	0.00	
	Large ($125 \leq n \leq 200$)	70,536.52	1749.1	0.00	68,844.68	4713.8	0.00	
Average		42,065.26	588.2	0.00	41,296.99	1755.0	0.00	0.0418
p-value		< 0.0001			< 0.0001			

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 decreasing the solution cost by up to 16.9% on average. Finally, comparing policies in Table 2 shows that a policy that takes future demand into consideration is capable of completely preventing stockouts. However, in both cases the average cost of the proactive policy is slightly higher than under the reactive policy, however only for the case with transshipment the averages are significantly different. We further explore this counter-intuitive result in the next section, where we allow the ALNS heuristic to refine solutions for longer periods of time, i.e., running for more iterations. As will be shown, the algorithm performs even better when allowed to run slightly longer.

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 simultaneously, 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.

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 [24,23], 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 run for fewer iterations in order 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 longer in Section 5.3.2.

In addition to the results presented above, we have investigated a number of other scenarios using the best of the proposed policies, i.e., that 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. The aim of this experiment is to assess whether one can obtain better solutions by allowing a longer computational time. We now allow the ALNS to iterate 40,000 times. Average solutions for the proactive policy without and with transshipments as well as significance tests 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 allowed to run for enough time, enabling it 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 (150 instances).

Instance	Without transshipments				With transshipments				p-value
	Cost	Time (s)	Avg. lost	Increase (%)	Cost	Time (s)	Avg. lost	Increase (%)	
Small ($5 \le n \le 25$)	9131.45	67.1	0.10	−7.20	8355.69	67.4	0.05	−2.16	
Medium ($50 \le n \le 100$)	30,137.81	888.3	0.00	−4.39	26,891.26	880.5	0.00	−4.23	
Large ($125 \le n \le 200$)	60,051.36	9248.9	0.00	−3.76	55,530.30	9196.0	0.00	−2.08	
Average	33,106.87	3401.4	0.03	−5.12	30,259.08	3381.3	0.02	−2.82	< 0.0001
p-value (with respect to regular)				< 0.0001				< 0.0001	

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 (150 instances).

Instance	Without transshipments				With transshipments				p-value
	Cost	Time (s)	Avg. lost	% increase over OU	Cost	Time (s)	Avg. lost	% increase over OU	
Small ($5 \le n \le 25$)	10,225.93	46.3	0.24	−0.78	7926.71	44.6	0.16	−9.96	
Medium ($50 \le n \le 100$)	30,360.66	452.7	0.01	−1.50	26,527.05	444.1	0.01	−3.78	
Large ($125 \le n \le 200$)	61,250.17	3860.1	0.01	−0.49	54,292.38	4100.1	0.00	−4.19	
Average	33,945.59	1453.0	0.09	−0.92	29,582.05	1529.6	0.06	−5.97	< 0.0001
p-value				< 0.0001				< 0.0001	

5.3.3. *Applying an ML inventory policy*

We have also examined how the ML inventory policy performs with respect to the OU policy. 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. Allowing transshipments enables all customers to share stockout risks and their respective inventories. This translates into less inventory being delivered to the customers, which in turn decreases inventory holding costs.

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.

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 and confirmed by significance tests, 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.

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, which translates into increased transshipment costs on average; 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 average total solution cost. The impact of changing the service level is, however, dependent on the instance size.

5.3.7. *Implementing consistency features in a dynamic environment*

We first apply the *vehicle filling rate* consistency feature ensuring that the vehicle is only used if it is at least γ% full, under the policy described in Section 3.2.2. We have tested the ML inventory policy with γ 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

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 (105 instances).

Change on inventory cost	Instance	Reactive policy				Proactive policy			
		Cost	Time (s)	Avg. lost	% increase (p-value)	Cost	Time (s)	Avg. lost	% increase (p-value)
Halved	Small ($5 \le n \le 25$)	14036.64	0.1	0.00	−2.52	8011.02	47.3	0.08	−8.29
	Medium ($50 \le n \le 100$)	39,703.27	18.9	0.00	−4.86	30,366.73	721.7	0.00	−6.72
Average		26,869.95	9.5	0.00	−3.69 (<0.0001)	19,188.87	384.5	0.04	−7.51 (<0.0001)
Doubled	Small ($5 \le n \le 25$)	15074.69	0.1	0.00	4.66	9291.39	47.4	0.05	8.26
	Medium ($50 \le n \le 100$)	45,346.88	19.3	0.00	8.45	36,087.193	707.3	0.00	10.62
Average		30,210.79	9.7	0.00	6.55 (<0.0001)	22,689.29	377.4	0.02	9.44 (<0.0001)

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 (105 instances).

Instance	Without transshipments				With transshipments			
	Cost	Time (s)	Avg. lost	% increase over $f=3$ (p-value)	Cost	Time (s)	Avg. lost	% increase over $f=3$ (p-value)
Small ($5 \le n \le 25$)	15,553.77	62.8	0.04	0.25	10,322.57	63.1	0.02	0.09
Medium ($50 \le n \le 100$)	45,963.58	1337.8	0.00	0.22	38,864.04	1341.1	0.00	0.16
Average	30,758.67	700.3	0.02	0.24 (<0.0001)	24,593.31	702.1	0.01	0.12 (<0.0001)

Table 8
Summary of cost savings when varying the service level for the dynamic and stochastic inventory-routing problem on the standard data set (105 instances).

Instance	Low service level ($1-\alpha=90\%$)				High service level ($1-\alpha=99\%$)			
	Cost	Time (s)	Avg. lost	% increase	Cost	Time (s)	Avg. lost	% increase
Small ($5 \le n \le 25$)	8774.46	47.0	0.07	3.57	8145.81	47.4	0.03	−10.77
Medium ($50 \le n \le 100$)	32,257.77	702.2	0.00	−0.48	33,566.23	738.6	0.00	2.55
Average	20,516.12	374.6	0.03	1.54	20,856.02	393.0	0.01	−4.11

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 (45 instances).

Instance	$\gamma=30$				$\gamma=50$				$\gamma=70$			
	Cost	Time (s)	Avg. lost	% increase	Cost	Time (s)	Avg. lost	% increase	Cost	Time (s)	Avg. lost	% increase
Medium ($50 \le n \le 100$)	41,233.80	707.8	0.00	31.06	41,680.54	731.4	0.00	31.39	42,474.15	744.6	0.00	34.22

solution cost. Adding this requirement to a deterministic environment [24] did not produce an increase of this magnitude.

Second, we apply a *quantity consistency* feature requiring that a customer can only be visited 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 increase 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 (45 instances).

Instance set	Under quantity consistency			
	Cost	Time (s)	Avg. lost	% increase in cost
Medium ($50 \le n \le 100$)	48,892.78	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, the use of transshipments does not make the problem more difficult to solve since these can easily be integrated within the min-cost network flow problem used to compute the delivery quantities. Its integration in the end of each ALNS iteration has proved successful as it acts as a recourse function for each of the created vehicle routes.

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 proposed algorithms 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.

Acknowledgments

The authors thank Raf Jans for his comments and suggestions. This work was partly supported by the Canadian Natural Sciences and Engineering Research Council under Grants 227837-09 and 39682-10. This support is gratefully acknowledged. We also thank the Calcul Québec for providing computing facilities. Thanks are due to the referees for their valuable comments.

References

[1] Adelman D. A price-directed approach to stochastic inventory/routing. Oper Res 2004;52(4):499–514.
[2] Adelman D, Klabjan D. Computing near-optimal policies in generalized joint replenishment. INFORMS J Comput 2012;24(1):148–64.
[3] Adulyasak Y, Cordeau J-F, Jans R. Optimization-based adaptive large neighborhood search for the production routing problem. Transp Sci 2014;48(1):20–45.
[4] Andersson H, Hoff A, Christiansen M, Hasle G, Løkketangen A. Industrial aspects and literature survey: combined inventory management and routing. Comput Oper Res 2010;37(9):1515–36.
[5] Archetti C, Bertazzi L, Laporte G, Speranza MG. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. Transp Sci 2007;41(3):382–91.
[6] Archetti C, Bertazzi L, Hertz A, Speranza MG. A hybrid heuristic for an inventory routing problem. INFORMS J Comput 2012;24(1):101–16.
[7] Arrow KJ, Harris T, Marschak J. Optimal inventory policy. Econometrica 1951;19(3):250–72.
[8] Avella P, Boccia M, Wolsey LA. Single item reformulations for a vendor managed inventory routing problem: computational experience with benchmark instances. Technical Report 2013/45, CORE, Louvain-la-Neuve, Belgium; 2013.
[9] Axsäter S. Inventory control – international series in operations research & management science, vol. 90. New York: Springer; 2006.
[10] Bard JF, Nananukul N. A branch-and-price algorithm for an integrated production and inventory routing problem. Comput Oper Res 2010;37(12):2202–17.
[11] Bartodziej P, Derigs U, Malcherek D, Vogel U. 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 Spectr 2009;31(2):405–29.
[12] Berbeglia G, Cordeau J-F, Laporte G. Dynamic pickup and delivery problems. Eur J Oper Res 2010;202(1):8–15.
[13] Bertazzi L, Paletta G, Speranza MG. Deterministic order-up-to level policies in an inventory routing problem. Transp Sci 2002;36(1):119–32.
[14] Bertazzi L, Savelsbergh M, Speranza MG. Inventory routing. In: Golden BL, Raghavan S, Wasil EA, editors. The Vehicle Routing Problem. New York: Springer; 2008. p. 49–72.
[15] Bertazzi L, Bosco A, Guerriero F, Laganà D. A stochastic inventory routing problem with stock-out. Transp Res Part C: Emerg Technol 2013;27(1):89–107.
[16] Bertsekas DP. Dynamic programming and optimal control: vol. 2. Belmont, MA: Athena Scientific; 2012.
[17] Bhatnagar R, Teo C-C. Role of logistics in enhancing competitive advantage: a value chain framework for global supply chains. Int J Phys Distrib Logist Manag 2009;39(3):202–26.
[18] Campbell AM, Clarke L, Kleywegt AJ, Savelsbergh MWP. The inventory routing problem. In: Crainic TG, Laporte G, editors. Fleet management and logistics pages. Boston: Springer; 1998. p. 95–113.
[19] Caplin AS. The variability of aggregate demand with (S,s) inventory policies. Econometrica 1985;53(6):1395–409.
[20] Coelho LC, Laporte G. The exact solution of several classes of inventory-routing problems. Comput Oper Res 2013;40(2):558–65.
[21] Coelho LC, Laporte G. A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. Int J Prod Res 2013;51(23–24):7156–69.
[22] Coelho LC, Laporte G. Improved solutions for inventory-routing problems through valid inequalities and input ordering. Int J Prod Econ forthcoming, 2014. http://dx.doi.org/10.1016/j.ijpe.2013.11.019.
[23] Coelho LC, Cordeau J-F, Laporte G. The inventory-routing problem with transshipment. Comput Oper Res 2012;39(11):2537–48.
[24] Coelho LC, Cordeau J-F, Laporte G. Consistency in multi-vehicle inventory-routing. Transp Res Part C: Emerg Technol 2012;24(1):270–87.
[25] Coelho LC, Cordeau J-F, Laporte G. Thirty years of inventory-routing. Transp Sci 2014;48(1):1–19.
[26] Dezső B, Jüttner A, Kovács P. LEMON – an open source C++ graph template library. Electron Notes Theor Comput Sci 2011;264(5):23–45.
[27] Eddelbuettel D, François R. RInside: C++ classes to embed R in C++ applications; 2012. URL: ⟨http://CRAN.R-project.org/package=RInside⟩. R Package Version 0.2.6.
[28] Ghiani G, Guerriero F, Laporte G, Musmanno R. Real-time vehicle routing: solutions concepts, algorithms and parallel computing strategies. Eur J Oper Res 2003;151(1):1–11.
[29] Goetschalckx M. Supply chain engineering, International series in operations research & management science, vol. 161. New York: Springer; 2011.
[30] Goodwin P, Önkal D, Thomson M. Do forecasts expressed as prediction intervals improve production planning decisions? Eur J Oper Res 2010;205(1):195–201.
[31] Hewitt M, Nemhauser GL, Savelsbergh MWP. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. INFORMS J Comput 2010;22(2):314–25.
[32] Hvattum LM, Løkketangen A, Laporte G. Scenario tree-based heuristics for stochastic inventory-routing problems. INFORMS J Comput 2009;21(2):268–85.
[33] Hyndman RJ, Khandakar Y. Automatic time series forecasting: the forecast package for R. J Stat Softw 2008;27(3):1–22.
[34] Hyndman RJ, Koehler AB, Ord JK, Snyder RD. Forecasting with exponential smoothing: the state space approach. Berlin: Springer-Verlag; 2008.
[35] Hyndman RJ, Razbash S, Schmidt D. Forecast: forecasting functions for time series and linear models, 2012. URL: ⟨http://CRAN.R-project.org/package=forecast⟩. R Package Version 3.19.
[36] Jaillet P, Bard JF, Huang L, Dror M. Delivery cost approximations for inventory routing problems in a rolling horizon framework. Transp Sci 2002;36(3):292–300.
[37] Kleywegt AJ, Nori VS, Savelsbergh MWP. The stochastic inventory routing problem with direct deliveries. Transp Sci 2002;36(1):94–118.

[38] Kleywegt AJ, Nori VS, Savelsbergh MWP. Dynamic programming approximations for a stochastic inventory routing problem. Transp Sci 2004;38(1):42–70.

[39] Laporte G, Musmanno R, Vocaturo F. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. Transp Sci 2010;44(1):125–35.

[40] Lee HL, Seungjin W. The whose, where and how of inventory control design. Supply Chain Manag Rev 2008;12(8):22–9.

[41] Makridakis SG, Wheelwright SC, Hyndman RJ. Forecasting: methods and applications. New York: Wiley; 1998.

[42] Maniezzo V, Stützle T, Voß S. Matheuristics: hybridizing metaheuristics and mathematical programming. New York: Springer; 2009.

[43] Nonås LM, Jörnsten K. Optimal solution in the multi-location inventory system with transshipments. J Math Model Algorithms 2007;6(1):47–75.

[44] Özer Ö. Inventory management: information, coordination, and rationality. In: Kempf KG, Keskinocak P, Uzsoy R, editors. Planning production and inventories in the extended enterprise, International series in operations research & management science, vol. 151. New York: Springer; 2011. p. 321–65.

[45] Paterson C, Kiesmüller G, Teunter R, Glazebrook K. Inventory models with lateral transshipments: a review. Eur J Oper Res 2011;210(2):125–36.

[46] Pepin A-S, Desaulniers G, Hertz A, Huisman D. A comparison of five heuristics for the multiple depot vehicle scheduling problem. J Sched 2009;12(1):17–30.

[47] Powell WB. Approximate dynamic programming, Wiley series in probability and statistics. New Jersey: Wiley; 2007.

[48] Psaraftis HN. Dynamic vehicle routing problems. In: Golden BL, Assad AA, editors. Vehicle routing: methods and studies. Amsterdam: North-Holland; 1998. p. 223–48.

[49] 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⟩.

[50] Rey D, Neuhäuser M. Wilcoxon-signed-rank test. In: Lovric M, editor. International encyclopedia of statistical science. Berlin: Springer; 2011. p. 1658–9.

[51] Ropke S, Pisinger D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp Sci 2006;40(4):455–72.

[52] Shaw P. A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, University of Strathclyde, Glasgow; 1997.

[53] Silver EA, Pyke DF, Peterson R. Inventory management and production planning and scheduling. New York: Wiley; 1998.

[54] Solyalı O, Süral H. A branch-and-cut algorithm using a strong formulation and an a priori tour based heuristic for an inventory-routing problem. Transp Sci 2011;45(3):335–45.

[55] Toriello A, Nemhauser GL, Savelsbergh MWP. Decomposing inventory routing problems with approximate value functions. Nav Res Logist 2010;57(8):718–27.