# Combinatorial Optimization and Machine Learning for Dynamic Inventory Routing

#### Toni Greif

Julius-Maximilians-Universität Würzburg, Germany toni.greif@uni-wuerzburg.de

#### Louis Bouvier

CERMICS, École des Ponts, Marne La Vallée, France, louis.bouvier@enpc.fr

#### Christoph M. Flath

Julius-Maximilians-Universität Würzburg, Germany christoph.flath@uni-wuerzburg.de

#### Axel Parmentier

CERMICS, École des Ponts, Marne La Vallée, France, axel.parmentier@enpc.fr

#### Sonja U. K. Rohmer

Department of Logistics and Operations Management, HEC Montréal, Canada, sonja.rohmer@hec.ca

#### Thibaut Vidal

CIRRELT & SCALE-AI Chair in Data-Driven Supply Chains, MAGI, Polytechnique Montréal, thibaut.vidal@polymtl.ca

We introduce a combinatorial optimization-enriched machine learning pipeline and a novel learning paradigm to solve inventory routing problems with stochastic demand and dynamic inventory updates. After each inventory update, our approach reduces replenishment and routing decisions to an optimal solution of a capacitated prize-collecting traveling salesman problem for which well-established algorithms exist. Discovering good prize parametrizations is non-trivial; therefore, we have developed a machine learning approach. We evaluate the performance of our pipeline in settings with steady-state and more complex demand patterns. Compared to previous works, the policy generated by our algorithm leads to significant cost savings, achieves lower inference time, and can even leverage contextual information.

Key words: Dynamic inventory routing, Stochastic demand, Combinatorial optimization, Machine learning

# 1. Introduction

The Inventory Routing Problem (IRP) is a challenging Combinatorial Optimization (CO) problem which simultaneously considers routing, inventory holding and stock-out costs. This problem is of high relevance for various real-world applications in transportation, logistics, and supply chain management. There has been extensive research on the deterministic IRP, resulting in various solution methods and heuristics. However, inventory management applications are typically characterized by different uncertainty sources which in turn affect different components of the IRP. In this study, we focus on uncertainty within the customer demands, assuming that the demands of the current period only become known once the replenishment decisions have been made. As such, we consider both the stochasticity of demand as well as the dynamic nature of the problem

requiring continuous inventory updates at the customer locations. Consequently, the problem state, which includes the inventory levels, historical demand observations, and possibly predictive contextual information for each customer, is consistently updated after each period. This combination of stochastic customer demands and inventory updates leads to the Dynamic and Stochastic Inventory Routing Problem (DSIRP), which may lead to stock-outs even under optimal decision-making (Coelho, Cordeau, and Laporte 2014a).

The goal of this paper is to introduce a learning-based policy for the DSIRP, and benchmark it against classic rolling-horizon policies based on single or multiple demand scenarios. Single-scenario policies are well established and easy to apply, but typically lack performance when demand patterns get more complex. Multi-scenario policies overcome these performance issues. However, their computational complexity increases rapidly with the number of scenarios and customers considered, which makes them impractical even for small instances. Our learning-based approach aims to combine the strengths of CO and Machine Learning (ML) in a hybrid pipeline to achieve optimal decision-making under uncertainty.

As a result, our research represents a pioneering effort in addressing the DSIRP using advanced algorithms. Vis-a-vis traditional solution approaches, our learning-based pipeline offers strong performance across a large number of problem configurations while retaining short inference times. This combination enables robust real-time decision-making in complex DSIRP settings. The strength of our pipeline lies in its resilience and adaptability. This is because it operates without being affected by the underlying demand distribution but instead relies solely on historical samples. Moreover, our approach has the advantage of effectively leveraging contextual information, which sets it apart from other algorithms that either only have limited capabilities to incorporate such information or do not scale well in terms of computational complexity. We provide the instances used in our research to support further studies on contextual demand in the DSIRP literature. In addition, while the potential of integrating hybrid ML and CO pipelines has been highlighted in simplified settings, our work is among the first attempts, together with a few notable exceptions, to address a complex IRP using such methods. By thoroughly analyzing the strengths and limitations of our approach, we offer guidance to both practitioners and researchers operating in this field.

This paper provides insights into the performance, robustness, and real-time decision-making capabilities of the presented methods, shedding light on the inherent trade-offs of selecting an appropriate approach to effectively solve the DSIRP for real-world scenarios. Focusing on a simplified setting, our numerical experiments allow us furthermore to evaluate the effectiveness of the most complex approaches in finding optimal solutions. Moreover, by isolating the heuristic solutions from other confounding factors, we are able to analyze and compare the relative strengths of these approaches.

We refer to our git repository https://github.com/tonigreif/InferOpt\_DSIRP.git for all the material necessary to reproduce the results outlined in this paper.

# 2. Related Work

Contributing to two research directions, we first review the IRP and Dynamic Vehicle Routing Problem (DVRP) literature before investigating the potential of ML-based approaches to address stochastic and dynamic problems. In the latter, we focus on the concept of *CO-enriched ML*, which presents a hybrid pipeline where ML models learn from many similar CO problems.

#### 2.1. Inventory Routing

The IRP is a well-known problem for which many extensions and modifications have been studied with the aim of representing a large variety of different real-world situations (Coelho, Cordeau, and Laporte 2014b). This has resulted in an abundance of solution approaches for the deterministic IRP, including exact methods such as branch-and-cut (Adulyasak, Cordeau, and Jans 2014, Archetti et al. 2007, Avella, Boccia, and Wolsey 2018, Bertazzi et al. 2019, Guimarães et al. 2019, Coelho and Laporte 2013a,b, Manousakis et al. 2021) and branch-and-price-and-cut (Desaulniers, Rakke, and Coelho 2016), as well as heuristics and metaheuristics (Adulyasak, Cordeau, and Jans 2014, Archetti et al. 2012, Archetti, Boland, and Grazia Speranza 2017, Bertazzi et al. 2019, Chitsaz, Cordeau, and Jans 2019, Guimarães et al. 2019, Bouvier et al. 2023+).

Given the importance of uncertainty in real-world settings, there is an increasing need to reconsider the deterministic nature of IRPs and adequately account for stochasticity when solving such problems. The review of Soeffker, Ulmer, and Mattfeld (2022) identifies in this context three key problem dimensions that may be affected by uncertainty: i) demand (requests, quantities, service time, etc.), ii) resources (vehicle availability, driver availability, range, etc.), and iii) the environment (travel times, fees, road closures, etc.). Arising in almost all practical settings, demand uncertainty is one of the most common sources of uncertainty considered in the scientific literature. Accordingly, this research will also focus on incorporating demand uncertainty and the most relevant studies in this area. A common approach to deal with demand uncertainty without information on the underlying probability distribution is to estimate customer demands based on historical samples. A straightforward strategy for addressing these CO problems is selecting multiple (demand) scenarios from these historical samples and formulating the Mixed Integer Linear Program (MILP). This approach involves introducing what we refer to as first-period linkage constraints. These constraints play a crucial role during the problem-solving phase, ensuring the uniqueness of the first-period routing decision across all scenarios. To handle the considerable increase in size and complexity, consensus optimization frameworks avoid using first-period linkage constraints. Instead, they decompose the *global consensus problem* into independent subproblems, iterating to achieve consensus on the solution (Nedic and Ozdaglar 2010, Boyd et al. 2011, Bertsekas and Tsitsiklis 2015). Under this paradigm, multiple IRP scenarios are solved independently, and consensus strategies like progressive hedging are employed to merge divergent routing solutions into a single decision (Hvattum and Løkketangen 2009).

The least complex option involves consolidating multiple scenarios into a single scenario before solving the problem, as demonstrated in the case of the DSIRP by Coelho, Cordeau, and Laporte (2014a). This option is often employed when the single-scenario problem is intricate in itself. As this approach may lack effectiveness, the study by Brinkmann, Ulmer, and Mattfeld (2019) proposes simulation techniques that involve multiple scenarios. In addition, different heuristics were proposed to solve this type of problem; examples of some notable methods, in this context, include ant colony optimization algorithms (Huang and Lin 2010), variable neighborhood search metaheuristic hybridized with simulation (Gruler et al. 2018), a hybrid rollout algorithm (Bertazzi et al. 2013), and benders decomposition (Li and Jiao 2022).

In addition to stochastic considerations, uncertainty may add dynamic considerations to a problem as the initial stochastic information is updated or revealed over time. Integrating dynamic aspects in IRPs is relatively new and has not been widely explored within the literature. For this reason we present a more general overview of dynamic vehicle routing problems. Examples of these problems in different routing contexts are provided by Ritzinger, Puchinger, and Hartl (2016), Gendreau, Jabali, and Rei (2016), Psaraftis, Wen, and Kontovas (2016), Oyola, Arntzen, and Woodruff (2018). A common strategy for solving these problems is the rolling-horizon approach, where a static solution is continually reoptimized as new information becomes available (Berbeglia, Cordeau, and Laporte 2010, Wong and Bell 2006, Bouvier and Parmentier 2023).

#### 2.2. Decision-Making with Machine Learning

In recent years, the use of ML techniques for decision-making has received growing interest in the scientific literature. Several studies highlight the potential of such techniques in addressing DVRPs (Joe and Lau 2020, James, Yu, and Gu 2019). Hildebrandt, Thomas, and Ulmer (2023) emphasize, in particular, the potential of Reinforcement Learning (RL) for two compelling reasons: First, the ability to model dynamic CO problems as Markov Decision Processes (MDP) (Powell 2019). Second, offline training of learning architectures to devise complex decision policies with short inference times. However, Hildebrandt, Thomas, and Ulmer (2023) also point out that ML techniques face challenges when dealing with complex, combinatorial action spaces. In such scenarios, the search for the action space is better performed by available CO solvers. Hence, there is untapped potential

in combining ML with CO within a hybrid pipeline. Such an approach can leverage the strengths of both paradigms and holds promise for developing combinatorial solutions while accounting for uncertainties.

These pipelines, known as CO-enriched ML pipelines, have demonstrated remarkable success in solving problems that were challenging for classic CO approaches. More specifically, they have shown excellent performance in addressing problems like single machine scheduling with release dates (Parmentier and T'Kindt 2021) and various multi-stage optimization problems. Examples of successful multi-stage optimization problems include the parameterization of the two-stage stochastic minimum spanning tree problem (Dalle et al. 2022), the stochastic vehicle scheduling problem (Parmentier 2021, 2022), and even the dispatching policy in a dynamic autonomous mobility-on-demand system (Jungel et al. 2023). Additionally, this method has proven effective in solving high-dimensional multi-stage stochastic optimization problems like the DVRP with time windows (Baty et al. 2023).

# 3. Problem Description

This paper considers the DSIRP, a dynamic variant of the IRP where information on customer demands and inventory updates are revealed progressively. Customer demands are assumed to be random variables and may be interpreted as a retailer's total order for a given period.

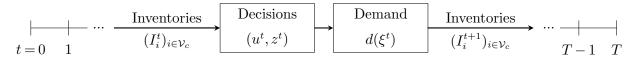


Figure 1 Process of making decisions, revealing demand, and updating inventories over time

The problem is defined over a given planning horizon, at the beginning of which each customer has an initial inventory. Inventory positions incur a customer-specific holding cost per period and unit. In the same fashion, inventory shortfalls result in customer-specific stock-out cost per unit of unsatisfied demand. We do not consider backlogging of unfulfilled demand. To determine the replenishment decisions for each customer we use an order-up-to policy which is widely used in IRP settings. Under this policy delivered quantities completely replenish the inventory capacity at the customer locations. Production capacity is unconstrained we do not account for inventory holding costs at the supplier. As common in the IRP literature, the distribution part of our problem consists of a single vehicle with a given capacity, which can perform one route per period. Each route starts at the supplier and covers a subset of customer locations, while traveling between locations incurs a travel cost. The problem's overall objective is to minimize the total routing cost

and the costs of inventory holding and stock-outs at the customer locations for the total *planning* horizon.

The problem can be defined on an undirected graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V} = \{0, ..., v\}$  presents the set of vertices and  $\mathcal{E} = \{(i, j) : i, j \in \mathcal{V}, i < j\}$  the set of feasible edges. Vertex 0 represents, in this context, the depot at which the supplier is located, and the vertices  $\mathcal{V}_c = \mathcal{V} \setminus \{0\}$  correspond to the customer locations.

#### 3.1. Markov Decision Process

To provide the necessary theoretical foundation for our learning-based pipeline, we formulate the DSIRP as a MDP. For this purpose, we define the vector  $I^t \in \mathbb{R}^{|\mathcal{V}_c|}_{\geq 0}$ , which represents the inventory levels of the customers at the beginning of period t, not taking into account the forthcoming realized demand or the replenishment activities. The inventory level of a specific customer i is then denoted by the element  $I^t_i$ . Furthermore, we denote the state before making a decision at period t as  $x^t$ . In the absence of additional contextual information,  $x^t$  corresponds to the customers' inventory levels at the beginning of the current period, along with the set of the most recent historical observations  $\hat{\mathcal{D}}^t = \{d^{t-1}, \dots, d^{t-50}\}$  for each customer  $i \in \mathcal{V}_c$ . However, when contextual information  $\Phi^t$  regarding the environment for period t is available, the state is augmented to  $x^t = (I^t, \hat{\mathcal{D}}^t, \Phi^t, \Phi^{t+1}, \dots)$  to encompass this supplementary information for the subsequent periods, with  $\Phi^t_i$  representing the context for customer i. We proceed from the premise that the context  $\Phi^t$  and the exogenous noise  $\xi^t$  capturing the uncertainty in demands  $d^t = d(\xi^t)$  are not only independent of each other, but also of all decisions made. This basic assumption ensures that we are in a Markovian setting.

We then denote by  $\mathcal{T}$  the set of tours (without subtours) that start and end in vertex 0. A tour can be encoded by a vector  $(u_{ij}^t)_{(i,j)\in\mathcal{E}}$ , where  $u_{ij}^t=1$  if edge (i,j) belongs to the tour and 0 otherwise. To encode if customer i belongs to the tour or not, we define

$$z^{t} = g(u^{t}) = \frac{1}{2} \left( \sum_{j \in \mathcal{V}, i > j} u_{ji}^{t} + \sum_{j \in \mathcal{V}, i < j} u_{ij}^{t} \right)_{i \in \mathcal{V}_{c}}.$$
 (1)

When the process occupies state  $x^t$ , the set of feasible decisions is then given by

$$\mathcal{U}(x^t) = \left\{ u^t \in \mathcal{T} \middle| \sum_{i \in \mathcal{V}_c} \left( C_i - I_i^t \right) z_i^t \le B \right\},\tag{2}$$

indicating that we can replenish all subsets of customers whose combined replenishment quantities, determined by the difference between inventory capacity  $C_i$  and current inventory  $I_i^t$ , are smaller than the vehicle capacity B. Furthermore, the calculation of routing costs is defined by

$$h(u^t) = \sum_{(i,j)\in\mathcal{E}} \gamma_{ij} u_{ij}^t, \tag{3}$$

where  $\gamma_{ij}$  are the usage costs of edge  $(i,j) \in \mathcal{E}$ . Given state  $x^t$  with inventory levels  $I^t$ , each decision is associated with a (negative) reward equal to its respective holding, stock-out, and routing costs:

$$\tilde{r}(x^t, u^t, \xi^t) = \sum_{i \in \mathcal{V}_c} \kappa_i \left\{ I_i^t (1 - z_i^t) + C_i z_i^t - d(\xi^t)_i \right\}^+ - \rho \kappa_i \left\{ I_i^t (1 - z_i^t) + C_i z_i^t - d(\xi^t)_i \right\}^- + h(u^t), \quad (4)$$

where  $\kappa_i$  are the holding costs per unit and the shortage penalty  $\rho$  indicates the multiplier by which the holding costs influence the stock-out costs. Due to the absence of recourse actions in case a customer runs out of stock, the deterministic transition to the next state  $x^{t+1}$  corresponds to updating the customer inventory levels

$$I_i^{t+1} = \left\{ I_i^t (1 - z_i^t) + C_i z_i^t - d(\xi^t)_i \right\}^+, \tag{5}$$

as well as the historical demand observations and, if available, also the contextual information, resulting in

$$x^{t+1} = (\underbrace{I^{t+1}}_{\text{inventories historical demand contextual information}}, \underbrace{\Phi^{t+1}, \Phi^{t+2}, \dots}_{\text{inventories historical demand contextual information}}). \tag{6}$$

A deterministic policy  $\delta$  then maps a state  $x^t$  to a feasible decision  $u^t \in \mathcal{U}(x^t)$ . We seek a policy in the set of all Markovian deterministic policies  $\Delta$  that minimizes the expected total (negative) reward conditional on the initial state  $x^0$ 

$$\underset{\delta \in \Delta}{\operatorname{arg\,min}} \mathbb{E} \left[ \sum_{t=0}^{T-1} \tilde{r}(x^{t}, \delta(x^{t}), \xi^{t}) \middle| x^{0} \right]. \tag{7}$$

# 4. Policy Encoded as Machine Learning Pipeline

The challenge of the DSIRP arises due to the combinatorially vast nature of both the state space  $\mathcal{X}$  and the decision space  $\mathcal{U}$ . While classic stochastic optimization policies can handle large  $\mathcal{U}$  when  $\mathcal{X}$  is small, and reinforcement learning can handle large  $\mathcal{X}$  in the context of small  $\mathcal{U}$ , settings with both large  $\mathcal{X}$  and large  $\mathcal{U}$  are still poorly addressed. As a result, solution methods for DVRPs often rely on rolling-horizon policies as discussed in Section 2. Although these policies are practical and can account for uncertainty by considering multiple historical samples through voting or consensus strategies, computational limitations commonly restrict the number of scenarios that can be considered, reducing their effectiveness.

To overcome these limitations, we propose a hybrid pipeline for encoding policies. The pipeline chains a statistical model  $\varphi_w$  with a standard CO problem for which well-established algorithms exist. We refer to this solution algorithm as oracle o. The oracle must share the same set of feasible solutions  $\mathcal{U}(x^t)$  and be parameterizable. With fixed parameters  $\theta$  in each state  $x^t$ , the oracle's decision  $u^t$  becomes a deterministic policy. This pipeline (Figure 2) is a CO-enriched ML pipeline.

It encodes a family of policies parametrized by w, which denotes the parameters of the statistical model  $\varphi_w$  to predict  $\theta$  given a state  $x^t$ . An important task in this context is choosing the parameters w such that the pipeline leads to a practically efficient policy. For this purpose, we introduce the learning algorithm, which is described in Section 5.

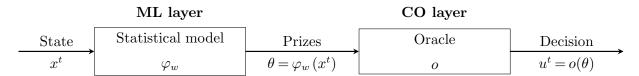


Figure 2 Our CO-enriched ML pipeline.

In the following subsections, we will first introduce and explain our choice for the oracle o, formulating it as a Capacitated Prize Collecting TSP (CPCTSP), and then our statistical model  $\varphi_w$ , a Physics-informed Neural Network (PINN).

#### 4.1. Combinatorial Optimization Layer

The CPCTSP is a deterministic vehicle routing problem variant in which customer deliveries are optional but rewarded by a prize. It aims to select a subset of the customers along with good delivery routes to maximize the sum of rewards minus transportation costs (distance). There exists close connections between this problem and the DSIRP since the set of feasible decisions  $\mathcal{U}(x^t)$ , at each state  $x^t$ , consists of tours that visit a subset of customers whose combined replenishment quantities do not exceed the vehicle capacity, as stated in Equation (2). This set aligns with the feasible solutions of a CPCTSP, parameterized by prizes  $\theta$  representing how desirable deliveries are to the different locations at the considered period. Moreover, solving a single-period CPCTSP is less complex than a multi-period IRP and benefits from well-established algorithms. As such, we define our problem as follows. Let  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  be the undirected graph defined in Section 3. Each vertex  $i \in \mathcal{V}_c$  has an associated prize  $\theta_i$ . For each edge (i,j) in  $\mathcal{E}$ , we have a travel cost  $\gamma_{ij} \geq 0$ . The task is then to determine the customers to be visited, by constructing a tour that begins at a depot. The goal is to maximize profit, calculated as the sum of the collected prizes minus the travel costs incurred on the traversed edges. Using Equation (1), the resulting CPCTSP can be formulated as

$$\max_{u \in \mathcal{U}} \sum_{i \in \mathcal{V}_c} \theta_i z_i^t - \sum_{\substack{(i,j) \in \mathcal{E} \\ \text{prizes collected}}} \gamma_{ij} u_{ij}^t.$$
 (8)

It is important to note that the routing costs are well-defined within our DSIRP context, but the prizes are not. However, we demonstrate that there always exists a prize vector  $\theta$  such that the optimal solution of the CPCTSP aligns with the optimal decision for the DSIRP.

**Proposition 1** For every state  $x^t$  there exists a prize vector  $\theta \in \mathbb{R}^{|\mathcal{V}_c|}$  such that any optimal solution of the CPCTSP (8) corresponds to an optimal decision in terms of the expected total (negative) reward of the MDP (7).

*Proof.* Since the planning horizon is finite and the set of feasible decisions at each period is also finite, there exists an optimal decision  $u^*$  for  $x^t$ . Let  $\bar{\mathcal{V}}_c$  be the subset of customers of  $\mathcal{V}_c$  that are replenished in  $u^*$ . Then any solution u which has lower or equally lower routing costs and covers  $\bar{\mathcal{V}}_c$  exactly is also optimal. This follows from the Bellman equation since the routes have no impact on the evolution of the state. We can construct u by solving a CPCTSP on  $\mathcal{V}_c$  with prizes denoted by

$$\theta_i = \begin{cases} +M & \text{if } i \in \bar{\mathcal{V}}_c, \\ -M & \text{otherwise,} \end{cases}$$
 (9)

where  $M = |\mathcal{V}_c| \cdot \max_{(i,j) \in \mathcal{E}} \gamma_{ij}$  is a large constant. The corresponding CPCTSP solution u clearly covers  $\bar{\mathcal{V}}_c$  and has at most the routing costs of  $u^*$ .

## 4.2. Machine Learning Layer

The goal of the ML layer is to predict for any state  $x^t$  a prize vector  $\theta$  in such a way that optimal solutions of the resulting CPCTSP align with good solutions of the DSIRP for this state.

To predict the prizes  $\theta$ , we design a PINN  $\varphi_w$ , which maps each state  $x^{\tau}$ , including the set of historical observations  $\hat{\mathcal{D}}_i^{\tau}$  for every customer  $i \in \mathcal{V}_c$ , to their corresponding prize values  $\theta_i$ . The PINN leverages our understanding of the cost structure and effectively addresses data scarcity concerns. For each quantile level  $p \in \mathcal{P}$ , we preprocess  $\hat{\mathcal{D}}_i$  into demand quantiles given by

$$Q_p(\hat{\mathcal{D}}_i) = \inf \left\{ d \in \hat{\mathcal{D}}_i : F(d) \ge p \right\}, \tag{10}$$

where F(d) refers to the cumulative distribution function of the set  $\hat{\mathcal{D}}_i$ . The first layer, denoted as  $\varphi^1$ , acts as a demand estimation layer, tailoring the projection of future demand to the contextual information  $\Phi_i$  for look-ahead horizons  $\mathcal{H} = \{0, ..., H-1\}$ . In our case, this corresponds to a linear combination of the individual features and their pairwise interactions as stated by

$$\varphi_p^1 \left( \hat{\mathcal{D}}_i, \Phi_i \right) = \text{ReLU} \left( Q_p(\hat{\mathcal{D}}_i) + w_1^\top \Phi_i + \Phi_i^\top w_2 \Phi_i \right), \tag{11}$$

where  $w_1$  is a vector, and  $w_2$  is a symmetric matrix with diagonal elements equal to zero. Subsequently, two parallel layers come into play to estimate the holding (12) and stock-out costs (13):

$$\varphi^{2} = \sum_{p \in \mathcal{P}} w_{3}^{p \top} \left( \operatorname{ReLU} \left( I_{i}^{\tau} - \sum_{t=\tau}^{\tau+h} \varphi_{p}^{1} \left( \hat{\mathcal{D}}_{i}^{\tau}, \Phi_{i}^{t} \right) \right) \kappa_{i} \right)_{h \in \mathcal{H}}$$

$$(12)$$

$$\varphi^{3} = \sum_{p \in \mathcal{P}} w_{4}^{p \top} \left( \text{ReLU} \left( \sum_{t=\tau}^{\tau+h} \varphi_{p}^{1} \left( \hat{\mathcal{D}}_{i}^{\tau}, \boldsymbol{\varPhi}_{i}^{t} \right) - I_{i}^{\tau} \right) \kappa_{i} \rho \right)_{h \in \mathcal{H}}$$

$$(13)$$

The resulting multi-layer statistical model  $\varphi_w = \varphi^2 + \varphi^3$ , with parameters  $w = \{w_1 \in \mathbb{R}^{|\Phi_i|}, w_2 \in \mathbb{R}^{|\Phi_i| \times |\Phi_i|}, w_3 \in \mathbb{R}^{|\mathcal{H}| \times |\mathcal{P}|}, w_4 \in \mathbb{R}^{|\mathcal{H}| \times |\mathcal{P}|}\}$ , is illustrated in Figure 3. This highly tailored design is justified by the inherent characteristics of the DSIRP's piecewise linear cost function, emerging from the distinct impacts of holding and stock-out costs, as inventory transitions from positive to negative. The statistical model  $\varphi_w$  can theoretically be replaced with any differentiable ML model. However, for the sake of interpretability, we make the choice to employ a statistical model where the core of layers  $\varphi_p^1, \varphi^2, \varphi^3$  is a generalized linear model offering streamlined training and involving a moderate number of interpretable parameters.

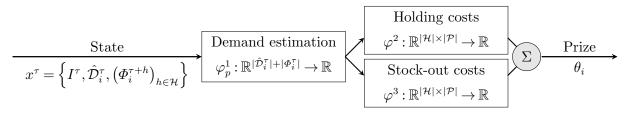


Figure 3 Layers of our statistical model.

# 5. Learning Algorithm

We now present our learning algorithm tailored to train policies for MDPs with large state and action spaces  $\mathcal{X}$  and  $\mathcal{U}$  of the following form:

$$\delta_w(x^t) = \underset{u \in \mathcal{U}(x^t)}{\arg \max} \theta^{\top} g(u) + h(u), \quad \text{where} \quad \theta = \varphi_w(x^t).$$
 (14)

The learning algorithm is designed to select the parameters w that result in an efficient policy  $\delta_w$ :  $x_t \mapsto u_t$ , where efficient is defined as minimizing the expected total (negative) reward. Section 5.1 introduces an abstract MDP setting, which clarifies the description of our learning algorithm, and highlights the main hypotheses needed to make it work. Section 5.2 introduces the imitated policy, and Section 5.3 formulates the learning problem. Next, Section 5.4 defines a suitable loss function for training, and Sections 5.5 and 5.6 discuss how to simultaneously train the model and include additional training samples with relevant states.

## 5.1. Setting

First, let's establish the generic setting to which our learning algorithm may be applied before delving into the specificities of our problem. For this purpose, we look at a generic MDP for which transitions (15) and rewards (16) are deterministic functions of the state  $x^t$ , the decision  $u^t$ , and some exogenous noise  $\xi^t$ .

$$x^{t+1} = f(x^t, u^t, \xi^t)$$
 (15)

$$r_t = \tilde{r}(x^t, u^t, \xi^t). \tag{16}$$

By exogenous, we mean that  $\xi^t$  is independent of the decisions taken. In other words, we consider an MDP of the following form:

$$\min_{\delta} \mathbb{E} \left[ \sum_{t=0}^{T-1} \tilde{r} \left( x^t, \delta(x^t), \xi^t \right) + \tilde{r}_T(x^T) \middle| x^0 \right] 
\text{s.t. } u^t = \delta(x^t) \text{ and } (15) \ \forall t \in \{0, \dots, T\}.$$
(17)

An episode then denotes a sequence  $\boldsymbol{\xi} = (\xi^0, \dots, \xi^{T-1})$  of noises. Given decisions  $u^0, \dots, u^{T-1}$ , we can compute the corresponding trajectory  $(x^0, u^0, \xi^0), \dots, (x^{T-1}, u^{T-1}, \xi^{T-1})$  followed by the system. To train our pipeline, we then require access to several episodes  $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_n$  of historical data, and an algorithm for the deterministic problem that arises from (17) when the distribution over  $\boldsymbol{\xi}$  is a Dirac, i.e., we have a single episode.

In the case of our manuscript, this generic setting takes a specific form as outlined in Section 3. Here,  $\xi^t$  represents the uncertainty in demands  $d(\xi^t)$ ,  $u^t$  corresponds to routing decisions, and  $x^t$  encapsulates the inventory, historical demand, and contextual information. The rewards (4) and transitions (6) are deterministic, and  $\tilde{r}_T = 0$ .

# 5.2. Imitated Policy

Since our training set contains only episodes  $\boldsymbol{\xi}_i = (\xi_i^0, \dots, \xi_i^{T-1})$ , we need to add decisions and turn these episodes into trajectories  $(x_i^0, u_i^0, \xi_i^0), \dots, (x_i^{T-1}, u_i^{T-1}, \xi_i^{T-1})$ . For that purpose, we determine the anticipative decisions  $\tilde{\boldsymbol{u}}^{\tau}(x^{\tau}; \boldsymbol{\xi}) = (\tilde{u}^t(x^{\tau}; \boldsymbol{\xi}))_{t \in [\tau, T-1]}$ . Given the current state  $x^{\tau}$  at time  $\tau$  and an episode  $\boldsymbol{\xi}$  for the rest of the time horizon,  $\boldsymbol{\xi}^{\tau}, \dots, \boldsymbol{\xi}^{T-1}$ , it is the solution of the following deterministic problem.

$$\tilde{\boldsymbol{u}}^{\tau}(x^{\tau};\boldsymbol{\xi}) \in \underset{u^{\tau},\dots,u^{T-1}}{\operatorname{arg\,min}} \sum_{t=\tau}^{T-1} \tilde{r}(x^{t}, u^{t}, \xi^{t}) + \tilde{r}_{T}(x^{T})$$
s.t.  $x^{t} = f(x^{t-1}, u^{t-1}, \xi^{t-1})$ , for all  $t \in \{\tau + 1, \dots, T\}$ 

A tailored algorithm is needed to solve this deterministic problem. In our case, this corresponds to a deterministic IRP, which we solve with an adaptation of the exact algorithm of Manousakis et al. (2021) that accommodates stock-outs. Any other available (heuristic or exact) algorithm could be used for this purpose. At time  $\tau$ , given the anticipative decisions  $\tilde{\boldsymbol{u}}^{\tau}(x^{\tau};\boldsymbol{\xi}) = (\tilde{u}^{t}(x^{\tau};\boldsymbol{\xi}))_{t\in[\tau,T-1]}$ , our anticipative policy corresponds to the first time-step  $\tilde{u}^{\tau}(x^{\tau};\boldsymbol{\xi})$  as defined below.

## 5.3. Learning Problem

Let us now consider the anticipative policy, denoted by

$$\delta_t^*(x^t; \boldsymbol{\xi}) \mapsto \tilde{u}^t(x^t; \boldsymbol{\xi}).$$

This policy  $\delta^*$  cannot be applied in practice since it requires information not available at time t. However, it can be recomputed a posteriori given an episode  $\boldsymbol{\xi}$ . Our learning algorithm trains our policy  $\delta_w$ , which uses only information available at time t, to imitate  $\delta^*$ . In order to train a policy by imitation, we minimize a surrogate loss  $\ell(u,\bar{u})$  that penalizes taking a decision u that does not correspond to the target decision  $\bar{u}$ . Given our pipeline, the decision taken by  $\delta_w$  is a deterministic function. Therefore, we can consider a loss  $\mathcal{L}(\theta,\bar{u})$  as a function of  $\theta$  and  $\bar{u}$ . When we train a policy  $\delta_w$  to imitate  $\delta^*$ , our objective is to find parameters w that minimize an expected surrogate loss under their induced distribution of state

$$\min_{w} \mathbb{E}_{\boldsymbol{\xi}, X \sim d_{w}} \left[ \mathcal{L} \left( \varphi_{w}(X), \delta^{*}(X; \boldsymbol{\xi}) \right) \right], \tag{19}$$

where  $d_w$  is the distribution on X induced by w, where we have aggregated the different time periods. Practically, we do not know how to evaluate the expectation presented in (19). As such, we iteratively construct a training set  $\mathcal{D}$  containing states x sampled from  $\delta_w$  and update the algorithm by solving the *empirical imitation learning problem* given by

$$\min_{w} \sum_{(x,\bar{u})\in\mathcal{D}} \mathcal{L}(\varphi_w(x),\bar{u}). \tag{20}$$

#### 5.4. Fenchel-Young Loss

Next, we describe the surrogate loss  $\mathcal{L}$  applied in our pipeline. Suppose we have a training set  $(x^1, \bar{u}^1), \ldots, (x^n, \bar{u}^n)$  composed of states  $x^k$  and their target decisions  $\bar{u}^k$ . We want our policy (14) to output  $\bar{u}^k$  when taking  $x^k$  as input. In other words, we want  $\bar{u}^k$  to be an optimum of the CO-layer. Consequently, it is natural to use as loss the non-optimality of  $\bar{u}^k$  as a solution of the CO-layer:

$$\max_{u \in \mathcal{U}} \theta^{\top} g(u) + h(u) - \theta^{\top} g(\bar{u}^k) - h(\bar{u}^k) \quad \text{where} \quad \theta = \varphi_w \left( x^k \right).$$

Based on this, we obtain a loss that is more robust to degeneracy when we perturb  $\theta$  with a standard Gaussian random variable Z, leading to the Fenchel-Young loss<sup>1</sup>

$$\mathcal{L}(\theta, \bar{u}^k) = \mathbb{E}\left[\max_{u \in \mathcal{U}} (\theta + Z)^\top g\left(u\right) + h\left(u\right) - \theta^\top g(\bar{u}^k) - h(\bar{u}^k)\right].$$

Several properties of this loss function are presented in Baty et al. (2023). For our purposes, we need to underline that it is strongly convex and that its gradient is given by,

$$\nabla_{\theta} \mathcal{L}(\theta, \bar{u}^k) = \mathbb{E} \left[ g \left( o(\theta + Z) \right) \right] - g(\bar{u}^k), \tag{21}$$

where o is the CO-layer's oracle that associates to  $\theta$  an optimal solution of  $\max_{u \in \mathcal{U}} \theta^{\top} g(u) + h(u)$ . Since we have such an oracle o by hypothesis, we can compute a stochastic gradient by sampling on Z. Stochastic gradients in w can, in this case, be computed by backpropagation (21) using automatic differentiation. The convexity of  $\mathcal{L}$  ensures that the empirical imitation learning problem (20) is convex when  $\varphi_w$  is convex.

<sup>&</sup>lt;sup>1</sup> Omitting the Fenchel conjugate term.

#### 5.5. Dataset Generation using DAgger Algorithm

To generate the training dataset  $\mathcal{D}$ , we use the DAgger algorithm proposed by Ross, Gordon, and Bagnell (2011). It is designed to ensure that the empirical learning problem (20) provides a good solution to problem (19). The general idea of the algorithm is to draw trajectories according to a (mixture) policy given by

$$\alpha \delta^* + (1 - \alpha) \delta_w, \tag{22}$$

where  $\alpha$  is a mixture parameter. Given an episode  $\boldsymbol{\xi} = (\xi^0, \dots, \xi^{T-1})$ , a trajectory from policy (22) can be sampled as follows. For t going from 0 to T-1, we select  $\delta^*$  with probability  $\alpha$ . If so, we solve the deterministic problem (18) to take decision  $u^t = \tilde{u}^t(x^t; \boldsymbol{\xi})$ . Otherwise, we set  $u^t = \delta_w(x^t)$ . We then compute  $x^{t+1}$  using the deterministic transition function (15).

# **Algorithm 1** DAgger with anticipative policy

- 1:  $\alpha_1, \ldots, \alpha_K$  a decreasing sequence of weights in  $[0, 1], \mathcal{D} = \emptyset, w$  undefined.
- 2: **for** i = 1, ..., K **do**
- 3: Sample  $\boldsymbol{\xi}_i$ , initial state  $x^0$ .
- 4: **for** t = 0, ..., T 1 **do**
- 5: Transition to  $x^{t+1}$  from policy (22) with  $\alpha_i$  and function (15).
- 6: Add  $(x^t, \tilde{u}^t(x^t; \boldsymbol{\xi}_i))$  to  $\mathcal{D}$ .
- 7: end for
- 8: Update the parameters w solving learning problem (20) with  $\mathcal{D}$ .
- 9: end for

Algorithm 1, referred to as Anticipative-DAgger, starts with a collection of states generated by  $\delta^*$  because  $\delta_w$  is an inferior policy at the beginning of the training algorithm. Iteration after iteration,  $\delta_w$  improves, and we can therefore give more and more weight to  $\delta_w$  in the mixture. In the last iterations, states come mostly from  $\delta_w$ , which is aligned with the use of the induced distribution  $d_w$  in (19).

An alternative would be the approach proposed in Baty et al. (2023), where the data set of states and target decisions to imitate is defined a priori, i.e., at the beginning. Following Baty et al. (2023)'s learning paradigm, we sample initial states (with focus on inventories), calculating the anticipative decision (18) for  $\tau$  once and adding all pairs  $(x^{\tau}, u^{\tau}), \dots, (x^{T-1}, u^{T-1})$  to  $\mathcal{D}$ . However, adding all pairs to the training set poses significant limitations in our IRP setting. Given the absence of an incentive to maintain inventory beyond period T-1, this triggers a reduction in delivered quantities during the final periods (as shown in Figure 5 in our computational experiments), which

biases the decisions. Consequently, we modify the learning paradigm of Baty et al. (2023) and include only the first-period pair  $(x^{\tau}, u^{\tau})$  of state and anticipative decision to  $\mathcal{D}$ . In this approach, termed the *Sampling* learning paradigm, states rely exclusively on the sampling strategy for initial states. This reliance may potentially lead to a lack of generalization.

Considering various look-ahead horizons between periods 1, ..., T-1 or even beyond T-1, to avoid the end-of-horizon effect, is theoretically possible. However, this implies a trade-off between the end-of-horizon effect, the impact of the sampling strategy for the initial states, and the computational time for the anticipative decision. These considerations make determining the best and most efficient look-ahead horizon challenging. The *Anticipative-DAgger* learning paradigm works well in practice for our application.

## 5.6. Voting Policy

Imitating the anticipative policy described in Section 5.2 may seem surprising at first glance since such policies are known to be often sub-optimal. An alternative would be to follow a voting policy, which consists in sampling several trajectories from any current state, solving them independently, and making them "vote" on the first decision by selecting the most frequent. Such an approach is generally more computationally demanding but leads to better policies. However, it cannot be applied to our setting due to the combinatorial size of the decision space; since the same decisions are highly unlikely to appear twice for different trajectories, a vote based on the highest frequency cannot be done. Nevertheless, to achieve a similar effect, we can replace step 6 of Algorithm 1 by the following steps, where M is the number of trajectories used for the vote.

## Algorithm 2 DAgger with voting policy modifications

Sample M trajectories  $\tilde{\boldsymbol{\xi}}_j$  for  $j \in \{1, \dots M\}$ .

Add  $(x^t, \tilde{u}^t(x^t; \boldsymbol{\xi}_j))$  to  $\mathcal{D}$  for all  $j \in \{1, \dots M\}$ .

The numerical experiments show that this improvement of Algorithm 1 is computationally expensive during training but improves the performance of the policy learned. Across all learning paradigms we employ a common ML technique known as early stopping. Using validation episodes  $(\bar{\boldsymbol{\xi}}_j = (\bar{\xi}_j^0, \dots, \bar{\xi}_j^{T-1}))_{j \in \{1,\dots,5\}}$  and an initial state  $x^0$ , we halt the training of our algorithm if there is no improvement in our current best parameters  $w^*$  in terms of the total costs  $\sum_{j=1}^{5} \sum_{t=0}^{T-1} \tilde{r}(x^t, \delta_{w^*}(x^t), \bar{\xi}_j^t)$ , with transitions (15). In addition, for both DAgger-like learning paradigms, we apply a reduction and subsampling strategy to the dataset. In general, we only consider the samples of the last ten epochs and furthermore select only 50% of the state-decision pairs from the past epochs.

# 6. Computational Experiments

All our experiments have been conducted using the Gurobi Optimization, LLC (2023) software as our MILP solver for the CPCTSPs (8) and the deterministic IRPs needed to generate anticipative decisions (18). For training, we used a heterogeneous CPU cluster for increased computational capacity. Afterward, the trained pipeline and all the other benchmark methods were evaluated on a single thread of an AMD EPYC 7713P 64-core CPU.

## 6.1. Instance Design

Instances from prior studies (e.g., Coelho, Cordeau, and Laporte 2014a) are limited in the characteristics of their demand scenarios. For a thorough experimental evaluation, we augmented these instances with additional demand scenarios and aligned their other parameters with those featured in other IRP datasets (Archetti et al. 2007, 2011, Coelho, Cordeau, and Laporte 2014a). These parameters include initial and maximum inventories, vehicle capacity, geographical locations, travel costs, and costs associated with inventory holding and stock-outs at customer locations. Furthermore, we assume an infinite production capacity and do not account for inventory holding costs at the supplier.

For an evaluation that permits even the most complex baseline methods (e.g., Sample Average Approximation (SAA)-based approaches) to produce solutions, we generated ten instances of each demand pattern with ten customer locations each. As explained in Section 4.2, we generated a set of 50 historical demand observations to enable initial estimates of demand quantiles. Moreover, we constructed two fixed sets of demand observations for each customer dedicated to validation and evaluation. Our final evaluation covers a horizon spanning ten periods, enabling the SAA-3 policy and anticipative baseline solutions to discover optimal solutions.

We also generate instances that have more varied demand characteristics than previous works focused on *normal* and *uniform* distribution. To that end, we incorporated instances featuring bimodal demand distributions. Additionally, we generate instances with contextual demand to gauge the effectiveness of our approach in handling contextual information. For these contextual demand scenarios, we introduced features that influence demand, a level of control not typically available with real datasets where assumptions about the underlying contextual effect must be made. All the specificities of these instances are thoroughly discussed in Appendix A.

#### 6.2. Benchmark Policies

Many previous approaches are limited in their scope of applicability. Some methods assume a normal demand distribution (Huang and Lin 2010), whereas others are designed exclusively for discrete demand scenarios (Li and Jiao 2022, Bertazzi et al. 2013) or involve stock-out costs that are at least equivalent to the travel costs for a separate round trip from the depot to the customer

(Gruler et al. 2018, Bertazzi et al. 2013). Consensus strategies (Hvattum and Løkketangen 2009, Bent and Van Hentenryck 2004), as an alternative to the multi-scenario policies with their linkage constraints, become impractical when confronted with the extensive combinatorial nature of actions inherent in the Stochastic Inventory Routing Problem (SIRP). Finally, in line with most IRP definitions, the DSIRP proposed by Coelho, Cordeau, and Laporte (2014a) does not consider possible stock-outs when optimizing over the rolling horizon. However, from a cost perspective in a stochastic context, it is imperative to consider the possibility of failing to fulfill demand, especially for distant customers with low stock-out costs.

Due to the above-mentioned limitations, our initial benchmark policies are rolling-horizon policies with a single demand episode inspired by Coelho, Cordeau, and Laporte (2014a). The first method, the MEAN policy, involves averaging all historical observations. The second method, denoted as SAA-1, directly selects a single historical observation. When contextual information is absent, we select the most recent observation. When contextual information is available, we select the observation with the smallest Euclidean distance based on the observed features. However, it is worth noting that these single-scenario methods have limitations: Averaging lacks the ability to consider contextual information, and relying on a single observation can lead to suboptimal decisions as it fails to capture the full range of demand uncertainty. Therefore, we incorporated the approach proposed by Bertazzi et al. (2013) into our set of baselines. To tailor it to our needs, we have adapted their multi-scenario approach by using multiple demand samples for the current period, while approximating all subsequent periods with the mean but leveraging a MILP formulation to solve it without the discrete demand assumption. This adaptation is called the SAA-3 policy, drawing inspiration from the SAA and employing three demand episodes. As a pipeline-based benchmark policy, we use the approach proposed by Baty et al. (2023).

#### 6.3. Operational Performance

We first the operational performance of our pipeline. Subsequently, we discuss the training time here in Section 6.4. Result 1 compares our pipeline, referred to as ML-CO policy and trained with the *Voting-DAgger* learning paradigm, to MEAN, SAA-1, SAA-3 policy. Result 2 confirms the end-of-horizon issue of Baty et al. (2023)'s learning paradigm. Result 3 compares the different learning paradigms. Result 4 evaluates the voting policy within a *DAgger-like* learning paradigm separately. As the primary performance metric, we use the performance relative to the anticipatory baseline solution, evaluated as:

Result 1 Our ML-CO policy outperforms all benchmarks on average across all penalties and demand patterns, with costs only 23.43% higher compared to the anticipatory baseline solution. The best benchmark, the MEAN policy, already incurs 32.55% additional costs. In addition, inference times are an order of magnitude smaller than our benchmark policies. Figure 4 and Table 1 summarize this major result.

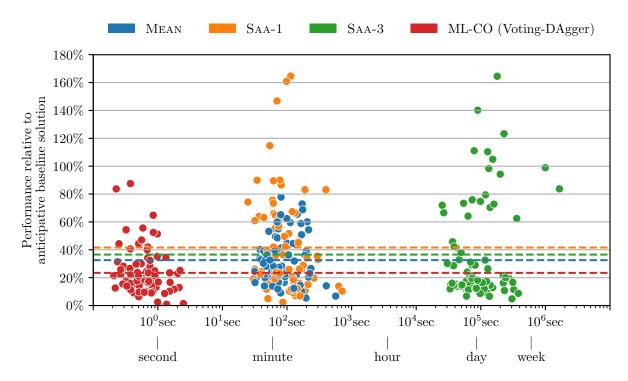


Figure 4 Performance and inference time per instance [horizontal line highlights the mean performance].

As detailed in Table 1, none of the single scenario policies outperform the ML-CO policy on average across all penalties in any demand pattern. For normal or uniform demand, the SAA-3 policy is slightly better in terms of the additional costs incurred (16.24% to 13.34% for normal demand and 30.13% to 27.27% for uniform demand). However, bimodal demand or contextual information completely transforms the landscape (39.60% to 92.07% for bimodal demand and 7.74% to 13.56% for contextual demand).

As depicted in Figure 4, our pipeline has significant advantages regarding the inference times for dynamic decision-making. The inference times of our ML-CO policy are below one second. Even single-scenario policies like MEAN and SAA-1 require minutes to hours for inference, whereas the SAA-3 policy requires inference times measured in days or weeks.

Table 1 Averag	erage performance relative to anticipative baseline solution (in %)				
		Mean	SAA-1	Saa-3	ML-CO
demand pattern	shortage penalty				
bimodal	low	43.46	78.42	80.18	37.47
	high	42.54	83.92	103.95	41.73
		43.00	81.17	92.07	39.60
contextual	low	26.25	18.34	12.93	8.07
	high	41.08	27.78	14.19	7.42
		33.66	23.06	13.56	7.74
normal	low	17.51	16.22	11.91	13.82
	high	17.95	18.39	14.78	18.65
		17.73	17.30	13.34	16.24
uniform	low	27.58	34.54	27.98	24.51
	high	44.07	55.30	26.56	35.76
		35.82	44.92	27.27	30.13
average across scenarios		32.55	41.61	36.56	23.43
standard deviation across scenarios		(16.91)	(34.48)	(35.85)	(18.84)

**Result 2** The learning paradigm of Baty et al. (2023) suffers from the end-of-horizon effect. Given the absence of an incentive to maintain inventory beyond the final period in the anticipatory problem, the number of visited customers and the delivery quantities decrease in the later periods, as highlighted in Figure 5.

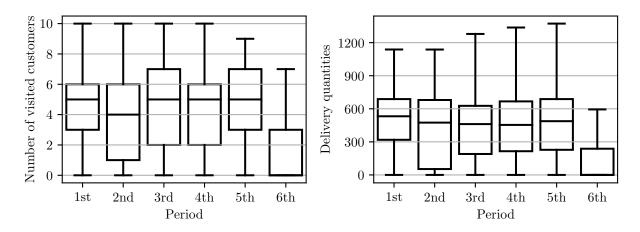


Figure 5 End-of-horizon effect using the learning paradigm of Baty et al. (2023).

Result 3 As summarized in Table 2 and Figure 6, both DAgger-based learning paradigms are superior. Regarding performance, Baty et al. (2023)'s learning paradigm is significantly worse. The Sampling learning paradigm is comparable in performance but lacks robustness and generalizability of explored states.

Table 2 Performance relative to anticipative baseline solution (in %)

1	11 .1 1	average	standard
learning paradigm	look-anead		deviation
Baty	6	56.13	(34.38)
Sampling	6	25.57	(23.21)
Anticipative-DAgger	6	24.79	(19.73)
Voting-DAgger	6	23.43	(18.84)

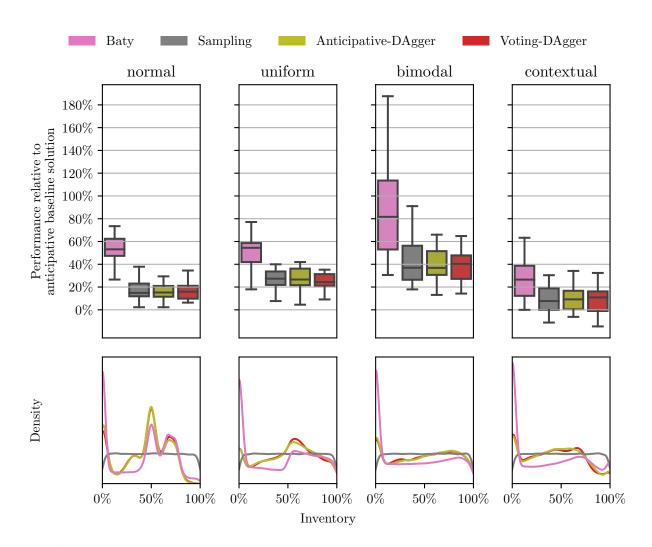


Figure 6 Performance per learning paradigm and demand pattern with its explored inventories.

As shown in Table 2, the learning paradigm of Baty et al. (2023) performs worst (56.13%), while the Sampling learning paradigm (25.57%) performs almost as good as the Anticipative-DAgger

(24.79%) and the *Voting-DAgger*  $(23.43\%)^2$  <sup>3</sup>. But as depicted in Figure 6, the investigated states (inventories) of the *Sampling* learning paradigm are based solely on the sampling strategy (here evenly distributed), lacking robustness and generalizability.

**Result 4** We improve the performance of the Anticipative-DAgger learning paradigm from 24.79% to 23.43% by using the voting policy with five scenarios per state, as shown in Table 2.

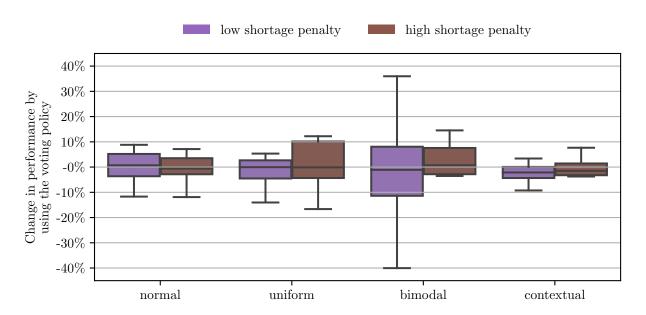


Figure 7 Impact of using Voting-DAgger with five demand scenarios per state instead of Anticipative-DAgger

As detailed in Figure 7, the voting policy with five scenarios instead of a single scenario per state is most valuable for demand with contextual information. No clear trend can be analyzed for the other demand patterns; this could be due to the reduced number of explored states using *Voting-DAgger*.

#### 6.4. Training Time

Training our pipeline demands substantial computational effort. Result 5 examines the training time of the pipeline, which may span several days or weeks. To conclude, we analyze the effect of a reduced look-ahead horizon in Result 6.

<sup>&</sup>lt;sup>2</sup> All learning paradigms use 600 samples in each epoch, except for the *DAgger-like* learning paradigms, where the 600 samples are not reached until the 10th epoch due to the iterative dataset updates. If the voting policy is used with five scenarios per state, the number of states is reduced accordingly.

<sup>&</sup>lt;sup>3</sup> To demonstrate that the suboptimal performance of Baty et al. (2023)'s learning paradigm is unrelated to the difference between the look-ahead horizon (6 periods) and the evaluation horizon (10 periods), we include results with an evaluation horizon of 6 periods in Appendix B. Expanding the look-ahead horizon is unfeasible due to computational constraints.

Result 5 As shown in Table 3, the learning paradigm of Baty et al. (2023) requires less time for sample generation (29.3 single-core CPU hours) compared to all other learning paradigms. This efficiency is achieved by extracting multiple samples from a single anticipatory decision problem. The time required for sample generation increases with the complexity of the learning paradigm and reaches 311.4 single-core CPU hours for Voting-DAgger. In terms of policy update, both DAgger-like learning paradigms require less time, as both use fewer samples up to the 10th epoch than the other learning paradigms.

Table 3 Average training time, normalized to a single CPU core (in hours).

		sample generation	policy update	others
learning paradigm	look-ahead			
Baty	6	29.3	1157.7	2.3
Sampling	6	162.4	1285.8	2.8
Anticipative-DAgger	6	304.4	776.8	4.8
Voting-DAgger	6	311.4	996.2	6.9

Moreover, we observe that almost 100% of the training time is spent on sample generation and policy updates, whereas the other steps take a time that is orders of magnitude smaller.

Result 6 As demonstrated in Table 4, a reduced look-ahead horizon of three periods instead of six periods for the anticipatory decision problem drastically accelerates sample generation from 311.4 single-core CPU hours to 10.2 single-core CPU hours. According to Table 5, this also leads to a better average performance. As detailed in Figure 8, consolidating routing costs over time becomes less important with high shortage penalties, and demand without contextual information. However, performance deteriorates for contextual demand patterns.

Table 4 Average training time, normalized to a single CPU core (in hours).

learning paradigm	look-ahead	sample generation	policy update	others
Voting-DAgger	3	10.2	802.5	3.3
	6	311.4	996.2	6.9

Coelho, Cordeau, and Laporte (2014a) report that the performance of their algorithm improves with a reduced look-ahead horizon. With our *Voting-DAgger* learning paradigm, we could also observe this for non-contextual demand. However, for contextual demand, which was not studied by Coelho, Cordeau, and Laporte (2014a), we observe a performance degradation. When setting

Table 5 Performance relative to anticipative baseline solution (in %)

learning paradigm	look-ahead	average	$\begin{array}{c} standard \\ deviation \end{array}$
Voting-DAgger	3	23.29	(14.86)
	6	23.43	(18.84)

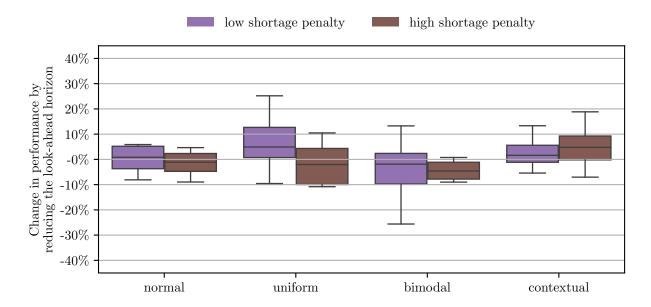


Figure 8 Impact of reducing the look-ahead horizon from 6 to 3 periods.

up the pipeline, it is important to test different look-ahead horizons because the definition of the look-ahead horizon should be related to the relationship between demand and maximum inventory capacity.

# 7. Conclusions

Interest in dynamic decision-making has been rapidly growing, prompting ongoing efforts to develop more efficient solvers and heuristics. Our contribution to this landscape involves the introduction of a learning-based policy designed for the DSIRP. Compared to the benchmarks, our ML-CO policy achieved state-of-the-art performance and demonstrated a significant advantage in inference times for real-time decision-making. This comes, however, at the price of a longer training process, which must be performed in advance. We have identified two primary bottlenecks: the IRP solver responsible for sample generation and the CPCTSP oracle employed during policy updates. Replacing these computationally intensive MILPs with efficient heuristics for the oracle, as done in Baty et al. (2023), could significantly accelerate training. Moreover, we suggest considering using ML techniques such as stochastic gradient descent or transfer learning in future research. This implies that advancements in both research directions, CO and ML, have the potential to enhance the effectiveness of our approach.

Future work could consider more complex demand patterns or instance parameters. Although we have trained and evaluated our pipeline on instances with the same number of customers, it could be applied to networks of varying sizes. A promising strategy could be to iteratively sample smaller subnetworks from large customer networks with thousands of customers, thereby creating a generalized model applicable to the entire network. The approach could also be extended to maximum-level policies, multi-depot settings, and multi-vehicle scenarios. Finally, beyond its application to the DSIRP, our learning algorithm can train any policy encoded by a neural network with a CO layer of the form (14). Indeed, it suffices to be in the setting described in Section 5.1 for our algorithm to be applicable. Future works could explore the efficiency of our approach on other problems modeled as MDPs with large state and action spaces. In conclusion, our approach achieves good performance for DSIRPs and is versatile, paving the way for numerous promising avenues in future research.

# Acknowledgments

This work was supported by a fellowship within the IFI program of the German Academic Exchange Service (DAAD).

## Appendix A: Stochastic Instances

We generated stochastic instances with *normal*, *uniform*, *bimodal* and *contextual* demand pattern, details are described in the following.

Normal and Uniform Demand Our stochastic instances with either normal or uniform demand for customer  $i \in \mathcal{V}_c$  have been generated according to the following rules:

- Mean demand  $\mu_i$  is randomly selected from the set  $\{10, \ldots, 100\}$ .
- Standard deviation  $\sigma_i$  is randomly chosen from the set  $\{2, \ldots, 10\}$ .
- Inventory capacity  $C_i$  is determined by  $\mu_i$  times a random number from the set  $\{2,3,4\}$ .
- Initial inventory level  $I_i^0$  is set as  $C_i \mu_i$ .
- Per-unit holding cost  $\kappa_i$  is randomly generated from a continuous uniform distribution within the interval [0.02, 0.10].
- Shortage penalty  $\rho$  determines the stock-out costs per unit, obtained by multiplying the holding cost  $\kappa_i$  by the shortage penalty  $\rho$ . We evaluate two different scenarios with multiples of (i)  $\rho = 200$  (low shortage penalty) and (ii)  $\rho = 400$  (high shortage penalty).
- Vehicle capacity B is calculated as 1.5 times the sum of the mean demands:

$$B = 1.5 \sum_{i \in \mathcal{V}_c} \mu_i$$

This ensures that the vehicle capacity is appropriately scaled to accommodate the aggregated demand of all customers.

- Routing costs  $\gamma_{ij}$ , representing distances between vertices i and j have been calculated using the Euclidean distance. The coordinates of each vertex  $i \in \mathcal{V}$  have been randomly sampled from a discrete uniform distribution within the interval [0,500].
- Normally-distributed demands have been generated using a truncated normal distribution with mean  $\mu_i$ , standard deviation  $\sigma_i$ , and bounds of 0 and  $C_i$ .
- Uniformly-distributed demands have been generated using a uniform distribution within the interval  $\left[0, \frac{C_i}{2}\right]$ .

**Bimodal Demand** Our stochastic instances with a bimodal demand distribution for each customer  $i \in \mathcal{V}_c$  have been generated according to the following rules. The two components of the general mixture model have been generated using a truncated normal distribution with mean  $\mu_i^1$ , standard deviation  $\sigma_i^1$  or mean  $\mu_i^2$ , standard deviation  $\sigma_i^2$ , and bounds of 0 and  $C_i$ .

- The mean difference  $\mu_{\Delta}$  has been randomly selected from the set  $\{4, \dots, 20\}$ .
- The mean  $\mu_i^1$  has been randomly selected from the set  $\{10, \dots, 50 \mu_{\Delta}\}$ .
- The mean  $\mu_i^2$  has been randomly selected from the set  $\{50 \mu_{\Delta}, \dots, 100\}$ .
- Standard deviations  $\sigma_i^1$  and  $\sigma_i^2$  were randomly chosen from the set  $\{2,\ldots,10\}$ .
- Inventory capacity  $C_i$  have been set as  $\frac{\mu_i^1 + \mu_i^2}{2}$  times a random number from the set  $\{2, 3, 4\}$ .
- The vehicle capacity B has been set to 1.5 times the sum of the mean demands:

$$B = 1.5 \sum_{i \in \mathcal{V}_c} \frac{\mu_i^1 + \mu_i^2}{2}$$

All other parameters were determined similarly to those for normal and uniform demand.

Contextual Demand We designed a synthetic dataset with eight distinct features, drawing inspiration from the generation process outlined by Pedregosa et al. (2011) in the *sklearn.datasets.make\_regression*() method. This generation method is widely adopted in the scientific literature, as evidenced by references such as Dessureault and Massicotte (2022) and Reddy and Claridge (1994). To ensure diversity in our dataset, we first randomly selected a number of informative features for each instance from the set  $\{2, \ldots, 6\}$ . This ensures that every instance includes at least two informative features, while the others remain non-informative.

We generated unscaled feature values within the range [-1,+1]. To diversify our instances, we employed one of three distributions (arcsin, uniform, or truncated normal) with an equal likelihood of selection for each distribution. These distributions exert varying influence on the data's tails and centers. These features are then scaled by a randomly chosen factor from the set  $\{\sqrt{10}, \ldots, \sqrt{100}\}$ , resulting in features  $\lambda^1$  through  $\lambda^8$  for each demand realization and each customer i  $in\mathcal{V}_c$ . Furthermore, we introduced an exogenous noise  $\xi$  generated in a manner consistent with the standard deviation of steady-state instances. Specifically, it follows a normal distribution with a mean of zero and a standard deviation randomly drawn from the set

 $\{2,\ldots,10\}$ . Let  $M=\{(n,m):n\in\{1,\ldots,8\},m\in\{n+1,\ldots,8\}\}$  represent the set of all unordered pairs of features. We constructed each *contextual demand* realization for each customer i as

$$\min \left\{ \max \left\{ \mu_i + \sum_{n=1}^8 \alpha_n \lambda^n + \sum_{(n,m) \in M} \alpha_{nm} \lambda^n \lambda^m + \xi, 0 \right\}, C_i \right\}.$$
 (23)

The coefficients  $\alpha_n$  and  $\alpha_{n,m}$  have been uniformly sampled once per instance in the interval [-1, +1]. These coefficients are the same among customers. When dealing with the product of features, as in Equation 23, there is an equal likelihood of them being either informative or non-informative. In the latter case, we set  $\alpha_{n,m} = 0$ . As visible in the equation, each demand is bounded within the range of 0 to  $C_i$ .

## Appendix B: Additional Results

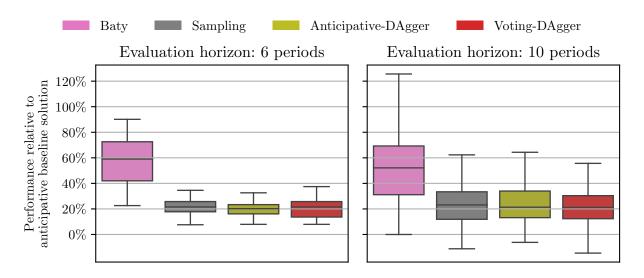


Figure 9 Performance per learning paradigm and evaluation horizon [for all demand patterns].

## References

- Adulyasak Y, Cordeau JF, Jans R, 2014 Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. INFORMS Journal on Computing 26(1):103–120.
- Archetti C, Bertazzi L, Hertz A, Speranza MG, 2012 A hybrid heuristic for an inventory routing problem. INFORMS Journal on Computing 24(1):101–116.
- Archetti C, Bertazzi L, Laporte G, Speranza MG, 2007 A branch-and-cut algorithm for a vendor-managed inventory-routing problem. Transportation science 41(3):382–391.
- Archetti C, Bertazzi L, Paletta G, Speranza MG, 2011 Analysis of the maximum level policy in a production-distribution system. Computers & Operations Research 38(12):1731–1746.
- Archetti C, Boland N, Grazia Speranza M, 2017 A matheuristic for the multivehicle inventory routing problem. INFORMS Journal on Computing 29(3):377–387.

- Avella P, Boccia M, Wolsey LA, 2018 Single-period cutting planes for inventory routing problems. Transportation Science 52(3):497–508.
- Baty L, Jungel K, Klein PS, Parmentier A, Schiffer M, 2023 Combinatorial optimization enriched machine learning to solve the dynamic vehicle routing problem with time windows.
- Bent RW, Van Hentenryck P, 2004 Scenario-based planning for partially dynamic vehicle routing with stochastic customers. Operations Research 52(6):977–987.
- Berbeglia G, Cordeau JF, Laporte G, 2010 Dynamic pickup and delivery problems. European journal of operational research 202(1):8–15.
- Bertazzi L, Bosco A, Guerriero F, Lagana D, 2013 A stochastic inventory routing problem with stock-out.

  Transportation Research Part C: Emerging Technologies 27:89–107.
- Bertazzi L, Coelho LC, De Maio A, Laganà D, 2019 A matheuristic algorithm for the multi-depot inventory routing problem. Transportation Research Part E: Logistics and Transportation Review 122:524–544.
- Bertsekas D, Tsitsiklis J, 2015 Parallel and distributed computation: numerical methods (Athena Scientific).
- Bouvier L, Dalle G, Parmentier A, Vidal T, 2023+ Solving a continent-scale inventory routing problem at renault. Transportation Science, in press.
- Bouvier L, Parmentier A, 2023 Dynamic multi-attribute inventory routing problem at Renault: dealing with the continental scale. ROADEF 2023, Best student paper prize section.
- Boyd S, Parikh N, Chu E, Peleato B, Eckstein J, et al., 2011 Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning 3(1):1–122.
- Brinkmann J, Ulmer MW, Mattfeld DC, 2019 Dynamic lookahead policies for stochastic-dynamic inventory routing in bike sharing systems. Computers & Operations Research 106:260–279.
- Chitsaz M, Cordeau JF, Jans R, 2019 A unified decomposition matheuristic for assembly, production, and inventory routing. INFORMS Journal on Computing 31(1):134–152.
- Coelho LC, Cordeau JF, Laporte G, 2014a Heuristics for dynamic and stochastic inventory-routing. Computers & Operations Research 52:55–67.
- Coelho LC, Cordeau JF, Laporte G, 2014b Thirty years of inventory routing. Transportation Science 48(1):1–19.
- Coelho LC, Laporte G, 2013a A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. International Journal of Production Research 51(23-24):7156-7169.
- Coelho LC, Laporte G, 2013b The exact solution of several classes of inventory-routing problems. Computers & Operations Research 40(2):558–565.
- Dalle G, Baty L, Bouvier L, Parmentier A, 2022 Learning with combinatorial optimization layers: a probabilistic approach. URL http://dx.doi.org/10.48550/ARXIV.2207.13513.

- Desaulniers G, Rakke JG, Coelho LC, 2016 A branch-price-and-cut algorithm for the inventory-routing problem. Transportation Science 50(3):1060–1076.
- Dessureault JS, Massicotte D, 2022 Dpdr: A novel machine learning method for the decision process for dimensionality reduction. arXiv preprint arXiv:2206.08974.
- Gendreau M, Jabali O, Rei W, 2016 50th anniversary invited article—future research directions in stochastic vehicle routing. Transportation Science 50(4):1163–1173.
- Gruler A, Panadero J, de Armas J, Pérez JAM, Juan AA, 2018 Combining variable neighborhood search with simulation for the inventory routing problem with stochastic demands and stock-outs. Computers & Industrial Engineering 123:278–288.
- Guimarães TA, Coelho LC, Schenekemberg CM, Scarpin CT, 2019 The two-echelon multi-depot inventory-routing problem. Computers & Operations Research 101:220–233.
- Gurobi Optimization, LLC, 2023 Gurobi Optimizer Reference Manual. URL https://www.gurobi.com.
- Hildebrandt FD, Thomas BW, Ulmer MW, 2023 Opportunities for reinforcement learning in stochastic dynamic vehicle routing. Computers & Operations Research 150:106071.
- Huang SH, Lin PC, 2010 A modified ant colony optimization algorithm for multi-item inventory routing problems with demand uncertainty. Transportation Research Part E: Logistics and Transportation Review 46(5):598–611.
- Hvattum LM, Løkketangen A, 2009 Using scenario trees and progressive hedging for stochastic inventory routing problems. Journal of Heuristics 15:527–557.
- James J, Yu W, Gu J, 2019 Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems 20(10):3806–3817.
- Joe W, Lau HC, 2020 Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. Proceedings of the international conference on automated planning and scheduling, volume 30, 394–402.
- Jungel K, Parmentier A, Schiffer M, Vidal T, 2023 Learning-based online optimization for autonomous mobility-on-demand fleet control.
- Li Z, Jiao P, 2022 Two-stage stochastic programming for the inventory routing problem with stochastic demands in fuel delivery. International Journal of Industrial Engineering Computations 13(4):507–522.
- Manousakis E, Repoussis P, Zachariadis E, Tarantilis C, 2021 Improved branch-and-cut for the inventory routing problem based on a two-commodity flow formulation. European Journal of Operational Research 290(3):870–885.
- Nedic A, Ozdaglar A, 2010 10 cooperative distributed multi-agent. Convex optimization in signal processing and communications 340.
- Oyola J, Arntzen H, Woodruff DL, 2018 The stochastic vehicle routing problem, a literature review, part i: models. EURO Journal on Transportation and Logistics 7(3):193–221.

- Parmentier A, 2021 Learning structured approximations of operations research problems. arXiv preprint arXiv:2107.04323.
- Parmentier A, 2022 Learning to approximate industrial problems by operations research classic problems.

  Operations Research 70(1):606–623.
- Parmentier A, T'Kindt V, 2021 Learning to solve the single machine scheduling problem with release times and sum of completion times.
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E, 2011 Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12:2825–2830.
- Powell WB, 2019 A unified framework for stochastic optimization. European Journal of Operational Research 275(3):795–821.
- Psaraftis HN, Wen M, Kontovas CA, 2016 Dynamic vehicle routing problems: Three decades and counting.

  Networks 67(1):3–31.
- Reddy T, Claridge D, 1994 Using synthetic data to evaluate multiple regression and principal component analyses for statistical modeling of daily building energy consumption. Energy and buildings 21(1):35–44.
- Ritzinger U, Puchinger J, Hartl RF, 2016 A survey on dynamic and stochastic vehicle routing problems.

  International Journal of Production Research 54(1):215–231.
- Ross S, Gordon G, Bagnell D, 2011 A reduction of imitation learning and structured prediction to no-regret online learning. Proceedings of the fourteenth international conference on artificial intelligence and statistics, 627–635 (JMLR Workshop and Conference Proceedings).
- Soeffker N, Ulmer MW, Mattfeld DC, 2022 Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. European Journal of Operational Research 298(3):801–820.
- Wong KI, Bell MG, 2006 The optimal dispatching of taxis under congestion: A rolling horizon approach. Journal of advanced transportation 40(2):203–220.