Bootstrap Enhanced Scenario Optimization, a Case Study in Two-Echelon Logistics

Livio Fenga[a], Vittorio Maniezzo[b,*]

[a] *Center for Simulation, Analytics and Modelling, University of Exeter Business School, Exeter, UK*
[b] *Department of Computer Science, University of Bologna, Bologna, Italy*

Abstract

We introduce the Bootstrap Enhanced Scenario Optimization (BESO) framework, a novel methodology for prescriptive analytics under uncertainty. BESO addresses the challenge of generating reliable demand scenarios from volatile, short, and non-stationary time series data. The core innovation lies in delegating the modeling of uncertainties to the Maximum Entropy Bootstrap (MEB) integrated with bagging, a technique that preserves the complex autocorrelation structure of empirical time series forecasts, enabling the generation of statistically coherent scenarios. We further extend MEB through an objective-augmented formulation that incorporates downstream recourse elements into the resampling mechanism. These statistically rich scenarios are then integrated into a deterministic equivalent model to derive accountable and optimized prescriptions. We demonstrate the framework's power using a real-world case study in tactical inventory allocation. Our findings, also validated against an extensive set of artificial benchmark instances, demonstrate the effectiveness of MEB-enhanced forecasting in optimization. BESO establishes a new, generalizable paradigm for robust decision-making derived from noisy univariate data series.

Keywords: Stochastic programming, Distributionally Robust Optimization, Forecasting, Maximum Entropy Bootstrap, Two-echelon logistics

1. Introduction

Decision-making under uncertainty is a pervasive challenge in operations research, particularly in complex systems such as logistics networks. Scenario-based optimization has long been used to address this challenge by capturing the variability of future events through a finite set of possible outcomes. However, the reliability of such methods depends heavily on how accurately these scenarios represent the underlying stochastic processes.

In this paper, we propose a novel method —- *Bootstrap Enhanced Scenario Optimization* (BESO) —- that integrates bootstrap-based forecasting with scenario-based linear optimization to enhance

*Corresponding author
Email addresses: L.Fenga@exeter.ac.uk (Livio Fenga), vittorio.maniezzo@unibo.it (Vittorio Maniezzo)

the robustness and realism of stochastic modeling. The approach can be framed within the Distributionally Robust Optimization (DRO) paradigm; however, unlike mainstream moment- or ambiguity-set-based formulations, it leverages statistical models fitted to empirical data to generate optimization-ready scenarios.

BESO introduces a twofold, decision-aware extension of the Maximum Entropy Bootstrap (MEB). First, the bootstrap objective is augmented with optimization cost components, allowing scenario generation to internalize downstream decision criteria. Second, we explicitly account for the effect of bootstrap-set cardinality on the resulting optimization outcomes, recognizing scenario size as a structural parameter shaping the effective ambiguity set. Together, these mechanisms create a feedback loop between the optimization model and the statistical resampling phase: the ambiguity set remains statistically coherent while its geometry and dispersion are partially shaped by optimization-relevant considerations. The framework is particularly suited to predictive settings where reliable historical time series are available.

The resulting data-driven methodology for empirically estimating the distribution of stochastic variables and embedding it in scenario generation for robust and stochastic optimization may offer significant advantages over standard distribution-aware approaches. Traditional methods often rely on strong assumptions about the underlying probability distribution — typically normality or other parametric forms — which may not accurately reflect the true behavior of real-world uncertainties. In contrast, data-driven techniques directly leverage historical or observed data to construct empirical distributions, capturing nuances such as asymmetry, multimodality, or heavy tails that standard models might overlook. This increased fidelity can lead to more realistic and effective scenario generation, improving both the robustness and relevance of the resulting optimization solutions. Furthermore, by adapting to the structure present in the data, data-driven approaches are naturally more flexible and can better accommodate nonstationarities or structural changes in the stochastic environment, providing better alignment with real operational environments.

To demonstrate the practical value and performance of BESO, we apply it to a real-world two-echelon logistics network problem. In this setting, uncertain outbound freight demands must be satisfied across multiple depots. By using BESO to define demand scenarios and solve for optimal allocation decisions and related inventory requests, we show how the method improves the solution robustness compared to conventional techniques, while remaining conceptually transparent, computationally efficient and straightforward to implement.

The remainder of the paper is structured as follows: Section 2 reviews previous works related to the different research areas intersected by this study. Section 3 describes the case study that motivated our research. Section 4 and 5 present the BESO methodology in detail. Section 6 discusses the computational results obtained, and Section 7 concludes with key insights and future research directions.

2. Literature review

The paper intersects different research threads, both in terms of the theoretical contribution and the application use case.

2.1. Predictive Prescriptions and Distributionally Robust Optimization

Stochastic programming has long served as a fundamental framework for modeling decision-making under uncertainty, primarily through expected-value models and recourse formulations [3, 11]. As the field matured, methods evolved to address model robustness. Robust optimization [4, 7] proposes a deterministic reformulation considering worst-case outcomes within a defined uncertainty set. A major development is *Distributionally Robust Optimization* (DRO), which seeks a solution most resilient against the worst-case probability distribution for uncertain parameters, typically within an *ambiguity set* [14].

The ambiguity set ($\mathcal{U}$) in DRO is a carefully constructed set of probability distributions that the true, but unknown, distribution of the uncertain parameters is assumed to belong to. The core of DRO is to optimize the worst-case performance over all distributions within this set, thus making the resulting solution robust to distributional uncertainty. Mainstream DRO approaches define this ambiguity set in different ways. Some utilize specific distance metrics to construct an ambiguity set centered around the empirical distribution, such as those based on the ϕ-divergence (or f-divergence) [6]. Others, such as in the work of Esfahani and Kuhn [24], define the set based on measurable moment constraints (e.g., bounds on the mean and covariance) of the uncertain parameters. This methodology provides strong theoretical guarantees but requires the designer to make choices on the ambiguity set's structure and size, which may not fully reflect the true statistical dynamics of the input data.

A primary technique for applying both stochastic programming and DRO is *Scenario-Based Optimization*. This approach discretizes the continuous probability distribution of uncertain parameters into a finite set of specific scenarios. The *Sample Average Approximation* (SAA) method is a pivotal technique here [45], approximating the true underlying distribution with an empirical distribution derived from a sample of independent and identically distributed realizations.

The integration of forecasting and scenario-based optimization - often framed as *predict-then-optimize* - is a growing body of literature [6, 40]. The typical workflow follows a two-stage paradigm: a forecasting model (e.g., SARIMA, GARCH, LSTM) first estimates future parameters, and a separate scenario generation phase simulates or samples future paths based on the forecast's predictive distribution. This sequential paradigm has been questioned because prediction errors, even small ones, can lead to suboptimal decisions when their impact on the downstream optimization problem is large. This motivated the development of methods that explicitly incorporate the optimization objective into the forecasting model's loss function [16], leading to the "Smart Predict, then Optimize" (SPO) framework [20], besides the aforementioned DRO contributions.

Unlike these traditional DRO and predict-then-optimize approaches, our work introduces a novel framework that fundamentally links the statistical modeling of input data to the generation of the uncertainty set. Our *Bootstrap Enhanced Scenario Optimization* (BESO) framework leverages bootstrapped time series to generate future scenarios. This approach is conceptually aligned with DRO, as it seeks robustness across a set of statistically plausible distributions. However, by using the Maximum Entropy Bootstrap (MEB), BESO generates scenarios that directly preserve the complex autocorrelation and non-stationarity observed in the historical data, eliminating the need for post-hoc parametric ambiguity set design. Scenarios are not a post-processing step but emerge directly from a statistically robust model of the input data's empirical structure. Furthermore, we show how the MEB objective function can be extended to incorporate the downstream optimization's recourse objective, thereby aligning the statistical model directly with the desired prescriptive outcome, similar to SPO methods. In addition, the bootstrap-set cardinality acts as an explicit structural control on the induced ambiguity, allowing the dispersion and effective geometry of the scenario set to be tuned in accordance with downstream optimization sensitivity.

2.2. 2-echelon logistics networks

Multi-echelon inventory management is a supply chain control strategy that coordinates inventory across multiple levels, called "echelons," of the chain [10]. The objective is to optimize inventory and routing costs across all levels of the supply chain rather than addressing each level in isolation. The literature on this strategy has expanded significantly, covering various aspects such as inventory control policies, replenishment strategies, and inventory risk management.

The successful implementation of effective management policies hinges on sound forecasting and optimization processes. Specifically, in the context of 2-echelon logistics, prescriptions grounded in predictive analytics are essential for refining demand forecasting, inventory allocation, and transportation planning [1, 32]. Considerable attention has been paid to demand forecasting [34, 46], however, precise demand forecasting, especially within the context of dynamic retail environments characterized by fluctuating trends and seasonal variations, remains a significant obstacle. Multi-echelon demand forecasting commonly incorporates a variety of models that frequently rely on assumptions of stationarity and specific parametric distributional characteristics. These assumptions are often violated in dynamic retail environments characterized by short-horizon, non-stationary demand series, leading to model mismatch and suboptimal prescriptions.

The application of prescriptive analytics in 2-echelon logistics systems thus faces a critical challenge: how to robustly model the underlying uncertainty to support efficient management decision processes. Most documented multi-echelon inventory policies in the literature rely on instances with generic demand, often generated according to parametric distributions [23, 21]. Our work addresses this methodological gap.

By introducing the BESO framework, we provide a statistically-grounded approach to tactical logistics planning. BESO leverages the MEB to generate demand scenarios that accurately preserve

the complex autocorrelation and non-stationary structure of empirical demand data. This novel method directly counters the limitations of traditional approaches that fail to capture the nuances of real-world volatility, enabling the distribution centers (DCs) to update their market forecast and anticipate the structural attributes of the model, ultimately providing valuable management insights [39]. We demonstrate this capability using a comprehensive real-world case study where all instance elements, from demand series to network configurations, are derived from actual records. This application provides a validation of how BESO improves both solution robustness and operational efficiency in multi-echelon inventory management.

3. The 2-echelon test case

The test case, that makes use of data that comes from an LSRT company that manages multiple retail stores in a large city in northern Italy, originally proposed in [36] for a different application. In this case, 52 stores provided aggregate sales time series data spanning at least 45 months prior to the forecast period, which is set at a three-month horizon, as detailed in the following. Each store maintains a small inventory, replenished by deliveries from one of four different distribution centers (DCs) in the area. The products are categorized into three lines: fresh, dry, and household. In this context, the focus has been exclusively on the logistics of dry and household goods, including product categories ranging from personal care items to basic electronics, from household goods to holiday decorations. For this product line, DCs are required to maintain an aggregate inventory sufficient to cover a demand for a period of two weeks.

The supply chain consists of a large distribution center (DC) in the northern suburbs of the city, a smaller DC in the south, and two small satellite facilities closer to the city center. The satellites have limited storage capacity, but they offer lower-cost and more timely service to the majority of the city's stores. Each store must be allocated to a distribution center (DC), ensuring that the total requests made to a DC do not exceed its storage capacity. It is assumed that the handling and transportation capacities will always be sufficient. Both storage capacities and requests are quantified in terms of pallets, with no further granularity at this analytical level. The cost of servicing a store from a DC is assessed based on the travel time required to transport goods between the two entities. Notably, the three smaller DCs are fully owned by the company, whereas the fourth DC is situated in a leased facility.

The ultimate goal of the analysis is to determine how much space to rent during the next peak demand period, the Christmas season. To achieve this, we first optimize the allocation of stores to DCs, taking into account transportation and storage costs, and from the solution it is immediate to derive the space required in each DC, therefore the space to rent in the largest one.

Another requirement characterizes the problem we are addressing: large stores located near small DCs often lead to rapid saturation of the capacity of central DCs. This forces nearby stores to be served by comparatively distant DCs. Therefore, it has been proposed to allow tentative

split service for certain selected stores to facilitate what-if analysis. The problem we address thus involves a parametric number of possible assignments for each store, where which store should be allowed multiple (double) assignments is an input parameter, not a decision variable.

The analysis was conducted during the summer of 2022 in preparation for the subsequent Christmas season. At the time of the analysis, the anticipated demand requests were unknown and required forecasting. The analysis underwent iterative refinements throughout the summer, culminating in the final forecast produced in September. Consequently, the data presented in this paper correspond to a three-month forecasting case.

All codes and data are available in an anonymized format from the project repository *(public upon acceptance)*. Specifically, the file containing the request time series can be downloaded from the repository, where all series are complete up to the most recent actual values. However, it should be noted that the last three values were unknown at the time the final forecast was generated.

According to the classification proposed in [13] our contribution is described by the following typology string: 2,D,D,G|F,C|D,G|b,F,N|SC|E,O standing for: *System specification*: 2 echelons, Divergent material structure, Discrete events, Global information; *Resources specifications*: bounded storage, Constant delivery time; *Market specifications*: Discrete stochastic demand, Guaranteed service; *Control type specifications*: installation base stock policy, Flexible release quantities, No other means to satisfy unexpected requirements; *Performance specifications*: meeting service requirements and minimization of costs; *Generic scientific aspects*: Exact techniques, Optimization.

4. The predictive module

Data-driven approaches to estimating the distribution of stochastic variables have gained increasing attention as alternatives to classical parametric modeling, particularly in settings where assumptions of stationarity or known distributional forms are invalid. Unlike traditional techniques that rely on a priori specification of distribution families (e.g., Gaussian or Poisson), data-driven methods infer distributional characteristics directly from historical or observed data, making them especially suited for real-world applications with complex, noisy, or limited information.

Among the most prominent techniques are resampling-based methods such as the Bootstrap [18], which generates empirical distributions by repeatedly sampling from the data with replacement. The Maximum Entropy Bootstrap (MEB) [47] extends this idea by producing replicates that preserve dependence structures in time series data, making it particularly valuable for short or non-stationary sequences.

Recent work has explored the integration of these empirical distribution estimates into stochastic and robust optimization pipelines. Notable contributions include nonparametric scenario generation [2], distributionally robust optimization using empirical Wasserstein distances [24].

In our proposal, we utilize the MEB methodology in conjunction with the ensemble learning technique known as *bagging* (Bootstrap Aggregating). This combination enhances the accuracy and

robustness of predictive models through the generation of multiple bootstrapped datasets from the original data. By training individual models on these datasets and aggregating their predictions through averaging, we adhere to MEB's principles while capturing the underlying statistical variability. Maximizing entropy under empirical constraints allows the integration of bagging with MEB to produce more robust models. In more details, each model trained on distinct bootstrapped samples incorporates uncertainty characteristics from the original data, allowing for a comprehensive representation of its variation.

Furthermore, this approach allows for dealing with noisy datasets and mitigating overfitting, as bagging inherently provides a mechanism to increase the robustness of the predictive model by leveraging the diversity across the bootstrapped samples. Since MEB generates bootstrapped datasets based on the maximum entropy principle, the resulting models avoid being overly sensitive to any one particular realization of the data.

Mathematically, if we denote the bootstrapped replicas of the dataset as $D_1^*, D_2^*, \ldots, D_B^*$, where B represents the number of bootstrapped samples, the resulting predictive framework can be expressed as: $\hat{y}_{BAG} = \rho(\hat{y}_b(x))$, where $\hat{y}_b(x)$ is the prediction made by the $b-th$ model trained on the bootstrapped sample D_b^* and ρ a generic centrality parameter. This averaging process effectively reduces variance by smoothing out the predictions of the individual models, leading to enhanced stability and generalization capabilities in the final aggregated model. The parameter ρ can be defined as either a simple average or the median, which leads to the following bagging predictors:

$$\hat{y}_{BAG} = \frac{1}{B}\sum_{b=1}^{B}\hat{y}_b(x) \qquad \hat{y}_{BAG} = \begin{cases} \hat{y}_b(x)\left[\frac{h+1}{2}\right] & \text{if } h \text{ is odd} \\ \frac{\hat{y}_b(x)\left[\frac{h}{2}\right]+\hat{y}_b(x)\left[\frac{h}{2}+1\right]}{2} & \text{if } h \text{ is even} \end{cases} \tag{1}$$

where, $\hat{y}_b(x)$ is sorted in ascending order whereas h is the number of future values to be predicted.

4.1. Resampling techniques

In recent years, resampling techniques, especially the bootstrap method, have become valuable for overcoming the limitations of conventional forecasting methods [29]. The bootstrap method creates simulated time series through resampling from observed data. This approach is useful for non-stationary, non-linear, intermittent, and asymmetric cyclical patterns, as well as low signal-to-noise scenarios [25, 33]. Moreover, resampling methods help address the effects of limited sample sizes [12] and model uncertainty [41].

The utility of bootstrap methods in model selection has been extensively validated. Fenga and Politis [26] introduced a bootstrap-based approach for autoregressive moving average (ARMA) order selection, demonstrating its effectiveness in enhancing model accuracy with finite sample sizes. Furthermore, they explored the LASSO technique for sparse autoregression, illustrating a bootstrap framework for model selection in high-dimensional contexts [27]. These studies collectively show how bootstrap methods can enhance order selection across statistical modeling frameworks, underscoring their relevance in modern statistical analysis.

The standard bootstrap assumes independent and identically distributed (i.i.d.) data — an assumption that fails for time series, where observations are dependent and ordered. This failure is more pronounced when the series is short, making dependence patterns more influential, the data exhibits autocorrelation, seasonality, or trend and when resampling destroys the structure and underrepresents variability. In these cases Maximum Entropy Bootstrap (MEB) guarantees the following advantages:

- *Preservetion of Dependence Structure* MEB maintains the temporal dependence (e.g., autocorrelation, trend, seasonality) by using rank-based and interpolation techniques that reorder and smooth the data. Unlike i.i.d. bootstrap, it doesn't break time series structure.
- *Better Handling of Short Time Series* In short samples, structural properties (e.g., local trends) are easily lost by standard resampling. MEB keeps these features by reconstructing plausible time series paths that are consistent with observed data and uncertainty.
- *No Need for Model Specification* Unlike block bootstrap or parametric resampling, MEB is nonparametric and doesn't require selecting block sizes or fitting ARMA models — a key benefit for short samples where such models may be unstable or overfitted.
- *Generation of Smooth Pseudo-Series* MEB produces continuous, smoothed pseudo-series through linear interpolation, making it well-suited for statistical inference tasks like forecasting, confidence intervals, and hypothesis testing.
- *Maximum Entropy Principle* The generated bootstraps are the least biased (i.e., most non-committal) estimates consistent with the known constraints (the observed data). This is especially valuable when little information is available — like in short series.
- *Support of Stationary and Non-Stationary Series* MEB is flexible with both stationary and non-stationary time series, whereas other bootstrap techniques often require stationarity or explicit transformation to achieve it.

The MEB resampling scheme adopted here offers a principled and flexible approach to bootstrapping time series data, leveraging the *Maximum Entropy Principle* (MEP) [43] to address the inherent limitations of conventional bootstrap methods, thus addressing many limitations of traditional block bootstrap methods. The essence of MEP is grounded in the idea that in the absence of specific information, the most rational probability distribution to adopt is one that maximizes entropy, reflecting the greatest uncertainty about the system [31]. This is mathematically articulated through Shannon entropy, defined as $S(p) = -\sum_{i=1}^{n} p_i \log(p_i)$, where p_i denotes the probability associated with the i-th outcome within a discrete sample space of size n.

The pivotal advancement provided by MEB lies in its ability to formulate bootstrapped samples that adhere to a spectrum of constraints dictated by the original dataset, such as predetermined moments, probabilities, or other statistical characteristics [5]. Specifically, under typical circumstances, these constraints can be specified in the form of normalized probabilities and empirical expectations, expressed as $\sum_{i=1}^{n} p_i = 1$ to ensure that the probabilities form a valid distribution, and $\sum_{i=1}^{n} p_i f_i = \bar{f}$, where f_i represents measurements or statistics relevant to the analysis, with

$\bar{f}$ being the empirical mean of these measurements. The formulation of the Maximum Entropy problem consequently leads to a constrained optimization framework. The optimization problem can be succinctly encapsulated in the Lagrangian:

$$L(p, \lambda, \mu) = S(p) + \lambda \left(1 - \sum_{i=1}^{n} p_i\right) + \mu \left(\bar{f} - \sum_{i=1}^{n} p_i f_i\right), \tag{2}$$

where λ and μ are Lagrange multipliers that adjust the balance under the constraints of normalizing the distribution and matching the empirical mean, respectively. The solution to this maximization problem yields a probability distribution, which appears as

$$p_i = \frac{e^{\mu f_i}}{\sum_{j=1}^{n} e^{\mu f_j}}, \tag{3}$$

where the parameter μ is determined through the constraint equations, specifically via a numerical root-finding approach that satisfies the moment condition $\sum_{i=1}^{n} p_i f_i = \bar{f}$.

Building on this classical formulation, we define an Objective-Augmented Maximum Entropy Bootstrap (OA-MEB) scheme that incorporates additional optimization-oriented structure directly into the resampling distribution. To do so, we introduce an auxiliary objective-related component g_i, representing scenario-wise quantities that are relevant to downstream decision or cost functions in stochastic programming (e.g., expected cost contributions, constraint violations, or risk-sensitive statistics). The OA-MEB extends the entropy maximization problem by embedding a tunable scalar parameter $\gamma \geq 0$ that controls the influence of this additional term while preserving the probabilistic and moment-matching constraints. The resulting augmented Lagrangian is given by:

$$L_{\mathrm{OA}}(p, \lambda, \mu, \gamma) = S(p) + \lambda \left(1 - \sum_{i=1}^{n} p_i\right) + \mu \left(\bar{f} - \sum_{i=1}^{n} p_i f_i\right) + \gamma \left(\sum_{i=1}^{n} p_i g_i\right), \tag{4}$$

where the additional inner product term $\sum_{i=1}^{n} p_i g_i$ promotes bootstrap resamples that remain statistically coherent while incorporating optimization-driven scenario emphasis. Solving the first-order optimality conditions yields a modified exponential-tilting solution:

$$p_i^{\mathrm{OA}} = \frac{\exp\left(\mu f_i + \gamma g_i\right)}{\sum_{j=1}^{n} \exp\left(\mu f_j + \gamma g_j\right)}, \tag{5}$$

showing that classical MEB is recovered as the special case $\gamma = 0$. Calibration of γ governs the trade-off between purely information-theoretic sampling and decision-aligned resampling, and may be performed via cross-validation, bilevel optimization, or risk-preference calibration strategies consistent with the downstream stochastic programming model.

This formulation transforms the problem into one that allows for flexible adaptation to various types of data and constraints imposed by the empirical observations. Once this extended maximum entropy distribution is established, resampling via bootstrapping can be accomplished by selecting m samples according to the derived p_i. This produces synthetic datasets that not only reflect the variability and uncertainty characteristic of the original data but also adhere to the specific constraints imposed, which may include higher moments or even domain-specific statistics.

This resampling process can be iterated to generate a comprehensive set of bootstrapped sam-

ples, enabling a multitude of statistical analyses, including confidence interval estimation, hypothesis testing, and model validation [19].

4.2. Bootstrap Cardinality and Decision–Ambiguity Feedback

Bootstrap-based ambiguity sets introduce an additional degree of freedom in data-driven DRO: the bootstrap cardinality m. Although often treated as a tuning parameter, m directly determines the geometry of the ambiguity set and therefore influences the optimizer. Conversely, the optimal decision induces a preference over m through validation or penalization criteria. This creates an intrinsic feedback loop between ambiguity construction and optimization, which the MEB-extension formalizes. We observe data $D_n = \{\xi_i\}_{i=1}^n$ with empirical measure

$$\widehat{P}n = \frac{1}{n}\sum i = 1^n \delta_{\xi_i}, \tag{6}$$

decision space $X \subset \mathbb{R}^d$, and cost function $c : X \times \Xi \to \mathbb{R}$. For a bootstrap scheme, let $\widehat{P}_n^{(1)}, \ldots, \widehat{P}_n^{(m)}$ denote m bootstrap empirical measures. A natural ambiguity set is the convex hull $\mathcal{U}_m := \mathrm{conv}\{\widehat{P}_n^{(1)}, \ldots, \widehat{P}_n^{(m)}\}$ where $\widehat{P}_n^{(j)}$ is the empirical distribution of the (j)-th bootstrap sample, and $\mathcal{U}_m$ is their convex hull in the space of measures whose diameter, extreme points, and support all depend on m. The associated DRO objective is

$$\Psi_m(x) := \sup_{P \in \mathcal{U}_m} \mathbb{E}_P[c(x, \xi)], \tag{7}$$

and the optimizer is defined by $x^\star(m) \in \arg\min_{x \in X} \Psi_m(x)$.

Since Ψ_m is the supremum of finitely many linear functionals $\mathbb{E}_{\widehat{P}_n^{(j)}}[c(x, \xi)]$, changing m changes the set of extreme points and therefore the optimizer. In general, $m \neq m'$ implies $\mathcal{U}_m \neq \mathcal{U}_{m'}$ and $x^\star(m) \neq x^\star(m')$.

If the ambiguity set depends on x—for example through residual-based resampling in the MEB-extension—write $\mathcal{U}_m(x)$ and consider the joint problem

$$\min_{x \in X,\, m \in \mathcal{M}} \left\{ \sup_{P \in \mathcal{U}_m(x)} \mathbb{E}_P[c(x, \xi)] + \lambda\, \mathrm{Pen}(m) \right\}. \tag{8}$$

Eliminating m yields the bilevel formulation

$$\min_{m \in \mathcal{M}} \Phi(x^\star(m), m) \quad \text{s.t.} \quad x^\star(m) \in \arg\min_{x \in X} \sup_{P \in \mathcal{U}_m(x)} \mathbb{E}_P[c(x, \xi)]. \tag{9}$$

This expresses the feedback loop: m shapes the ambiguity faced by x, while x determines which m minimizes the joint objective.

Under standard bootstrap consistency assumptions, if $m = m(n) \to \infty$ with $m(n)/n \to 0$, then $\mathcal{U}m(n) \Rightarrow P^\star$ and $\Psi m(n)(x) \to \mathbb{E}_{P^\star}[c(x, \xi)]$ uniformly on compact X, implying $x^\star(m(n)) \to x^\star$. For finite n,

$$\sup_{x \in X} \left| \Psi_m(x) - \Psi_{m'}(x) \right| \leq L\, \mathrm{Haus}\big(\mathcal{U}m, \mathcal{U}m'\big), \tag{10}$$

so changes in m induce controlled perturbations in the DRO objective and optimizer.

The MEB-extension incorporates decision-dependent ambiguity and regularizes m within the joint objective, replacing the classical "forecast–then–optimize" pipeline with a unified formulation

that captures the $m \leftrightarrow x^{\star}(m)$ feedback.

5. The prescriptive module

The predictive module provides a structural component of the prescriptive module, which aims to optimize the allocation of stores to distribution centers (DCs), thereby enabling the quantification of space to rent in third-party DCs. The results of the optimization are in turn used to evaluate the modeled *scenario*, where scenarios are defined for specific what-if analyses. For example, the objective of the what-if analyses could be to examine the effect of freezing some allocations while leaving free the others, or possibly accepting some multiple – double in the test case – allocations for some selected stores.

The core optimization problem is a *split allocation problem*. This variant of the basic allocation problem has already received attention in the optimization literature, mainly in the context of survivable communication network design [22, 38], and more generally as the splittable capacitated multiple allocation hub location problems [37]. It finds applications in fiber optic access networks [48], split delivery routing [17], and even in school timetabling [8]. The problem we are interested in is related to one that is presented in [49], but it is modelled in a very different way.

The mathematical model we use is a direct extension of the standard model for allocation problems. As our proposed framework is valid beyond the specific case study described in the paper, we will refer to the application elements (stores and data centers) using standard allocation terminology in what follows. To emphasize the generality of the analysis, we will use the more generic term 'clients' to denote the stores and 'servers' to denote the centers. In our context, assignment constraints can accept a parametric number of assignments, enabling requests to be serviced across multiple servers. The notation used in the model is as follows:

- n number of clients (stores)
- J index set of clients, $J = \{1, \ldots, n\}$
- m number of servers (DCs)
- I index set of servers, $I = \{1, \ldots, m\}$
- c_{ij} global cost for allocating client $j \in J$ to server $i \in I$
- Q_i storage capacity of server $i \in I$
- d_i unit storage cost of server $i \in I$
- req_j total request of client j.
- b_j maximum number of servers (DCs) client j can be assigned to
- x_{ij} binary decision variable equal to 1 iff client j is allocated to server i, $i \in I$ and $j \in J$
- q_{ij} integer decision variable representing the amount (number of pallets) of request of client j provided from server i

In the final test case, the amount requested by each client was always set to equal the corresponding predicted value. However, in some scenarios, some clients' requests could be split between

two servers, meaning the requested $q_(ij)$ amounts could vary.

The mathematical model used for this problem is as in the following formulation (FD).

$$
\begin{aligned}
(FD) \qquad z_{FD} = \textbf{min} \ & \sum_{i\in I}\sum_{j\in J}(c_{ij}x_{ij} + d_i q_{ij}) && && (11)\\
\textbf{subject to} \ & \sum_{i\in I} q_{ij} = req_j && j \in J && (12)\\
& \sum_{j\in J} q_{ij} \le Q_i && i \in I && (13)\\
& \sum_{i\in I} x_{ij} = b_j && j \in J && (14)\\
& q_{ij} \le req_j x_{ij}; \quad x_{ij} \in \{0,1\}; \quad q_{ij} \in \mathbb{Z}_0^+ && i \in I, j \in J && (15)
\end{aligned}
$$

The constraints in equation 14 are formulated as equalities due to the need to impose specific splits in 'what if' scenarios. However, if this is not of particular interest, these constraints could be replaced by less-than-or-equal-to inequalities combined with inequalities that ensure allocations are made to at least one server.

Note that when $b_j = 1$ for all clients, i.e., when each client $j \in J$ can be served by only one server, constraints 15 and 12 force the entire quantity requested by client j, req_j, to be supplied by a single server. The constraints 15 thus become equations, the constraints 12 become redundant, and the constraints 13 can be expressed by replacing $\sum_{j\in J} q_{ij}$ with $\sum_{j\in J} req_j x_{ij}$. Problem P reduces to the standard GAP and can be solved by any of the methods described in Maniezzo et al. [35].

However, we are interested in the general case that allows for multiple assignments. Since the cost is derived from these assignments, the problem can be considered a fixed-cost split-assignment allocation problem. Furthermore, as future client requests can only be forecast, it is more convenient to formulate the problem as a stochastic optimization problem, where the future requests are random variables ρ_j, $j \in J$, whose distributions are derived from the forecast results. The resulting stochastic problem is formulated as in formulation (FS), where the values $R(\rho_j)$ in the constraints 17 and 20 refer to any same realization of the random variable ρ_j.

$$
\begin{aligned}
(FS) \qquad z_{FS} = \textbf{min} \ & \sum_{i\in I}\sum_{j\in J}(c_{ij}x_{ij} + d_i q_{ij}) && && (16)\\
\textbf{subject to} \ & \sum_{i\in I} q_{ij} = R(\rho_j) && j \in J && (17)\\
& \sum_{j\in J} q_{ij} \le Q_i && i \in I && (18)\\
& \sum_{i\in I} x_{ij} = b_j && j \in J && (19)\\
& q_{ij} \le R(\rho_j) x_{ij}; \quad x_{ij} \in \{0,1\}; \quad q_{ij} \in \mathbb{Z}_0^+ && i \in I, j \in J && (20)
\end{aligned}
$$

Different methods can be used to deal with this case [44]. We transformed the formulation (FS) into its *deterministic equivalent* [11] using the forecasts computed on the bootstrap sets. We generated multiple scenarios each containing a realization of the stochastic variable ρ, which is defined as a forecast of one series from the bootstrap set in use. For each scenario s, we thus created a version of the constraint with b_i^s as the right-hand side. The final solution must satisfy these constraints across all scenarios.

The objective function now incorporates the quantities to be delivered to each client in different scenarios. Therefore, we consider the expected cost as a weighted average across these scenarios and include it in the objective function. The resulting formulation is in formulation (DE). The random variables ρ_j are sampled in a finite number of scenarios. Let ρ_j^s denote the realization of ρ_j in scenario s, $s \in S$, and let $p_s = \frac{1}{|S|}$ be the probability associated with scenario $s \in S$. The variables ϵ_{js} are slack variables for the customer service constraints.

$$
\begin{aligned}
(DE) \quad z_{DE} = \quad & \textbf{min} \sum_{s \in S} \left(p_s \sum_{j \in J} \left(\sum_{i \in I} (c_{ij} x_{ij} + d_i q_{ij}^s) + M \epsilon_{js} \right) \right) && && (21)\\
& \textbf{s.t.} \sum_{i \in I} q_{ij}^s + \epsilon_{js} = \rho_j^s && j \in J, s \in S && (22)\\
& \sum_{j \in J} q_{ij}^s \leq Q_i && i \in I, s \in S && (23)\\
& \sum_{i \in I} x_{ij} = b_j && j \in J && (24)\\
& q_{ij}^s \leq \rho_j^s x_{ij}; \quad q_{ij}^s \in \mathbb{Z}_0^+ && i \in I, j \in J, s \in S && (25)\\
& x_{ij} \in \{0,1\} && i \in I, j \in J && (26)\\
& \epsilon_{js} \in \mathbb{Z}_0^+ && j \in J, s \in S && (27)
\end{aligned}
$$

In this formulation, since each variable ρ_j^s derives from a specific realization of the corresponding random variable ρ_j, the amounts to be delivered to each store $j \in J$ are scenario-specific. However, it is required to determine a unique allocation that allows to serve all stores in any proposed scenario. This last requirement may be infeasible, so the slack variables ϵ_{js} are needed in the formulation. They are lexicographically penalized in the objective function by appropriately high costs, in order to implement the primary concern of achieving feasibility.

Optimizing formulation (DE) makes it possible to determine the quantities that each server must be able to supply to each client, i.e. the total quantity that each server must handle. In terms of our logistical application, since each DC must maintain sufficient inventory to meet the anticipated demand during the designated period, the optimization of the client (retailer) allocation translates directly into a quantification of the DC inventory and therefore of its required minimum size. Thus, solving the allocation problem also solves the dimensioning question that motivated this research.

6. Computational results

This section reports on the results of validating the predictive and prescriptive modules in accordance with the specifications previously described.

6.1. Predictive module tests

The objective of the predictive module is to forecast demand for each store, asking for a forecast three months ahead. The available data consists of a time series for each of the 52 stores. Each series contains 45 values corresponding to four years of monthly data on relevant requests for each store. The values for the last three months of the last year are missing, as these are to be forecast.

The core approach set out in this paper is the bootstrap method, which is described in Section 4. The results obtained using this method were validated against those obtained using several alternative forecasting algorithms: SARIMAX, Holt-Winters, MLP, RNN (LSTM), Random Forest and Gradient Boosting (XGBoost). The relevant hyperparameters were determined by grid search for algorithms with integer-valued parameters, such as SARIMAX, and by Hammersley sampling [28] for real-valued parameters.

All series were pre-processed using a 'log-diff' transform to approach stationarity, and a max–min scaler was applied to the artificial intelligence-derived methods. Python version 3.11 was used for all implementations and the codes were run on a Windows PC workstation equipped with four Intel Core i7-4790 CPUs running at 3.60 GHz and with 32 GB of RAM.

The focus of this part of the research is on the bootstrap method. To this end, we tested the approach with the following settings:

- bootstrap sets containing 75, 125 and 175 series;
- bootstrap sets generated by simple autoregressive AR(p) with p=5 model (the default in python's statsmodels), where parameters are estimated using Yule-Walker equations [9], and by ARIMA, where autoarima is run on each of the seed series and the optimized orders are then applied to each boosted series;
- the forecasting of all the boosted series generated as above to obtain the boosted forecasts was tested with AR(p), Random Forest or ARIMA. In the case of ARIMA, autoarima was only repeated on the first series of each bootstrap set. In the case of AR, the parameter p was set for each series after a loop on different p values, keeping the one that maximized the corresponding AIC value.

We also attempted to generate bootstrap sets with and without backcasting, as well as with simple extraction and repetitions among the residuals of the seed series, and compared these with their scrambling. We did not report the results of these tests because backcasting worsened the results, while scrambling the residuals showed no difference compared to extraction with repetitions. Therefore, in line with the literature, we decided to use the latter approach.

The forecasts were compared using three cost functions: mean absolute error (MAE), mean squared error (MSE) and bias. The results are reported by ranking the corresponding forecasts for

	YW_AR_75			YW_AR_125			YW_AR_175		
	MAE	MSE	bias	MAE	MSE	bias	MAE	MSE	bias
fcast_avg	2.67	2.67	0.08	2.58	2.58	0.05	3.83	3.83	0.04
fcast_50	3.04	3.04	0.06	3.19	3.19	0.04	2.50	2.50	0.03
yhw	5.42	5.42	0.08	6.06	6.06	0.04	5.88	5.88	0.03
yarima	5.73	5.73	0.05	6.04	6.04	0.03	6.08	6.08	0.02
ysvm	5.88	5.88	0.08	5.50	5.50	0.05	5.50	5.50	0.03
ylstm	6.02	6.02	0.06	5.52	5.52	0.04	6.27	6.27	0.02
yar	6.06	6.06	0.04	6.54	6.54	0.02	6.79	6.79	0.02
ymlp	6.38	6.38	0.05	5.98	5.98	0.03	6.17	6.17	0.02
yrf	6.46	6.46	0.11	6.31	6.31	0.06	5.48	5.48	0.04
yxgb	7.33	7.33	0.12	7.29	7.29	0.07	6.50	6.50	0.05

Table 1: Average rankings for different boostset size.

each series according to the quality of the considered function to emphasize how the algorithms perform relative to each other. Then, the rank of each function is averaged over all series. Appendix Appendix A reports the absolute values of the error measures, rather than the rank-based values, for all boostset sizes.

Table 1 presents the results of comparing the forecasts obtained using standard algorithms with those obtained using boosting. The boosted series were computed using an AR(5) model based on Yule-Walker equations. The forecasts of these series were obtained using a simple autoregressive model. The boosted sets consisted of 75, 125, and 175 series, which are labelled YW_AR_75 (standing for Yule-Walker, AutoRegressive, 75 series bootstrap set), YW_AR_125, and YW_AR_175, respectively.

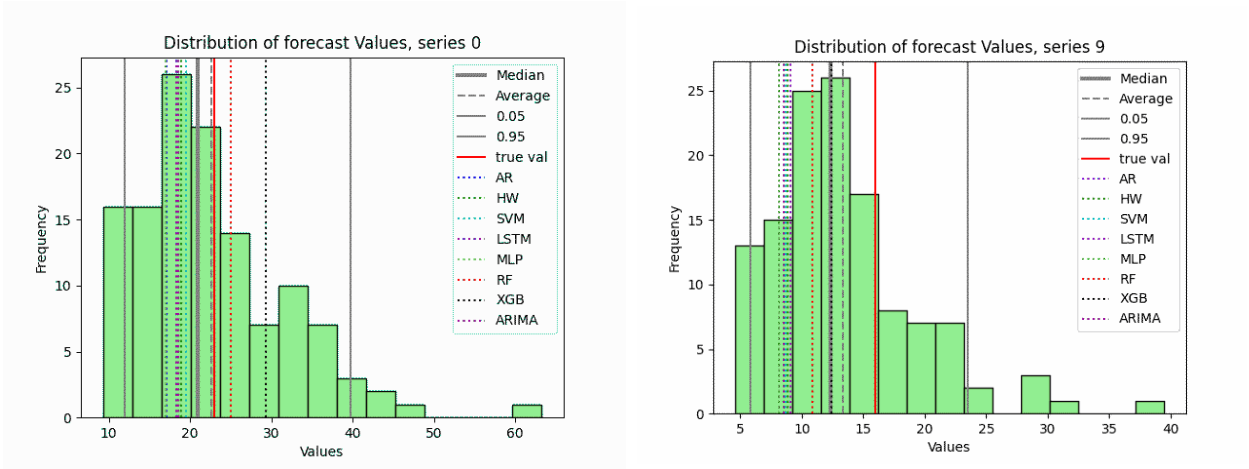
The columns show the average ranking of each cost function (MAE, MSE or bias) for each boostset size, while on the rows we report:

- $fcast\_avg$: boost forecast obtained as the average of the corresponding ranks;
- $fcast\_50$: boost forecast obtained as the median of the corresponding ranks;
- yar: non-boost forecast obtained by an AR model;
- $yarima$: non-boost forecast obtained by an ARIMA model;
- yhw: non-boost forecast obtained by a Holt and Winters model;
- $ymlp$: non-boost forecast obtained by a multilayer perceptron model;
- $ysvm$: non-boost forecast obtained by a SVM model;
- $ylstm$: non-boost forecast obtained by a LSTM model;
- yrf: non-boost forecast obtained by a random forest model;
- $yxgb$: non-boost forecast obtained by a XGboost model;

The table shows that boosting is consistently the most reliable method for minimizing forecast errors, even when forecasts are obtained from a model as simple as AR(p), while ARIMA produces the least bias.

Figure 1 shows two examples of forecast distributions obtained for two different series. The green bars correspond to the distribution of the results obtained by the AR[p] model over the differently

recolored alternatives of a series. The distribution is superimposed with lines representing the results of other forecasting algorithms applied to the original series.



(a) 75 bootset *(b) 125 boostset*

Figure 1: Distribution of bootstrap forecasts.

The relative merits of the different forecasting approaches are better highlighted by critical difference diagrams based on the non-parametric Friedman test followed by a post-hoc Nemenyi test. These diagrams allow for a visual comparison of performance ranks and indicate whether the differences between classifiers are statistically significant [15]. In our case, the graphs were generated using the SciKit version [42] and plot the average rank across all series for each algorithm along the x-axis. Horizontal bars connect those that could not be considered statistically different.

Figures 2 and 3 show a comparison of the ranks on MAE for the 75 series boostset and on MSE for the 175 series boostset. In both cases, as with all those tested, the accuracy of the boost forecasts is significantly higher than that obtained with alternative models. The tables in Appendix A confirm that the comparison yields consistent results across all boostset sizes and cost functions, except for bias. The bias rankings are in fact unrelated to those obtained with the other cost functions. This is because the bias is very small in all cases, less than 1% of the average of the empirical values, which ensures that all the forecasting models are free of systematic errors on our data and makes the difference in the rankings insignificant.

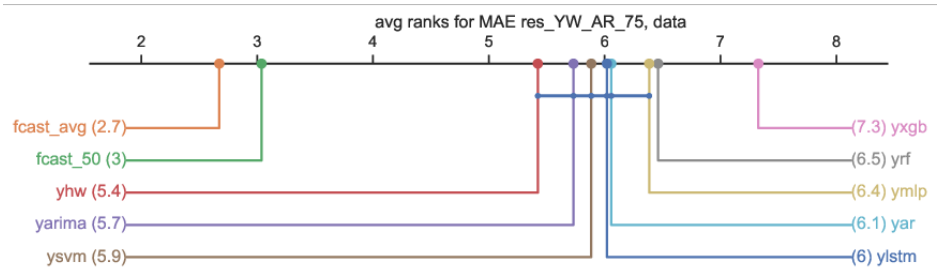


Figure 2: Average ranks for MAE on instance res_YW_AR_75

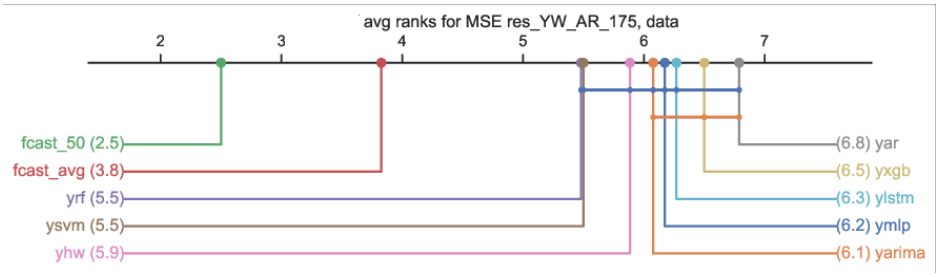


Figure 3: Average ranks for MSE on instance res_YW_AR_175

Our relatively short time series require a forecasting approach that can handle both limited data and non-stationarity. Although sophisticated machine learning models such as recurrent neural networks (RNNs) and gradient boosting machines (GBMs) have become more popular in recent years, they can perform less well when there is limited data available, particularly when the data has non-stationary properties. This can lead to overfitting. The autoregressive (AR) model combined with maximum entropy bootstrap offers a powerful and promising alternative. This technique performed excellently in the context of this study, consistently outperforming more complex machine learning models. It offers an approach based on robust statistical principles that mitigates the risk of overfitting inherent in many AI approaches. Moreover, the AR model combined with the bootstrap methodology provides reliable forecasts ideally suited to integration into the subsequent optimization module.

In our case, the choice of an autoregressive (AR) model not only offers superior accuracy, but also several crucial advantages over more complex machine learning approaches. Firstly, the AR model's inherent simplicity makes it easy to implement and interpret. Unlike many machine learning algorithms, which operate as 'black boxes', the parameters of the AR model have clear statistical interpretations that provide insight into the underlying dynamics of the time series data. Secondly, the well-established theoretical framework of the AR model and the readily available diagnostic tools allow for robust validation and assessment of model performance.

6.2. Prescriptive module tests

The bootstrap forecasts results reported in section 6.1 provided the basis for instantiating the optimization phase. The prescriptive module was first run in a deterministic setting, to define a benchmark solution, then extended to the stochastic setting described in section 5.

When the problem is modelled using the (FD) formulation and the average predictions obtained by boosting are used as deterministic data, it can be easily solved by any state-of-the-art MILP solver. This demonstrates the feasibility of dealing with a significant deterministic equivalent of the stochastic formulation (FS). The optimal deterministic value of $z_{FP} = 15553$ is used as a benchmark in the following tests.

We moved on to the (DE) formulation generating a value for each of the stochastic variables ρ_j in correspondence to each bootstrap forecast. Given these input data, we ran experiments to assess

the impact of the variation of the remaining sets of parameters of the formulation, i.e., cardinality of the boostsets, number of splittable clients, and assumptions on inventory costs.

All reported tests were run on the same machine as in subsection 6.1, a Windows PC workstation equipped with four Intel Core i7-4790 CPUs running at 3.60 GHz and 32 GB of RAM, but in this case the implementation language was C++ to allow finer control over the MILP solver used, which was CPLEX v. 22.11.

This section reports on the sensitivity of the solution to variations in the instance coefficients. Appendix B provides an initial study of the fitness landscape for the split-assignment problem, using a set of artificial benchmark instances.

Increasing boostset size

The size of the boostset helps to stabilize the results during forecasting, as introduced in subsection 4.2, but it also has a deep impact on the optimization. In fact, the number of variables and constraints in the DE formulation do not depend on the size of the boostset, larger boostsets are more likely to contain unusual values or combinations thereof. This makes the constraints in equation (22) progressively harder to satisfy, which increases the cost and may render the instance infeasible. To ascertain this, we conducted experiments using 75, 125 and 175 series boostsets, with a time limit of 600 seconds on the optimization process. Figure 4 shows a boxplot diagram depicting the distribution of the optimized cost over a number of tests equal to 1/5 of the number of series in the boostset, i.e., 15, 25, 35, respectively.

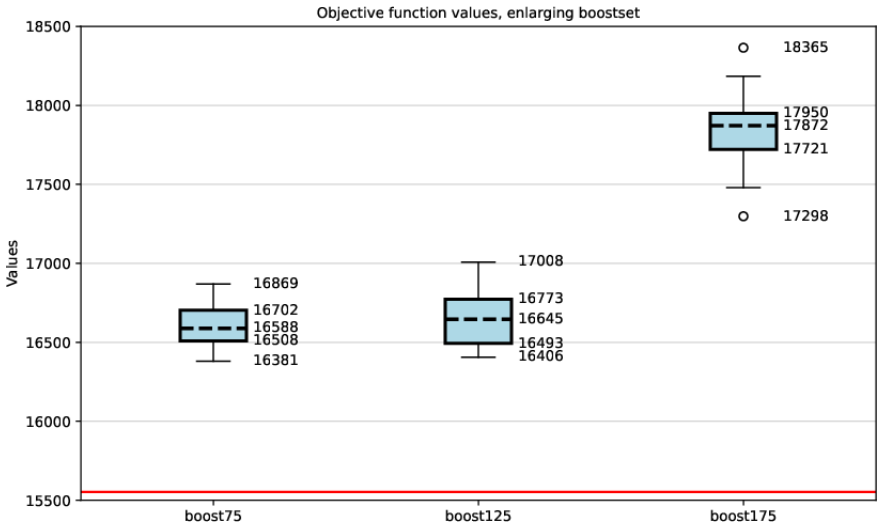


Figure 4: Allocation cost for increasing boostset size

Figure 4 has a red horizontal line at the cost value for the deterministic case. Therefore, the increase in cost can be considered a quantification of the value of information about storage requests. We observe that the cost remains relatively stable for the 75 and 125 series sets, but increases significantly for the 175 boostset. This is due to the increased difficulty of the larger boostset instances, which meant that some could only be solved heuristically within the time limit.

While all instances were feasible, seven out of 35 still had a gap between the lower and upper bounds after 600 seconds of CPU time.

Finally, we note that, as expected, taking the optimal solution obtained with the coefficients imposed by the 75 series and evaluating it with the coefficients of the 125 series results in a more costly solution than that computed with the 125 series coefficients. More interestingly, however, this solution is not feasible with the 175 series coefficients.

Increasing number of splittable clients

A defining feature of the problem we face is the possibility of accepting split service for some clients. This is unusual for allocation problems and results from the requirement to perform specific what-if analyses on the results. The number of clients allowed to split, and which ones they are, is given in input. In the actual case, the service to a client, if splittable, could be split between at most two servers, and the quantities delivered had to be integers, as specified in all formulations.

The clients to be split are chosen from the most requesting clients. We ran a series of tests based on 75 series boostsets, allowing the 0, 4, 8, and 12 most requesting clients to be split. Figure 5 shows the corresponding boxplots.

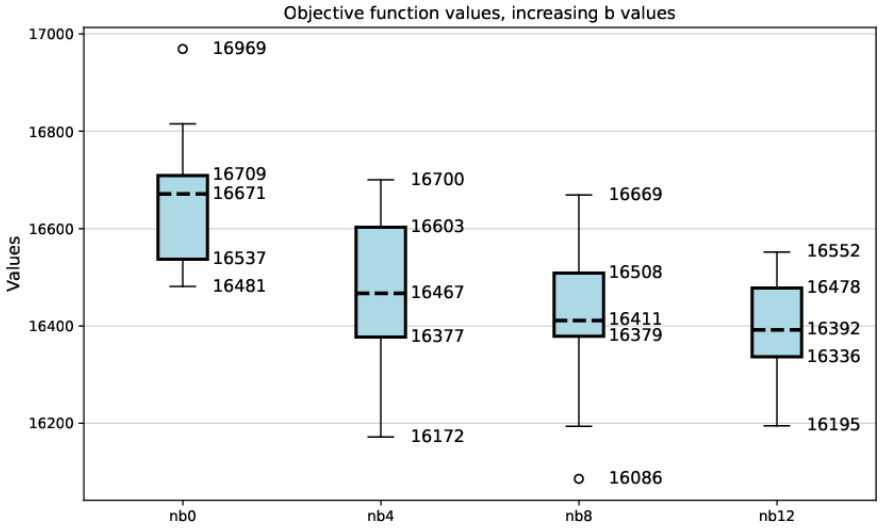


Figure 5: Allocation cost for increasing number of splittable clients

Allowing a service to be split for a client is a partial relaxation of the corresponding assignment constraint, so the expected optimal cost is lower. This can indeed be seen in figure 5, where the more clients that are permitted to split, the lower all distribution moments become. However, the marginal decrease becomes smaller as the number of clients who can split increases, since splitting a small client may result in only a small contribution, if any.

Impact of different inventory cost assumptions

A further set of tests was conducted to determine the impact of inventory costs d_i, $i \in I$ on the search process. Recall that in the original case study, one of the objectives was to determine how

n	m	n_{boost}	n_q	n_{inf}	GAP lin	GAP fin	t.CPU
52	4	75	0	0	0.01	0.00	333.95
52	4	75	1	1	0.18	0.17	600
52	4	75	1	0	0.15	0.14	1200
52	4	75	1	0	0.14	0.13	3600
52	4	75	2	0	0.04	0.04	600
52	4	75	2	0	0.01	0.03	1200
52	4	75	2	0	0.01	0.03	3600
52	4	75	3	0	0.01	0.00	211.03
52	4	75	4	0	0.01	0.00	13.00

Table 2: Gap and CPU time for increasing number of leased servers.

much space should be rented in the only DC not owned by the company, where storage space needs to be leased. The data of the case study proposed in Maniezzo and Zhou [36] did not report storage costs. Here, we consider storage costs that are inversely proportional to the average distance of the DC from the stores. We consider three settings, one with no storage costs, one where only the most distant DC incurs costs, and one where all DCs need to lease space. Costs are proportional to the quantities stored and are therefore influenced by the allocation policy.

All the tests were performed using the 75-series boost set. The focus of the analysis is not on the increase in optimal costs, since an increase in the costs of the decision variables clearly leads to an increase in the total costs. Rather, the focus is on the variation in the complexity of the search, as reflected in the CPU time required to reach the final result, and on the optimality or feasibility of the solutions. To evaluate the effectiveness of increasingly lengthy searches, we present results after 600, 1200, and 3600 seconds of CPU time. The results are summarized in the table 2. All results are based on 15 repetitions, and the columns show:

- n: number of clients (stores);
- m: number of servers (DCs);
- n_{boost}: number of series in the boostset;
- n_q: number of servers incurring leasing costs;
- n_{inf}: number of infeasible solutions at the end of the search;
- GAP lin: average gap between the linear bound and the best solution at the end of the search;
- GAP fin: average gap between the best lower bound and the best solution at the end of the search;
- t.CPU: average CPU time (in sec) of the search process.

Table 2 illustrates an interesting feature of the fitness landscape of this split-cost allocation problem: the complexity of the search is not monotonic with respect to the number of leased servers. In fact, CPLEX was able to solve all instances to optimality for 0, 3 and 4 leased servers. However, for 1 or 2 leased servers, it was never able to prove the optimality of the best solution found within the time limit of 3600 seconds. Furthermore, instances with one leased server are more

challenging than those with two, as demonstrated by the gap between the lower and upper bounds, both between the linear relaxation bound and the best solution found and between the best final lower bound and the best solution. This feature is discussed further in Appendix B.

As a final note, we point out that the objective function was structured to lexicographically first strive for feasibility and then optimize within the feasible domain. In fact, all instances have always resulted in feasible solutions. Reducing the cost coefficient of the request slack variables allows us to explore the trade-off between solution cost and the probability of reaching an infeasible solution. However, we did not explore this further.

Test case solution

To further evaluate the benefits of the proposed approach, we compared the effectiveness of the robust, BESO-optimized solution with that obtained using a parametric method which assumed a Gaussian distribution of forecasts from the request datasets. This comparison was conducted using scenarios derived from a Gaussian distribution based on the confidence intervals of the forecasts provided by an Holt-Winters ETS model [30].

Figure 6 shows the distributions of the forecasts of the 75 boostset obtained by BESO and by ETS. Note that the BESO distribution, given its MEB foundation, spans a wider range of values than the ETS distribution.

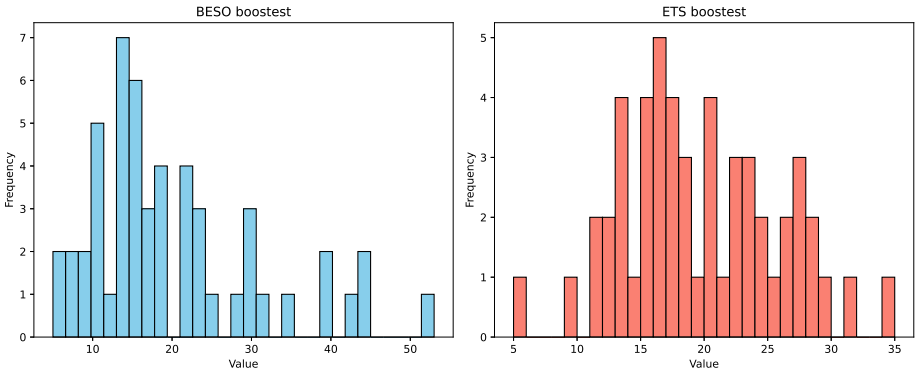


Figure 6: Comparison of distributions on 75 bootstrapped series

The wider distribution has an impact on the proposed allocation. We had empirical proof of this when we tested the BESO and ETS solutions on the validation data obtained by the end of the year — i.e. when we finally received the data for which we were asked to provide a forecast.

Table 3 presents a comparison of solutions in the first two columns when no rental cost was added for the use of leased floor space and in the last two columns when the rental cost was included in the objective function. As expected, given that the constraints were less restrictive, the ETS solution is less expensive in both cases. However, in both cases, the solution was infeasible as it did not provide all the goods requested by one of the retailers. This result was obtained in a single instance; therefore, it may not have happened. However, it is significant that, even in the first

	No rental cost		With rental cost	
	BESO	ETS	BESO	ETS
Assign. Cost	17781	15423	5446720	4378380
Optimality gap%	0	0	16.07	0.69
Infeasibilities	0	1	0	1
t.Cpu	33	29	3600	3600

Table 3: Solution cost with and without rental cost

	No rental cost		With rental cost	
	BESO	ETS	BESO	ETS
DC 0	67 / 120	119 / 120	82 / 120	116 / 120
DC 1	571 / 1000	430 / 1000	532 / 1000	428 / 1000
DC 2	257 / 300	301 / 300	257 / 300	303 / 300
DC 3	128 / 180	173 / 180	152 / 180	176 / 180

Table 4: DC usage with and without rental cost

instance in which we tested our model, its robustness proved important in dealing with a limit case.

Table 4 details the proposed usage of the DCs in the different cases. It appears that DC 2, the biggest owned one, is the overloaded one in both cases, even though by small amounts.

Finally, we note that Table 3 shows that including the rental cost makes otherwise identical instances much harder to solve. In fact, the time limit of 3600 CPU seconds was too short to reach optimality, and a duality gap was left in the proposed solutions, a substantial one in the case of the BESO instance.

7. Conclusions and future directions

This paper introduces Bootstrap Enhanced Scenario Optimization (BESO), a framework for establishing predictive prescriptions when reliable data series are available. The method couples a statistically principled forecasting engine with scenario-based optimization by generating future scenarios through bootstrapped autoregressive time series, and explicitly accounts for feedback from the downstream optimization phase in the forecasting process. This is demonstrated through a two-echelon logistics case study, in which bootstrapped time series are used to forecast retailer requests and guide the allocation of stores to distribution centers (DCs). By capturing the empirical dynamics embedded in historical data, BESO supports the derivation of inventory requirements, DC capacity decisions and allocation strategies that reflect cost structures and forecast uncertainty.

A key contribution of BESO lies in offering a credible alternative to prevailing Distributionally Robust Optimization (DRO) methodologies. Classical DRO approaches rely on ambiguity sets defined through moment bounds, Wasserstein balls, or ϕ-divergences, which—while theoretically elegant—impose distributional geometry that coul not match real empirical patterns, especially when the data exhibit nonstationarity, heavy tails, or structural breaks. In contrast, BESO avoids the explicit construction of ambiguity sets by directly generating empirical scenario distributions

through the Maximum Entropy Bootstrap (MEB). Moreover, the framework allows for a lightweight objective-aware extension of MEB, in which elements derived from downstream optimization costs can be incorporated into the entropy maximization step, thereby biasing the resampling distribution toward patterns that are more related to decision quality. This produces scenario sets that are simultaneously statistically coherent and operationally aligned, thus offering a data-centric and computationally transparent alternative to established DRO formulations. In addition, the bootstrap-set cardinality provides an explicit and tractable lever on the effective ambiguity geometry, enabling controlled modulation of dispersion and robustness without resorting to exogenous parametric set design.

Through both real logistics data and controlled artificial instances, we show how BESO yields interpretable and robust prescriptions. The artificial benchmarks also revealed how the complexity of an instance depends on its different elements, outlining the fitness landscape and illustrating the framework's adaptability to diverse uncertainty environments. These results suggest that integrating forecasting and optimization within a unified empirical loop can offer a solution stability and an operational performance that rivals classical scenario-generation or DRO-based approaches.

Several promising research directions emerge. Future work could explore incorporating MEB into preprocessing pipelines, such as Box–Cox transformations, to enhance predictive accuracy. Ensemble strategies, including weighted or Bayesian model averaging, may strengthen robustness in environments with shifting dynamics. On the methodological level, deeper investigation into bootstrap-driven ambiguity representations could position BESO as a data-implicit contribution to DRO, retaining robustness while avoiding restrictive geometric assumptions. Finally, the closure properties of the MEB framework could facilitate the development of hybrid schemes that integrate external knowledge or Bayesian priors, while further refining the incorporation of optimization-aware elements within the resampling process. Crucially, further work is required to extend BESO's applicability to environments where reliable time-series forecasting models cannot be established, thereby removing the current framework's reliance on predictable data dynamics.

Appendix A. Bootstrap sets error costs

This section reports the absolute values of the various cost functions, which were calculated using the forecast errors obtained by averaging the corresponding errors for each of the 52 test case series described in the paper.

The functions reported in the columns are: bias, Means Absolute Percentage Error (MAPE), Mean Error (ME), Mean Absolute Error (MAE), Mean Percentage Error (MPE) and Root Mean Square Error (RMSE). Table A.5 gives the cost values for the case of a boost set composed of 75 series, table A.6 for the 125 series and table A.7 for the 175 series sets.

All tables show significantly consistent results across all boost set sizes.

model	bias	MAPE	ME	MAE	MPE	RMSE
fcast_50	-0.27	0.09	-1.02	1.63	-0.05	2.12
fcast_avg	0.02	0.07	0.07	1.25	0.01	1.68
yar	0.36	0.34	1.36	5.86	0.07	9.75
yhw	0.63	0.34	2.41	5.99	0.13	10.59
ysvm	0.69	0.36	2.66	6.20	0.14	10.99
ylstm	0.56	0.35	2.13	6.01	0.11	10.41
ymlp	0.47	0.36	1.79	6.26	0.09	10.60
yrf	1.50	0.44	5.73	7.79	0.31	13.54
yxgb	2.28	0.58	8.74	10.12	0.49	17.05
yarima	0.44	0.34	1.67	5.80	0.09	9.77

Table A.5: Error costs for the 75 series boostset.

model	bias	MAPE	ME	MAE	MPE	RMSE
fcast_50	-0.56	0.09	-1.08	1.59	-0.06	2.25
fcast_avg	0.02	0.07	0.05	1.18	0.01	1.65
yar	-0.59	0.25	-1.13	4.36	-0.07	6.31
yhw	-0.13	0.24	-0.26	4.33	-0.02	6.39
ysvm	-0.06	0.24	-0.11	4.18	-0.01	6.39
ylstm	-0.30	0.24	-0.58	4.15	-0.04	6.34
ymlp	-0.47	0.24	-0.90	4.22	-0.05	5.99
yrf	1.27	0.28	2.44	5.01	0.14	7.48
yxgb	2.52	0.38	4.85	6.52	0.28	9.46
yarima	-0.40	0.25	-0.76	4.44	-0.05	6.83

Table A.6: Error costs for the 125 series boostset.

Appendix B. Extended deterministic benchmark set

All analyses reported in sections 6.1 and 6.2 refer to the case study at the heart of this paper, but the interest of the method is not limited to this particular case study as it can be effective on any instance of this 2-echelon supply chain problem. To test the generality of the results, we generated a benchmark set with varying dimensions and coefficients. The study was conducted only with respect to the deterministic setting, and is therefore significant for assessing the computational complexity of the split-allocation problem as a deterministic combinatorial optimization problem.

We generated 5 instances for each configuration of number of clients (n), number of servers (m), and number of splittable clients. The travel costs were obtained by randomly selecting m rows and n columns from a large distance matrix, whose values were computed as the actual road distances between points located in the same area as the test case. Server inventory costs were inversely proportional to the average travel cost from the server to the clients and client requests were inversely proportional to the average travel cost from the client to the servers. Note that the distance matrix is asymmetric, as it was computed within a city area. The number n_b of splittable clients was progressively increased as the number n_q of servers with nonzero inventory costs. Specifically, the range of the defining parameters were:

- n: {50, 52, 100, 200, 300};
- m: {4, 5, 10, 25, 50};
- n_b: {0, 4, 5, 10, 25, 50}

model	bias	MAPE	ME	MAE	MPE	RMSE
fcast_50	-0.04	0.07	-0.07	1.22	0	1.51
fcast_avg	0.99	0.12	1.90	2.04	0.12	2.76
yar	-0.22	0.35	-0.42	5.84	-0.03	7.64
yhw	0.25	0.34	0.48	5.75	0.02	7.90
ysvm	0.30	0.33	0.58	5.66	0.03	7.76
ylstm	0.10	0.34	0.20	5.76	0	7.74
ymlp	-0.19	0.34	-0.36	5.69	-0.02	7.57
yrf	1.72	0.37	3.31	6.39	0.18	9.66
yxgb	3.01	0.46	5.78	7.96	0.33	11.83
yarima	-0.04	0.34	-0.08	5.69	-0.01	7.63

Table A.7: Error costs for the 175 series boostset.

- n_q: {0, 1, 2, 3, 4, 5}

Not all combinations of values were generated, the larger values were used only for instances with higher number of clients. In particular, the values 4 and 52 were only used to generate instances with the same dimensionality as the case study, in order to check the generality of the results obtained. The first instance of this set was kept to be the case study one.

The computational results in the case of no costly server, reported as averages over the 5 generated instances, are summarized in table B.8, whose columns show:

- n, m: number of clients and number of servers;
- n_b: maximum number of splittable clients;
- *ncols*, *nrows*: number of columns and number of rows in formulation FD;
- *gaplb*: average percentage gap between the linear relaxation lower bound and the cost of the best solution found;
- $gapfinal$: average percentage gap between the best lower bound at the end of the search (smallest cost of an unexpanded node) and the cost of the best solution found;
- $ninf$: number of instances for which no feasible solution was found;
- *nopt*: number of instances solved to proven optimality;
- *tcpu*: average CPU time for solving the instance, time to optimality or 3600 if optimality could not be proven;

The table shows two trends. One, with respect to instance size, shows an expected increase in complexity. While instances the size of our test case are easily solved to optimality by modern solvers, increasing the size, as defined by the number of variables and the number of constraints, quickly leads to instances that could not be solved to optimality within the 3600-second time limit. The number of splittable clients has an important effect on the complexity of the instance. Instances of the same absolute size, but with more splittable clients, usually have a looser linear relaxation (LP) bound and have more difficulty reaching optimality, as can be seen by the decreasing number of instances out of the 5 from each group that could be solved to optimality.

Note that the optimality gap of the LP bound is usually small, except for the n=100 and m=50 instances, but the search still struggles to close this gap. Interestingly, the LP bound gets tighter as n increases, but this does not make the search any easier. The least-cost solution when optimality

n	m	n_b	$ncols$	$nrows$	$gaplb$	$gapfinal$	$ninf$	$nopt$	$tcpu$
52	4	0	416	316	0.21	0.01	0	5	0
50	5	0	500	355	0.39	0.00	0	5	3
50	5	5	500	355	0.40	0.00	0	5	4
50	10	0	1000	610	1.32	0.00	0	5	383
50	10	5	1000	610	1.34	0.00	0	5	378
50	10	10	1000	610	1.35	0.13	0	4	1150
100	10	0	1000	1210	0.50	0.12	0	2	1920
100	10	5	1000	1210	0.54	0.05	0	3	1514
100	10	10	1000	1210	0.57	0.09	0	3	2063
100	50	0	10000	5250	5.23	3.06	0	0	3600
100	50	5	10000	5250	4.77	2.70	0	0	3600
100	50	10	10000	5250	4.55	2.42	0	0	3600
100	50	25	10000	5250	4.52	2.61	0	0	3600
200	10	0	4000	2410	0.17	0.05	0	2	2369
200	10	5	4000	2410	0.17	0.09	0	1	2893
200	10	10	4000	2410	0.18	1.01	0	0	3600
200	50	0	20000	10450	1.66	1.26	0	0	3600
200	50	10	20000	10450	1.68	1.29	0	0	3600
200	50	25	20000	10450	1.71	1.37	0	0	3600
300	10	0	6000	3610	0.09	0.03	0	3	1798
300	10	5	6000	3610	0.09	0.07	0	0	3600
300	10	10	6000	3610	1.02	0.47	0	0	3600
300	50	0	30000	15650	0.94	0.73	0	0	3600
300	50	10	30000	15650	0.93	0.76	0	0	3600
300	50	25	30000	15650	0.96	0.84	0	0	3600

Table B.8: Extended benchmark set, deterministic case, no costly server

was not proven was usually found within the first 600 seconds of CPU time. However, we note that designing efficient heuristics for this problem is beyond the scope of this research. We could have tweaked the solver configuration to favor heuristic search at the expense of efficiency in proving optimality, or designed custom heuristics, but the focus here is on assessing the fitness landscape when road network costs are involved.

Table B.9 reports results about tests varying the number of costly servers, the columns show:

- n, m: number of clients and number of servers;
- $ncols$, $nrows$: number of columns and number of rows in formulation FD;
- lb0, t0: linear relaxation bound and CPU time in the case of no costly servers;
- lb1, t1: linear relaxation bound and CPU time in the case of 1 costly server;
- lb2, t2: linear relaxation bound and CPU time in the case of 2 costly servers;
- lb3, t3: linear relaxation bound and CPU time in the case of 3 costly servers;
- lb4, t4: linear relaxation bound and CPU time in the case of 4 costly servers;
- lb5, t5: linear relaxation bound and CPU time in the case of 5 costly servers;

The table reports results on two sets of instances. The top 4 rows contain aggregate results on a set of instances, 5 for each configuration, where inventory costs are directly proportional to average travel distances, while the bottom 4 rows have inventory costs inversely proportional to average travel distances.

The complexity of solving instances increases non-monotonically with the number of costly

n	m	ncols	nrows	lb0	t0	lb1	t1	lb2	t2	lb3	t3	lb4	t4	lb5	t5
52	4	416	316	0.86	0.2	0.17	0.81	0.08	0.22	0.01	0.02	0.00	0.02	-	-
50	5	500	355	1.33	0.06	2.77	0.06	1.46	0.13	0.11	14.31	0.06	7	0.05	0.93
100	5	1000	705	0.41	0.24	0.09	6.39	0.06	3.2	0.01	0.06	0.02	4.8	0.02	2.71
200	5	2000	1405	0.14	1.43	0.01	1.79	0.01	0.25	0.01	0.17	0.01	0.11	0.01	7.84
52	4	416	316	0.86	0.36	0.24	0.22	0.22	0.14	0.26	0.09	0.05	1.9	-	-
50	5	500	355	1.33	0.16	0.52	0.07	0.50	5.8	0.13	10.11	0.05	1.37	0.02	0.22
100	5	1000	705	0.41	0.33	0.19	8.58	0.04	6.14	0.02	0.41	0.02	0.58	0.02	3.78
200	5	2000	1405	0.14	1.55	0.07	39.07	0.04	9.44	0.01	0.12	0.02	4.48	0.01	6.66

Table B.9: Extended benchmark set, deterministic case, increasing number of costly servers

servers, for both increasing and decreasing inventory costs. While the 0-cost instances are comparatively easy to solve, the CPU time required to prove optimality increases with the addition of the first costly servers and then decreases as more servers are paid to store inventory. However, this is not true of the gap between the linear relaxation bound and the cost of the optimal solution, where the instances with no costly servers always have the largest gap. As expected, the CPU time required to prove optimality increases with the dimension of the case, and the case with an inverse correlation between storage cost and travel cost is more difficult to solve than the case with a positive correlation.

References

[1] Aviv, Y., 2001. The effect of collaborative forecasting on supply chain performance. Management Science 47, 1326–1343.

[2] Bayraksan, G., Morton, D.P., 2011. A sequential sampling procedure for stochastic programming. Operations research 59, iii–1058.

[3] Beale, E.M.L., 2018. On minimizing a convex function subject to linear inequalities. Journal of the Royal Statistical Society: Series B (Methodological) 17, 173–184.

[4] Ben-Tal, A., Nemirovski, A., 1998. Robust convex optimization. Mathematics of Operations Research 23, 769–805.

[5] Berger, J., 1985. Statistical Decision Theory and Bayesian Analysis. Springer.

[6] Bertsimas, D., Kallus, N., 2019. From predictive to prescriptive analytics. Management Science 66, 1025–1044. doi:10.1287/mnsc.2018.3253.

[7] Bertsimas, D., Sim, M., 2004. The price of robustness. Operations Research 52, 35–53.

[8] Beyrouthy, C., Burke, E.K., Landa-Silva, D., McCollum, B., McMullan, P., Parkes, A.J., 2007. The teaching space allocation problem with splitting, in: Burke, E.K., Rudová, H. (Eds.), Practice and Theory of Automated Timetabling VI, Springer, Heidelberg. pp. 228–247.

[9] Brockwell, P.J., Davis, R., 1991. Time Series: Theory and Methods. Springer.

[10] Clark, A., Scarf, H., 1960. Optimal policies for a multi-echelon inventory problem. Management Science 6, 475–490.

[11] Dantzig, G., 1955. Linear programming under uncertainty. Management Science 1, 197–206.

[12] Davison, A.C., Hinkley, D.V., 1997. Bootstrap Methods and Their Application. Cambridge University Press.

[13] de Kok, T., Grob, C., Laumanns, M., Minner, S., Rambau, J., Schade, K., 2018. A typology and literature review on stochastic multi-echelon inventory models. European Journal of Operational Research 269, 955–983.

[14] Delage, E., Ye, Y., 2010. Distributionally robust optimization under moment uncertainty with application to data-driven problems. Operations Research 58, 595–612.

[15] Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine learning research 7, 1–30.

[16] Donti, P., Amos, B., Kolter, J.Z., 2017. Task-based end-to-end model learning in stochastic optimization, in: Guyon, I., Luxburg, U.V., et al., S.B. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc.

[17] Dror, M., Trudeau, P., 1989. Savings by split delivery routing. Transportation Science 23, 141–145.

[18] Efron, B., 1979. Bootstrap methods: Another look at the jackknife. The Annals of Statistics 7, 1–26.

[19] Efron, B., Tibshirani, R.J., 1993. An Introduction to the Bootstrap. Chapman and Hall/CRC.

[20] Elmachtoub, A.N., Grigas, P., 2022. Smart "predict, then optimize". Management Science 68, 9–26. URL: https://doi.org/10.1287/mnsc.2020.3922.

[21] Eppen, G., Schrage, L., 1981. Centralized ordering policies in a multi-warehouse system with lead times and random demand, in: Schwarz, L. (Ed.), Multi-Level Production/Inventory Control Systems, North Holland. pp. 51–69.

[22] Ernst, A., Krishnamoorthy, M., 1998. Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem. European Journal of Operational Research 104, 100–112.

[23] Escorcia-Caballero, J., Amaya-Mier, R., Soto-Ferrari, M., Chams-Anturi, O., 2020. Multi-echelon inventory management policies: A case study for a two-echelon supply chain, in: proc. International Conference on Industrial Engineering and Operations Management, pp. 165–175.

[24] Esfahani, P.M., Kuhn, D., 2017. Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. URL: https://arxiv.org/abs/1505.05116.

[25] Feliu, J.M., 2024. Enhancing demand forecasting: an analysis of factors impacting sales and implementation of improved methodologies for accurate prediction. B.S. thesis. Universitat Politècnica de Catalunya.

[26] Fenga, L., Politis, D., 2011. Bootstrap-based arma order selection. Journal of Statistical Computation and Simulation .

[27] Fenga, L., Politis, D.N., 2017. Lasso order selection for sparse autoregression: a bootstrap approach. Journal of Statistical Computation and Simulation 87, 319–334.

[28] Hammersley, J., 1960. Monte carlo methods for solving multivariable problems. Annals of the New York Academy of Sciences 86, 844–874.

[29] Hasni, M., Aguir, M., Babai, M.Z., Jemai, Z., 2019. Spare parts demand forecasting: a review on bootstrapping methods. International Journal of Production Research 57, 4791–4804.

[30] Hyndman, R., Koehler, A., Ord, J., Snyder, R., 2008. Forecasting with Exponential Smoothing: The State Space Approach. Springer, Switzerland.

[31] Jaynes, E., 1957. Information theory and statistical mechanics. Physical Review 106, 620–630.

[32] Kim, D., Kim, S., 2010. A multi-echelon inventory model for supply chain management using the supply hub in industrial park (ship) concept. International Journal of Production Economics 123, 1–12.

[33] Kreiss, J.P., Lahiri, S.N., 2012. Bootstrap methods for time series, in: Handbook of statistics. Elsevier. volume 30, pp. 3–26.

[34] Ma, Z., Li, H., Wang, Y., 2015. Combining forecasts in supply chain forecasting: A review. Omega .

[35] Maniezzo, V., Boschetti, M., Stützle, T., 2021. Matheuristics, algorithms and implementations. EURO Advanced Tutorials on Operational Research, Springer, New York.

[36] Maniezzo, V., Zhou, T., 2023. Integrated forecast and optimization for retailer allocation in a two-echelon inventory system, in: Bringas, P., et al. (Eds.), proc. of 18th Int. Conf. on Soft Computing Models in Industrial and Environmental Applications, Springer. pp. 279 – 289.

[37] Marín, A., 2005. Formulating and solving splittable capacitated multiple allocation hub location problems. Computers & Oper. Research 32, 3093–3109.

[38] Mokhtar, H., Krishnamoorthy, M., Ernst, A.T., 2019. The 2-allocation p-hub median problem and a modified benders decomposition method for solving hub location problems. Computers & Operations Research 104, 375–393.

[39] Papier, F., 2016. Supply allocation under sequential advance demand information. Operations Research 64, 341–361.

[40] Römisch, W., 2018. Stochastic Programming, Scenario Generation In. John Wiley and Sons, Ltd. pp. 1–5.

[41] Sarris, D., Spiliotis, E., Assimakopoulos, V., 2020. Exploiting resampling techniques for model selection in forecasting: an empirical evaluation using out-of-sample tests. Operational Research 20, 701–721.

[42] Scikit, 2024. posthocs critical difference diagram. Https://scikit-posthocs.readthedocs.io/en/latest/generated/ scikit_posthocs.critical_difference_diagram.html.

[43] Shannon, C., 1948. A mathematical theory of communication. Bell System Technical Journal 27, 379–423.

[44] Shapiro, A., Dentcheva, D., Ruszczyński, A., 2009. Lectures on Stochastic Programming: Modeling and Theory. SIAM.

[45] Shapiro, A., Homem-de Mello, T., 1998. A simulation-based approach to two-stage stochastic programming with recourse. Mathematical Programming 81, 301–325.

[46] Shenstone, A.G., Dekker, R., 2001. A literature review on approaches to the forecasting of intermittent demand. Journal of the Oper. Research Society .

[47] Vinod, H.D., Kourentzes, N., 2006. Maximum entropy bootstrap for predictive inference. Journal of Forecasting 25, 49–63.

[48] Youngjin, K., Youngho, L., Junghee, H., 2011. A splitter location–allocation problem in designing fiber optic access networks. Eur. Journal of Operational Research 210, 425–435.

[49] Zhu, S., Hu, X., Huang, K., Yuan, Y., 2021. Optimization of product category allocation in multiple warehouses to minimize splitting of online supermarket customer orders. Eur. Jou. of Oper. Research 290, 556–571.