PRESCRIPTIVE ANALYTICS IN MARITIME TRANSPORTATION

XUECHENG TIAN

PhD

The Hong Kong Polytechnic University

2024

The Hong Kong Polytechnic University

Department of Logistics and Maritime Studies

Prescriptive Analytics in Maritime Transportation

Xuecheng Tian

A thesis submitted in partial fulfilment of the
requirements for the degree of Doctor of Philosophy

May 2024

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Xuecheng Tian (Name of student)

Abstract

The maritime transportation sector is increasingly relying on optimization and analytics to maximize the potential of the extensive data generated in the industry. This thesis leverages real-world data to optimize ship maintenance planning, port state control officer (PSCO) routing, and container ship bunkering decisions by using advanced prescriptive analytics methodologies.

The first study introduces a smart predict-then-optimize (SPO) framework, using an ensemble of SPO trees (SPOTs) for ship maintenance planning. This method significantly enhances the traditional predict-then-optimize (PO) approach by integrating operational, repair, and risk costs into the machine learning (ML) model. It effectively reduces total operating expenses of ship inspections by approximately 1% over the PO-based scheme and at least 3% over non-ML-based schemes. The SPO-based plans also contribute to more efficient port operations by reducing the necessity for intensive PSC inspections, thus alleviating port congestion.

The second study explores the PSCO routing problem in the face of uncertain ship conditions. By embedding this optimization problem into the ML model training process through a decision-focused learning framework, this study addresses the limitations of the traditional two-stage framework that separates prediction

and optimization. A compact model using undominated inspection templates and a surrogate decision loss function based on noise-contrastive estimation enhances solution efficiency and decision accuracy, underscoring the importance of integrating decision making directly into the learning process.

The third study optimizes container ship bunkering decisions under uncertain fuel prices. It employs a two-channel long short-term memory model to capture spatiotemporal correlations among multi-port fuel prices, significantly outperforming traditional predictive models. Two prescriptive analytics frameworks are compared: a two-stage contextual deterministic programming model and a multistage contextual stochastic programming model. The suitability of these frameworks varies based on the variance of inter-port fuel prices and the number of ports on each shipping service, offering crucial insights for maritime operators.

This thesis collectively demonstrates the efficacy of integrating ML with mathematical optimization to address critical operational challenges in maritime transportation, leading to substantial economic benefits and operational efficiencies. The approaches applied in this thesis pave the way for innovative practices in maritime logistics, emphasizing the transformative potential of data-driven decision making in complex operational contexts.

Keywords: maritime transportation; port state control; routing; container shipping; prescriptive analytics; data-driven optimization; machine learning; stochastic programming.

Publications and Working Papers

Arising from the Thesis

- Tian, X., Yan, R., Wang, S., Liu, Y., 2023. Tutorial on prescriptive analytics for logistics: What to predict and how to predict. Electronic Research Archive 31(4), 2265–2285.
- Tian, X., Yan, R., Liu, Y., Wang, S., 2023. A smart predict-then-optimize method for targeted and cost-effective maritime transportation. Transportation Research Part B: Methodological 172(2), 32–52.
- Tian, X., Yan, R., Wang, S., Laporte, G., 2023. Prescriptive analytics for a maritime routing problem. Ocean & Coastal Management 242(1), 106695.
- Tian, X., Wang, S., Liu, Y., Yang, Y. Data-driven optimization for container ship bunkering management under fuel price uncertainty. Under major revision at Transportation Research Part B.

Others

- Tian, X., Wang, S., Laporte, G., Yang, Y., 2024. Determinism versus uncertainty: Examining the worst-case expected performance of data-driven policies.

European Journal of Operational Research 318(1), 242–252.

- Tian, X., Shangguan, Y., Pang, K.W., Guo, Y., Lyu, M., Wang, S., Huang, G.Q. Carbon emission allocation policy making in liner shipping: A novel approach toward equitable and efficient maritime sustainability. Ocean & Coastal Management, in press.

- Wang, T., Meng, Q., Tian, X., 2024. Dynamic container slot allocation for a liner shipping service. Transportation Research Part B: Methodological 174, 102874.

- Shangguan, Y., Tian, X., Pang, A., Ruan, Q., Jin, Y., Wang, S, 2024. Navigating the green shipping: Stochastic hydrogen hub deployment in inland waterways. Transportation Research Part D: Transport and Environment 129, 104126.

- Yan, R., Tian, X., Wang, S., Peng, C., 2024. Development of computer vision informed container crane operator alarm methods. Transportmetrica A: Transport Science 20(2), 2145862.

- Wang, H., Yi, W., Tian, X., Zhen, L., 2023. Prescriptive analytics for intelligent transportation systems with uncertain demand. ASCE Journal of Transportation Engineering, Part A: Systems 149(12), 04023118.

- Tian, X., Wang, S., Zhen, L., Shen, M.Z.J. k-Tree: Crossing sharp boundaries to find neighbors. Submitted.

- Tian, X., Wang, S., Laporte, G., Yang, Y. Contextual stochastic optimization: machine learning-driven estimation of conditional distributions. Submitted.

- Tian, X., Wang, S., Laporte, G. Multioutput shrunken regression tree. Submitted.

Acknowledgements

As I organize my thesis, revisiting previously published or under-review studies, I often discover new issues—such as imprecise mathematical formatting and unclear statements—that I may not have recognized a year or two ago. These experiences mark tangible traces of my PhD journey, making me profoundly grateful for the support that has made them possible.

First, I would like to express my deepest gratitude to my supervisor, Prof. Shuaian (Hans) Wang. Hans is the most intelligent and hard-working individual I have encountered. His passion for absorbing and transferring knowledge has provided me with many innovative and unexpected perspectives. I will forever cherish the exciting moments when he shared his creative ideas for my research, including the k-Tree universe and a series of insights on deficiencies in prescriptive analytics frameworks. Hans's patience and care for his students have been a constant source of encouragement, especially when I faced challenges. I am incredibly fortunate to have been supervised by him during my PhD studies. His expertise and kindness have made this journey smoother and faster than I anticipated. I hope to continue learning from Hans and aspire to emulate his creativity, rigor, and lifelong dedication to research.

Additionally, I am thankful to Prof. Tingsong Wang for recommending me for

the PhD program and for his guidance during my master's studies. I extend my appreciation to all of my co-authors, particularly Prof. Gilbert Laporte from HEC Montréal and Dr. Ran Yan from Nanyang Technological University, for their efforts and valuable advice. I am also grateful to all of the faculty and administrative members at LMS for fostering a supportive learning and research environment. My thanks go as well to my friends, both those far away and those I met at PolyU, who have brightened my spare time.

Last but not least, many thanks go to my parents for their endless support. Although they may not be able to understand the specifics of my research, they try to comprehend my challenges and provide understanding and support. Their unconditional love and absence of pressure have allowed me the freedom to pursue my passions.

List of Abbreviations

(**in the order of appearance**)

Abbreviations in Chapter 2

- port state control (PSC)
- port state control officer (PSCO)
- machine learning (ML)
- International Maritime Organization (IMO)
- predict-then-optimize (PO)
- smart predict-then-optimize (SPO)
- smart predict-then-optimize tree (SPOT)
- smart predict-then-optimize forest (SPOF)
- random forest (RF)
- ship risk profile (SRP)
- tree augmented naive (TAN)
- support vector machine (SVM)
- Bayesian network (BN)
- Memorandum of Understanding (MoU)
- weighted sample average approximation (w-SAA)

- classification and regression tree (CART)
- area under the receiver operating characteristic curve (ROC AUC)
- random inspection scheme (RIS)
- no inspection scheme (NIS)
- overall inspection scheme (OIS)
- predict-then-optimize scheme (POS)
- smart predict-then-optimize scheme (SPOS)

Abbreviations in Chapter 3

- International Maritime Organization (IMO)
- port state control (PSC)
- port state control officer (PSCO)
- Kwai-Tsing Container Terminal (KTCT)
- River Trade Terminal (RTT)
- machine learning (ML)
- noise-contrastive estimation (NCE)
- mean squared error (MSE)
- artificial neural network (ANN)
- maximum a posteriori (MAP)
- Memorandum of Understanding (MoU)

Abbreviations in Chapter 4

- high-sulfur fuel oil (HSFO)
- metric tonne (MT)

- International Maritime Organization (IMO)
- very low-sulfur fuel oil (VLSFO)
- machine learning (ML)
- two-channel long short-term memory (TC-LSTM)
- Monte Carlo (MC)
- sample average approximation (SAA)
- centistoke (CST)
- liquefied natural gas (LNG)
- recurrent neural network (RNN)
- predict-then-optimize (PTO)
- estimate-then-optimize (ETO)
- Two-stage contextual Deterministic framework with Point predictions (TDP)
- Multistage contextual Stochastic framework with Distributional estimates (MSD)
- spatial-temporal (ST)
- intra-sequence temporal (IST)
- inter-sequence spatial (ISS)
- fully connected (FC)
- root mean square error (RMSE)
- mean absolute error (MAE)
- mean absolute percentage error (MAPE)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Maritime transportation is a backbone of global trade, accounting for over 80% of the world's trade by volume due to its cost efficiency and large capacity (Ng, 2015; Sun and Zheng, 2016). The complexity of the maritime transportation network, consisting of interconnected routes and ports, supports extensive activities such as cargo handling, logistics operations, and ship maintenance. However, this sector is characterized by significant uncertainties including unpredictable sea and ship conditions, variable shipping demand, and fluctuating fuel prices. These uncertainties challenge traditional deterministic models used in maritime logistics, which often fail to capture the uncertain nature of maritime operations, leading to suboptimal decision making and inefficiencies.

Ports act as critical nodes within the shipping network where operational delays and inefficiencies can significantly affect the global supply chain. Factors like uncertain ship arrival times, variable cargo volumes, and limited port facilities con-

tribute to operational uncertainties. For example, delays in one port can propagate through the network, causing further delays and increased costs. Furthermore, the volatile nature of fuel prices adds a layer of economic uncertainty, influencing operational strategies such as ship routing, sailing speed, and bunkering decisions, which in turn impacts fuel consumption and greenhouse gas emissions.

Like many other industries, maritime transportation is more and more driven by optimization and analytics to make the best out of the wealth of data generated through modern technologies (Fagerholt et al., 2023). Figure 1.1 depicts the workflow of business analytics (He et al., 2022). The workflow is motivated by the business problem, which consists of the collection, preprocessing, and interpretation of the data, the selection and refinement of predictive analytics methods, and the modeling for decision making in prescriptive analytics. The integration of prescriptive analytics into maritime transportation offers a transformative potential to tackle uncertain optimization problems effectively. By combining advanced data analytics, machine learning, and optimization techniques, prescriptive analytics provides actionable insights that enable shipping companies and port authorities to make more informed decisions.

This thesis aims to explore the implementation of prescriptive analytics across three critical areas within maritime transportation: designing cost-effective and targeted ship maintenance plans for ship operators, enhancing ship inspection routing to maximize inspection benefits, and managing ship bunkering plans to mitigate the effects of fuel price volatility. Each of these areas represents a vital aspect of maritime operations where prescriptive analytics can bring substantial benefits, driving the industry towards more intelligent practices.

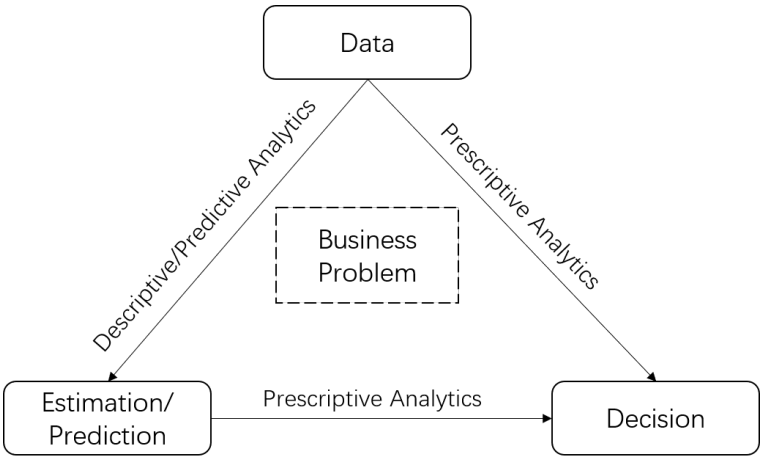

Figure 1.1: A general workflow of business analytics (He et al., 2022)

1.2 Thesis Outline

This thesis comprises three studies that demonstrate the application of prescriptive analytics in different aspects of maritime transportation. The studies are structured as follows:

- **Study 1: Prescriptive Analytics for Ship Maintenance Optimization**[1]

  Chapter 2 uses port state control inspection data to design cost-effective and targeted ship maintenance plans for ship operators. This study designs a smart predict-then-optimize framework that innovatively incorporates the operational, repair, and risk costs into the training process of the machine learning model. By optimizing maintenance schedules based on decision-oriented predictions, this approach aims to minimize operational costs, reduce detention rates, and enhance the efficiency of inspections.

[1]This study has been published: Tian, X., Yan, R., Liu, Y., Wang, S., 2023. A smart predict-then-optimize method for targeted and cost-effective maritime transportation. Transportation Research Part B: Methodological 172, 32–52.

- **Study 2: Prescriptive Analytics for a Maritime Routing Problem**[2]

  Chapter 3 explores the optimization of port state control officer routing—a key problem in operational logistics for port authorities. Traditional methods often do not consider the impacts of ship deficiency prediction outcomes on inspector routing efficiency. This study designs a decision-focused learning framework that embeds the optimization problem directly into the machine learning training process. Consequently, this study improves the prescriptive accuracy for inspector routing decisions and ultimately reduces unnecessary inspections and port congestion.

- **Study 3: Prescriptive Analytics for Container Ship Bunkering Optimization**

  Chapter 4 tackles the optimization of container ship bunkering decisions in the face of uncertain fuel prices. Given the volatility of fuel prices and their significant impact on operational costs, this study uses two prescriptive frameworks to model and optimize these decisions. The study assesses a two-stage contextual deterministic programming model and a multistage contextual stochastic programming model, examining their effectiveness in various scenarios. This comparison aims to identify the most effective strategy to minimize fuel costs and ensure economic efficiency in maritime operations.

At last, Chapter 5 concludes the thesis and discusses several future research directions.

[2]This study has been published: Tian, X., Yan, R., Wang, S., Laporte, G., 2023. Prescriptive analytics for a maritime routing problem. Ocean & Coastal Management 242(1), 106695.

Chapter 2

Prescriptive Analytics for Ship Maintenance Optimization

2.1 Introduction

Since 1982, port authorities have used port state control (PSC) to inspect foreign visiting ships as the last line of defense against substandard ships. A ship condition found to be non-compliant with the relevant convention(s) during a PSC inspection is deemed a deficiency.[1] When PSC officers (PSCOs) identify critical deficiencies, the port state may detain the ship and may require the ship to rectify these deficiencies before departing. Most studies of PSC have used machine learning (ML) methods and PSC data to design inspection schemes for port authorities, examining how to implement PSC inspections more efficiently, and discussing the effects of PSC inspections (Yan and Wang, 2019). However, to the best of our

[1] See https://wwwcdn.imo.org/localresources/en/KnowledgeCentre/IndexofIMOResolutions/AssemblyDocuments/A.1119(30).pdf.

knowledge, no studies have been conducted to examine how ship operators can benefit from using ML technologies to analyze PSC record data.

For ship operators, ship detention is the most negative outcome of a PSC inspection, as it indicates that the ship is in poor condition and is at risk of incidents and accidents. Furthermore, ship detention delays shipping schedules (Yan et al., 2021b) and maritime transportation delays are costly. Additionally, by undermining the reputation of the ship's flag state, recognized organization, and company, ship detention leads to higher inspection rates of their ships in the future (Yan et al., 2021b). Hence, given the impact of ship detention on safety, cost, and reputation, being able to identify high-risk ships that have a greater number of deficiencies and conduct targeted ship maintenance before formal PSC inspections is of profound value to ship operators. Furthermore, targeted ship maintenance is not only beneficial to ship operators but can improve overall ship conditions, thereby reducing the resources needed for formal PSC inspections (Xu and Lee, 2016). Reducing the number of detained ships will also alleviate port congestion and improve the efficiency of port operations (Wang et al., 2018). In the long run, targeted ship maintenance will improve maritime safety, the marine environment, and the living and working conditions of the ships' crew, thus achieving the objective of PSC. Therefore, there is a need for studies using PSC data to design ship maintenance schemes that focus on ship operators' decisions about which deficiency codes need to be inspected before formal PSC inspections.

When carrying out ship maintenance before formal PSC inspections, ship operators are concerned with ship deficiencies, which are commonly assumed to be the primary cause of ship detentions (Wu et al., 2022). When ship operators have limited ship maintenance resources, they are likely to prioritize high-risk deficien-

cies that may warrant detentions. However, to the best of our knowledge, there are no guidelines for detainable deficiencies except for the general descriptions given in International Maritime Organization (IMO) Resolution A.1119(30).[2] Hence, before developing a framework for designing ship maintenance schemes, we first conduct a preliminary analysis to examine the contribution of each deficiency code to the detention outcome—the detention contribution of the deficiency items under each code. We then transform this detention contribution into the risk cost of having deficiency items under each code.

Eruguz et al. (2017) classify the different maintenance strategies as corrective (failure-based), preventive (schedule-based), and predictive (condition-based) maintenance. Our focus on how to design maintenance schemes before formal PSC inspections using ML models and PSC data falls into the third category. Traditionally, the predict-then-optimize (PO) framework is used for the third category, and it consists of two stages: 1) predicting the probability of having deficiency items under each code using ML models and 2) deciding whether to inspect each deficiency code by solving the downstream optimization model built from the predictions. In the PO framework, the prediction model focuses on minimizing the prediction error and ignores the impact of the prediction on the downstream decision. In contrast, this study focuses on the ship operators' objective of obtaining near-optimal ship maintenance decisions while paying less attention to prediction error. We thus train ML models using a loss function that minimizes the decision error by measuring the sub-optimality of the decisions generated by the predictions.

[2]See https://wwwcdn.imo.org/localresources/en/KnowledgeCentre/IndexofIMOResolutions/AssemblyDocuments/A.1119(30).pdf.

To integrate the prediction task with the optimization task, Elmachtoub and Grigas (2021) propose a smart predict-then-optimize (SPO) loss function for a broad class of decision-making problems. However, they find that training ML models using SPO loss is likely to be impossible because of the nonconvex and discontinuous characteristics of the SPO loss function. The convex surrogate loss function that they propose, referred to as the SPO+ loss, does not guarantee optimal decisions but only provides an approximation for computational feasibility. In a subsequent study, Elmachtoub et al. (2020) propose an algorithm for training decision trees using SPO loss, called SPO trees (SPOTs). In our study, we build on this tractable framework proposed by Elmachtoub et al. (2020) and construct tailored SPOTs by exploiting certain structural properties of the optimization problem analyzed in this study. To improve decision performance, we also train an ensemble of SPOTs referred to as an SPO forest (SPOF).

Our *contributions* can be summarized as follows. First, our study innovatively uses PSC data to design ship maintenance plans for ship operators. When simulating ship operators' decision-making process, we consider the impacts of ship detention on them and identify three types of operational costs associated with each deficiency code: inspection cost, repair cost, and risk cost. The ship maintenance plans consider all of the three costs and the occurrence probability of having deficiency items under each code. Second, the risk cost of each deficiency code is determined by the detention contribution of the deficiency items under each code, obtained from a random forest (RF) model of ship detention prediction. Deficiency codes with higher detainable risks are assigned higher risk costs. Third, rather than using the PO framework, we adopt an SPO framework using an ensemble of SPOTs, which minimizes the SPO loss by leveraging the optimization

problem's properties. By comparing our proposed method to other ship maintenance schemes, we demonstrate its superiority.

The remainder of this chapter is organized as follows. Section 2.2 reviews the related research. Section 2.3 introduces the preliminary concepts, including the ship maintenance planning problem, the basic introduction about ML models that we use in this study, and the method for determining the detention contribution of the deficiency items under each code. Section 2.4 depicts the traditional PO framework, and based on this framework, Section 2.5 shows the improved SPO framework. Section 2.6 describes the computational experiments used to compare different types of ship maintenance schemes. Section 2.7 concludes the study.

2.2 Literature Review

Most studies of PSC have focused on improving the inspection efficiency of port authorities, as current ship risk profile (SRP) schemes do not efficiently identify substandard ships. As a PSC inspection can result in the identification of deficiencies and detention, many studies have sought to improve inspection efficiency by developing methods for identifying ships with more deficiencies or higher detention probabilities. For example, to assist the port state in identifying high-risk ships with more deficiencies, Wang et al. (2019) develop a tree augmented naive (TAN) Bayes classifier to predict the number of deficiencies of each ship. Chung et al. (2020) and Yan et al. (2021c) use the Apriori algorithm to determine the type and sequence of deficiency codes that should be inspected. In recent years, ship deficiency prediction models have been used to inform the allocation of scarce inspection resources. That is known as the PSCO scheduling problem and has been

studied by Yan et al. (2020) and Yan et al. (2021a). In their research frameworks, Yan et al. (2020) first compare the results of three RF models with different loss functions that predicted ship deficiency numbers under four deficiency categories and then establish optimization models for efficiently matching officers' expertise with ship deficiency conditions. Subsequently, Yan et al. (2021a) improve prediction performance by integrating shipping domain knowledge into an XGBoost model, and the downstream PSCO scheduling models were modified to be more consistent with practice.

Xu et al. (2007a) are the first to develop a support vector machine (SVM) model to predict ship detention using both generic and historic factors. A subsequent study by Xu et al. (2007b) enhances the prediction performance by introducing new features extracted from web mining technology into the SVM model. Gao et al. (2008) further integrate the SVM model with the k-nearest neighbor model and bag-of-words model to predict ship detention. Yang et al. (2018a) propose a data-driven Bayesian network (BN) model based on TAN learning to predict the ship detention probability of bulk carriers under the Paris Memorandum of Understanding (MoU). To determine the optimal inspection policy for a port, Yang et al. (2018a) incorporate the results of the ship detention prediction model into a game model that considers both port authorities and ship operators. In a recent study, Wu et al. (2022) use an SVM model to predict ship detention, where input features are selected using an analytic hierarchy process and a grey relational analysis to improve prediction accuracy. Yan et al. (2021b) use a balanced RF model to predict ship detention to address the issues caused by the low-probability detention outcome.

As mentioned above, some recent studies of PSC have integrated prediction

and optimization (Yan et al., 2020, 2021a). As we also combine prediction and optimization by training SPOTs, we briefly review the research on training decision trees for personalizing decisions from a finite set of possible options. Kallus (2017) trains trees with a loss function to maximize the effectiveness of the predictions rather than minimize the prediction error. Bertsimas et al. (2019) study a similar treatment recommendation problem but adopt a weighted loss function to combine prediction and decision error. Elmachtoub et al. (2020) consider a more general class of decision-making problems that could involve a large number of decisions represented by a general feasible region. To train decision trees under SPO loss, they propose a tractable methodology called SPOTs. They claim that SPOTs could benefit from the interpretability of decision trees, allowing for an interpretable segmentation of a set of contextual features with different optimal solutions to the optimization problem of interest. In a recent study, Kallus and Mao (2022) also study how to fit the forest policies in contextual stochastic optimization problems to directly minimize the optimization costs. There are two major differences between Elmachtoub et al. (2020) and Kallus and Mao (2022). The first one is the different splitting rules for the nodes in a tree. Elmachtoub et al. (2020) split the nodes greedily, searching all possible values of all the chosen features to find the optimal split. Kallus and Mao (2022) term this method the oracle splitting criterion. However, this method may lead to the problem of burdensome re-optimization for every candidate split when applied to large-scale problems. Therefore, Kallus and Mao (2022) propose a computationally efficient node splitting method by leveraging a second-order perturbation analysis of stochastic optimization for large-scale optimization problems and datasets. It is possible that an increase in training efficiency is at the cost of a decrease in decision quality be-

cause Kallus and Mao (2022) use approximate rather than optimal splitting methods. Therefore, we apply oracle splitting rules that do not consume large computational resources for our problem because of its moderate size and structural properties, and we generate high-quality local-optimal splits. The second difference lies in the methods of prescriptive analytics. In our work, the PO and SPO frameworks still first predict the uncertain parameters in the optimization problem and then derive the final decisions. However, Kallus and Mao (2022) do not utilize the point predictions of the uncertain parameters; instead, they predict the distributions of the uncertain parameters. Their method resembles the weighted sample average approximation (w-SAA) method in Bertsimas and Kallus (2020), which is mainly suitable for nonlinear optimization objectives (Wang et al., 2022). However, the forest construction algorithms of Bertsimas and Kallus (2020) and Kallus and Mao (2022) are different, where Bertsimas and Kallus (2020) use prediction loss and Kallus and Mao (2022) use decision loss. A detailed comparison between Bertsimas and Kallus (2020) and Kallus and Mao (2022) can be found in Kallus and Mao (2022).

Our research study contributes to both the literature on PSC and the literature on maintenance and service logistics management. According to Eruguz et al. (2017), our research problem pertains to the subdomains of maintenance strategy selection and maintenance planning. The maintenance strategy selection subdomain focuses on selecting the best maintenance strategy for a system, part, or component to find the optimal balance between the benefits of maintenance and related costs (Eruguz et al., 2017; Goossens and Basten, 2015). The maintenance planning subdomain focuses on maintenance-related tasks, such as inspections, replacements, repairs, and overhauls (Eruguz et al., 2017; Giorgio et

al., 2015). As there is no related study using PSC data to design maintenance schemes, our study is a specific application of maintenance and service logistics management research, mainly tailored to PSC inspections. Furthermore, as indicated by the "small amount of failure-related data" characteristic of the maritime sector (Eruguz et al., 2017), scholars face a contradiction where successful preventive maintenance entails preventing the collection of the historical data which we think we need in order to decide what preventive maintenance we ought to be doing (Moubray, 1997). However, with open-source PSC data, the conditions of all inspected ships can be reviewed and shared for academic research. More literature surveys about other subdomains on the topic of maintenance and service logistics in the maritime sector can be also found in Eruguz et al. (2017).

In summary, to the best of our knowledge, there are no existing studies that evaluate the value of PSC data to ship operators, who are directly affected by the lack of attention to operations management in global supply chains (Yang and Qu, 2016). As PSC inspection data are public, ship operators can apply prescriptive analytics methods to these data to improve their decision performance in ship maintenance planning. Identifying detainable deficiencies during ship maintenance would eliminate the adverse impacts of ship detention on ship operators and improve the efficiency of port operations. Therefore, in this study, we address three research gaps. First, the detention contribution of the deficiency items under each deficiency code is innovatively studied to provide useful inputs for the design of ship maintenance plans. Second, a new optimization objective function that minimizes the operational costs of conducting ship maintenance is proposed, which is different from that of previous PSC-related studies. Third, we innovatively apply the SPO framework in providing inspection suggestions for ship operators, which

is different from the ML methods used in previous PSC-related studies.

2.3 Preliminaries

In this section, we present the ship maintenance planning problem mathematically in Section 2.3.1. Then, Section 2.3.2 describes basic information about classification and regression tree (CART) and RF. At last, Section 2.3.3 introduces how to use the structure of decision trees to measure the detention contribution of the deficiency items under each code, serving as a basis for determining the risk costs associated with the deficiency items under each code.

2.3.1 Ship maintenance planning problem

We denote the total number of ship deficiency codes as K and an individual ship deficiency code as k ($k = 1, ..., K$). A customized ship maintenance plan for an individual ship consists of k recommended decisions, which are denoted as a column vector $\mathbf{W} = [w_1, ..., w_k, ..., w_K]^\top$, where w_k is a binary decision variable that equals 1 if an inspection is recommended for k, and 0 otherwise. To simulate ship operators' decision-making process, we innovatively propose three types of operational costs: inspection cost (denoted by c_1), repair cost (denoted by c_2), and risk cost (denoted by c_3^k).

- The inspection cost c_1 includes manpower and material expenses associated with inspecting items required by a deficiency code. If a ship operator decides to inspect the deficiency items under a deficiency code, it incurs an inspection cost.

- The repair cost c_2 includes the manpower and material expenses if the deficiency items under a code are identified. Considering that identified deficiencies of a ship in a PSC inspection may lead to detention and thus cause huge reputational and economical losses (which will be considered in the risk cost in the following) to a ship operator, we assume that a ship operator conducts repair work if deficiency items are found under each deficiency code. For simplicity, we assume that all deficiency codes have the same inspection cost c_1 and repair cost c_2.

- The risk cost c_3^k represents the loss to a ship operator when a PSCO finds specific deficiency items under code k during a formal PSC inspection. The risk cost of a deficiency code contains two components: indirect and direct costs. Indirect costs refer to the reputational damage to a ship operator from the discovery of deficiency items under code because these identified deficiency items are recorded in the public PSC system. Direct costs refer to the economic costs caused by a delay in the shipping schedule if a ship is detained because of the identified deficiency items. Therefore, we assume that the direct economic costs are related to the detention contribution of the deficiency items under each deficiency code. Recall that there are no specific descriptions of detainable deficiencies except for the general guidelines. In addition, according to the "List of Tokyo MoU Deficiency Codes,"[3] all the deficiency items considered are extracted from important international regulations and conventions that are proposed to monitor ship conditions from different perspectives. The deficiency items under differ-

[3]Available at `https://www.tokyo-mou.org/doc/Tokyo%20MOU%20deficiency%20codes%20(December%202019).pdf`

ent codes are guaranteed to be of different nature and are exclusive to each other. As deficiencies under different codes are classified independently, and having any deficiency would definitely increase the detention probability of a ship, their contributions to the detention decision can thus be regarded as independent as well. Therefore, we assume that each deficiency code has a distinct and independent risk cost, derived from the PSC inspection records using the feature importance method, which will be introduced in Section 2.3.3. Furthermore, although PSCOs can fail to detect all deficiencies because of their negligence and lack of experience or their personal preference based on domain knowledge, it is difficult to obtain accurate data on this issue because we do not know the ground truth of the inspected ships' deficiency conditions. Actually, such real-life conditions are nearly impossible to obtain because the determination of a deficiency by a PSCO can be highly subjective. We can only use the deficiencies identified and recorded on each ship as the ground truth. Therefore, to fully consider the possible risks that potential deficiencies may bring to ship operators and to reduce the influence of PSCOs' subjectivity, we assume that all items under each code will be inspected and all deficiencies will be identified if a ship is chosen for inspection.

Let us consider the decision for a particular deficiency code k as an example. Here, given that we do not have the ground-truth data for intertwined decisions, we assume dependence among the decisions for all of the codes. We let $\mathbf{S}_k = \{1, 0\}$ denote the feasible region for the decision of whether to inspect code k or not. The

decision-making problem can be defined as

$$\begin{aligned}\min_{w_k \in \mathbf{S}_k} z_k &= \min_{w_k \in \mathbf{S}_k} \mathbb{E}\left[(c_1 + c_2\tilde{p}_k)w_k + (c_2 + c_3^k)\tilde{p}_k(1 - w_k)\right] \\ &= \min_{w_k \in \mathbf{S}_k} \left[c_2 w_k + (1 - w_k)\left(c_2 + c_3^k\right)\right]\mathbb{E}\left[\tilde{p}_k\right] + c_1 w_k,\end{aligned} \tag{2.1}$$

where $\tilde{p}_k$ is the uncertain probability that a ship has deficiency items under code k. In Optimization Problem (2.1), $c_1 + c_2\tilde{p}_k$ represents the uncertain costs that a ship operator needs to pay if he/she decides to inspect k, and $(c_2 + c_3^k)\tilde{p}_k$ otherwise. After observing Optimization Problem (2.1), we find that the underlying decision problem always has a unique solution except when $\mathbb{E}\left[\tilde{p}_k\right] = c_1/c_3^k$ that leads to two solutions 0 and 1 with identical objective function values $c_1(1 + c_2/c_3^k)$. We then let

$$\mathbf{W}_k^* = \arg\min_{w_k \in \mathbf{S}_k} \left\{\left[c_2 w_k + (1 - w_k)\left(c_2 + c_3^k\right)\right]\mathbb{E}\left[\tilde{p}_k\right] + c_1 w_k\right\} \tag{2.2}$$

denote the set of optimal solutions for code k, and let w_k^* denote an arbitrary individual member of the set $\mathbf{W}_k^*$.

To derive the optimal decision for k, we need to obtain the following knowledge: the expected value of $\tilde{p}_k$, c_1, c_2, and c_3^k. As mentioned above, costs c_1 and c_2 are assumed to be identical for all deficiency codes and the risk cost c_3^k can be determined based on the detention contribution of the deficiency items under each code, elaborated in Section 3.3. Now, only the expected value of $\tilde{p}_k$ is unknown when solving Optimization Problem (2.1). To solve it, the classic approach is a two-stage PO framework, where the first stage uses an ML model that minimizes the prediction error to predict the expected value of $\tilde{p}_k$ and the second stage

solves an optimization problem that integrates the predicted expected value of $\tilde{p}_k$ with c_1, c_2, and c_3^k. However, this sequential PO framework cannot guarantee that practitioners can obtain near-optimal decisions because the predicting (first) stage focuses on minimizing the prediction error rather than on minimizing the decision error. Our study therefore uses an SPO framework that modifies the loss function used in the ML prediction model to minimize the excess operational costs that result from a (potential) sub-optimal decision under the prediction model over the optimal decision under perfect information. These two frameworks will be introduced in detail in Section 2.4 and Section 2.5, respectively.

2.3.2 Introduction of CART and RF

We mainly use RF and its variant SPOF in this study. RF is an ensemble of CARTs. For one single CART, all of the training examples (data points with input features and output target values) are first stored in the root node. The root node is then split into daughter nodes containing the subsets of all training examples to reduce node impurity (Breiman, 2001). The splitting rule for classification is to maximize the decrease in the Gini index at each split. A split generally depends on a selected feature and one of its values. Constructing a CART requires splitting the nodes to build a binary decision tree recursively and binarily (Yan et al., 2021b). This process stops when all of the nodes contain training examples of the same output value. However, this may lead to an overfitting decision tree. Therefore, we need to set stopping criteria to prevent trees from becoming too complicated. We use two stopping criteria widely considered to control tree dimension: the maximum depth of a tree (denoted by `max_depth`) and the minimum number of training ex-

amples per leaf (denoted by `min_samples_leaf`) (Breiman, 2001). The training process terminates when no node can be further split or one of the stopping criteria is met. For a detailed explanation of CART construction, refer to Breiman (2001) and Yan et al. (2021b).

Ensemble models are used to improve the performance of decision trees. RF is a typical ensemble model that consists of multiple CARTs as weak learners. Compared with a traditional CART, there are two components of randomization in an RF model. The first is that each tree is grown from a bootstrap sample of the original dataset. The second is that a random subset of all the features is chosen to split each node in a tree (Breiman, 2001). Therefore, an RF has two more hyperparameters to ensure the randomness in the RF construction process relative to the two hyperparameters used in constructing a single decision tree: the number of CARTs in the RF (denoted by `n_estimators`) and the number of features considered in each split (denoted by `max_features`) (Breiman, 2001).

2.3.3 Measuring the detention contribution of the deficiency items under each deficiency code

As mentioned above, we assume that the identified deficiency items under each code have a positive and independent effect on a ship's detention. To assist ship operators in allocating their limited ship maintenance resources to deficiency codes with higher risks, we determine the risk cost of having deficiency items under each code by relating the risk cost to the detention contribution of the deficiency items under each code. Before we introduce the PO and SPO frameworks for solving Optimization Problem (2.1), we first illustrate how to determine the de-

tention contribution of deficiency items under code k, which provides the criteria for determining the risk cost c_3^k.

We adopt the Gini index measure, which is a common feature importance method (Tjoa and Guan, 2021), to capture the detention contribution of the deficiency items under each code, obtained by training an RF model to predict the detention outcome using a given dataset $\{(\mathbf{y}_i, d_i)\}_{i=1}^n$, where n is the number of samples. In this dataset, $\mathbf{y}_i = [y_{1,i}, \cdots, y_{k,i}, \cdots, y_{K,i}]^\top$, where $y_{k,i}$ takes a value of 1 if ship i has deficiency items under code k ($k \in \{1, ..., K\}$ and 0 otherwise, and $d_i \in \{1, 0\}$ denotes a class label that takes the value of 1 if ship i is detained and 0 otherwise. Therefore, this preliminary prediction is a classification task whose feature importance can indicate the detention contribution of the deficiency items under each code. In general, any feature in the node that can lead to a large decrease in the Gini index is considered important. Accordingly, the Gini importance of a feature (whether there are deficiency items under code k) in this task is first determined by summing all of the Gini index decreases at the nodes where the feature k is used for splitting in the RF and then normalized by the number of input features, namely the total number of deficiency codes.

We now show how to compute the feature importance of code k. The Gini index for node v is defined as

$$\text{Gini}\,(v) = \sum_{d \in \{0,1\}} \text{RCF}\,(v, d)\,(1 - \text{RCF}\,(v, d))\,, \tag{2.3}$$

where RCF(v, d) denotes the relative class frequency for class d in node v and is

calculated as follows:

$$\text{RCF}(v,d) = \frac{g(v,d)}{\sum_{d' \in \{0,1\}} g(v,d')}, \tag{2.4}$$

where $g(v,d)$ measures the number of training examples belonging to class d that fall into the same node v.

We then let v_{L} and v_{R} denote the left and right daughter nodes of node v. The Gini indexes for nodes v_{L} and v_{R} are denoted as Gini (v_{L}) and Gini (v_{R}), respectively. Then, the Gini index decrease resulting from splitting node v into nodes v_{L} and v_{R}, denoted by $\Gamma(v)$, is calculated as follows:

$$\Gamma(v) = \text{Gini}(v) - \left[\frac{N_{\text{L}}}{N}\text{Gini}\,(v_{\text{L}}) + \frac{N_{\text{R}}}{N}\text{Gini}\,(v_{\text{R}})\right], \tag{2.5}$$

where N, N_{L}, and N_{R} represent the numbers of data examples falling into nodes N, N_{L}, and N_{R}, respectively.

To determine the feature importance of k using an RF model, we denote $\mathbf{V}_k$ as the set that contains nodes in the forest in which feature k is used for splitting. The feature importance of k is denoted by FI (k) and is calculated as follows:

$$\text{FI}\,(k) = \sum_{v \in \mathbf{V}_k} \Gamma(v)\,. \tag{2.6}$$

Finally, the normalized feature importance of code k, denoted by NFI (k), is as follows:

$$\text{NFI}\,(k) = \frac{\text{FI}\,(k)}{\sum_{k' \in \{1,\ldots,K\}} \text{FI}\,(k')}. \tag{2.7}$$

2.4 The PO Framework

Suppose that we have obtained the three types of operational costs c_1, c_2, and c_3^k. To derive w_k^* where the expected value of $\tilde{p}_k$ is unknown, we need to obtain the predicted probability of having deficiency items under code k (i.e., the predicted expected value of $\tilde{p}_k$), which is denoted by $\hat{p}_k$. As mentioned above, there are two frameworks for designing ship maintenance plans that combine $\hat{p}_k$ with c_1, c_2, and c_3^k. In this section, we focus on designing ship maintenance plans using the PO framework.

In the PO framework,obtaining $\hat{p}_k$ is achieved by training an independent RF model for code k, denoted as f_k, using a given dataset $\{(\mathbf{x}_i, y_{k,i})\}_{i=1}^n$. In this dataset, $\mathbf{x}_i \in \mathbb{R}^q$ denotes a vector of q ship-related features for ship i and $y_{k,i}$ denotes a class label indicating whether ship i has deficiency items under code k. Note that $y_{k,i}$ used as input in Section 2.3.3 now is used as output in this task. Then, we use the following metric to evaluate the prediction accuracy:

$$l_{\text{Brier}}^k := \sum_{i=1}^{n} (\hat{p}_{k,i} - y_{k,i})^2 . \tag{2.8}$$

This metric is termed Brier score (Brier, 1950), a strictly proper score function that measures the accuracy of probabilistic predictions.

This study uses the relative class frequency method to obtain the estimated $\hat{p}_k$ of a ship, detailed below. Denote the feature vector of a ship to be predicted as $\mathbf{x}$, the trees in the ensemble for obtaining $\hat{p}_k$ of the ship as $t_1, ..., t_{M_k}$, where M_k denotes the number of trees. The relative class frequency of $y_k = 1$ in the leaf

node where $\mathbf{x}$ falls of an individual decision tree t $(t \in \{t_1, ..., t_{M_k}\})$ is denoted as

$$\text{RCF}_k\left(t, \mathbf{x}, y_k = 1\right) = \frac{g_k\left(t, \mathbf{x}, y_k = 1\right)}{\sum_{y_k' \in \{0,1\}} g_k\left(t, \mathbf{x}, y_k'\right)}, \tag{2.9}$$

where $g_k\left(t, \mathbf{x}, y_k\right)$ measures the number of training examples in class y_k that fall into the same leaf node as $\mathbf{x}$ in decision tree t. Then, $\hat{p}_k$ can be obtained as follows:

$$\hat{p}_k = \frac{\sum_{m=1}^{M_k} \text{RCF}_k\left(t_m, \mathbf{x}, y_k = 1\right)}{M_k}. \tag{2.10}$$

While training the CARTs in the RF for obtaining $\hat{p}_k$, we still adopt the splitting rule that maximizes the Gini index reduction at each split. After the training of f_k is completed, for a new feature vector $\mathbf{x}_0$, $\hat{p}_{k,0}$ can be computed by averaging the relative class frequencies of the leaves where $\mathbf{x}_0$ falls into. With $\hat{p}_{k,0}$, c_1, c_2, and c_3^k, $w_{k,0}^*\left(\hat{p}_{k,0}\right)$ can be derived by solving Optimization Problem (2.1).

2.5 The SPO Framework

As the evaluation metric in the PO framework only measures the prediction error, this section introduces the SPO framework. Section 2.5.1 demonstrates SPO loss. Section 2.5.2 introduces a simplified method to train decision trees using SPO loss. Section 2.5.3 describes the general method for constructing an ensemble of SPOTs.

2.5.1 SPO loss

By definition, SPO loss is measured not by the prediction error, but by the quality of the decisions that are derived from the predictions. In our problem, the decision error should be measured by the extra operational costs resulting from a (potential) sub-optimal decision under the prediction over the optimal decision made under perfect information. Mathematically, we denote by $\hat{p}_k^{'}$ the predicted probability of having deficiency items under code k obtained from an ML model using SPO loss. We then denote by $c_k\left(w_k^*\left(\hat{p}_k^{'}\right), y_k\right)$ the actual incurred costs that the ship operator needs to pay if the (potential) sub-optimal decision is $w_k^*\left(\hat{p}_k^{'}\right)$ when the true label is y_k. For our ship maintenance planning problem, $c_k\left(w_k^*\left(\hat{p}_k^{'}\right), y_k\right)$ can be categorized into the following four forms:

$$c_k\left(w_k^*\left(\hat{p}_k^{'}\right), y_k\right) = \begin{cases} c_1, & \text{if } \; w_k^*\left(\hat{p}_k^{'}\right) = 1, y_k = 0; \\ 0, & \text{if } \; w_k^*\left(\hat{p}_k^{'}\right) = 0, y_k = 0; \\ c_1 + c_2, & \text{if } \; w_k^*\left(\hat{p}_k^{'}\right) = 1, y_k = 1; \\ c_2 + c_3^k, & \text{if } \; w_k^*\left(\hat{p}_k^{'}\right) = 0, y_k = 1. \end{cases} \tag{2.11}$$

In the first two forms, if a ship is free from deficiency items under code k ($y_k = 0$), and the prescribed optimal decision is to not inspect deficiency items under code k ($w_k^*\left(\hat{p}_k^{'}\right) = 0$), the ship operator does not incur any cost. However, if $w_k^*\left(\hat{p}_k^{'}\right) = 1$ and $y_k = 0$, a ship operator only has to pay c_1 for an inspection. In the third and fourth forms, if a ship has deficiency items under code k ($y_k = 1$), and the prescribed optimal decision is to inspect the deficiency items under code k ($w_k^*\left(\hat{p}_k^{'}\right) = 1$), the ship operator has to pay for the corresponding $c_1 + c_2$. However, if $w_k^*\left(\hat{p}_k^{'}\right) = 0$ and $y_k = 1$, the deficiency items under code k may be discovered

during a PSC inspection, which leads to a penalty of $c_2 + c_3^k$ to the ship operator. In summary, we define $\mathbf{c}_k = [c_k^1, c_k^2]^\top$ as a cost vector where $c_k^1 = c_1 + c_2 y_k$ and $c_2^k = (c_2 + c_3^k) y_k$, and $\top$ represents the transpose of a column vector. We then define $\mathbf{w}_k^*\left(\hat{p}_k^{'}\right) = \left[w_k^*\left(\hat{p}_k^{'}\right), 1 - w_k^*\left(\hat{p}_k^{'}\right)\right]$ as a prescribed decision column vector derived from the prediction $\hat{p}_k^{'}$. Note that $\mathbf{W}_k^*\left(\hat{p}_k^{'}\right) = \left\{w_k^*\left(\hat{p}_k^{'}\right)\right\}$ may contain more than one optimal solution associated with $\hat{p}_k^{'}$. Hence, the SPO loss is defined with respect to the worst-case decision from a predicted $\hat{p}_k^{'}$ as follows:

$$\begin{aligned} l_{\text{SPO}}^k\left(\hat{p}_k^{'}, y_k\right) &= \max_{w_k^*\left(\hat{p}_k^{'}\right) \in \mathbf{W}_k^*\left(\hat{p}_k^{'}\right)} \left\{c_k\left(w_k^*\left(\hat{p}_k^{'}\right), y_k\right)\right\} - z_k^* \\ &= \max_{w_k^*\left(\hat{p}_k^{'}\right) \in \mathbf{W}_k^*\left(\hat{p}_k^{'}\right)} \left\{\mathbf{w}_k^*\left(\hat{p}_k^{'}\right) \mathbf{c}_k\right\} - z_k^*, \end{aligned} \tag{2.12}$$

where $\mathbf{w}_k^*\left(\hat{p}_k^{'}\right) \mathbf{c}_k$ represents the cost incurred from a prescribed optimal decision $w_k^*\left(\hat{p}_k^{'}\right)$ and z_k^* represents the perfect optimal cost if y_k is known (Elmachtoub and Grigas, 2021).

According to Elmachtoub and Grigas (2021), training ML models under the SPO loss (2.12) is likely to be impossible, as this loss function is nonconvex and discontinuous in terms of the predicted probabilities and the associated operational costs. In this study, inspired by the framework proposed by Elmachtoub et al. (2020), training decision trees under the SPO loss $l_{\text{SPO}}^k\left(\hat{p}_k^{'}, y_k\right)$ can be greatly simplified by Theorem 2.1, which is discussed in the following section.

2.5.2 Construction of SPOT

We now describe a tractable method to train decision trees under SPO loss based on Theorem 2.1, which states that letting $\hat{p}_k^{'}$ be equal to the relative class frequency

of $y_k = 1$ to a node minimizes the SPO loss in the node.

Theorem 2.1 *Consider any node e and any code k. Let $\mathbf{R}_{k,e}$ denote the set of examples falling into the node e for code k, $\bar{y}_{k,e} = RCF_k\left(t, \mathbf{x}, y_k = 1\right) = \frac{g_k(t,\mathbf{x},y_k=1)}{\sum_{y_k' \in \{0,1\}} g_k\left(t,\mathbf{x},y_k'\right)} = \left(|\mathbf{R}_{k,e}|\right)^{-1} \sum_{i' \in \mathbf{R}_{k,e}} y_{k,i'}$ denote the relative class frequency of $y_k = 1$ within node e, and $\bar{\mathbf{c}}_{k,e} = \left[c_1 + c_2 \bar{y}_{k,e}, \left(c_2 + c_3^k\right) \bar{y}_{k,e}\right]^\top$ denote the average cost vector within node e. If Optimization Problem (2.1) corresponding to $\bar{y}_{k,e}$ has a unique minimizer, then $\bar{y}_{k,e}$ minimizes the within-node SPO loss.*

Proof. We show that the within-node SPO loss associated with prediction $\bar{y}_{k,e}$ is a lower bound of that associated with any other prediction $\hat{p}_{k,e}^{''}$. The following holds for any $\hat{p}_{k,e}^{''}$:

$$
\begin{aligned}
&\frac{1}{|\mathbf{R}_{k,e}|} \sum_{i' \in \mathbf{R}_{k,e}} l_{\text{SPO}}^k \left(\bar{y}_{k,e}, y_{k,i'}\right) - \frac{1}{|\mathbf{R}_{k,e}|} \sum_{i' \in \mathbf{R}_{k,e}} l_{\text{SPO}}^k \left(\hat{p}_{k,e}^{''}, y_{k,i'}\right) \\
=&\frac{1}{|\mathbf{R}_{k,e}|} \sum_{i' \in \mathbf{R}_{k,e}} \max_{w_{k,e}^*(\bar{y}_{k,e}) \in \mathbf{W}_{k,e}^*(\bar{y}_{k,e})} \left\{\mathbf{w}_{k,e}^* \left(\bar{y}_{k,e}\right) \mathbf{c}_{k,i'}\right\} \\
&- \frac{1}{|\mathbf{R}_{k,e}|} \sum_{i' \in \mathbf{R}_{k,e}} \max_{w_{k,e}^*(\hat{p}_{k,e}^{''}) \in \mathbf{W}_{k,e}^*(\hat{p}_{k,e}^{''})} \left\{\mathbf{w}_{k,e}^* \left(\hat{p}_{k,e}^{''}\right) \mathbf{c}_{k,i'}\right\} \\
=&\frac{1}{|\mathbf{R}_{k,e}|} \sum_{i' \in \mathbf{R}_{k,e}} \mathbf{w}_{k,e}^* \left(\bar{y}_{k,e}\right) \mathbf{c}_{k,i'} - \frac{1}{|\mathbf{R}_{k,e}|} \sum_{i' \in \mathbf{R}_{k,e}} \max_{w_{k,e}^*(\hat{p}_{k,e}^{''}) \in \mathbf{W}_{k,e}^*(\hat{p}_{k,e}^{''})} \left\{\mathbf{w}_{k,e}^* \left(\hat{p}_{k,e}^{''}\right) \mathbf{c}_{k,i'}\right\} \\
&\text{(Because } \mathbf{W}_{k,e}^* \left(\bar{y}_{k,e}\right) = \left\{w_{k,e}^* \left(\bar{y}_{k,e}\right)\right\} \text{ is a singleton.)} \\
\leq&\frac{\mathbf{w}_{k,e}^* \left(\bar{y}_{k,e}\right)}{|\mathbf{R}_{k,e}|} \sum_{i' \in \mathbf{R}_{k,e}} \mathbf{c}_{k,i'} - \max_{w_{k,e}^*(\hat{p}_{k,e}^{''}) \in \mathbf{W}_{k,e}^*(\hat{p}_{k,e}^{''})} \left\{\frac{1}{|\mathbf{R}_{k,e}|} \sum_{i' \in \mathbf{R}_{k,e}} \mathbf{w}_{k,e}^* \left(\hat{p}_{k,e}^{''}\right) \mathbf{c}_{k,i'}\right\} \\
=&\mathbf{w}_{k,e}^* \left(\bar{y}_{k,e}\right) \bar{\mathbf{c}}_{k,e} - \max_{w_{k,e}^*(\hat{p}_{k,e}^{''}) \in \mathbf{W}_{k,e}^*(\hat{p}_{k,e}^{''})} \left\{\mathbf{w}_{k,e}^* \left(\hat{p}_{k,e}^{''}\right) \bar{\mathbf{c}}_{k,e}\right\} \\
\leq&0 \text{ (by the definition of } w_{k,e}^* \left(\bar{y}_{k,e}\right)).
\end{aligned}
\tag{2.13}
$$

We show above that $\bar{y}_{k,e}$ achieves a minimum within-node SPO loss, thereby proving the theorem. □

Recall that $\bar{y}_{k,e}$ leads to two optimal solutions when it equals c_1/c_3^k. Empirically, to guarantee the uniqueness of the optimal solution given $\bar{y}_{k,e}$, we can add a small noise term to every cost vector in the training set (Elmachtoub et al., 2020). Therefore, we assume that $\mathbf{W}_{k,e}^*(\bar{y}_{k,e})$ is a singleton for any $\bar{\mathbf{c}}_{k,e}$ below.

Under the SPO framework and with the utilization of Theorem 2.1, assume that the objective of any decision tree training algorithm is to partition the training examples into L_k leaves for code k, and then the training examples in the L_k leaves are represented as $\mathbf{R}_{k,1}, ..., \mathbf{R}_{k,l}, ..., \mathbf{R}_{k,L_k} := \mathcal{R}_{1:L_k}$, whose predictions minimize the following loss function:

$$\min_{\mathcal{R}_{1:L_k} \in \mathbf{T}_k} \frac{1}{n} \sum_{e=1}^{L_k} \sum_{i \in \mathbf{R}_{k,e}} \left(\mathbf{w}_{k,e}^*(\bar{y}_{k,e})\, \bar{\mathbf{c}}_{k,e} - z_k^*(y_{k,i}) \right), \tag{2.14}$$

where $\mathbf{T}_k$ denotes the set of all possible tree structures given the dataset for code k (Elmachtoub et al., 2020). We next use the same procedure as in CARTs to find a reliable and quick sub-optimal solution to Optimization Problem (2.14); that is, we use the recursive partitioning method to find the decision tree that minimizes the decision error in the training set. Define $x_{i,j}$ as the jth input feature component corresponding to the ith training sample. Beginning with the entire training set at the root node, consider a decision tree split (j_k, s_k) that represents a splitting feature component j_k and a splitting point s_k to partition the training examples

into a left child node l and a right child node r for code k:

$$\mathbf{R}_{k,l}\left(j_k, s_k\right) = \{i \in \{1, ..., n\} | x_{i,j} \leq s_k\} \text{ and } \mathbf{R}_{k,r}\left(j_k, s_k\right) = \{i \in \{1, ..., n\} | x_{i,j} > s_k\}, \tag{2.15}$$

if feature j_k is numeric, and

$$\mathbf{R}_{k,l}\left(j_k, s_k\right) = \{i \in \{1, ..., n\} | x_{i,j} = s_k\} \text{ and } \mathbf{R}_{k,r}\left(j_k, s_k\right) = \{i \in \{1, ..., n\} | x_{i,j} \neq s_k\}, \tag{2.16}$$

if feature j_k is categorical. The first split of the decision tree is chosen by finding the pair (j_k, s_k) that minimizes the following optimization problem:

$$\min_{j_k, s_k} \frac{1}{n} \left(\sum_{i \in \mathbf{R}_{k,l}} \left(\mathbf{w}^*_{k,l}\left(\bar{y}_{k,l}\right) \mathbf{c}_{k,i} - z^*_k\left(y_{k,i}\right) \right) + \sum_{i \in \mathbf{R}_{k,r}} \left(\mathbf{w}^*_{k,r}\left(\bar{y}_{k,r}\right) \mathbf{c}_{k,i} - z^*_k\left(y_{k,i}\right) \right) \right), \tag{2.17}$$

where $\bar{y}_{k,l}$ and $\bar{y}_{k,r}$ represent the relative class frequencies of $y_k = 1$ within the left child node l and the right child node r, respectively.

Optimization Problem (2.14) can be solved by finding the split that has the minimum objective function value among every possible split (j_k, s_k). From Theorem 2.1, the objective value of a split can be computed as follows: 1) splitting the training examples according to a possible criteria, 2) determining the relative class frequencies of $y_k = 1$ in two child nodes l and r, and the associated $w^*_{k,l}\left(\bar{y}_{k,l}\right)$ and $w^*_{k,r}\left(\bar{y}_{k,r}\right)$, 3) deriving the corresponding incurred cost $\mathbf{w}^*_{k,l}\left(\bar{y}_{k,l}\right) \mathbf{c}_{k,i}$ or $\mathbf{w}^*_{k,r}\left(\bar{y}_{k,r}\right) \mathbf{c}_{k,i}$ for each sample i in the two daughter nodes, and 4) summing the SPO losses of each node and dividing the sum by the total number of samples in the two daughter nodes. After the first split is chosen, the greedy split selection approach is then recursively applied to the resulting nodes. During the training

process, two stopping criteria widely considered in an RF are also applied to control the dimension of an SPOT, namely the maximum depth of a tree (`max_depth`) and the minimum number of training examples per leaf (`min_samples_leaf`) (Breiman, 2001). The training process terminates when no node can be further split or one of the stopping criteria is met.

2.5.3 Construction of SPOF

To improve the performance of SPOTs, we further consider training an ensemble of SPOTs, denoted as SPOF. The procedure for constructing an SPOF is similar to constructing a classic RF and uses the same two components of randomization. The first component is that each SPOT is grown from a bootstrap sample of the training dataset. The second is that a random subset of all of the features is chosen to split each node in an SPOT. After the training procedure is completed, given a new feature vector $\mathbf{x}_0$, the relative class frequencies of $y_k = 1$ in the leaves that $\mathbf{x}_0$ falls into in the forest are averaged, and the optimal decision on whether to inspect code k is determined based on the three types of operational costs. At last, the SPOF is subject to two more hyperparameters similar to an RF, namely the number of SPOTs contained in the SPOF (`n_estimators`) and the number of features considered in each split (`max_features`) (Breiman, 2001).

2.6 Computational Experiments

Section 2.6.1 describes the dataset that we use and the settings for each ML model. In Section 2.6.2, we evaluate the detention contribution of the deficiency items under each code. We then show how to determine the three types of operational

costs, especially the risk cost. In Section 2.6.3, we compare the average total operational costs and the single-code costs of five ship maintenance schemes. In Section 2.6.4, we conduct a sensitivity analysis of risk costs.

2.6.1 Data description

The dataset for this study contains 3,026 records of PSC initial inspections during January 2015 to December 2019 period at the Hong Kong Port and the corresponding ship-related factors. Hong Kong Port is a member of the Tokyo MoU that governs the Asia-Pacific region. There are 17 deficiency codes required by the Tokyo MoU,[4] as shown in Table 2.1. We focus on the first 16 deficiency codes that describe specific inspection items and areas. The PSC inspection records are retrieved from the Asia Pacific Computerized Information System[5] provided by Tokyo MoU, and the ship-related factors are obtained from the World Shipping Register database.[6]

Table 2.1: Description of deficiency codes in the Tokyo MoU

Code	Meaning	Code	Meaning
D1	Certificates and documentation	D10	Safety of navigation
D2	Structural condition	D11	Life saving appliances
D3	Water/Weathertight condition	D12	Dangerous goods
D4	Emergency system	D13	Propulsion and auxiliary machinery
D5	Radio communication	D14	Pollution prevention
D6	Cargo operations including equipment	D15	International Safety Management (ISM)
D7	Fire safety	D18	Labour conditions
D8	Alarms	D99	Other
D9	Working and living conditions		

Before training ML models, we randomly divide the whole dataset into a train-

[4] Available at `https://www.tokyo-mou.org/doc/ANN20-f.pdf`.
[5] See `https://apcis.tmou.org/public/`.
[6] See `https://world-ships.com/`.

ing set (8%, 2420 records) and a test set (20%, 606 records). We primarily use the RF or SPOF in this study, and we fix n estimators in each ML model to be 200. A tuple of three hyperparameters needs to be tuned for these models: max depth, max features, and min samples leaf. We use a grid search with 5-fold cross validation on the training set to tune these hyperparameters in each ML model. The proposed models are constructed using the training set and their performance is validated by using the test set.

2.6.2 Measuring each deficiency code's risk cost

The preliminary analysis results show that the dataset is highly imbalanced, as it only contains 100 records showing detention (78 detentions in the training set and 22 detentions in the test set). Thus, instead of using accuracy to evaluate the performance of the developed RF model, we adopt the area under the receiver operating characteristic curve (ROC AUC) as our main metric for evaluating the prediction performance. We follow Yan et al. (2021b) to determine the search ranges for the three hyperparameters. The ROC AUC score for the predictor on the test set is 0.97, which verifies that it performs 94% better than random guessing. Then, we output the normalized detention contribution of the deficiency items under each code based on the feature importance method, as shown in Figure 2.1.

Figure 2.1 shows that the five deficiency codes with the highest risk are D10, D15, D7, D1, and D3, and the deficiency items under these codes are more likely to lead to detentions. Accordingly, it is reasonable to assign higher risk costs to them. In contrast, the five deficiency codes with the lowest risk are D12, D8, D18, D6, and D13; accordingly, their risk costs are lower. As we cannot obtain accu-

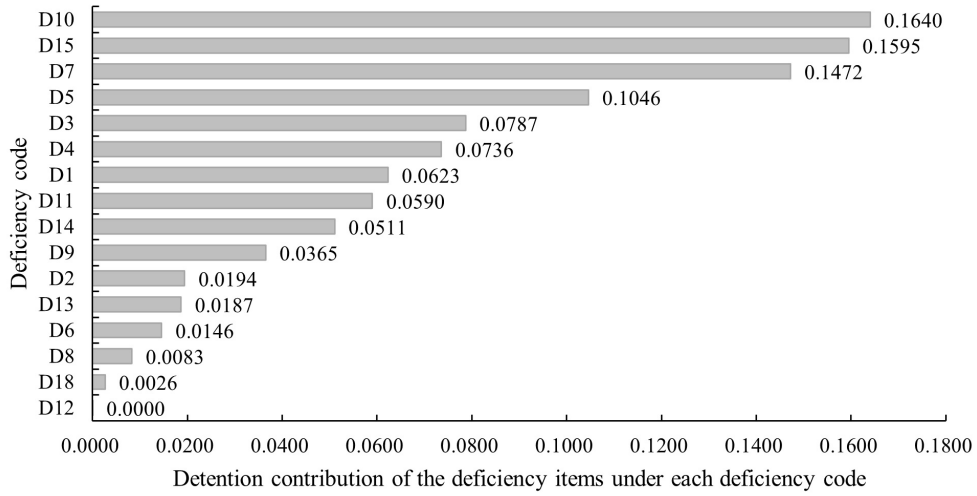

Figure 2.1: Detention contributions of the deficiency items under 16 deficiency codes

rate values for the three types of operational costs, we measure these costs in units. Following Knapp (2007), who estimated that the average inspection cost is USD 506 when there are no deficiencies and USD 759 when deficiencies (which may involve more procedures and repair work) are found in a PSC inspection at the port, we approximate the inspection and repair costs at two units and three units, respectively (because $506 : 759 \approx 2 : 3$). As repair work requires more human and material resources than inspection, it is reasonable to assign a higher value to repair costs than inspection costs. As mentioned in Sections 2.1 and 2.3.1, risk cost is divided into reputation cost and detention cost. Reputation cost refers to the reputational damage to a ship operator brought about by the recording of deficiency items in the public PSC information system. The recorded deficiency items indicate that the ship operator cannot guarantee the navigation safety of its ships and can thus lead to higher rates ship inspections in the future because the detention and deficiency history of all ships in an operator's fleet affects the ship operator's

performance.[7] As the effect of detention on reputation is long-lasting and cannot be easily erased once the deficiency items are recorded in the system, we first estimate reputation cost, denoted as c_r, to be five units in benchmark experiments, which is greater than the inspection cost and the repair cost, respectively. We will conduct sensitivity analysis in Section 2.6.4 to analyze the impact of reputation cost. Next, the detention cost of the deficiency items under each deficiency code is approximated as its detention contribution multiplied by an adjustable factor λ. This adjustable factor converts the normalized detention contribution into detention cost and can be adjusted according to the ship operators' preference. If a ship operator is risk-averse, it would consider improving the risk costs by increasing the value of λ. Conversely, if a ship operator is risk-neutral or risk-appetitive, it would consider decreasing the risk costs by decreasing the value of λ. We will conduct sensitivity analysis in Section 2.6.4 to analyze the impact of λ. Mathematically, the parameters c_1, c_2, and c_3^k for k are as follows:

$$\begin{aligned} c_1 &= 2 \text{ (units)}, \\ c_2 &= 3 \text{ (units)}, \\ c_3^k &= c_r + \text{NFI}(k) \times \lambda \text{ (units)}. \end{aligned} \tag{2.18}$$

In our benchmark experiments, we set $c_r = 5$ and $\lambda = 100$, which means that if the deficiency items under a code causes detention, its detention cost is 100 units. Therefore, the detention cost is 50 times greater than the inspection cost, 33.33 times greater than the repair cost, and 20 times greater than the reputation cost. Based on the detention contributions shown above, the risk costs of having

[7] See `https://www.tokyo-mou.org/inspections_detentions/NIR.php`

deficiency items under each code when $\lambda = 100$ are shown in Figure 2.2.

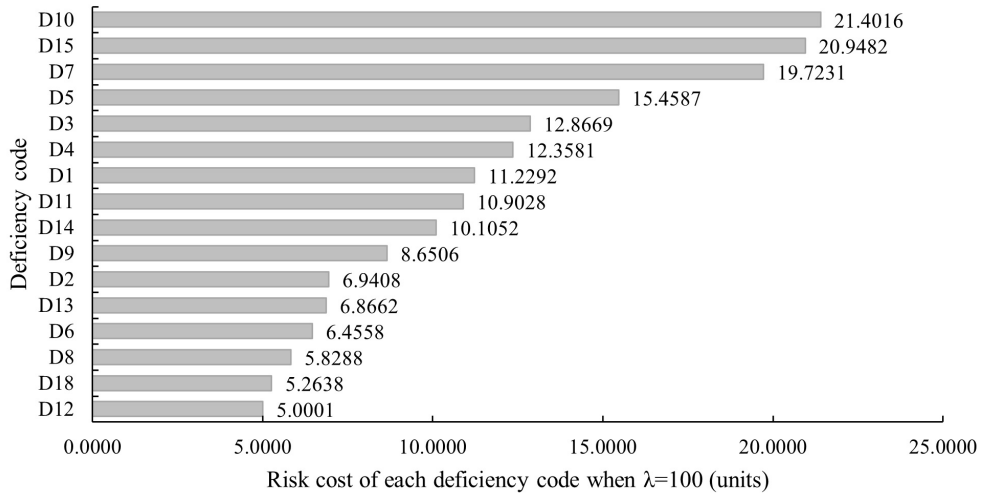

Figure 2.2: Risk costs of 16 deficiency codes when $\lambda = 100$

2.6.3 Comparison of ship maintenance schemes

Considering the three types of operational costs, we make ship maintenance decisions for ships in the test set. We compare five different types of ship maintenance schemes. The first scheme is the random inspection scheme (RIS), where each deficiency code has a 50% chance of being inspected. The second scheme is the no inspection scheme (NIS), where ship operators do not perform any ship maintenance. The third scheme is the overall inspection scheme (OIS), where ship operators inspect all deficiency codes. The fourth scheme is the PO scheme (POS). The fifth scheme is the SPO scheme (SPOS). In the benchmark experiments, we set $\lambda = 100$. For RF and SPOF models, we set `n_estimators` $= 200$ and set the search range for `max_features` as $\{2, 3, 4, 5, 6, 7\}$, for `max_depth` as $\{2, 3, 4, 5, 6, 7, 8, 9, 10\}$, and for `min_samples` leaf as $\{1, 2, 3, 4, 5, 6, 7\}$. We then

use the optimal hyperparameters to construct each ML model on the whole training set and compute the single-code actual costs incurred by the decision $w_k^*\left(\hat{p}_k'\right)$ for a ship in the test set. The total incurred operational costs for a ship would be $\sum_{k\in\{1,\ldots,K\}} c_k\left(w_k^*\left(\hat{p}_k', y_k\right)\right)$. The average total incurred operational costs of five ship maintenance schemes for the test set are shown in Figure 2.3.

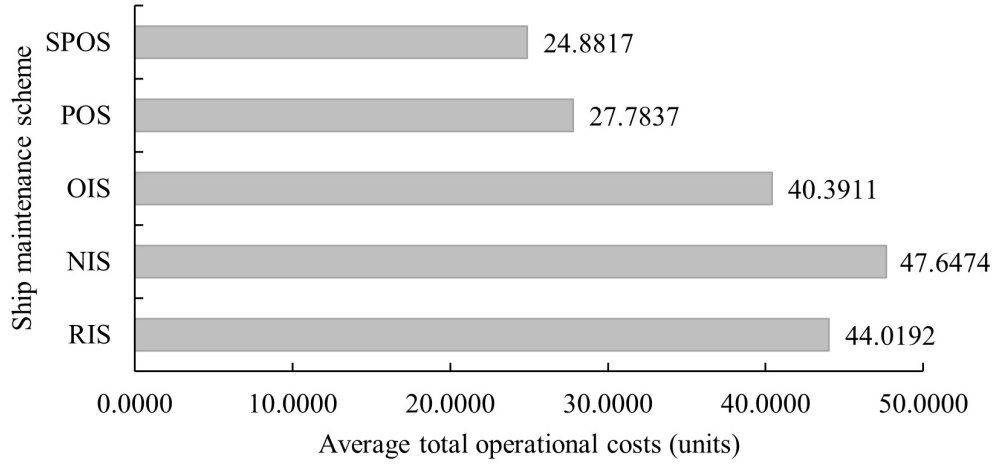

Figure 2.3: Average total operational costs of five ship maintenance schemes for the test set

As Figure 2.3 shows, the SPOS outperforms the other four ship maintenance schemes in terms of the average total operational costs. The SPOS reduces the average total operational costs by 43.22% on average compared with ship maintenance schemes that do not apply artificial intelligence methods, such as the RIS, NIS, and OIS. Compared to the POS, the SPOS reduces the average total operational costs by 10.44%, which demonstrates the superiority of SPO loss function that considers decision error. Next, we list the average single-code costs of the five ship maintenance schemes for the test set in Table 2.2; the proportions indicate the percentage of ships in the test set with deficiency items under each code

and gaps 1 to 4 are the cost savings of the SPOS over the RIS, NIS, OIS, and POS, respectively.

The findings from Table 2.2 are as follows. First, the SPOS outperforms the other four ship maintenance schemes on all of the deficiency codes. This result indicates that the average total operational costs of maintaining all of the codes using the SPOS must be lower than the average total costs of randomly adopting the other four schemes to maintain these codes. Second, after analyzing the cost savings of the SPOS over the other four maintenance schemes on all 16 deficiency codes, we find some obvious differences between the deficiency codes related to their proportions and respective risk costs. We further illustrate the values of gaps 1 to 4 of each deficiency code in Figure 2.4. Note that the purpose of this line chart is not to reflect the development trend of values from gaps 1 to 4, but to highlight the differences and facilitate comparison. The fluctuations in these lines are also used to classify deficiency codes in the following analysis. Moreover, the smoothness of the lines, as indicated by the standard deviation shown in Table 2.2, is also one of the criteria for classification. Therefore, based on the deficiency codes' proportions, risk costs, gap values, and standard deviations, we roughly classify the deficiency codes into three categories, illustrated using the different lines in Figure 2.4. Table 2.3 summarizes the codes in each category and their main characteristics.

Category I contains deficiency codes that have a high average proportion and average risk cost and includes D3, D7, D10, and D11. The cost savings from adopting the SPOS are relatively modest for these deficiency codes, especially as indicated by the values of gap 3 and gap 4. This is because cost-conscious and risk-averse ship operators will increase their efforts to inspect these risky defi-

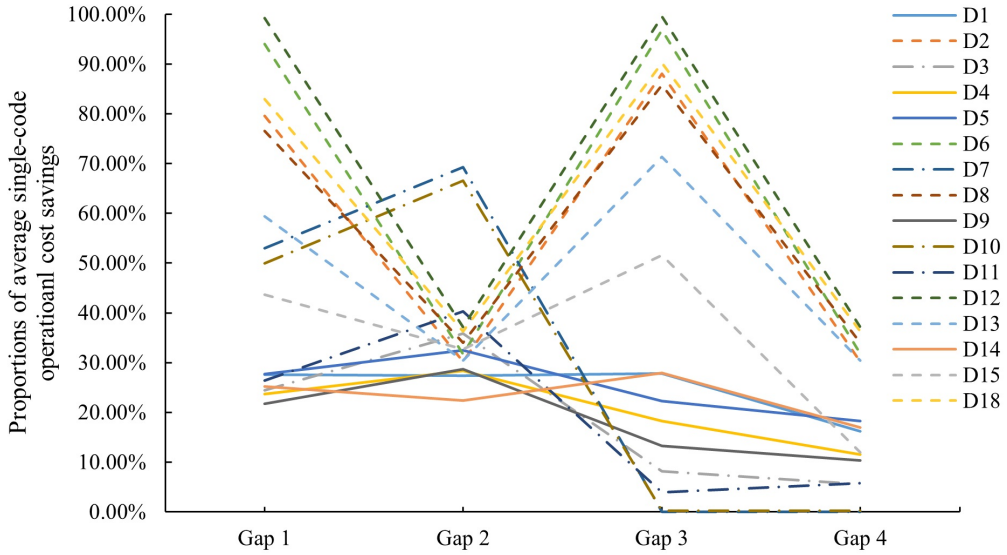

Figure 2.4: Cost savings of average single-code operational costs for 16 deficiency codes

ciency codes and minimize risk costs. Category II contains D1, D4, D5, D9, and D14, which have a medium average proportion and average risk cost. Furthermore, as shown by their smooth lines in Figure 4, the standard deviations of the four gaps of these codes are within 0.1. This result indicates that for these deficiency codes, the SPOS has a relatively stable advantage over the other four ship maintenance schemes. Category III contains D2, D6, D8, D12, D13, D15, and D18, which have a low average proportion and average risk cost. As shown by the values of gap 1 to gap 4, the SPOS can generate significant cost savings for the third category of deficiency codes over the other four schemes. When managing these deficiency codes, ship operators must consider that the NIS may introduce certain risks, while the OIS and the RIS may result in a waste of ship maintenance resources. Although ship operators can use the ML method to predict the probabilities of having deficiency items under these deficiency codes, the POS assigns

Table 2.2: The average single-code costs of five ship maintenance schemes for the test set

Deficiency code	Proportion	Risk cost (units)	Average single-code cost (units)					Gap 1	Gap 2	Gap 3	Gap 4	Standard deviation of 4 gaps
			RIS	NIS	OIS	POS	SPOS					
D1	16.72%	11.2292	2.5211	2.5124	2.5297	2.1785	1.8262	27.56%	27.31%	27.81%	16.17%	0.06
D2	3.44%	6.9408	1.2349	0.3609	2.1089	0.3609	0.2520	79.59%	30.17%	88.05%	30.17%	0.31
D3	23.46%	12.8669	3.3350	3.9275	2.7426	2.6652	2.5188	24.47%	35.87%	8.16%	5.49%	0.14
D4	20.22%	12.3581	2.7570	2.9398	2.5743	2.3800	2.1055	23.63%	28.38%	18.21%	11.53%	0.07
D5	16.69%	15.4587	2.6466	2.8328	2.4604	2.3385	1.9129	27.72%	32.47%	22.25%	18.20%	0.06
D6	1.02%	6.4558	1.0617	0.0936	2.0297	0.0937	0.0639	93.98%	31.75%	96.85%	31.80%	0.37
D7	47.32%	19.7231	7.4520	11.3991	3.5050	3.5050	3.5050	52.97%	69.25%	0.00%	0.00%	0.36
D8	4.86%	5.8288	1.3123	0.4662	2.1584	0.4663	0.3078	76.55%	33.98%	85.74%	33.99%	0.27
D9	28.42%	8.6506	3.2242	3.5375	2.9109	2.8160	2.5246	21.70%	28.63%	13.27%	10.35%	0.08
D10	38.60%	21.4016	6.2904	9.4224	3.1584	3.1584	3.1518	49.90%	66.55%	0.21%	0.21%	0.34
D11	32.49%	10.9028	3.9984	4.9325	3.0644	3.1248	2.9447	26.35%	40.30%	3.90%	5.76%	0.17
D12	0.33%	5.0001	1.0091	0.0132	2.0050	0.0132	0.0083	99.18%	37.13%	99.59%	37.12%	0.36
D13	8.96%	6.8662	1.6157	0.9443	2.2871	0.9447	0.6572	59.32%	30.40%	71.27%	30.43%	0.21
D14	16.56%	10.1052	2.4484	2.3572	2.5396	2.2037	1.8305	25.24%	22.34%	27.92%	16.94%	0.05
D15	6.11%	20.9482	1.8894	1.5807	2.1980	1.2077	1.0640	43.69%	32.69%	51.59%	11.90%	0.17
D18	4.49%	5.2638	1.2230	0.3273	2.1188	0.3271	0.2085	82.95%	36.29%	90.16%	36.26%	0.29

Table 2.3: The category of deficiency codes and their characteristics

Category	Deficiency codes	Characteristics
I	D3, D7, D10, D11	High average proportion (35.47%) High average risk cost (16.22 units) "Medium-high-low-low" shape of gap values High average standard deviation (0.25)
II	D1, D4, D5, D9, D14	Medium average proportion (19.72%) Medium average risk cost (11.56 units) Smooth shape of gap values Low average standard deviation (0.06)
III	D2, D6, D8, D12, D13, D15, D18	Low average proportion (4.17%) Low average risk cost (8.19 units) "High-medium-high-medium" shape of gap values High average standard deviation (0.28)

low predicted probabilities to them because of their low probabilities of occurrence; in this way, the POS prescribes decisions similar to those prescribed by the OIS. Therefore, the superiority of the SPOS is demonstrated by the fact that the minimum cost savings associated with the deficiency codes in category III are greater than 30% compared with the other four schemes.

In summary, the SPOS helps ship operators reduce operational costs in ship maintenance for all deficiency codes. Considering the different cost savings brought about by the SPOS for different deficiency codes, we strongly recommend that ship operators invest more monetary and human resources in developing intelligent SPO alarm systems for the deficiency codes in categories II and III because they can always be overlooked due to their medium proportions and risk costs. Installing intelligent alarm systems for these two categories will help ship operators identify these infrequent deficiencies in advance and save on maintenance costs. For the codes in category I, due to their proportions and risk costs, intelligent alarm systems combined with regular maintenance is the best way to prevent detentions.

2.6.4 Sensitivity analysis of risk costs

We now adjust the values of c_r and λ, respectively, to conduct sensitivity analysis on risk costs. For reputation cost c_r, the candidate value is from set $\{3, 5, 8, 10\}$. For parameter λ, the candidate value is from set $\{50, 100, 150, 200, 250\}$. Figures 2.5 and 2.6 show the cost savings of the average total operational costs under different values of c_r and λ, respectively. The following observations and conclusions can be drawn. First, the results confirm that the SPOS consistently outperforms the other four schemes. Second, with the increase of and λ, respectively, both gaps 1 and 2 show an upward trend. Intuitively, we see that as the risk costs increase, ship operators pay more for inaction, which is similar to the RIS and NIS. In contrast, both gaps 3 and 4 show a downward trend with the increase of reputation cost and , respectively, which means that the cost savings gained from the SPOS over the OIS and POS decrease as risk costs escalate. If the risk costs approach infinity, both the POS and SPOS would resemble the OIS; that is, the best strategy would be to inspect all deficiency codes. Furthermore, as the cost associated with ship maintenance and repair operations is around 10% of the total operating expenses of a ship and it can increase up to 20–30% for old ships,[8] the SPOS can save approximately 1% more of the total operating expenses than the POS and at least 3% more than the schemes that do not use ML methods. To conclude, the above results verify the superiority of SPOS in reducing the overall operational costs over other schemes.

[8] See `https://www.seaplace.es/maintenance-cost-how-can-owners-reduce-it/`.

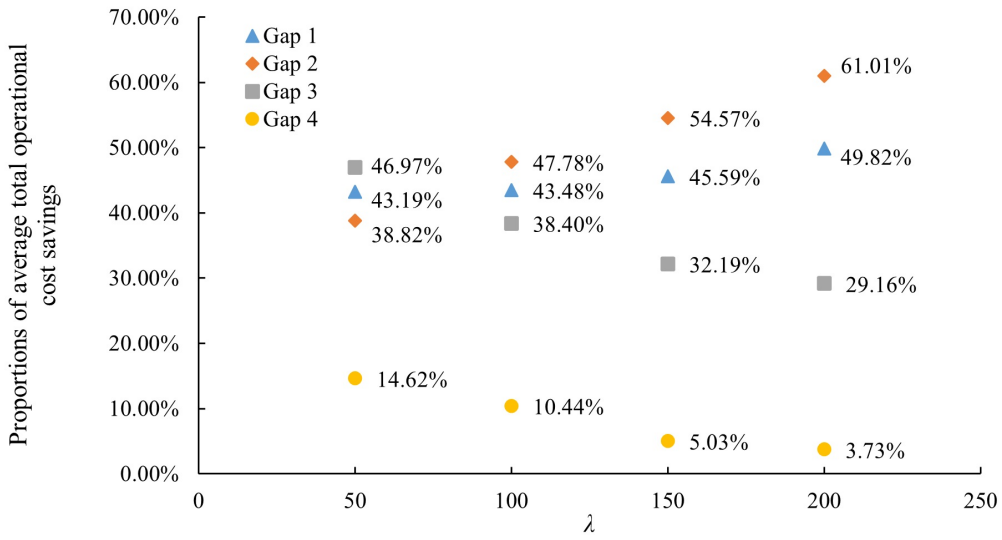

Figure 2.5: Cost savings of the average total operational costs under different values of λ

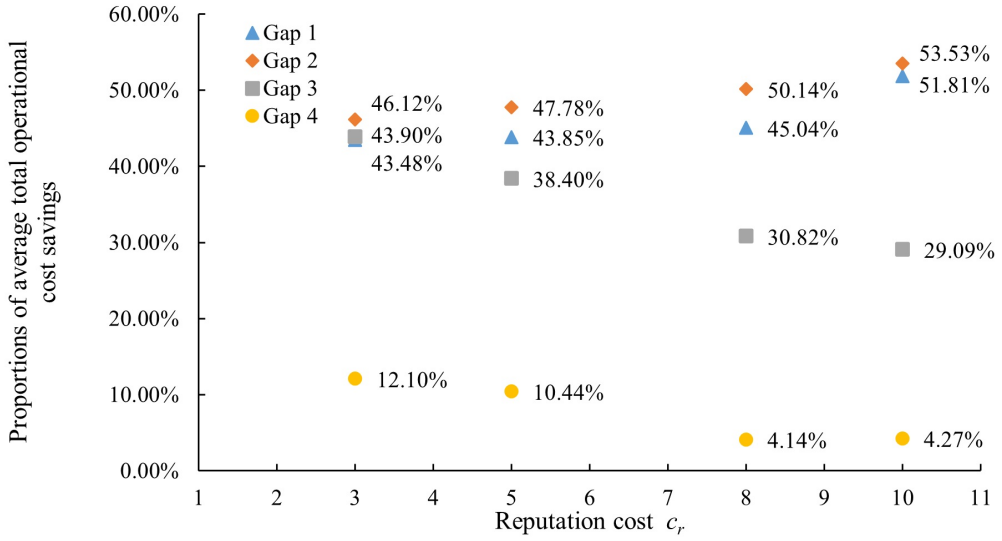

Figure 2.6: Cost savings of the average total operational costs under different values of reputation cost

2.7 Conclusions

Due to the development of machine learning technologies and the availability of PSC data, many studies have been conducted to improve the efficiency of PSC inspections for port authorities. In contrast, this study designs ship maintenance plans for ship operators with the aim at minimizing the overall operational costs. The targeted and cost-effective ship maintenance plans would also improve the efficiency of port operations by reducing the resources needed for formal PSC inspections and relieving port congestion. When simulating ship operators' decision-making process, we consider the impacts of ship detention on ship operators and innovatively examine three types of operational costs separately: inspection cost, repair cost, and risk cost. In particular, the risk cost of having deficiency items under each code is determined by its detention contribution. Instead of using a PO framework, we propose an SPO framework that adopts an SPO loss function that minimizes the decision error. Computational experiments demonstrate that when the detention cost is 50 times greater than the inspection cost, the average total operational costs derived from the proposed SPO-based scheme are on average 43.22% lower than those derived from ship maintenance schemes that do not use artificial intelligence algorithms and are also 10.44% lower than those derived from the PO-based scheme. Analyses of the average single-code cost savings under the SPO-based scheme indicate that the SPO-based scheme is significantly superior to the other ship maintenance schemes. Furthermore, through a sensitivity analysis of risk costs, we find that the SPO-based scheme can reduce the total operating expenses of a ship by approximately 1% compared to the PO-based scheme and at least 3% compared to the schemes that do not use ML methods. Fi-

nally, the superiority of the SPO framework is further verified by analyzing the trade-off between maintenance cost and detention probability of the SPO-based scheme and the PO-based scheme.

Chapter 3

Prescriptive Analytics for a Maritime Routing Problem

3.1 Introduction

The International Maritime Organization (IMO) has implemented resolutions on ship maintenance and operations to enhance maritime safety, allowing port authorities to inspect foreign visiting ships through port state control (PSC) to ensure compliance with international regulations. However, with fewer than 5% of ships inspected due to high costs and limited port state control officers (PSCOs) Yan et al. (2021b), effectively selecting substandard ships for inspection is essential. Existing research, such as Yan et al. (2020, 2021a), has primarily focused on ship selection without considering PSCO routing. Given varied berthing locations and arrival and departure times of arriving ships, and PSCO constraints such as lunch breaks and working hours, the number of ships that can be inspected strongly depends on the routing selected by the PSCOs (see Example 3.1). There-

fore, this study innovatively uses PSC inspection data for the PSCO routing problem to maximize the number of deficiencies identified while considering various practical constraints.

Example 3.1 *Assume that a PSCO at the Hong Kong port spends two hours inspecting a ship and one hour traveling from the Kwai-Tsing Container Terminal (KTCT) to the River Trade Terminal (RTT). Hence, in six hours, a PSCO can inspect either one ship at the KTCT and another at the RTT, or three ships at a single terminal (i.e., KTCT or RTT). Assume that we know in advance that two foreign visiting ships at the RTT have six and eight deficiencies, respectively, and three foreign visiting ships at the KTCT have nine, four, and two deficiencies, respectively. Then, if routing is not considered, the optimal inspection scheme is to inspect the two ships at the RTT that have a combined 14 deficiencies, and the ship at the KTCT that has nine deficiencies, resulting in a total of 23 detected deficiencies. However, if the traveling time from the RTT to the KTCT is considered, the proposed inspection scheme set forth above is infeasible when the PSCO only has six working hours available. Thus, considering the PSCO traveling time, the modified optimal inspection scheme should inspect all three of the ships at the KTCT, resulting in a total of 15 detected deficiencies.*

Most port authorities' ship selection schemes still rely on the sum of the weighted points of risk factors, such as ship age and type, that determines ships' selection scores based on expert knowledge. However, the subjective value of these scores can be biased, thus compromising their effectiveness. To overcome the shortcomings of the current ship selection schemes, maritime researchers have applied a two-stage prescriptive analytics framework to solve ship selection problems. Pre-

scriptive analytics utilizes a combination of predictive and optimization techniques to generate informed recommendations based on accessible auxiliary data. In the context of ship selection for PSC inspections, most of the previous studies first use machine learning (ML) models to predict the number of deficiencies or the probability of detention for foreign visiting ships, and the predictions are used to guide ship selection for inspection through mathematical optimization models (Yang et al., 2018a,b; Yan et al., 2020, 2021a) and Yan et al. (2021b). However, it is important to note that predictive models only aim to minimize the prediction error, while the impact of prediction results on the downstream decisions is totally ignored (see Example 3.2), leading to suboptimal decisions. Therefore, to overcome this drawback, a more appropriate prescriptive analytics framework is developed to integrate the prediction and optimization tasks, aiming at creating a decision-focused learning framework to improve the decision quality (Mulamba et al., 2021).

Example 3.2 *Suppose that a port authority needs to choose between ship A and ship B for inspection. Ship A has five deficiencies and ship B has 10 deficiencies, but these numbers are unknown prior to the inspection. In an environment of perfect information, ship B should be inspected. Suppose that we have two models: model I and model II. Model I predicts eight deficiencies in ship A and seven in ship B, and model II predicts two deficiencies in ship A and three in ship B. It follows that model I outperforms model II in terms of prediction accuracy. However, when using model I, ship A is selected for inspection, because it is predicted to have more deficiencies than ship B. In contrast, when using model II, ship B is selected for inspection, showing that the predictive model with worse performance in terms of*

prediction accuracy leads to a better decision.

In this study, we adopt such a framework that considers the interactions between ships in both the optimization model and the predictive model. Specifically, we adopt a prescriptive analytics framework that uses the criterion of minimizing the decision error measured by the suboptimality of the decisions generated by the predictions during the training process, rather than minimizing the prediction error to improve the decision quality. Because the PSCO routing problem applies the team orienteering problem, which has been proven to be NP-hard (Vansteenwegen et al., 2009), computational complexity and scalability are two major obstacles to putting this method into practice (Mulamba et al., 2021). To remove these obstacles, we first exploit the structure of the PSCO routing problem by designing undominated inspection templates (Yan et al., 2021a), allowing the decision-focused learning framework to efficiently solve the PSCO routing problem. We then adopt a new family of surrogate loss functions motivated by the noise-contrastive estimation (NCE) literature (Gutmann and Hyvärinen, 2010) for the decision-focused learning framework (Mulamba et al., 2021). These surrogate loss functions require building a solution pool with suboptimal solutions, which are regarded as noise samples. This solution pool can be interpreted as the convex hull of the feasible solutions, and its use avoids frequent reoptimizations when training the predictive model (Mulamba et al., 2021).

Our scientific contributions can be summarized as follows. First, our study innovatively uses PSC inspection data for the PSCO routing problem. Second, we compare two prescriptive analytics frameworks, which are the two-stage framework and the decision-focused learning framework. Specifically, the decision-

focused learning framework considers the PSCO routing problem in the ship deficiency prediction model. To overcome the obstacles of computational complexity and scalability for the decision-focused learning framework, we first transform the original PSCO routing model and then adopt a family of surrogate loss functions. Third, through computational experiments using real PSC inspection records, we compare the performance of the two-stage framework with that of the decision-focused learning framework to answer the following questions: 1) Does a good prediction lead to a good decision? 2) Does the decision-focused learning framework outperform the two-stage framework for the PSCO routing problem? 3) How can we achieve a balance between solution quality and solution efficiency in the decision-focused learning framework?

The remainder of this chapter is organized as follows. Section 3.2 reviews the related literature. Section 3.3 first formulates the PSCO routing problem, and then uses a route generation method to transform the original combinatorial model into a more compact model. Third, we compare the two models and discuss the results of our comparison. Section 3.4 describes the traditional two-stage framework. Based on this framework, Section 3.5 describes a decision-focused learning framework that uses a new family of noise-contrastive loss functions. Section 3.6 describes the results of our computational experiments that compare the traditional two-stage framework with the decision-focused learning framework and conducts a sensitivity analysis. Section 3.7 concludes the study.

3.2 Literature Review

Because current ship risk profile schemes do not efficiently identify substandard ships, most PSC studies have aimed at improving inspection efficiency by using ML technologies to identify ships with more deficiencies or higher detention probabilities. For example, Wang et al. (2019) developed a tree augmented naive Bayes classifier to identify high-risk ships with more deficiencies. Chung et al. (2020) and Yan et al. (2021c) used the Apriori algorithm to determine the type and sequence of ship items that should be inspected. In recent years, maritime researchers have begun to adopt ship deficiency prediction models to allocate scarce inspection resources. This issue, which is known as the PSCO scheduling problem, was studied by Yan et al. (2020) and Yan et al. (2021a). Yan et al. (2020) first compared the results of three random forest models with different loss functions to predict the number of ship deficiencies under four deficiency categories and then developed optimization models to efficiently match officers' expertise with ship deficiency conditions. Subsequently, Yan et al. (2021a) improved the prediction performance by integrating shipping domain knowledge into an XGBoost model, and then modified the downstream PSCO scheduling models to be more consistent with practice. In their study, the authors considered practical constraints concerning ship berthing time windows and PSCO lunch break requirements but did not consider either the berthing locations of foreign visiting ships or the time it took PSCOs to travel between two locations. Therefore, their optimization model did not fully capture the PSCO routing problem. Accordingly, our study represents an advance in the field. For other PSC-related studies, please refer to the literature review in Chapter 2.

As mentioned above, some PSC studies have used both prediction and optimization methods to make ship selection decisions (Yan et al., 2020, 2021a, 2023) by considering structural characteristics of the downstream optimization problem when training their predictive models. Tian et al. (2023a) is the first study to use PSC data to design ship maintenance schemes for ship operators. Recently, an emerging stream of literature has combined prediction and optimization in developing prescriptive analytics frameworks for many domains, such as human resource planning Berk et al. (2019), charging infrastructure planning Brandt et al. (2021), vehicle routing (Soeffker et al., 2022), and queuing (Notz et al., 2023). Detailed reviews of prescriptive analytics frameworks are available in He et al. (2022), Qi and Shen (2022), and Tian et al. (2023b).

The predictions involved in the decision-focused learning framework have been most trained by gradient-descent predictive models in the computer science field. The main obstacle to plugging the optimization problem into the training process of gradient-descent predictive models is that the discrete and discontinuous solution space prevents the algorithm from easily differentiating the decision loss over the predicted values, and so it is infeasible to pass back the gradients to inform the predictive model with respect to how it should adjust its weights to improve the decision quality of the solutions it prescribes (Ferber et al., 2020). To overcome this problem, Wilder et al. (2019) added a quadratic regularization term to the objective function of the relaxed form of the combinatorial problem, but this method can only be applied to combinatorial problems with a totally unimodular matrix. Ferber et al. (2020) strengthened this method by employing a cutting-plane solution approach, which tightened the continuous relaxation by adding constraints to eliminate fractional solutions. Instead of computing the real decision loss by di-

rectly solving the combinatorial problem during the training process, some studies, including Elmachtoub and Grigas (2021) and Mandi et al. (2020), have designed a class of surrogate loss functions based on subgradient. One issue common to these approaches is that they need to repeatedly solve the (possibly relaxed) optimization problem, imposing a huge burden on computational efficiency. In contrast, Mulamba et al. (2021) used a noise-contrastive approach by viewing suboptimal solutions as noise examples and caching them, thus replacing optimization calls with a look-up table in the solution cache.

In summary, to the best of our knowledge, no studies have used PSC inspection records to support PSCO routing. Because PSC inspection data are public, port authorities can apply prescriptive analytics methods to improve decision performance in ship selection and PSCO routing. Identifying ships with more deficiencies and routing PSCOs to maximize the number of identified deficiencies while satisfying the required constraints would eliminate the adverse impacts of substandard ships on maritime transportation and improve the efficiency of port operations. Therefore, we bridge the following research gaps. First, we innovatively use PSC data to inform the decisions of PSCO routing while considering multiple practical constraints, including the berthing locations and berthing time windows of foreign visiting ships, and the lunch breaks of PSCOs. Because the PSCO routing problem is NP-hard, we recast this problem into a more compact combinatorial problem by generating undominated inspection templates. Second, we use a traditional two-stage framework to investigate this problem and apply the decision-focused framework to plug the transformed PSCO routing problem into the training process of the predictive model. This approach enables us to consider the impact of the predictions on downstream decisions. Our study is the first

PSC-related study to fully combine prediction and combinatorial optimization.

3.3 PSCO Routing Problem

Section 3.3.1 mathematically presents the PSCO routing problem. Section 3.3.2 transforms the original PSCO routing problem into a more compact formulation by generating undominated inspection templates. Finally, Section 3.3.3 conducts several groups of computational experiments to compare the solution efficiency of the two proposed models.

3.3.1 PSCO routing model M1

The PSCO routing model stems from the PSCO scheduling problem, which involves selecting the set of ships to be inspected and assigning these ships to PSCOs with the goal of maximizing the number of deficiencies identified among the inspected ships (Yan et al., 2021a). Based on the PSCO scheduling problem, the proposed PSCO routing problem considers not only the matching of ships and PSCOs, but also the inspection sequence of the ships assigned to each PSCO. To solve this problem, human resources, time resources, the ships' predicted deficiency numbers, and the ships' berthing locations and time windows are considered simultaneously.

Denote the number of foreign visiting ships that need to be inspected on a given working day by N and the ships by $i = 1, ..., N$. Each ship i is characterized by a vector of features $\mathbf{a}_i$, an arrival time O_i, a departure time C_i, a fixed berthing location P_i, and the duration required for an inspection t_i'. The arrival and departure times constitute the berthing time window $[O_i, C_i]$ of ship i during which

the ship is available for inspection. Following Yan et al. (2021a), we assume that for all of the inspected ships, a typical PSC inspection lasts two hours, that is, $t_i^{'} = 2$ $(i = 1, ..., N)$. Denote the number of available PSCOs on duty for a working day by M and the PSCOs by $m = 1, ..., M$. The PSCOs generally work from 8:00 to 11:00 and from 14:00 to 17:00. They depart from the office to perform inspections, and the office is denoted by the index $i = 0$. The fact that the PSCOs can leave the office to perform inspections at any time between 8:00 and 17:00 implies that the time window during which their office can be visited as a starting location is denoted by $[O_0, C_0] = [8, 17]$. Between 11:00 and 14:00, the PSCOs spend one hour having a lunch break and another two hours working. The lunch break index is denoted by $i = N + 1$ and the duration required for the lunch break is denoted by $t_{N+1}^{'} = 1$. Similarly, the time window during which the PSCOs can have lunch at location $N + 1$ is denoted by $[O_{N+1}, C_{N+1}] = [11, 14]$. When the PSCOs finish their assigned work or when the working day is over, the PSCOs return to the office. Commonly, the PSCOs both start and finish their work at the office. However, for modeling convenience, we denote the office location by two indices. Different from the index $i = 0$ when the office is regarded as the starting location for the PSCOs' daily work, the office is denoted by $i = N + 2$ when it is regarded as the ending location. Similarly, because the PSCOs can return to the office at any time between 8:00 and 17:00 in a working day, the time window during which the ending location can be visited is denoted by $[O_{N+2}, C_{N+2}] = [8, 17]$. Therefore, there are $N + 3$ location indices to be considered in this problem, including N berthing locations of foreign visiting ships, the location of start of work $(i = 0)$, the location of end of work $(i = N + 2)$, and the location of lunch break $(i = N + 1)$. Furthermore, the duration spent at the starting and ending locations

is denoted by $t_i^{'} = 0$ $(i = 0, N + 2)$. Therefore, each location i $(i = 1, ..., N)$ is characterized with a time window $[O_i, C_i]$ and a duration $t_i^{'}$. Finally, we denote the travel time from location i to location j by t_{ij} $(i, j = 0, ..., N + 2)$.

Before solving the PSCO routing problem, information concerning each ship's berthing time window and berthing location, and the PSCO's travel time between different locations is known to the port authority; the deficiency conditions of the foreign visiting ships are not known. We denote the set of uncertain numbers of ship deficiencies by $\tilde{\mathbf{d}} := \{\tilde{d}_i | i = 1, ..., N\}$. The sets of decision variables are denoted by $\mathbf{x} := \{x_{ijm} | i, j = 0, ..., N + 2; m = 1, ..., M\}$, where $x_{ijm} = 1$ if a visit to location i is followed by a visit to location j by PSCO m and 0 otherwise; $\mathbf{y} := \{y_{im} | i = 1, ..., N; m = 1, ..., M\}$, where $y_{im} = 1$ if ship i is assigned to be inspected by PSCO m and 0 otherwise; and $\mathbf{s} := \{s_{im} | i = 0, ..., N + 2; m = 1, ..., M\}$, where s_{im} represents the start time of the visit to location i by PSCO m. We then use P to denote a large constant. The PSCO routing problem is as follows:

$$\max \mathbb{E}\left[z(\tilde{\mathbf{d}}, \mathbf{y})\right] = \max \mathbb{E}\left[\sum_{m=1}^{M}\sum_{i=1}^{N} \tilde{d}_i y_{im}\right] \tag{3.1}$$

subject to

$$\sum_{m=1}^{M}\sum_{j=1}^{N+2} x_{0jm} = \sum_{m=1}^{M}\sum_{i=0}^{N+1} x_{i,N+2,m} = M \tag{3.2}$$

$$\sum_{i=0}^{N+1} x_i km = \sum_{j=1}^{N+2} x_{kjm} = y_{km} \quad k = 1, ..., N + 1; m = 1, ..., M \tag{3.3}$$

$$s_{im} + t_{ij} + t_i^{'} - s_{jm} \leq P(1 - x_{ijm}) \quad i, j = 0, ..., n + 2; m = 1, ..., M \tag{3.4}$$

$$\sum_{m=1}^{M} y_{im} = 1 \quad i = 1, ..., N \tag{3.5}$$

$$y_{im} = 1 \quad i = 0, N+1, N+2; m = 1, ..., M \tag{3.6}$$

$$O_i - P(1 - y_{im}) \leq s_{im} \quad i = 0, ..., N+2; m = 1, ..., M \tag{3.7}$$

$$s_{im} + t_i^{'} \leq C_i + P(1 - y_{im}) \quad i = 0, ..., N+2; m = 1, ..., M \tag{3.8}$$

$$x_{ijm}, y_{im} \in \{0, 1\} \quad i, j = 0, ..., N+2; m = 1, ..., M \tag{3.9}$$

$$s_{im} \geq 0 \quad i, j = 0, ..., N+2; m = 1, ..., M. \tag{3.10}$$

Objective function (3.1) maximizes the expected total number of deficiencies identified among all foreign visiting ships to be inspected. Constraint (3.2) guarantees that all PSCOs start work at location 0 and end work at location $N + 2$. Constraints (3.3) and (3.4) guarantee each PSCO's connectivity and timeline. Constraints (3.5) mean that each ship is inspected no more than once. Constraints (3.6) ensure that all PSCOs must visit locations 0, $N+2$, and $N+1$ to start and end their work and have their lunch breaks. Constraints (3.7) and (3.8) restrict the visit to each ship's time window and the lunch break to a specific period; that is, a ship can only be inspected during its berthing time window and the PSCOs can only have lunch break during the specified time window. Constraints (3.9)–(3.10) define the domains for decision variables.

3.3.2 PSCO routing model M2

Observing the structure of model M1, we note that it is a practical application of the team orienteering problem with time windows (Vansteenwegen et al., 2009). The team orienteering problem with time windows is a highly constrained problem that is difficult to solve. Golden et al. (1987) proved that the orienteering problem

is NP-hard. Therefore, it is reasonable to believe that model M1 is unlikely to be solved to optimality using a polynomial-time algorithm. To improve the solution efficiency of the PSCO routing problem M1, we adopt a route generation method to transform model M1 into a more compact model M2.

Recall that the total daily working time of a PSCO is eight hours, and the duration of an inspection is two hours. This implies that a PSCO can inspect zero, one, two, or three ships in one day, considering both the duration of the lunch break and the time spent travelling between different locations. Therefore, the PSCO routing problem can be reformulated as the problem of selecting and assigning the sets of ships that can be inspected to all available PSCOs. Define the number of ships inspected by a PSCO during a working day as L, where $L \in \{0, 1, 2, 3\}$. Given that L ships are selected from N visiting foreign ships, the total number of combinations is $C_N^L = N!/\left(L!\left(N-L\right)!\right)$. Denote a combination of L ships by the set $\mathbf{S}$, where $|\mathbf{S}| = L$. We then define set $\mathbf{S}$ as an inspection template when it is feasible for one PSCO to inspect all the ships in the set in a single working day. Then, the PSCO routing problem is modified to first select the set of inspection templates that maximize the number of deficiencies identified, while ensuring that each ship is inspected no more than once by a PSCO, and then to route the PSCOs based on the selected inspection templates. The method of selecting the inspection templates is illustrated in Algorithm 3.1, which follows Yan et al. (2021a). The method of routing the available PSCOs is illustrated in Algorithm 3.3.

We first illustrate the basic idea of Algorithm 3.1 as follows. Although we can obtain C_N^L combinations (sets of templates), not every combination is feasible considering the hard constraints on the visiting time windows and the travel time between two locations. To examine whether it is feasible for a single PSCO to

inspect all of the ships in **S**, we first need to verify whether there exists a feasible route that satisfies all of the constraints. For a feasible route, a PSCO needs to visit $L+3$ locations (including the office as a starting location, the berthing locations of L ships, the lunch break location, and the office as an ending location) to finish all inspection work during a day. To this end, we define α as a location, and each location is labeled with a duration time t_α, an earliest start time λ_α, and a latest end time $\bar{\lambda}_\alpha$. If a PSCO visits a location where ship i is berthed for an inspection, then $t_\alpha = 2$, $\lambda_\alpha = O_i$, and $\bar{\lambda}_\alpha = C_i$; if a PSCO visits the office when starting work ($i = 0$) or finishing work ($i = N+2$), then $t_\alpha = 0$, $\lambda_\alpha = 8$, and $\bar{\lambda}_\alpha = 17$; if a PSCO visits the lunch break location, then $t_\alpha = 1$, $\lambda_\alpha = 11$, and $\bar{\lambda}_\alpha = 13$. Considering that the starting and ending location, which is the office, is indifferent in each set, there are $(L+1)!$ candidate routes for the PSCOs to complete their tasks (note that not all of the candidate routes are feasible, because we do not consider the travel time between two locations and the different time windows of foreign visiting ships). For a particular route, we define the locations visited by a PSCO as $(\alpha_1, ..., \alpha_{L+3})$, where α_l $(l = 1, ..., L+3)$ is the lth location to visit; t_{α_l}, λ_{α_l}, and $\bar{\lambda}_{\alpha_l}$ are the visiting duration, the earliest visiting time, and the latest visiting time, respectively, for location α_l. To ensure that $L+3$ locations can be visited in the defined sequence of a route within the specified working time limit, we define the decision variable ζ_l as the start time of visiting location α_l. Then, $L+3$ locations can be visited in the above sequence by one PSCO if and only if there is a set of solutions ζ_l $(l = 1, ..., L+3)$, that satisfies the following constraints:

$$\zeta_l \geq \lambda_{\alpha_l} \quad l = 1, ..., L+3 \tag{3.11}$$

$$\zeta_l + t_{\alpha_l} \leq \bar{\lambda}_{\alpha_l} \quad l = 1, ..., L+3 \tag{3.12}$$

$$\zeta_{l+1} \geq \zeta_l + t_{\alpha_l} + t'_{l,l+1} \quad l = 1, ..., L+3, \tag{3.13}$$

where $t'_{l,l+1}$ denotes the travel time from location α_l to α_{l+1}.

Proposition 3.1 *For a candidate route, whether Constraints* (3.11)– (3.13) *have a feasible solution is guaranteed by the following conditions: for location* α_1*, let its start time* $\zeta_1 = \lambda_{\alpha_1} = 8$*; for location* α_l $(l = 2, ..., L+3)$*, let its start time* $\zeta_l = \max\{\zeta_{l-1} + t_{\alpha_{l-1},l}, \lambda_{\alpha_l}\}$*; if* $\zeta_l \leq \bar{\lambda}_{\alpha_l} - t_{\alpha_l}$ $(l = 1, ..., L+3)$*, then the candidate route is feasible, otherwise it is infeasible.*

Property 3.1 *For two inspection templates* $\mathbf{S}$ *and* $\mathbf{S}'$*, if* $\mathbf{S} \subseteq \mathbf{S}'$*, then inspection template* $\mathbf{S}$ *is dominated by inspection template* $\mathbf{S}'$*. If inspection template* $\mathbf{S}'$ *does not contain any other inspection template, it is considered an undominated inspection template because inspecting it can always identify no fewer deficiencies than inspecting any other inspection template contained within it.*

Therefore, the overall procedure to generate the set of all undominated inspection templates, denoted by $\mathbf{H}$, is shown in Algorithm 3.1. After obtaining the set of undominated inspection templates $\mathbf{H}$ and parameters η_i^h $(i = 1, ..., N; h \in \mathbf{H})$, we further introduce a binary decision variable y'_h, which equals 1 if an undominated inspection template $h \in \mathbf{H}$ is assigned to one PSCO, and a binary decision variable u_i $(i = 1, ..., N)$, which equals 1 if and only if ship i is inspected by one PSCO. Then, model M2 is as follows:

$$\max \mathbb{E}\left[z(\tilde{\mathbf{d}}, \mathbf{u})\right] = \max \mathbb{E}\left[\sum_{i=1}^{N} \tilde{d}_i u_i\right] \tag{3.14}$$

subject to

$$\sum_{h \in \mathbf{H}} y_h^{'} \leq M \tag{3.15}$$

$$u_i \leq \sum_{h \in \mathbf{H}} \eta_i^h y_h^{'} \quad i = 1, ..., N \tag{3.16}$$

$$y_h^{'} \in \{0, 1\} \quad h \in \mathbf{H} \tag{3.17}$$

$$u_i \in \{0, 1\} \quad i = 1, ..., N, \tag{3.18}$$

where $\mathbf{u} = [u_1, ..., u_N]^\top$. Objective function (3.14) maximizes the expected total number of deficiencies identified. Constraint (3.15) provides that the maximum number of adopted inspection templates cannot exceed the number of available PSCOs. Constraints (3.16) indicate the relationship between u_i and $y_h^{'}$. Constraints (3.17)–(3.18) represent the domains for decision variables.

We note that M2 is more compact than M1, but solving M2 can only yield the selection of the optimal inspection templates, and does not provide information on how to route the PSCOs. Next, we describe how to route the available PSCOs based on these selected inspection templates. Because there may be several feasible routes to finish the tasks in a selected inspection template, to determine the optimal route, we first define an optimal route as the one with the earliest return time to the office. Given an optimal inspection template $\mathbf{S}^*$ containing $|\mathbf{S}^*|$ ships, adopting this inspection template requires visiting $|\mathbf{S}^*| + 3$ locations and generating $(|\mathbf{S}^*| + 1)!$ candidate routes for the PSCOs to complete their tasks. The above-defined earliest return time to the office is denoted by $\zeta_{|\mathbf{S}^*|+3}$. Algorithm 3.2 is introduced to find the optimal route for a selected inspection template.

After obtaining the optimal routes for all of the selected inspection templates,

Algorithm 3.1 Generate undominated inspection templates

Input: Set of locations $\{0, ..., N+2\}$; duration spent at each location t_i $(i \in \{0, ..., N+2\})$; time window of each location $[O_i, C_i]$ $(i \in \{0, ..., N+2\})$.

Output: The set of undominated inspection templates **H**, binary variable parameter η_i^h indicating whether ship i is contained in inspection template h $(i \in \{0, ..., N+2\}; h \in \mathbf{H})$.

Initialize $\mathbf{H} = \varnothing$.
for $L = 0, 1, 2, 3$ **do**
    Formulate all of the possible combinations that select L ships from all of the visiting ships, denoted by **Q**.
    for each combination $q \in \mathbf{Q}$ **do**
        Initialize `feasibility` $=$ `False`.
        Define set **V** that contains all of the candidate routes of starting work, inspecting the ships in q, having lunch break, and ending work.
        for each candidate route $v \in \mathbf{V}$ **do**
            Test the feasibility of v using Proposition 3.1.
            if v is feasible **then**
                $\mathbf{H} \leftarrow \mathbf{H} \cup \{q\}$.
                For locations i included in q, set $\eta_i^q = 1$; otherwise, $\eta_i^q = 0$.
                Update `feasibility` $=$ `True`.
                Break.
            end if
        end for
        if `feasibility` $=$ `True` **then**
            Continue.
        end if
    end for
end for
Delete dominated inspection templates in **H** using Property 3.1.

Algorithm 3.2 Find the optimal route for an inspection template.

Input: A selected inspection template $\mathbf{S}^*$; set of locations $\{0, ..., N+2\}$; duration spent at each location t_i $(i = 0, ..., N + 2)$; time window of each location $[O_i, C_i]$ $(i = 0, ..., N + 2)$.

Output: `Optimal_route`.

1: Define $\mathbf{V}$ as a set that contains all of the candidate routes of starting work, inspecting the ships in $\mathbf{S}^*$, having lunch break, and ending work. Denote a candidate route as p_j $(p_j \in \mathbf{V}; j \in \{1, ..., (|\mathbf{S}^*| + 1)!\})$ and the earliest return time to the office for p_j as ζ_j.
2: Initialize `Optimal_route` $= \varnothing$, `Earlist_finish_time` $= \infty$.
3: **for** $j = 1, ..., (|\mathbf{S}^*| + 1)!$ **do**
4: Test the feasibility of a candidate route p_j using Proposition 3.1.
5: **if** p_j is feasible and $\zeta_j \leq$ `Earlist_finish_time` **then**
6: `Earlist_finish_time` $\leftarrow \zeta_j$.
7: `Optimal_route` $\leftarrow p_j$.
8: **end if**
9: **end for**

we cannot directly assign the determined optimal routes to available PSCOs, because model M2 cannot guarantee that each ship is inspected no more than once, since two inspection templates may contain the same ship. Hence, to avoid duplicate inspections for a ship while adopting optimal routes for the selected inspection templates, we further require that a selected ship be inspected only once. Therefore, we introduce a set $\mathcal{S}$ containing the ships that have already been inspected. The overall procedure of PSCO routing after solving M2 is shown in Algorithm 3.3.

Finally, by observing Constraint (3.15), we note that it is possible that in an optimal solution, some PSCOs are not assigned to any inspection task. In other words, not all PSCOs have ships to inspect according to an optimal solution. To ensure temporal fairness in work assignments, we recommend that the port state shuffle the index of PSCOs each working day so that each PSCO has a prioritized opportunity to be assigned inspection work.

Algorithm 3.3 PSCO routing after solving M2.

Input: Set of undominated inspection templates $\mathbf{H}$, optimal solutions $y_h^{'}$ $(h \in \mathbf{H})$.
Output: Scheduling results.

Define $\mathbf{H}^{'} = \{h | h \in \mathbf{H}, y_h^{'} = 1, \}$ as the set of selected inspection templates.
Denote $\mathcal{S} = \varnothing$ as the set of inspected ships.
Initialize $i = 0$.
for $h \in \mathbf{H}'$ **do**
    if $i = 0$ **then**
        Determine the optimal route p for the inspection template h by applying Algorithm 3.2.
        Assign p to PSCO 1.
        $i \leftarrow i + 1$.
        Initialize $h' \leftarrow h$.
    else
        Update $\mathcal{S}$ by merging the last inspection template: $\mathcal{S} \leftarrow \mathcal{S} \cup \{h'\}$.
        Delete $\mathcal{S}$ from the current inspection template h: $h \leftarrow h \setminus \mathcal{S}$.
        Determine the optimal route p for inspection template h by applying Algorithm 3.2.
        Assign p to PSCO $i + 1$.
        $h' \leftarrow h$.
    end if
end for
for $m = |\mathbf{H}^{'}| + 1, ..., M$ **do**
    Not assign any route to PSCO m.
end for

3.3.3 Comparison of M1 and M2

To facilitate the introduction of the decision-focused learning framework proposed for the PSCO routing problem, which requires high computational efficiency, we first conduct a basic computational experiment to compare the solution efficiency of M1 and M2. To obtain the travel time t_{ij} between each pair of locations (i, j), we divide the port area of concern into five parts, with each location belonging to one part. We then introduce e_i $(i = 0, ..., N+2)$ as the part index of each location i and an auxiliary index $t'_{e_i e_j}$ indicating the travel time between a pair of area parts (e_i, e_j). We further assume that the location of the office and the lunch break is in part 1, indexed by $e_0, e_{N+1}, e_{N+2} = 1$, and the berthing locations of foreign visiting ships are randomly set to parts 2 to 5. Then, the travel time t_{ij} can be obtained by mapping (e_i, e_j) with $t'_{e_i e_j}$. For example, assuming that ships 1 and 2 berth at parts $e_1 = 2$ and $e_2 = 4$, respectively, and the travel time between parts 2 and 4, t'_{24}, is one hour, we can thus obtain that the travel time between ships 1 and 2, t_{12}, is one hour. The travel times, $t'_{e_i e_j}$, between two area parts (e_i, e_j) are shown in Table 3.1. Furthermore, we randomly set the values of the time window $[O_i, C_i]$, part indexes e_i, and number of deficiencies d_i for ship i following the ranges shown in Table 3.2.

Table 3.1: Travel time (hour) $t_{e_i e_j}$ between two area parts (e_i, e_j)

$t_{e_i e_j}/(e_i, e_j)$	1	2	3	4	5
1	0	0.2	0.4	0.3	0.2
2	0.2	0	0.1	0.3	0.2
3	0.4	0.1	0	0.4	0.2
4	0.3	0.3	0.4	0	0.3
5	0.2	0.2	0.2	0.3	0

Table 3.2: The ranges of input integer parameters for models M1 and M2

Parameter	O_i	C_i	p_i	d_i
Range	$8 \sim 11$	$13 \sim 17$	$2 \sim 5$	$0 \sim 10$

To compare the solution efficiency of models M1 and M2 with different groups of input parameters and under different instance sizes, we design four groups of instances. One group is denoted by (M, N), where M represents the number of available PSCOs with values of 2, 4, 6, and 8, and N represents the number of foreign visiting ships with values of 10, 15, 20, and 25, respectively. For each instance, we rerun the experiments with 10 groups of randomly generated input parameters. The programs are coded in Python and the models are solved by Gurobi. Because the PSCO routing problem is NP-hard, it may be impractical to solve large-scale instances. To save computational efforts, we limit the solution time of each model to 200 seconds. That is, given a model M1 or M2, Gurobi stops when a specified time limit is reached. Next, the current best solution found by Gurobi is retrieved as the near-optimal solution. The computational results for models M1 and M2 are shown in Table 3.3.

As we can see from Table 3.3, for model M1, the average solution time increases greatly as the instance size increases. For instance size (8,25), nearly all of the models cannot be solved to optimality within 200 seconds. Furthermore, the standard deviations of problems with different groups of parameters in the same instance size show that their solution time may vary greatly due to the difference parameters involved in the model. For example, assigning visiting ships with longer time windows can increase the computing time required to solve M1, because doing so can increase the search space of the optimal solutions. Com-

Table 3.3: Computational results for models M1 and M2

Instance			(2,10)	(4,15)	(6,20)	(8,25)
M1	Model solution time (s)	Average (Avg.)	2.22	92.16	164.55	200.93
		Standard deviation (Std.)	3.94	93.48	75.81	0.08
	Number of groups where the optimal solution is not found within 200s		0	4	8	10
M2	Time spent on Algorithm 3.1 (s)	Avg.	0.29	1.11	2.68	5.61
		Std.	0.04	0.16	0.30	0.19
	Model solution time (s)	Avg.	0.02	0.08	0.27	0.52
		Std.	0.02	0.05	0.11	0.13
	Time spent on Algorithm 3.3 (s)	Avg.	0.00	0.01	0.01	0.01
		Std.	0.00	0.00	0.00	0.00
	Overall CPU time (s)	Avg.	0.32	1.20	2.95	6.14
		Std.	0.04	0.11	0.20	0.09
	Average objective function value gap of M2 to M1		0.00%	0.00%	0.00%	0.09%

pared with the average solution time of M1 under different instance sizes, solving M2 takes a much shorter time. For M2, the time spent on Procedure B.1 to find undominated inspection templates is the determinant of the overall CPU time, followed by the solution time for M2. However, solving M2 under different instance sizes never costs more than one second and the solution time for M2 does not fluctuate very much in different groups of input parameters under the same instance size. Finally, by observing the average gap of the objective functions between M2 and M1, the solution qualities of M1 and M2 only show a slight difference under instance size (8,25). Although the solution qualities of M1 and M2 are nearly the same, because M2 is much more computationally efficient than M1, the rest of the analysis in the study will use M2 as the target optimization problem, which is plugged into the ML algorithm in the following sections.

3.4 The Two-Stage Framework

Before the port state authority routes the PSCOs, the number of deficiencies of foreign visiting ships is unknown. Fortunately, the authority has access to a historical dataset $\mathcal{D} = \{(\mathbf{a}_i, d_i)\}_{i=1}^{R}$ with an R number of PSC inspection records, where $\mathbf{a}_i \in \mathbb{R}^q$ denotes a vector of q features for ship i, and d_i is an integer indicating the ship's number of deficiencies. One traditional and straightforward way to solve the PSCO routing problem is to first train an ML model $f(\boldsymbol{\omega}, \mathbf{a})$ with $\mathbf{a}$ as the input and $\boldsymbol{\omega}$ as the weights (parameters) to predict the value of d, denoted by $\hat{d}$. This ML model is trained to minimize a specified loss function using dataset $\mathcal{D}$, such as the mean squared error (MSE) loss function L_{MSE} for a regression task defined as follows:

$$L_{\text{MSE}} = \frac{1}{R} \sum_{i=1}^{R} \left(\hat{d}_i - d_i \right)^2. \tag{3.19}$$

Given the loss function L_{MSE}, ML model f is trained by solving the following optimization problem to obtain the optimal $\boldsymbol{\omega}^*$:

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} L_{\text{MSE}} = \arg\min_{\boldsymbol{\omega}} \frac{1}{R} \sum_{i=1}^{R} \left(\hat{d}_i - d_i \right)^2. \tag{3.20}$$

Then, when presented with a new example with feature vector $\mathbf{a}_0$, model f with the optimal $\boldsymbol{\omega}^*$ can be applied to predict the number of deficiencies of the new example $\hat{d}_0 = f(\boldsymbol{\omega}^*, \mathbf{a}_0)$. Finally, the predicted values of all of the foreign visiting ships that may be inspected are put into the PSCO routing problem to derive the routing results. This framework is generally termed either the predict-then-optimize framework or the two-stage framework. As described in Section 3.2,

gradient-descent ML algorithms are most commonly used in the decision-focused learning framework. To facilitate our comparison of the two-stage framework and the decision-focused learning framework, we specify that the ML algorithm used in our study is the artificial neural network (ANN) (Yegnanarayana, 2009) because of its high popularity and good performance.

An ANN generally has three layers: an input layer, a hidden layer, and an output layer (Yegnanarayana, 2009). The outputs of the input and hidden layers act as the input to the ANN's direct downstream layer. Training an ANN refers to adjusting its weights (i.e., ω as mentioned above) that connect the neurons of consecutive layers, with the goal of minimizing the loss. Backpropagation is the most widely used training algorithm for ANNs. It is a way of computing the gradients of the loss on the weights by recursively applying chain rules such that the current prediction loss in the output layer can be reversely passed to the preceding layers, and the weights can be adjusted to minimize the loss. Backpropagation computes gradients in an efficient manner, making it feasible to use the gradient-descent method to train multilayer ANNs. The hyperparameters considered in an ANN mainly include learning rate, epochs (number of iterations), and batch size, which together deal with the problems of underfitting and overfitting. The learning rate controls the speed of weight update by determining the step size at each iteration when moving toward a minimum loss value. Epochs refer to the number of times the whole training dataset is trained. The batch size refers to the number of examples in a mini-batch, and a mini-batch is a strict and non-empty subset of the whole training set. Examples in a mini-batch are passed to the network at one time to update the weights. Therefore, the total number of batches in an epoch is equal to the ratio of the size of the whole training set to the batch size. For a

more detailed introduction to ANN, please refer to Yegnanarayana (2009). Algorithm 3.4 depicts the two-stage framework including a standard gradient-descent learning procedure.

Algorithm 3.4 Two-stage framework.

Input: Training data $\mathcal{D} = \{(\mathbf{a}_i, d_i)\}_{i=1}^{R}$, learning rate α, epochs, batch size.

for each epoch **do**
  for each batch **do**
    for each sample **a do**
      Predict the number of deficiencies of **a**, denoted by $\hat{d}$.
    end for
    Calculate the accumulated MSE L_{MSE} for the set of samples in a batch.
    Update $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \alpha \frac{dL_{\text{MSE}}}{d\hat{\mathbf{d}}} \frac{d\hat{\mathbf{d}}}{d\boldsymbol{\omega}}$, where $\hat{\mathbf{d}}$ denotes the vector of predicted values in a batch.
  end for
end for

3.5 The Decision-Focused Learning Framework

This section introduces the decision-focused learning framework. Section 3.5.1 defines the regret loss. Section 3.5.2 introduces a new family of noise-contrastive loss functions. Section 3.5.2 describes the gradient-descent decision-focused learning framework using the noise-contrastive losses proposed in Section 3.5.2. We mainly follow the decision-focused learning framework proposed by Mulamba et al. (2021)

3.5.1 The decision loss

One possible disadvantage of the two-stage framework is that it does not consider the impact of the predictions on the downstream optimization problem, which

consequently generates sub-optimal decisions. Therefore, a more appropriate approach is to integrate the prediction and the decision procedures when training the ML model, which requires using a decision-focused loss to take decision errors into account. For the PSCO routing problem under the decision-focused learning framework, another ML model, denoted by $f'\left(\boldsymbol{\omega}', \mathbf{a}_i\right)$, is trained to generate the prediction $\hat{d}_i'$ for ship i that can provide optimal decisions with respect to the real value of d_i. To measure the accuracy of the prescribed decisions, instead of using the loss function (16), we adopt the decision regret loss denoted by L_{regret}. Unlike the traditional loss function, which is computed by summing the prediction error of each data example, the regret loss is computed based on the instance level, that is, summing the decision error of $T = \lfloor R/N \rfloor$ instances (recall that there are R PSC inspection records and an instance contains N foreign visiting ships). Because we plug model M2 into the decision-focused learning framework and recall that the objective function of model M2 is $z(\mathbf{d}, \mathbf{u})$, finding the optimal parameters in $\boldsymbol{\omega}'$ over a set of T training instances can be established as

$$\boldsymbol{\omega}'^{*} = \arg\min_{\boldsymbol{\omega}'} L_{\text{regret}} = \arg\min_{\boldsymbol{\omega}'} \frac{1}{T} \sum_{j=1}^{T} \left[z(\mathbf{d}_j, \mathbf{u}^*(\mathbf{d}_j)) - z(\mathbf{d}_j, \mathbf{u}^*(\hat{\mathbf{d}}_j')) \right], \quad (3.21)$$

where $\hat{\mathbf{d}}_j'$ is the N-dimensional vector of the predicted numbers of deficiencies for jth instance and $\mathbf{u}^*(\hat{\mathbf{d}}_j')$ denotes the corresponding optimal solution. By observing Formula (3.21), we find that the regret loss is the sum of the difference between the objective function values derived from 1) the perfect solution $\mathbf{u}^*(\mathbf{d}_j)$ under the real deficiency number vector $\mathbf{d}_j$ and 2) the optimal solution $\mathbf{u}^*(\hat{\mathbf{d}}_j')$ under the predicted deficiency number vector $\hat{\mathbf{d}}_j'$. As mentioned above, we use an ANN as our predictive model. However, we cannot directly use the original regret loss

L_{regret} during its training process because model M2 involves integer variables and L_{regret} cannot differentiate over the arg min on $\hat{\mathbf{d}}'_j$. Therefore, our goal is to find a differentiable and efficient-to-compute loss function for the decision-focused learning framework, which is introduced in the next section.

3.5.2 Contrastive losses

We note that model M2 cannot be easily embedded in the ANN training algorithm as it cannot be easily differentiated due to its structure and discontinuity. Inspired by the contrastive losses proposed by Mulamba et al. (2021), we design the following decision-focused learning framework for the PSCO routing problem. Contrastive losses are proposed based on the fact that probabilistic models can define a parametric probability distribution over feasible solutions for an optimization problem, and maximum likelihood estimation can be used to find the distribution parameters, making the observed perfect solution appear with the greatest probability. The exponential distribution is ubiquitous in ML research among popular probabilistic models, as it has the required form of the optimal solution to maximum entropy problem (Berger et al., 1996). This study thus proposes an exponential distribution to fit model M2.

Let $\mathcal{U}$ denote the state space of the feasible solutions of an optimization problem $\max_{\mathbf{u}\in\mathcal{U}} z(\mathbf{d},\mathbf{u})$, where $\mathbf{u}\in\mathcal{U}$ is a feasible solution. Then, we define the following exponential distribution over $\mathcal{U}$ under the prediction $\hat{\mathbf{d}}$, which represents the probability of deriving solution $\mathbf{u}$ under the prediction $\hat{\mathbf{d}}$:

$$\mathbb{P}(\mathbf{u}|\hat{\mathbf{d}}) = \frac{1}{Z}\exp(z(\hat{\mathbf{d}},\mathbf{u})), \tag{3.22}$$

where Z normalizes the distribution over the solution space $\mathcal{U}$ and is expressed as

$$Z = \sum_{\mathbf{u}' \in \mathcal{U}} \exp(z(\hat{\mathbf{d}}, \mathbf{u}')). \tag{3.23}$$

If $\mathbf{u}^*(\hat{\mathbf{d}})$ is the maximizer of $\max_{\mathbf{u} \in \mathcal{U}} z(\mathbf{d}, \mathbf{u})$ (i.e., $\mathbf{u}^*(\hat{\mathbf{d}}) = \mathbf{u}^*(\mathbf{d})$), it can maximize Formula (3.22) among all $\mathbf{u} \in \mathcal{U}$. This implies that if we can learn the parameters in $\boldsymbol{\omega}'$ that can maximize the likelihood of $\mathbb{P}(\mathbf{u}^*(\mathbf{d})|\hat{\mathbf{d}})$, we can obtain the true perfect solution $\mathbf{u}^*(\mathbf{d})$ with the highest probability under the prediction $\hat{\mathbf{d}}$. Consequently, for all training instances, our goal is to learn the parameters in $\boldsymbol{\omega}'$ that can maximize the probability that the true perfect solutions are prescribed.

However, obtaining an accurate Z is almost impossible for most integer programming problems because it is necessary to find all of the possible solutions belonging to $\mathcal{U}$. Therefore, we apply NCE (Mikolov et al., 2013) to obtain an estimation of Z, which requires establishing a solution pool with a limited number of noise samples that are feasible solutions to the optimization problem. In Section 3.5.3, we describe the method of establishing the solution pool. Below, we introduce four forms of loss functions based on NCE following Mulamba et al. (2021).

NCE-basic loss. We first define noise samples as solutions to the optimization problem that are feasible but different from the perfect solution $\mathbf{u}^*(\mathbf{d})$ and that belong to the subset $\mathcal{U}'$, where $\mathcal{U}' \subset \mathcal{U} \setminus \mathbf{u}^*(\mathbf{d})$. These noise samples constitute the solution pool that can be used to approximate Z. Therefore, $\mathcal{U}'$ is the solution pool that we need. Next, our goal is to learn the parameters in $\boldsymbol{\omega}'$ by maximizing the product of the ratios between the probability of the perfect solution $\mathbf{u}^*(\mathbf{d})$ under the prediction $\hat{\mathbf{d}}$ and the probability of any noise sample $\mathbf{u}'$ in $\mathcal{U}$ under the prediction

$\hat{\mathbf{d}}$ for all instances, which is expressed as

$$\begin{aligned}\boldsymbol{\omega}'^{*} &= \arg\max_{\boldsymbol{\omega}'} \log \prod_{j=1}^{T} \prod_{\mathbf{u}' \in \mathcal{U}'} \frac{\mathbb{P}(\mathbf{u}_j^*(\mathbf{d}_j)|\hat{\mathbf{d}}_j)}{\mathbb{P}(\mathbf{u}'|\hat{\mathbf{d}}_j)} \\ &= \arg\max_{\boldsymbol{\omega}'} \log \prod_{j=1}^{T} \prod_{\mathbf{u}' \in \mathcal{U}'} \frac{\exp(z(\hat{\mathbf{d}}_j, \mathbf{u}_j^*(\mathbf{d}_j)))}{\exp(z(\hat{\mathbf{d}}_j, \mathbf{u}'))} \\ &= \arg\max_{\boldsymbol{\omega}'} \sum_{j=1}^{T} \sum_{\mathbf{u}' \in \mathcal{U}'} \left[z(\hat{\mathbf{d}}_j, \mathbf{u}_j^*(\mathbf{d}_j)) - z(\hat{\mathbf{d}}_j, \mathbf{u}') \right].\end{aligned} \tag{3.24}$$

To minimize the decision loss, the above equation can be transformed into the following NCE-basic loss function:

$$L_{\text{NCE}} = \sum_{j=1}^{T} \sum_{\mathbf{u}' \in \mathcal{U}'} \left[z(\hat{\mathbf{d}}_j, \mathbf{u}') - z(\hat{\mathbf{d}}_j, \mathbf{u}_j^*(\mathbf{d}_j)) \right]. \tag{3.25}$$

This NCE-basic loss function can be easily embedded in the training process for ML algorithms, as both $\mathbf{u}'$ (noise samples) and $\mathbf{u}_j^*(\mathbf{d}_j)$ can be computed before the training process if we can formulate a solution pool $\mathcal{U}'$, and they can be regarded as constants.

MAP-basic loss. Furthermore, instead of considering all of the noise samples in $\mathbf{u}'$ when estimating the regret loss, we consider a special form of NCE called maximum a posteriori (MAP) estimation (Goodfellow, 2015). MAP estimation only considers the noise sample that has the highest probability of achieving the optimal objective function value for each instance under the corresponding prediction. Specifically, we define $\dot{\mathbf{u}}_j^* = \arg\max_{\mathbf{u}' \in \mathcal{U}'} z(\hat{\mathbf{d}}_j, \mathbf{u}')$. Similarly, learning the parameter $\boldsymbol{\omega}'$ by maximizing the product of the ratios between the probability of deriving the perfect solution $\mathbf{u}_j^*(\mathbf{d}_j)$ and that of deriving the noise sample $\dot{\mathbf{u}}_j^*$ given

the prediction can be expressed as

$$\begin{aligned}\boldsymbol{\omega}'^{*} &= \arg\max_{\boldsymbol{\omega}'} \log \prod_{j=1}^{T} \frac{\mathbb{P}(\mathbf{u}_j^*(\mathbf{d}_j)|\hat{\mathbf{d}}_j)}{\mathbb{P}(\dot{\mathbf{u}}_j^*|\hat{\mathbf{d}}_j)} \\ &= \arg\max_{\boldsymbol{\omega}'} \log \prod_{j=1}^{T} \frac{\exp(z(\hat{\mathbf{d}}_j, \mathbf{u}_j^*(\mathbf{d}_j)))}{\exp(z(\hat{\mathbf{d}}_j, \dot{\mathbf{u}}_j^*))} \\ &= \arg\max_{\boldsymbol{\omega}'} \sum_{j=1}^{T} \left[z(\hat{\mathbf{d}}_j, \mathbf{u}_j^*(\mathbf{d}_j)) - z(\hat{\mathbf{d}}_j, \dot{\mathbf{u}}_j^*) \right].\end{aligned} \tag{3.26}$$

Accordingly, the above equation can be transformed into the following MAP-basic loss function:

$$L_{\text{MAP}} = \sum_{j=1}^{T} \left[z(\hat{\mathbf{d}}_j, \dot{\mathbf{u}}_j^*) - z(\hat{\mathbf{d}}_j, \mathbf{u}_j^*(\mathbf{d}_j)) \right]. \tag{3.27}$$

NCE-variant loss and MAP-variant loss. We note that the objective function of model M2 is a linear function. Therefore, the original Formulas (3.25) and (3.27) can be rewritten in the following linear form:

$$L_{\text{NCE}} = \sum_{j=1}^{T} \sum_{\mathbf{u}' \in \mathcal{U}'} (\hat{\mathbf{d}}_j)^{\top} \left(\mathbf{u}' - \mathbf{u}_j^*(\mathbf{d}_j) \right). \tag{3.28}$$

$$L_{\text{MAP}} = \sum_{j=1}^{T} (\hat{\mathbf{d}}_j)^{\top} \left(\dot{\mathbf{u}}_j^* - \mathbf{u}_j^*(\mathbf{d}_j) \right). \tag{3.29}$$

By observing Formulas (3.28) and (3.29), we find that if $\hat{\mathbf{d}}_j = \mathbf{0}$, L_{NCE} and L_{MAP} equal zeros, which are the minimum loss. To avoid this case, we introduce variants of Formulas (3.25) and (3.27) by replacing $\hat{\mathbf{d}}_j$ with $\hat{\mathbf{d}}_j - \mathbf{d}_j$. This modification can be regarded as adding a regularization term to keep $\hat{\mathbf{d}}_j$ close to $\mathbf{d}_j$. Therefore, we

can obtain the NCE-variant loss and MAP-variant loss as follows:

$$L_{\text{NCE-v}} = \sum_{j=1}^{T} \sum_{\mathbf{u}' \in \mathcal{U}'} (\hat{\mathbf{d}}_j - \mathbf{d}_j)^\top \left(\mathbf{u}' - \mathbf{u}_j^*(\mathbf{d}_j) \right). \tag{3.30}$$

$$L_{\text{MAP-v}} = \sum_{j=1}^{T} (\hat{\mathbf{d}}_j - \mathbf{d}_j)^\top \left(\dot{\mathbf{u}}_j^* - \mathbf{u}_j^*(\mathbf{d}_j) \right). \tag{3.31}$$

By observing Formulas (3.30) and (3.31), we find that the NCE-variant loss and MAP-variant loss cannot be minimized by predicting $\mathbf{d}_j$ to be $\mathbf{0}$. These two losses can only be minimized by letting $\hat{\mathbf{d}}_j$ be close to $\mathbf{d}_j$. In this way, the prescribed solution under $\hat{\mathbf{d}}_j$ can approach the perfect solution $\mathbf{u}_j^*(\mathbf{d}_j)$, which is the ultimate goal of decision-focused learning.

3.5.3 Gradient-descent decision-focused learning with noise samples

Following the above introduction of the four types of contrastive losses, the main problem to be solved now is how to formulate the solution pool of noise samples $\mathcal{U}'$. We note that any feasible solution in $\mathcal{U}$ is a noise sample in $\mathcal{U}'$. However, finding all of the feasible solutions in $\mathcal{U}$ is time-consuming and nearly impossible, especially for large-scale combinatorial problems. Therefore, we first initialize $\mathcal{U}'$ by solving models using the real values of $\mathbf{d}$ before the training process, and then expand it while obtaining a new solution by solving model M2 using the predicted $\hat{\mathbf{d}}$ during the training process. Algorithm 3.5 shows the procedure of the gradient-descent decision-focused learning framework with noise samples.

There are three main differences between the proposed Algorithm 3.4 and Al-

Algorithm 3.5 Decision-focused learning framework

Input: Auxiliary parameters in model M2; training data $\mathcal{D}^{'} = \{(\mathbf{A}_j, \mathbf{d}_j)\}_{j=1}^{T}$, a fixed parameter p_{solve}, learning rate α', epochs, batch size; here $\mathbf{A}_j$ denotes the feature matrix of the jth instance.

Initialize $\boldsymbol{\omega}', \mathcal{U}' = \{\mathbf{u}^*(\mathbf{d}_j) | (\mathbf{A}_j, \mathbf{d}_j) \in \mathcal{D}^{'}\}$.
for each epoch **do**
    for each batch **do**
        for each instance **A do**
            Predict the numbers of deficiencies of the samples in instance **A**, denoted by $\hat{\mathbf{d}}$.
        end for
        if a random number between 0 and 1 is smaller than p_{solve} **then**
            Obtain $\mathbf{u}^*(\hat{\mathbf{d}})$ by solving model M2 with $\hat{\mathbf{d}}$.
            $\mathcal{U}' \leftarrow \mathcal{U}' \cup \mathbf{u}^*(\hat{\mathbf{d}})$.
        end if
        Calculate the accumulated surrogate decision losses L_{regret} for the set of instances in a batch.
        Update $\boldsymbol{\omega}' \leftarrow \boldsymbol{\omega}' - \alpha' \frac{dL_{\text{regret}}}{d\hat{\mathbf{D}}} \frac{d\hat{\mathbf{D}}}{d\boldsymbol{\omega}'}$, where $\hat{\mathbf{D}}$ denotes the matrix of predicted values in a batch.
    end for
end for

gorithm 3.5. First, the data records used to train the ML model in Algorithm 3.5 are averagely divided into T instances, with each instance represented by a feature matrix $\mathbf{A}$ and a vector of deficiency numbers $\mathbf{d}$. Instantiating this training dataset is to compute the regret loss at the instance level and formulate the solution pool using these training instances. Second, in addition to initializing the parameters in ω', Algorithm 3.5 initializes the solution pool $\mathcal{U}'$ by solving all of the training instances beforehand. An initialized solution pool can be expanded by adding a new solution $\mathbf{u}(\hat{\mathbf{d}})$ after obtaining the predicted $\hat{\mathbf{d}}$ and solving the optimization model M2 using $\hat{\mathbf{d}}$. Furthermore, the solution pool can be regarded as an inner approximation of $\mathcal{U}$, because the noise samples in $\mathcal{U}'$ can represent the convex hull of $\mathcal{U}$ if it contains all potentially optimal solutions. When more new solutions are added to the solution pool, we expect to obtain a tighter inner approximation for $\mathcal{U}$. Third, we introduce a parameter p_{solve} to represent the probability of calling a solver to obtain the current prescribed solution in Algorithm 3.5. If a random number between 0 and 1 is smaller than p_{solve}, the algorithm needs to call the solver and add the solution to the solution pool if it is not in the solution pool. This parameter may have an influence on the trade-off between efficiency and accuracy. That is, a larger p_{solve} leads to more intensive calls to the solver so that computational time is increased but decision error may be decreased.

3.6 Computational Experiments

This section presents the results of our computational experiments. In Section 3.6.1, we describe our dataset and the settings for each ML model. In Section 3.6.2, we compare the performance of the two-stage framework and the decision-focused

learning framework and present several interesting findings. In Section 3.6.3, we conduct a sensitivity analysis on p_{solve}.

3.6.1 Data description

This study uses a dataset with 3,026 PSC initial inspection records from January 2015 to December 2019 at the Hong Kong Port and the corresponding ship-related factors of the inspected ships. Hong Kong Port is a member of the Tokyo Memorandum of Understanding (MoU), which governs the Asia-Pacific region. The PSC inspection records are retrieved from the Asia Pacific Computerized Information System provided by the Tokyo MoU, and the ship-related factors are obtained from the World Shipping Register database . The main work of this study is to route the PSCOs to maximize the number of deficiencies identified on the foreign visiting ships selected for inspection. This is achieved by training ML models whose input is the ships' auxiliary features and whose output is the predicted number of deficiencies. This study considers 14 auxiliary features that are closely related to ship condition, according to Yan et al. (2020, 2021b), namely, ship age, gross tonnage, length, depth, beam, type, flag performance, recognized organization performance, company performance in the Tokyo MoU, last PSC inspection date in the Tokyo MoU, the number of ship deficiencies identified in the last inspection in the Tokyo MoU, the number of detentions in all historical PSC inspections, flag change times, and whether a ship has had a casualty in the last five years. We follow the data processing method used by Yan et al. (2020) and Yan et al. (2021b) for these features.

Because the regret loss is computed at the instance level, we need to instantiate

the original dataset by dividing the original dataset into same-sized instances. We note that due to the limited size of the dataset, if we divide the original dataset into instances with different sizes, we can obtain different numbers of instances. For example, for instances in sizes (2,10) and (4,15), we can obtain a maximum of 302 ($\lfloor 3026/10 \rfloor = 302$) and 201 ($\lfloor 3026/15 \rfloor = 201$) instances, respectively. To maintain identical numbers of training and test instances under different sizes in the following experiments, we generate more instances by adopting a bootstrap sampling method, which is a statistical procedure that resamples a single dataset with replacement to generate more simulated examples. Using this method, we generate 300 instances under each instance size and divide them into a training instance set (80%, 240 instances) and a test instance set (20%, 60 instances). The predictive model used in this study is an ANN with a hidden layer of 100 neurons and ReLU (short for rectified linear unit) as the activation function implemented by PyTorch. A tuple of the following three hyperparameters needs to be tuned for these models: learning rate, epoch, and batch size. We use a grid search with five-fold cross validation on the training set to tune these hyperparameters in each ML model. All of the ANN models are trained using ADMM (short for alternating direction method of multipliers), which is an algorithm that solves convex optimization problems by breaking them into smaller pieces, each of which is easier to handle (Kingma and Ba, 2015). The proposed models are constructed using the training instance set, and their performance is validated using the test instance set.

3.6.2 Comparison of the two-stage framework and the decision-focused learning framework

Recall that model M2 has higher solution efficiency than model M1, and thus the following experiments use model M2 as the target optimization problem. Following the parameter settings for model M2 in Section 3.3.3, we compare the performance of the two-stage framework and the decision-focused learning framework using different contrastive losses under four instance sizes, namely, (2,10), (4,15), (6,20), and (8,25). For the ANN models, we set the search range for the learning rate at $\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.3, 0.5, 0.7\}$, the search range for the epochs at $\{5, 10, 15, 20, 25\}$, and the search range for the batch size at $\{1, 2, 4, 8, 16, 32\}$.

We then use the best hyperparameters to construct the ML models with different loss functions and different instance sizes on the whole training instance set and compute the regret loss based on the decision and the MSE based on the prediction on the test instance set. We rerun the models under the same instance size 30 times with different groups of randomly generated parameters for M2 in the same ranges as those introduced in Section 3.3.3. We then calculate the average regret loss and the average MSE of each model. Furthermore, we design a vote mechanism to compare the regret loss and the MSE of models with different loss functions and under the same instance size of 30 groups of experiments. For the five experiments under the same instance size and using the same group of randomly generated parameters for M2 but with different loss functions, the experiment with the lowest regret loss or the lowest MSE obtains one point. If there is a tie among several experiments, all of the experiments with the lowest score

can obtain one point. The final results are shown in Table 3.4.

Table 3.4: The computational results of 30 experiments using models with different loss functions under different instance sizes

Instance size	Method	Loss function	Metric		Vote	
			Average regret	Average MSE	Lowest regret	Lowest MSE
(2,10)	Decision-focused learning	NCE_basic	5.224	20.400	1/30	0/30
		NCE_variant	4.958	23.668	0/30	0/30
		MAP_basic	5.879	35.668	2/30	0/30
		MAP_variant	**3.447**	53.228	**23/30**	0/30
	Two-stage	MSE	3.814	**13.370**	4/30	**30/30**
(4,15)	Decision-focused learning	NCE_basic	6.79	23.38	2/30	0/30
		NCE_variant	6.01	27.27	1/30	0/30
		MAP_basic	5.63	34.52	0/30	0/30
		MAP_variant	**2.88**	77.97	**23/30**	0/30
	Two-stage	MSE	3.15	**13.56**	6/30	**30/30**
(6,20)	Decision-focused learning	NCE_basic	5.84	16.98	5/30	0/30
		NCE_variant	3.86	25.32	8/30	0/30
		MAP_basic	7.22	28.24	0/30	0/30
		MAP_variant	2.80	26.09	12/30	0/30
	Two-stage	MSE	**2.15**	**13.21**	**15/30**	**30/30**
(8,25)	Decision-focused learning	NCE_basic	5.71	25.42	9/30	0/30
		NCE_variant	4.70	30.39	10/30	0/30
		MAP_basic	12.50	61.12	2/30	0/30
		MAP_variant	1.82	27.10	**14/30**	0/30
	Two-stage	MSE	**1.25**	**15.83**	8/30	**30/30**

From the above results, we draw the following findings, which is consistent with recent studies (Hu et al., 2022).

A good prediction may not lead to a good decision. A better prediction is indicated by a lower MSE. This metric shows that the two-stage framework significantly outperforms the decision-focused learning framework with respect to prediction performance, as the two-stage framework can always obtain the lowest MSE loss under each of the four instances. However, when we compare the regret loss, which indicates the decision error, the decision-focused learning framework

is superior. The ratios of the total votes of the decision-focused learning framework to the total votes of two-stage framework under these four instance sizes are 26:4, 26:6, 25:15, and 35:8. Among the four types of contrastive losses in the decision-focused learning framework, except for (6,20), the MAP-variant loss obtains the highest votes under the other three instance sizes, thus validating its superiority. This result is explained by the format of the MAP-variant loss shown in Formula (3.31). For example, compared with the NCE-variant loss, the MAP-variant loss does not consider unimportant noise samples when computing the loss, thus helping the ML algorithm focus on the tightest inner approximation. In addition, unlike the MAP-basic loss, the MAP-variant loss tries to keep its predictions close to the real values.

The quality of the decision-focused learning framework may depend on the size of the optimization problem and on the size of the training dataset. Another obvious tendency shown in Table 3.4 is that the superiority of the MAP-variant loss declines as the instance size increases. This is indicated by the decreasing number of least-regret votes for the MAP-variant loss function when the instance size increases from (2,10) and (4,15) to (6,20) and (8,25). Furthermore, the average regret values of the two-stage framework under the instance sizes (6,20) and (8,25) are lower than those of the decision-focused learning framework. This result indicates that the decision-focused learning framework may not always perform better than the traditional two-stage framework. This finding was also obtained by Hu et al. (2022), and further explanations are provided below for the optimization problem and the dataset that we use in this study.

When the instance size increases, it is much more difficult to obtain an accurate convex hull of the solution pool. It is easy to imagine that if the solution pool

contains the whole set of feasible solutions, the size of the solution pool increases exponentially when the instance size increases. Assume that there are approximately 50 and 600 feasible undominated inspection templates to select under the instance sizes (2,10) and (8,25), respectively. Choosing two and eight inspection templates to constitute a feasible routing scheme generates $C_{50}^{2} = 1225$ and $C_{600}^{8} = 3.96 \times 10^{17}$ feasible outcomes, respectively. That is, the full size of the solution pool under the instance size (2,10) is only 1,225, but the full size of the solution pool under the instance size (8,25) reaches an extremely large number. Therefore, it becomes exponentially difficult to obtain an accurate and tight inner approximation under the instance size (8,25). To visually represent this point, we count the mean (represented by points) and standard deviation (represented by bands) of the number of noise samples added to the solution pool in each epoch for the 30 experiments using the MAP-variant loss function under different instance sizes, and the results are presented in Figure 3.1.

As shown in Figure 3.1, the solution pool under a larger instance size can take in more new noise samples generated during the training process. Furthermore, the structure of the solution pool under a larger instance size can be more complex and unstable, as shown by a larger standard deviation. For example, for instance sizes (2,10) and (4,15), the number of added noise samples quickly converges to 0 after epoch 2, indicating that the solution pool stops expanding. However, for instance sizes (6,20) and (8,25), although the number of added noise samples decreases as the training process proceeds, the size of the solution pool continues to grow until epoch 5, indicating that the current solution pool is not tight enough. Therefore, the size of the optimization problem plays a vital role in the prescribed decision quality of the decision-focused learning framework, which is mainly influenced

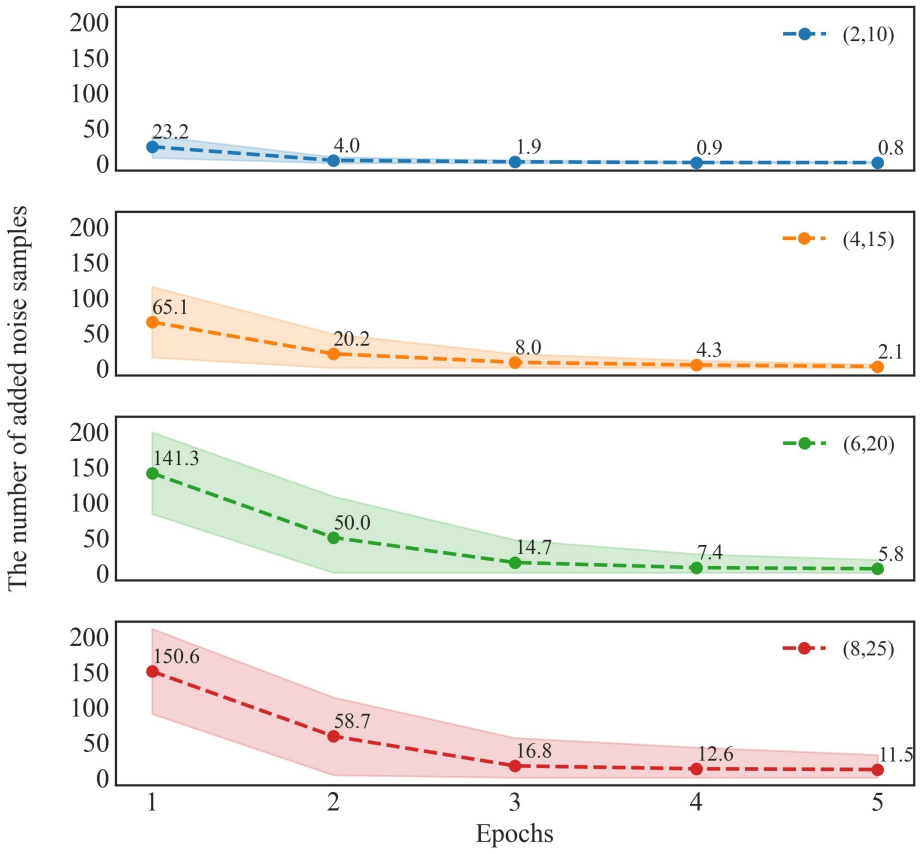

Figure 3.1: The number of added noise samples in each epoch for experiments using the MAP-variant loss under different instance sizes

by the size and structure of the solution pool.

Furthermore, due to the limited size of the dataset, we use bootstrap sampling to generate more records. This method may create a "same-ships-but-different-decisions" situation because it is difficult for the auxiliary parameters (i.e., berthing locations, berthing time windows) of two "same" ships (with the same feature values and the same number of deficiencies) to be the same, which can lead to different decisions. This does not influence the training process of the two-stage framework, but introduces difficulty and noise to the training process in the decision-focused learning framework. When the instance size increases, a ship is more likely to be resampled, and the prescribed quality of the decision-focused learning framework is adversely affected.

3.6.3 The influence of p_{solve}

In Algorithm 3.5, parameter p_{solve} is used to control the frequency of calling the solver to solve model M2 to expand the solution pool. A larger p_{solve} represents a greater possibility of calling the solver, increasing the possibility of adding a new solution to the solution pool. However, calling a solver does not necessarily result in new information that can tighten the convex hull of the inner approximation, as the newly derived solution may have existed in the solution pool; however, this approach definitely increases the training time of Algorithm 3.5. To investigate the influence of p_{solve} on the trade-off between efficiency and accuracy, we change the value of p_{solve} from 0.2 to 1 with a step size of 0.2, and train the models using the MAP-variant loss under different instance sizes with each value of p_{solve}. We rerun models 30 times with different groups of randomly generated parameters

for M2. Then, we obtain the average regret and average training time of the 30 experiments for each model. The final results are shown in Figures 3.2 and 3.3.

Figure 3.2 shows that the average regret may have a downward trend when p_{solve} increases, especially under instance sizes (6,20) and (8,25). However, it does not show an obvious linear relationship, indicating that not all of the solutions obtained by calling the solver are new to the solution pool. Furthermore, Figure 3.2 shows that the downward trend under instance sizes (6,20) and (8,25) is more significant than under instance sizes (2,10) and (4,15). As explained in Section 3.6.2, this result is due to the difficulty of finding a good inner approximation for large-sized optimization problems. In contrast, under instance sizes (2,10) and (4,15), the initial solution pool already functions well as a good inner approximation. Accordingly, increasing the value of p_{solve} does not have a significant effect on decreasing regret for models under these instance sizes.

Figure 3.3 shows that the average training time has a strict upward trend when p_{solve} increases. As the model solving time shown in Section 3.3.3 does not exhibit a linear relationship as the instance size increases, the slope of the fitted line connecting the scatters becomes steeper as the instance size increases. Therefore, these results indicate that for models under instance sizes (2,10) and (4,15), calling the solver to add new solutions to the solution pool by sampling seems to result in little improvement in decision quality, as the initial solution can function well as a good inner approximation. Therefore, to reduce the training time of ML models, we recommend setting a small value for under instance sizes (2,10) and (4,15). However, for models under instance sizes (6,20) and (8,25), because the initial solution pool may not function as a good inner approximation, we recommend setting a moderate value for p_{solve} between 0.4 and 0.8, thus achieving a balance

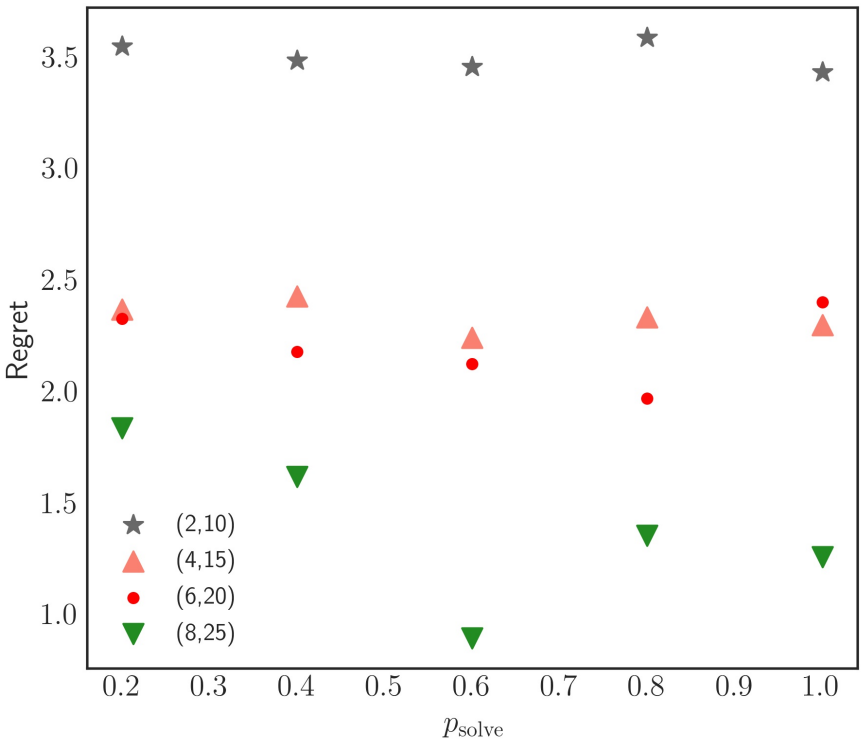

Figure 3.2: The average regret of the 30 experiments using models adopting the MAP-variant loss with different p_{solve} and different instance sizes

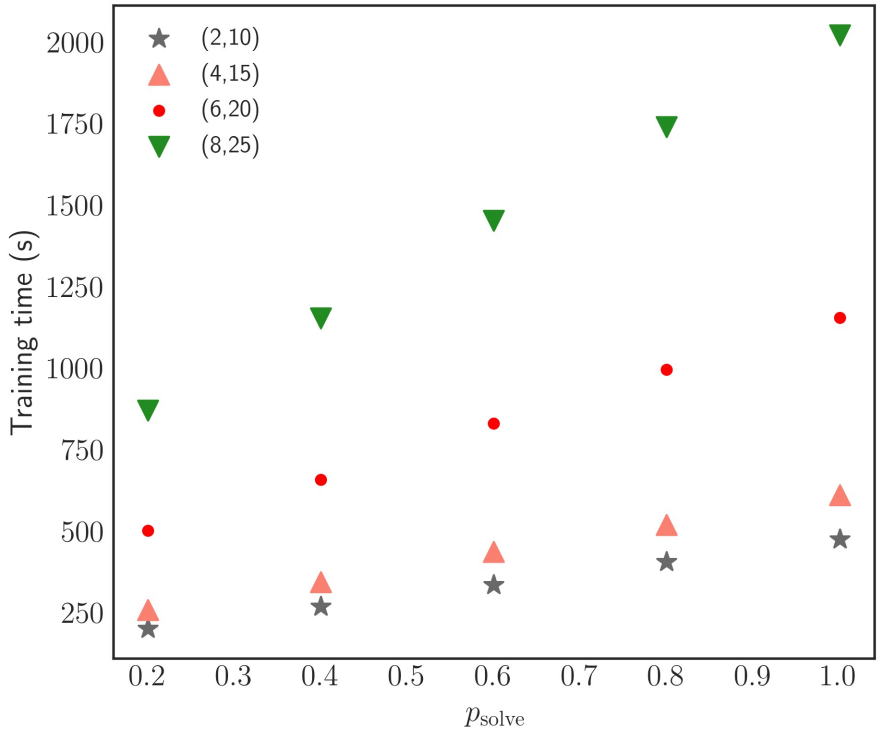

Figure 3.3: The average training time of the 30 experiments using models adopting the MAP-variant loss with different p_{solve} and different instance sizes

between efficiency and accuracy.

3.7 Conclusions

With the development of ML technologies and the availability of PSC data, this study investigates the PSCO routing problem with the aim of maximizing the number of deficiencies that can be identified from inspected ships considering practical constraints. Because ship condition is not known to port authorities when they route PSCOs, the traditional solution to this problem involves a two-stage framework that first predicts the number of deficiencies of each foreign visiting ship and then uses that prediction to solve the PSCO routing problem. However, the loss function used in this framework does not consider the issue of the decision error. Therefore, we adopt a decision-focused learning framework to solve the PSCO routing problem by plugging the optimization problem directly into the training process of the ML model. Under this framework, the PSCO routing problem must be solved tens of thousands of times. Given that the original PSCO routing problem is NP-hard, computational complexity and scalability are two major obstacles to putting this decision-focused learning framework into practice. To overcome these two issues, we first transform the original PSCO routing problem to be more compact by designing undominated inspection templates and then use a family of surrogate loss functions based on NCE. Our computational experiments result in several interesting findings. First, a good prediction may not lead to a good decision, which is seen from the fact that under some instance sizes, the decision-learning framework is superior to the two-stage framework with respect to decision quality. Second, the quality of the decision-focused learning framework may de-

pend on the size of the optimization problem and on the size of the dataset. This quality decreases as the instance size increases, because it is difficult for ML models to learn the structural properties of large-scale optimization problems. Third, the adopted decision-focused learning framework with a solution pool containing noise samples can guarantee a balance between training efficiency and decision quality; thus, it does not require frequent reoptimizations during the training process.

Chapter 4

Prescriptive Analytics for Container Ship Bunkering Optimization

4.1 Introduction

Liner shipping, a crucial component of global trade, transports containerized goods from origin to destination ports in accordance with a fixed-schedule shipping service (Meng et al., 2014; Wang et al., 2018; Wang and Meng, 2021; Zhang et al, 2022). To sustain these services, container ships need to procure and replenish bunker fuel at some ports of call. As fuel costs account for 20% to 61% of ship operating costs, shipping companies persistently strive to reduce their fleet's fuel costs (Wang et al., 2015; Meng et al., 2016, 2017). According to data from the Shipping Intelligence Network,[1] the average global price of high-sulfur fuel oil (HSFO) was $452 per metric tonne (MT) from 2012 to 2017. In the same period, the "Fourth Greenhouse Gas Study 2020" of the International Maritime Organi-

[1] See `https://sin.clarksons.net/`, last accessed date: April 25, 2023.

zation (IMO) reports that the annual usage of HSFO in international shipping was 174 million MTs.[2] Multiplying these two figures, we arrive at an average annual expenditure of $78.65 billion on HSFO for maritime transportation. This underscores the substantial influence of fuel costs on shipping operations management. Moreover, with the goal of reducing shipping emissions, the IMO has mandated that from 2020, the global sulfur content in marine fuel cannot exceed 0.5%, requiring ships to use very low-sulfur fuel oil (VLSFO). Historical data indicate that the average price of VLSFO is approximately 40% higher than that of traditional HSFO.[3] Thus, ships' fuel costs increase significantly when they use VLSFO. Consequently, the primary operational goal of shipping companies in the green shipping era is cost reduction while adhering to emission reduction requirements. To reduce fuel costs, shipping companies need to monitor fuel price changes in real time and adjust ship bunkering strategies, i.e., where to bunker and how much to bunker, accordingly. However, considerable uncertainties influencing fuel prices create substantial challenges in ship bunkering management for shipping companies.

The price of fuel is an external cost factor faced by every shipping company. Fluctuations in fuel prices are influenced by multiple factors, such as crude oil prices, political stability, and weather, and may lead to distinct variations in ship operating costs (Wang et al., 2018). Fuel prices vary significantly across different ports worldwide, with such variances possibly depending on geographical proximity. For instance, Figure 4.1 shows variations in VLSFO prices at the ports of Rotterdam, Singapore, Shanghai, and Hong Kong between November 22, 2019

[2]See https://maritimecyprus.com/wp-content/uploads/2021/03/4th-IMO-GHG-Study-2020.pdf, last accessed date: April 30, 2023.

[3]See https://shipandbunker.com/prices.

and February 10, 2023. Due to the long distance between the port of Rotterdam and the other three ports (Singapore, Shanghai, and Hong Kong), the fuel price difference (or inter-port fuel price variance) between the port of Rotterdam and the other three ports is large. Historically, the fuel prices at the port of Rotterdam have been the lowest among the four ports. In contrast, the ports of Shanghai, Singapore, and Hong Kong are geographically close, resulting in smaller price differences and more significant *ranking* fluctuations in their historical fuel prices. Consequently, for a container ship following the route Rotterdam → Singapore → Shanghai → Hong Kong → Rotterdam, fueling up in Rotterdam could generally reduce total fuel costs. However, for a ship following the route Singapore → Shanghai → Hong Kong → Singapore, the bunkering decision becomes more challenging due to a more volatile ranking of the fuel prices of the three ports led by the smaller inter-port fuel price variance.

Furthermore, over the past two years or more, VLSFO prices at specific ports have fluctuated dramatically between $200/MT and $1,200/MT, indicating a high level of uncertainty. While fuel prices at different ports generally follow the same trend of ups and downs, and can be considered correlated random variables, operational ship bunkering decisions do not rely on a *contemporaneous* comparison of fuel prices at specific ports of call. Instead, they involve anticipating the prices at each port when the ship is scheduled to dock, complicating the bunkering decision-making process. Therefore, it is of great practical importance to study how to formulate the optimal ship bunkering strategy under highly uncertain future conditions.

To model and solve optimization problems involving uncertainty, various frameworks have been developed. In general, traditional methods, such as stochastic

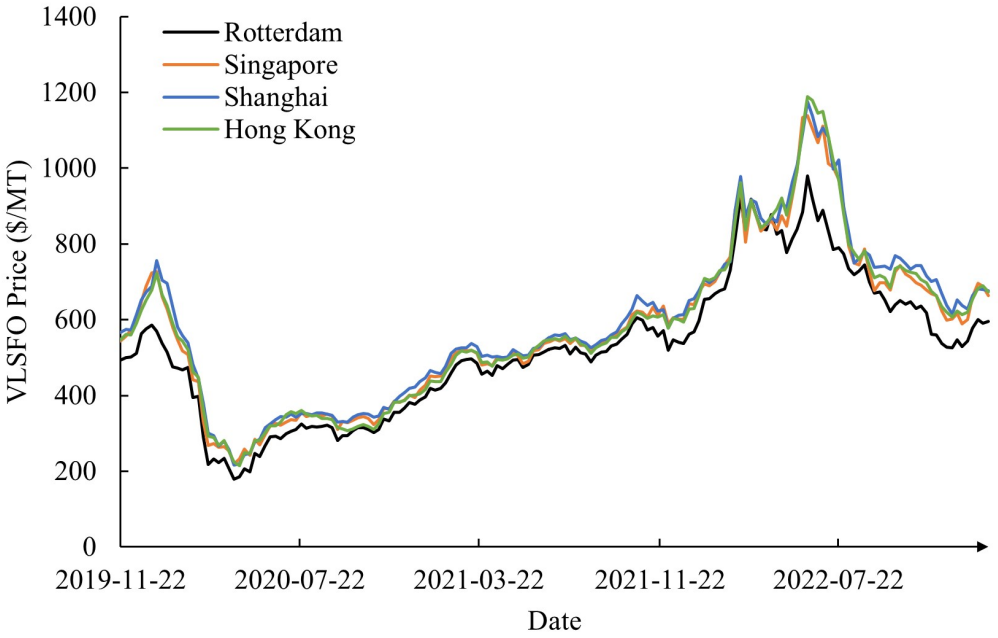

Figure 4.1: VLSFO prices of four ports

programming (Birge and Louveaux, 2011) and robust programming (Ben et al., 2009; Bertsimas et al., 2011), predefine information on distributions for uncertain parameters. However, it is difficult for decision makers to know the ground-truth distributions of uncertain parameters (Qi and Shen, 2022). In light of this limitation, alternative frameworks that take data as a primitive could offer more accurate and actionable solutions for optimization problems with uncertainty. Specifically, due to the development of Internet technologies and various machine learning (ML) techniques, prescriptive analytics frameworks that integrate ML and optimization have emerged (Bertsimas and Kallus, 2020; Bertsimas and Koduri, 2021; Tian et al., 2023a). These frameworks can provide a more realistic approach to modeling uncertainty by predicting uncertain parameters using diverse data sources rather than making assumptions about the underlying distributions. As a result, decision makers can benefit from optimization strategies that are more

reliable and effective in the presence of uncertainty than earlier strategies, ultimately leading to improved solution performance. Thus, this study applies prescriptive analytics frameworks to explore the value of real ship fuel price data for ship bunkering management.

The remainder of this chapter is organized as follows. Section 4.2 reviews related literature and summarizes our contributions. Section 4.3 mathematically formulates the ship bunkering management problem in liner shipping. Section 4.4 presents two prescriptive analytics optimization models for the problem. Section 4.5 describes the predictive models, including the two-channel long short-term memory (TC-LSTM) for predicting multi-port fuel prices and the MC dropout technique for estimating distributions in deep learning. Section 4.6 uses both real-world and synthetic data to compare the prediction performance of the TC-LSTM and the traditional LSTM, as well as the decision performance of the two prescriptive analytics frameworks. Section 4.7 concludes the study.

4.2 Literature Review

In this section, we introduce the literature on ship bunkering management, ship fuel price prediction, and prescriptive analytics for business analytics, and discuss the corresponding research gaps.

Ship bunkering management. Ship bunkering management has been widely studied in the literature. As our study primarily investigates the impact of uncertain fuel prices on ship bunkering strategies, we categorize the literature into two streams based on whether the studies have considered uncertain fuel prices. Yao et al. (2012) highlight bunkering port selection and bunkering amount deter-

mination as two key components of ship bunkering management and propose a planning-level model to determine the optimal bunkering strategy. Wang et al. (2019) build upon this work by proposing a detailed model and using linearization techniques to handle nonlinear terms. To address the issue that the actual sailing speed may deviate from the planned speed, Wang and Meng (2015) propose a robust optimization model that optimizes the sailing speed and bunkering strategy of container ships under the worst-case fuel consumption scenarios. De et al. (2020) study sustainable ship routing and bunkering management, whereas De et al. (2021b) analyze ship bunkering strategies that address environmental concerns related to fuel consumption and carbon emissions. Notably, none of the above studies has considered the impact of uncertain ship fuel prices.

The second stream of the literature utilizes stochastic programming approaches to investigate the impact of uncertain fuel prices on ship bunkering management. For example, Wang and Teo (2013) construct a scenario decision tree to account for the uncertainty of ship fuel prices and study integrated hedging and network planning for liner shipping's bunkering management. By using a scenario tree to represent stochastic fuel prices, Sheng et al. (2014) investigate dynamic vessel speed optimization and bunkering port selection for liner shipping. Meng et al. (2015) analyze a liner shipping company's ability to hedge against the uncertainty of fuel prices by purchasing fuel from both the contract and spot markets. They develop and solve a mean-variance minimization stochastic programming model using an approximation method that integrates random variable sampling techniques, scenario tree generation, and quadratic programming approximation. Ghosh et al. (2015) and Sheng et al. (2015) propose dynamic programming models to optimize ship bunkering management with service contracts and speed optimization,

respectively. These studies use Monte Carlo (MC) simulation and sample average approximation (SAA) methods to solve the problems. Zhen et al. (2017) propose a dynamic programming model for optimal ship bunkering policies, assuming that uncertain fuel prices at each port follow known distributions. Wang et al. (2018) jointly optimize the sailing speeds on shipping voyages and the bunkering strategy at each port of call in the presence of correlations among fuel prices at these ports. Gu et al. (2019) develop a stochastic programming model for tactical and operational decisions in ship bunkering management, including routing and speed optimization, subject to the latest regulations for reducing shipping emissions. Their model represents the uncertainty of fuel prices using scenarios. De et al. (2021a) study a practical problem with speed optimization and ship bunkering management under different fuel price scenarios, and propose an approximation algorithm.

As highlighted in the current body of research on ship bunkering management, the uncertainty of fuel prices has been a focal point, with many studies employing stochastic programming methods. However, these approaches have notable limitations. First, a common practice in existing studies is to make assumptions about the distribution of fuel prices. In reality, the exact distribution of fuel prices is often unknown and can be challenging to predict accurately. Assumed distributions might not truly capture the real-world dynamics of fuel price changes, potentially leading to decision-making strategies that are less effective in practical scenarios. The discrepancy between assumed and actual fuel price distributions can result in significant deviations in optimal decision making. Second, much of the existing literature on ship bunkering decisions is framed at a static level, largely disconnected from the decision time. This approach tends to overlook the dynamic nature of fuel price fluctuations across multiple decision periods, which is a pivotal

aspect of operational-level decision making in bunkering. Unlike static decisions, operational-level decisions need to account for not only the current state of fuel prices but also their potential changes over time. This temporal dimension is vital for accurately capturing the volatility of fuel prices and is indispensable for optimizing bunkering decisions in the operational context under uncertain conditions.

Ship fuel price prediction. Stefanakos and Schinas (2014) conduct a ship fuel price forecast analysis using weekly HSFO 380 centistoke (CST) prices from major fuel supply ports, such as Rotterdam, Fujairah, Singapore, and Houston. They use a vector autoregressive moving average model and find that the error in medium-term predictions of the next 52 weeks was within 20%. Stefanakos and Schinas (2015) use a fuzzy time series model to analyze the prices of multiple types of fuel in the same ports, including HSFO 180 CST, marine diesel oil, and marine gas oil. Choi (2017) uses system dynamics to analyze HSFO 380 CST prices based on the annual fuel data of the port of Singapore. They identify the variables that affect fuel prices, including crude oil production and consumption, the West Texas Intermediate crude oil price, global gross domestic product, exchange rates, cargo demand, vessel supply, demand/supply ratio, and freight rates. Recently, deep learning methods have become increasingly popular for time series prediction. Kim et al. (2022) show that liquefied natural gas (LNG) prices are influenced by various factors, and that recurrent neural network (RNN) models, such as the LSTM model, can forecast short-term LNG bunker prices and effectively manage ship operating costs. Furthermore, as marine fuel is a byproduct of the refining process of crude oil, the main factor affecting fuel oil prices is the price of crude oil. Studies that have focused on predicting crude oil prices include Orojo et al. (2019), Gupta and Nigam (2020), and Chiroma et al. (2015).

Research has focused on predicting crude oil or ship fuel prices using weekly prices and employing various prediction methods. However, studies have usually predicted fuel prices for each port *separately*, without considering the spatiotemporal correlation between fuel prices at multiple ports or the impact of the prediction target on bunkering operations. Given that the main objective of predicting ship fuel prices is to support ship bunkering operations, it is crucial to predict and compare oil prices at multiple ports at different future time points, according to the ship's schedule. Although current models can account for the temporal dependencies of time series, predicting fuel prices for individual ports separately fails to consider the *spatial* dependencies of fuel prices among different ports. This limitation necessitates a more comprehensive approach to addressing both temporal and spatial correlations in fuel prices across multiple ports, thereby providing more accurate and actionable insights than current ship bunkering management studies.

Prescriptive analytics for business analytics. Driven by the business problem, the workflow of business analytics involves collecting, preprocessing, and interpreting data, selecting and refining predictive analytics methods, and modeling for decision making in prescriptive analytics. Numerous business analytics scenarios involve the application of big data techniques (Yang et al., 2017; Adler et al., 2022). The goal of business analytics is to make informed decisions, with a focus on prescriptive analytics to provide recommendations for action (Bertsimas and Kallus, 2020). In prescriptive analytics, the predict-then-optimize (PTO) framework is widely used. This approach involves generating point predictions of uncertain parameters and using them to make decisions through downstream optimization problems. However, point predictions may not be suitable for op-

timization problems with nonlinearity (Qi and Shen, 2022; Tian et al., 2023b). To address this issue, it may be preferable to estimate the marginal distributions of uncertain parameters and solve the resulting contextual stochastic optimization problem (Muñoz et al., 2022); this approach is termed the estimate-then-optimize (ETO) framework (Qi and Shen, 2022). One of its strengths lies in its ability to reveal all information necessary for solving the stochastic problem when a perfect estimate of the underlying conditional distribution can be obtained (Qi and Shen, 2022).

In terms of methods for estimating the conditional distribution, Bertsimas and Kallus (2020) introduced nonparametric ML models, such as k-nearest neighbors and decision tree methods, to derive weights from the features, and then optimize the decision based on a re-weighting of the data. This approach has been termed weighted SAA by Notz and Pibernik (2022). However, local learning methods such as kNN and decision trees may discard data that are not closely related to the observation, requiring a sufficient amount of data to ensure accurate results (Bertsimas and Koduri, 2021). As an alternative, Wang and Yan (2022) propose a global method based on quantile regression to estimate the marginal distribution of a univariate parameter, by taking all data into account. Despite these developments, the literature on prescriptive analytics discussing how to predict the distribution of time series data using deep learning methods is limited. Our study fills this gap. Detailed reviews of prescriptive analytics can be found in Qi and Shen (2022) and Tian et al. (2023b).

The main contributions of this study are threefold. First, the study contributes to the literature on ship bunkering management. We harness real-world data for predicting uncertain fuel prices, diverging from the common practice in existing

literature which relies heavily on stochastic programming methods and necessitates assumptions about fuel price distributions. Our approach utilizes ML models *without* presupposing any specific price distribution. We introduce two prescriptive analytics frameworks: the **T**wo-stage contextual **D**eterministic framework with **P**oint predictions (TDP) and **M**ultistage contextual **S**tochastic framework with **D**istributional estimates (MSD). These frameworks, tested with both real-world and synthetic data, reveal the impact of inter-port fuel price variances and the number of ports on their decision performance. For both frameworks, our models uniquely capture the dynamic nature of fuel prices over time, connecting each bunkering decision to its specific decision time, which is the moment of the ship's arrival at each port. To consider dynamic and operational decisions, we predict fuel prices for downstream ports at *different* future time periods, integrating temporal aspects into each decision.

Second, the study contributes to the literature on ship fuel price prediction. We develop a TC-LSTM model for predicting multi-port fuel prices, considering *intra*-dependencies within individual ports and *inter*-dependencies across multiple ports. This model outperforms the traditional LSTM model in prediction accuracy. This study is pioneering in the realm of ship fuel price prediction by simultaneously accounting for the temporal and spatial correlations in fuel prices across multiple ports, leading to more precise predictions for shipping companies.

Finally, the study contributes to the literature on prescriptive analytics. The TDP and MSD frameworks contribute to the literature by relating to PTO and ETO frameworks, respectively. Notably, we apply the MC dropout technique in the TC-LSTM model within the MSD framework to derive distributional estimates. This research is the first in the prescriptive analytics field to utilize deep learning mod-

els for estimating the conditional distribution of correlated uncertain parameters, indicating a new direction for future studies related to the ETO framework using deep learning. Additionally, this study is the first to apply prescriptive analytics frameworks in the context of ship bunkering management, enhancing the practicality and relevance of these frameworks in real-world scenarios.

4.3 Problem Description

Consider a liner shipping service with a set of ports of call $\mathcal{N} = \{1, ..., N\}$, each indexed by i. The service route forms a loop, represented as $1 \rightarrow 2 \rightarrow ... \rightarrow N \rightarrow 1'$, where $1'$ signifies the return to port of call 1 in the subsequent round trip. This notation aids in distinguishing between consecutive visits to the same port. We define a shipping leg as the voyage between two adjacent ports, uniquely identified by the starting port index $i \in \mathcal{N}$. Given this one-to-one correspondence between ports and shipping legs, the same indexing system is applied to both. For each time period $t = 1, 2, ...$, the vector of fuel prices at all ports of call is $\mathbf{x}_t := (x_t^1, ..., x_t^N)$, with x_t^i representing the fuel price at port i during period t. It is assumed that a container ship, upon arrival at port i during period t^i (where $t^i \geq 1$), has a *known* fuel inventory R^i. The historical fuel prices at all N ports from period 1 to t^i are recorded as $\mathbf{X}_{t^i} := [\mathbf{x}_1, ..., \mathbf{x}_{t^i}] \in \mathbb{R}^{N \times t^i}$. Lastly, the maximum fuel capacity of a container ship is denoted by V.

In addressing the problem at hand, we establish the following assumptions. First, given the fixed shipping schedules of liner shipping services (Song and Dong, 2013; Bell et al., 2013; Meng et al., 2014) and our focus on the impact of fluctuating fuel prices, we exclude extreme uncertainties like unpredictable sail-

ing conditions and port congestion. Consequently, the sum of travel time for each shipping leg i and the dwell time at port $i-1$ is assumed deterministic, denoted by τ^i, where $\tau^i = t^i - t^{i-1}$. Second, while acknowledging that ships use diesel oil for auxiliary power, our analysis concentrates solely on the fuel consumption of main engines, a significant component of bunker costs. Third, following the approach in Zhen et al. (2017), we treat the fuel consumption for each shipping leg i, denoted by G^i, as a deterministic value. This assumption allows us to concentrate on the implications of uncertain fuel prices on operational bunkering decisions. Fourth, for a given liner shipping service, the initial fuel inventory R^1 is a known parameter, with a requirement that $R^1 = R^{1'}$ to ensure service sustainability. However, for more extended time horizons or complex routes, this assumption can be relaxed to only necessitate that the container ship reaches the final port of call, without compromising the applicability of subsequent models and methods. Fifth, we assume the maximum tank capacity of a container ship V to exceed the fuel consumption for each shipping leg G^i.

In the context of a specific liner shipping service, upon the container ship's arrival at port of call i, where $i < N$, the ship operator can ascertain the current port's fuel price. However, the future fuel prices at downstream ports of call j ($j \in \{i+1, ..., N\}$), corresponding to the periods t^j when the ship is scheduled to visit these ports, remain uncertain. These unknown prices are represented by $\tilde{x}^j_{t^j}$ ($j \in \{i+1, ..., N\}$). We denote the vector of these *uncertain* future fuel prices as $\tilde{\mathbf{x}}^N_{i+1} := (\tilde{x}^{i+1}_{t^{i+1}}, ..., \tilde{x}^N_{t^N})$. The uncertain nature of future fuel prices significantly influences the bunkering decisions at the current port. The central aim of this study is to develop a strategy that minimizes the total bunkering costs for the entire voyage, taking into account the service requirements and the variability in fuel

prices. The decision variable in this scenario is $z^i(\mathbf{X}_{t^i}, R^i)$, abbreviated as z^i, which signifies the amount of fuel to be purchased at port i. This decision is based on the observed fuel prices $\mathbf{X}_{t^i}$ and the existing fuel inventory R^i.

4.4 Prescriptive Analytics Frameworks

This section delineates two prescriptive analytics models devised to address the challenge of managing ship bunkering in the context of uncertain future fuel prices. In Section 4.4.1, we explore the implementation of two-stage contextual deterministic models. These models use point predictions of future fuel prices. Subsequently, Section 4.4.2 delves into multistage contextual stochastic optimization models. Unlike the TDP framework, these models employ distributional estimates of future fuel prices. Each framework offers a unique lens through which the ship bunkering management problem can be analyzed and solved, reflecting the different features of their prescriptive strategies.

4.4.1 The TDP framework

The TDP framework, as detailed in this section, leverages historical fuel price data to forecast future prices at downstream ports of call. This forecasting is accomplished through point predictions, with the methodology elaborated in Section 4.5.2. Concretely, upon the arrival of a container ship at port i $(i < N)$,[4] and equipped with the historical fuel prices $\mathbf{X}_{t^i}$, the framework approximates the un-

[4]Upon the ship's arrival at the final port of call (i.e., when $i = N$), the requirement to ensure a return to the first port of call simplifies to verifying if the remaining fuel inventory suffices to meet the fuel consumption needs of the last shipping leg. This verification bypasses the need for employing optimization techniques typically used at other ports of call.

certain future fuel prices for downstream ports, $\tilde{\mathbf{x}}_{i+1}^{N} := (\tilde{x}_{t^{i+1}}^{i+1}, ..., \tilde{x}_{t^N}^{N})$, via point predictions. These predictions are generated using ML models, resulting in the estimated fuel price vector $\hat{\mathbf{x}}_{i+1}^{N} := (\hat{x}_{t^{i+1}|t^i}^{i+1}, ..., \hat{x}_{t^N|t^i}^{N})$.

Incorporating point predictions is fundamental to the two-stage contextual deterministic framework. As detailed in Adulyasak et al. (2015), within this framework, the fuel prices for the upcoming segments of the journey at downstream ports of call are effectively rendered "known" (via predictions) subsequent to the finalization of the bunkering decision at the current stage. This methodology effectively converts a potentially stochastic challenge into a deterministic one at each stage of the journey, thereby streamlining the decision-making process. By applying these point predictions of future fuel prices, the two-stage contextual deterministic model can be formulated for each port of call i $(i < N)$ as follows:

[**Model-TDP**]

$$\min_{\mathbf{z},\mathbf{R}_{i+1}^{N}} z^i x_{t^i}^i + \sum_{j=i+1}^{N} z^j \hat{x}_{t^j|t^i}^j \tag{4.1}$$

subject to

$$z^l \geq G^l - R^l \quad \forall l \in \{i, ..., N\} \tag{4.2}$$

$$z^l \leq V - R^l \quad \forall l \in \{i, ..., N\} \tag{4.3}$$

$$R^{l+1} = R^l + z^l - G^l \quad \forall l \in \{i, ..., N-1\} \tag{4.4}$$

$$R^{1'} = R^N + z^N - G^N \tag{4.5}$$

$$R^{1'} = R^1 \tag{4.6}$$

$$z^l \geq 0 \quad \forall l \in \{i, ..., N\} \tag{4.7}$$

$$R^l \geq 0 \quad \forall l \in \{i+1, ..., N\}. \tag{4.8}$$

In our model, we define $\mathbf{z} := (z^i, \mathbf{z}_{i+1}^N)$, where $\mathbf{z}_{i+1}^N := (z^{i+1}, ..., z^N)$ and $\mathbf{R}_{i+1}^N := (R^{i+1}, ..., R^N)$. It is crucial to note that for downstream ports j $(j = i+1, ..., N)$, z^j and R^j, as components of $\mathbf{z}_{i+1}^N$ and $\mathbf{R}_{i+1}^N$ respectively, are considered *temporary dummy* decision variables. This is because the container ship has not yet reached these ports, and consequently, the bunkering decisions and the fuel inventory levels for these ports remain undetermined. Objective function (4.1) aims to minimize the total fuel costs for the remaining voyage. This calculation incorporates the observed fuel price $x_{t^i}^i$ at the current port and the predicted fuel prices $\tilde{x}_{t^j}^j$ for downstream ports $j = i+1, ..., N$. Constraints (4.2) and (4.3) ensure that the bunkering amount at each port of call for the remainder of the voyage adheres to service requirements and does not exceed the tank's capacity limit. Constraints (4.4) and (4.5) describe the state transition relationships of the fuel inventory between adjacent ports, reflecting the consumption and replenishment of fuel. Constraint (4.6) is pivotal in guaranteeing the sustainability of services across voyages.

It is important to emphasize that for each optimization model evaluated at a given port of call $i > 1$, the remaining fuel amount R^i must be calculated in advance. This computation is based on the remaining fuel amount at the previous port R^{i-1}, the amount of fuel replenished at that port z^{i-1}, and the fuel consumption G^{i-1} for the preceding shipping leg $i-1$. These factors are considered known parameters, aligning with practical considerations in liner shipping operations. The temporary dummy decision variables for downstream ports z^j $(j = i+1, ..., N)$, when the container ship is at port i, should be iteratively updated as the shipping service progresses. The ultimate decision regarding the amount of fuel to bunker at

each port will be based on the actual observed fuel prices upon the ship's arrival at those ports. Furthermore, enhancing the decision-making quality necessitates the re-prediction of future fuel prices for downstream ports (i.e., ports $i+2, ..., N$) when the ship arrives at port $i+1$. This re-prediction should utilize updated datasets, reflecting the continuous updating of fuel price data during the voyage. Consequently, the two-stage contextual deterministic model must be re-solved at each port, considering both the observed fuel price at the current port and the newly predicted prices for downstream ports. The total fuel costs for the entire round trip are as follows:

$$\sum_{i=1}^{N} x_{t^i}^{i} z^{i*}, \tag{4.9}$$

where z^{i*} denotes the final determined bunkering amount when the container ship visits port i during period t^i. This cumulative cost represents the sum of the product of the fuel price and the decided bunkering amount at each port, capturing the essence of optimizing fuel costs across the voyage.

4.4.2 The MSD framework

In the previously discussed Model-TDP, we employed point predictions to estimate uncertain fuel prices at downstream ports. This approach, while easy-to-implement, overlooks the inherent uncertainty associated with these predictions. Notably, the reliance on the two-stage deterministic framework presumes that fuel prices at downstream ports for future calling periods are *simultaneously* realized upon making the current decision. This assumption often diverges from real-world scenarios. In practical terms, the ship bunkering management problem is inherently a multistage decision-making process. The fuel price at each port of call

is only revealed subsequent to the decision making at the preceding stage. This sequential unfolding of information implies that even with "perfect" point predictions of fuel prices, deriving optimal solutions can be challenging within a multistage context. To demonstrate this, we provide an intuitive example below.

Example 4.1 *Consider a shipping route with four ports, as depicted in Figure 4.2. The container ship on this route has a maximum fuel capacity of 200 MTs. Assume that ports 1, 2, and 3 are in close proximity, resulting in no fuel consumption for leg 1 (port 1 to port 2) and leg 2 (port 2 to port 3). At port 1, the ship's fuel inventory is empty (0 MTs), and the observed fuel price is \$400/MT. To reach port 4, the ship must bunker at least 100 MTs of fuel at one of the first three ports. We consider two equally probable scenarios for future fuel prices at ports 2 and 3: Scenario 1,* $\Pr\left((\tilde{x}^2_{t^2}, \tilde{x}^3_{t^3}) = (300, 800)\right) = 1/2$*, and Scenario 2,* $\Pr\left((\tilde{x}^2_{t^2}, \tilde{x}^3_{t^3}) = (800, 300)\right) = 1/2$*. A "perfect" point prediction gives* $(\hat{x}^2_{t^2}, \hat{x}^3_{t^3}) = (550, 550)$*, equal to the expected values of the ground-truth distribution. Under Model-TDP, this prediction suggests that the optimal strategy is to bunker 100 MTs of fuel at port 1. However, when considering the ground-truth distribution and the multistage nature of decision making, the optimal approach would be not to refuel at port 1. Instead, if the real distribution is predicted, the ship should bunker at port 2 under Scenario 1, or at port 3 under Scenario 2, upon reaching port 2.*

The insights gleaned from Example 4.1 underscore the necessity of predicting the distribution of future fuel prices, prompting the development of multistage contextual stochastic optimization models. The methodology for these predictions is elaborated in Section 4.5.3. Given the historical fuel prices $\mathbf{X}_{t^i}$ for $i < N$, we

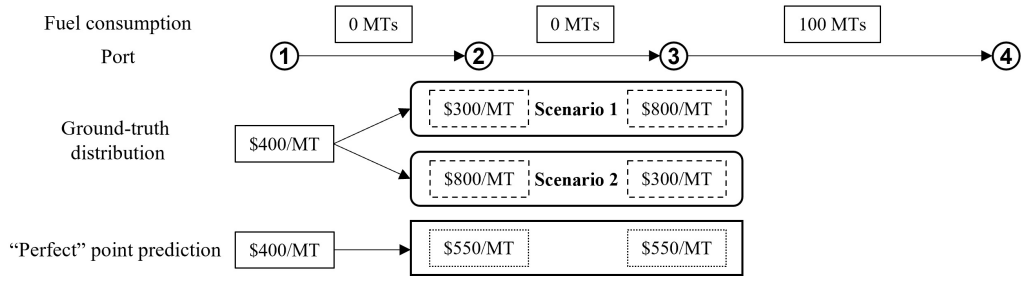

Figure 4.2: An illustrative example of the multistage ship bunkering problem

posit that the empirical marginal distribution of the uncertain future fuel prices $\tilde{\mathbf{x}}_{i+1}^{N} := \left(\tilde{x}_{t^{i+1}}^{i+1}, ..., \tilde{x}_{t^N}^{N}\right)$ can be represented by a set of scenarios, denoted as $\mathcal{S}$. For each scenario $s \in \mathcal{S}$, the vector of *estimated* fuel prices for downstream ports is expressed as $\hat{\mathbf{x}}_{i+1,s}^{N} := (\hat{x}_{t^{i+1}|t^i,s}^{i+1}, ..., \hat{x}_{t^N|t^i,s}^{N})$. Each scenario s is assigned an equal probability, $\lambda_s = 1/|\mathcal{S}|$. In line with this stochastic framework, dummy decision variables z^j and R^j for downstream ports $j = i+1, ..., N$ at port i are tailored to each scenario $s \in \mathcal{S}$. These variables are denoted as z_s^j and R_s^j, respectively, for each scenario. This approach allows for a nuanced consideration of different potential future states and their corresponding impacts on the decision-making process at each stage of the journey.

In the context of distributional estimates for fuel prices, it is important to note that while scenarios are independent, the (dummy) bunkering decisions for downstream ports across different scenarios are *interdependent*. This is due to the multistage nature of the problem, where the decision to bunker at a port is based solely on the information available up to that point, rather than on future developments. Consequently, if different scenarios share identical predicted prices for a given future period t^j ($j = i+1, ..., N$), the (dummy) bunkering decisions during period t^j at port j should be consistent across these scenarios, not independent and

varied. To enforce this consistency, nonanticipativity constraints are introduced, as outlined in (Adulyasak et al., 2015). These constraints ensure that bunkering decisions align across different scenarios, preventing decisions based on future information not yet revealed. A scenario tree can effectively represent the full set of predicted scenarios for downstream fuel prices. In this tree, let H_s^t denote the index of the scenario node at period t for scenario s, and let $\dot{z}^j_{H_s^{tj}}$ represent the decision variable z_s^j at node H_s^{tj}. The nonanticipativity constraints can then be formulated as

$$z_s^l = \dot{z}^l_{H_s^{tl}} \quad \forall l \in \{i+1, ..., N\}, s \in \mathcal{S}. \tag{4.10}$$

For an applied understanding of these constraints, consider Example 4.2.

Example 4.2 *Imagine a shipping route encompassing four ports, as illustrated in Figure 4.3. Upon reaching port 1, the ship observes a fuel price of \$400/MT. Using an ML model, eight scenarios are predicted for future fuel prices at ports 2, 3, and 4. These scenarios are organized into a scenario tree, with each node indexed near the respective price label. Nonanticipativity constraints, as shown on the right-hand side of Figure 4.3, ensure decision consistency. For instance, scenarios 1 and 2, which converge at the same node index (1) at period t^2, necessitate identical decisions for period t^2 (i.e., $z_1^1 = z_2^2$), linked through the auxiliary decision variable $\dot{z}_1^2$.*

Considering the aforementioned estimated marginal distribution of future fuel prices at downstream ports, the multistage contextual stochastic ship bunkering problem can be formulated as follows:

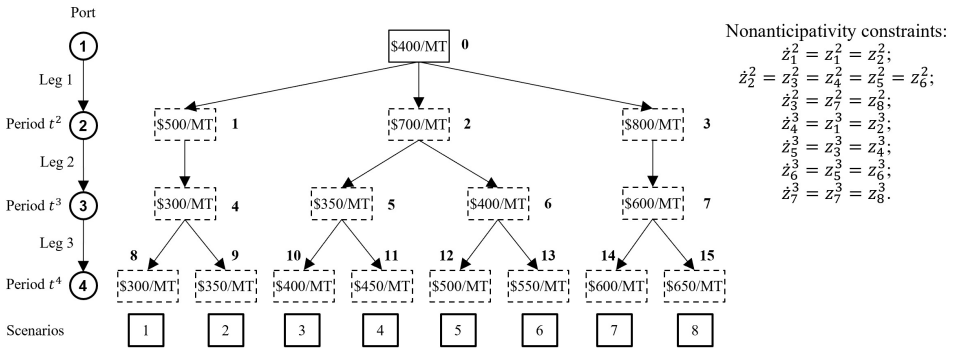

Figure 4.3: An illustrative example of nonanticipativity constraints

[**Model-MSD**]

$$\min_{\mathbf{z}',\mathbf{R}'} z^i x^i_{t^i} + \sum_{s\in\mathcal{S}} \lambda_s \sum_{j=i+1}^{N} z^j_s \hat{x}^j_{t^j|t^i,s} \tag{4.11}$$

subject to

$$z^i \geq G^i - R^i \tag{4.12}$$

$$z^l_s \geq G^l - R^l_s \quad \forall l \in \{i+1, ..., N\}, s \in \mathcal{S} \tag{4.13}$$

$$z^i \leq V - R^i \tag{4.14}$$

$$z^l_s \leq V - R^l_s \quad \forall l \in \{i+1, ..., N\}, s \in \mathcal{S} \tag{4.15}$$

$$R^{i+1}_s = R^i + z^i - G^i \quad \forall s \in \mathcal{S} \tag{4.16}$$

$$R^{l+1}_s = R^l_s + z^l_s - G^l \quad \text{if } i < N-1, \forall l \in \{i+1, ..., N-1\},\ s \in \mathcal{S} \tag{4.17}$$

$$R^{1'} = R^N_s + z^N_s - G^N \quad \forall s \in \mathcal{S} \tag{4.18}$$

$$R^{1'} = R^1 \tag{4.19}$$

$$z^i \geq 0 \tag{4.20}$$

$$z^l_s \geq 0 \quad \forall l \in \{i+1, ..., N\}, s \in \mathcal{S} \tag{4.21}$$

$$R_s^l \geq 0 \quad \forall l \in \{i+1, ..., N\}, s \in \mathcal{S} \tag{4.22}$$

and Constraints (4.10). Here, $\mathbf{z}' := (z^i, z_1^{i+1}, ..., z_{|\mathcal{S}|}^{i+1}, ..., z_1^N, ..., z_{|\mathcal{S}|}^N)$ and $\mathbf{R}' := (R_1^{i+1}, ..., R_{|\mathcal{S}|}^{i+1}, ..., R_1^N, ..., R_{|\mathcal{S}|}^N)$. Constraints (4.12)–(4.22) represent the scenario-based counterparts for Constraints (4.2)–(4.7).

4.5 Predictive Methods

In this section, we delve into the development of predictive models, crucial for informed decision making in the ship bunkering management problem. Section 4.5.1 lays the groundwork by defining the specific prediction problem at hand. In Section 4.5.2, we introduce the TC-LSTM model, designed to capture both *spatial* and *temporal* dependencies in fuel price time series across various ports. In Section 4.5.3, we utilize the MC dropout framework to estimate the conditional distribution of multi-port fuel prices, offering a probabilistic perspective that aligns with the multistage contextual stochastic nature of the bunkering decision problem.

4.5.1 Prediction problem definition

In the context of our study, the prediction problem emerges when a container ship docks at port i during period t^i. At this juncture, we gather historical data $\mathbf{X}_{t^i} := [\mathbf{x}_1, ..., \mathbf{x}_{t^i}] \in \mathbb{R}^{N \times t^i}$, encompassing consecutive periods up to the current one. This dataset comprises fuel price vectors $\mathbf{x}_t := (x_t^1, ..., x_t^N)$ for each period $t = 1, ..., t^i$ across all N ports.

The primary objective here is to generate accurate predictions for future fuel prices at downstream ports j ($j = i+1, ..., N$) for their respective periods t^j. These

predictions can be in the form of point estimates, denoted as $(\hat{x}^{i+1}_{t^{i+1}|t^i}, ..., \hat{x}^{N}_{t^{N}|t^i})$, which provide specific predicted values for fuel prices. Alternatively, we may consider distributional estimates, represented as $(\hat{x}^{i+1}_{t^{i+1}|t^i,s}, ..., \hat{x}^{N}_{t^{N}|t^i,s})$ for each scenario $s \in \mathcal{S}$. The latter approach aims to capture the range of possible future outcomes, reflecting the inherent uncertainty in fuel price fluctuations.

To streamline the notation and generalize the prediction task, we define N' as the number of correlated time series under consideration. In the context of our specific problem, N' corresponds to the number of downstream ports, which is $N - i$. Each of these time series comprises T available historical values, where in our case, T is equivalent to t^i, the current period when the ship is at port i. We denote the entire historical dataset as $\mathbf{X}'_T := [\dot{\mathbf{x}}'_1, ..., \dot{\mathbf{x}}'_T] \in \mathbb{R}^{N' \times T}$. Here, each vector $\dot{\mathbf{x}}'_t := (\dot{x}^1_t, ..., \dot{x}^{N'}_t)$ for $t = 1, ..., T$ represents the historical values across all N' time series for a given historical period t. Specifically, $\dot{x}^n_t$ $(n = 1, ..., N'; t = 1, ..., T)$ denotes the historical value of the n-th time series during the period t.

Given the historical dataset $\mathbf{X}'_T$, our objective is to predict the values for the next τ periods for all relevant time series. Directly using the entire dataset $\mathbf{X}'_T$ as input for a prediction model poses significant challenges. These challenges stem from the high dimensionality and potential redundancy in the data, which can lead to computational inefficiency and difficulties in capturing the most relevant trends and patterns. To address these issues, we focus on a subset of the data, selecting the previous d values of each time series up to and including period T. This selection is represented as $[\dot{\mathbf{x}}'_{T-d+1}, ..., \dot{\mathbf{x}}'_T]^\top \in \mathbb{R}^{d \times N'}$. This approach not only reduces the complexity of the input data but also allows for capturing the most recent and therefore potentially most relevant trends and dependencies. The parameter d, representing the number of periods considered, is a hyperparameter that can be

tuned to balance the trade-off between capturing sufficient historical context and maintaining model manageability.

To transform the historical dataset $\mathbf{X}'_T$ into a format suitable for ML model training, we adopt a sliding window approach. Starting from period 1, the first *input* for training is selected as $[\dot{\mathbf{x}}'_1, ..., \dot{\mathbf{x}}'_d]^\top \in \mathbb{R}^{d \times N'}$, capturing the initial d periods of the time series. Correspondingly, the first *output*, representing the next τ periods, is $[\dot{\mathbf{x}}'_{d+1}, ..., \dot{\mathbf{x}}'_{d+\tau}]^\top \in \mathbb{R}^{\tau \times N'}$. To generate subsequent input-output pairs, we define an interval e, which determines how the window slides over the time series. The second input-output pair, for instance, is extracted starting from period $1+e$, and is denoted by $([\dot{\mathbf{x}}'_{1+e}, ..., \dot{\mathbf{x}}'_{d+e}]^\top, [\dot{\mathbf{x}}'_{d+1+e}, ..., \dot{\mathbf{x}}'_{d+\tau+e}]^\top)$.

Given the dataset $\mathbf{X}'_T$, this sliding window process yields a total of $T' = \lfloor 1 + (T-(d+\tau))/e \rfloor$ input-output pairs. This calculation assumes that $(T-(d+\tau))/e$ is an integer. In cases where this division does not yield an integer (as detailed in Case 2 in Example 4.3), we adjust the starting point by omitting the first $(T-(d+\tau)) \mod e$ periods of values. The resulting set of training input-output pairs is denoted as $\{(\ddot{\mathbf{x}}_t, \ddot{\mathbf{y}}_t)\}_{t=1}^{T'}$. Each input $\ddot{\mathbf{x}}_t$ is a matrix $[\dot{\mathbf{x}}'_{1+e(t-1)}, ..., \dot{\mathbf{x}}'_{e(t-1)+d}]^\top$, and each output $\ddot{\mathbf{y}}_t$ is $[\dot{\mathbf{x}}'_{e(t-1)+d+1}, ..., \dot{\mathbf{x}}'_{e(t-1)+d+\tau}]^\top$.

To optimize the data utilization and maintain continuity in the input-output pairs for training, it is practical to set $\tau = e \leq d$. This requirement is based on the following considerations:

1. If the interval e is set larger than the number of periods d considered for each input, the sliding window skips over certain historical periods. This skipping results in the loss of potentially valuable information, as detailed in Case 3 in Example 4.3. To prevent such information gaps, we ensure that $e \leq d$.

2. Setting e greater than τ leads to a situation where the data fetching pace outstrips the rate at which new data is generated. In other words, the model would be required to predict further into the future than the data available, which is impractical and violates the constraints of real-time data availability, as explained in Case 4 in Example 4.3.

3. Conversely, if e is set smaller than τ, it implies that the sliding window moves too slowly. In such a scenario, part of the predicted target data may already be realized. This situation diminishes the practical utility of the predictions. This issue is elaborated in Case 5 in Example 4.3.

Example 4.3 (Cases of the Reconstruction of Time Series Data) *Figure 4.4 depicts five cases. In Case 1, we can obtain four training input–output pairs* $\{(\ddot{\mathbf{x}}_t, \ddot{\mathbf{y}}_t)\}_{t=1}^{4}$ *and use* $\ddot{\mathbf{x}}_5$ *to obtain the point prediction (or distributional estimates) of the subsequent two periods. In Case 2, as* $\big(T-(d+\tau)\big)/e = 7/2$ *is not an integer, we cut the values of the first period to constitute four training input–output pairs. In Case 3, as* $\big(T-(d+\tau)\big)/e = 7/2$ *is not an integer, we first cut values of the first period. Then, since* $e > d$, $\dot{\mathbf{x}}'_3$, $\dot{\mathbf{x}}'_5$, $\dot{\mathbf{x}}'_7$, *and* $\dot{\mathbf{x}}'_9$ *are never treated as inputs, leading to information ignorance. In Case 4, as* $e > \tau$, *although we train the model using* $\{(\ddot{\mathbf{x}}_t, \ddot{\mathbf{y}}_t)\}_{t=1}^{3}$, *the prediction input* $\ddot{\mathbf{x}}_4$ *requires values from period 12 that are to be predicted and not available, violating reality. In Case 5, as* $e < \tau$, *although we train the model using* $\{(\ddot{\mathbf{x}}_t, \ddot{\mathbf{y}}_t)\}_{t=1}^{3}$, *it is unable to predict the values of periods 11, 12, and 13 using* $\ddot{\mathbf{x}}_4$.

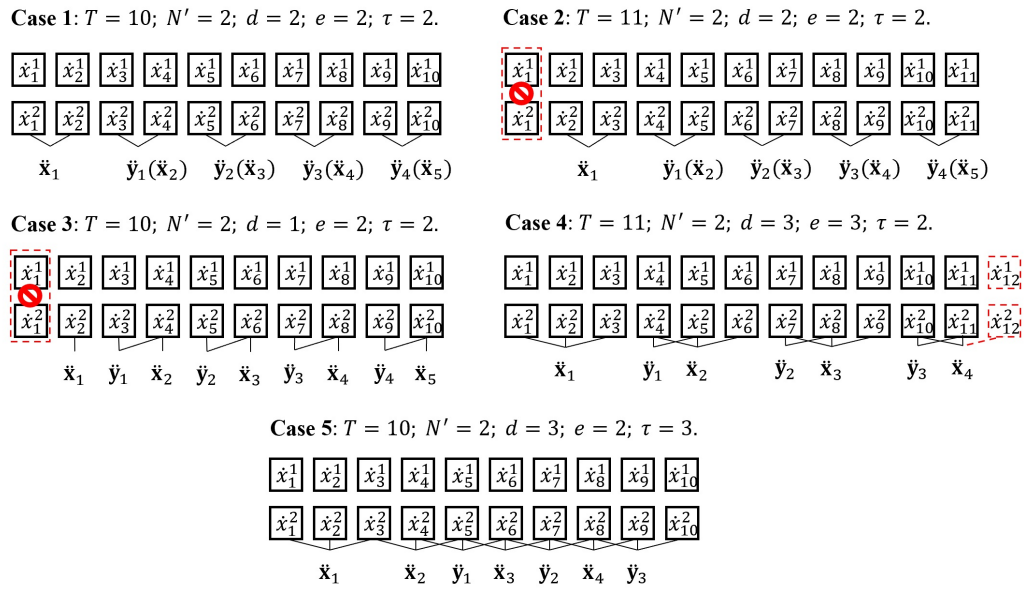

Figure 4.4: The reconstruction of time series data

4.5.2 Correlated ship fuel price prediction: TC-LSTM

Deep RNNs have achieved breakthroughs in processing sequential data, leading to various models based on these networks for time series analysis (Medsker and Jain, 2001). The most well-known model is the LSTM model (Hochreiter and Schmidhuber, 1997), which captures the temporal patterns of sequences by retaining the history of a sequence in its hidden units. LSTM models have shown good performance in single time series forecasting over traditional time series prediction methods (Siami-Namini et al., 2018). However, basic LSTM models are inadequate for predicting correlated time series, as they only model intra-sequence temporal dependencies and ignore inter-sequence spatial dependencies among multiple time series (Wan et al., 2020). Although intra-sequence temporal dependencies reveal the changing patterns in a single series, inter-sequence spatial dependencies provide reasonable explanations for the abnormal changes in the series caused by

other time series (Wan et al., 2020). Therefore, it is critical to explicitly model both intra-sequence temporal dependencies and inter-sequence spatial dependencies to make accurate correlated time series predictions. To avoid a redundant introduction of the LSTM, refer to Hochreiter and Schmidhuber (1997) and Siami-Namini et al. (2018).

In this study, we not only consider the intra-dependencies of each port's fuel prices but also address the inter-dependencies of fuel prices at different downstream ports. To achieve this, we use a TC-LSTM model that can simultaneously capture the spatial and temporal dependencies of correlated time series following Wan et al. (2020).

The TC-LSTM has two main parts: (1) a spatial-temporal (ST) cell, and (2) an ST fusion, the whole structure of which is shown in Figure 4.5, which shows how to train the model using $\{(\ddot{\mathbf{x}}_t, \ddot{\mathbf{y}}_t)\}_{t=1}^{T'}$. Specifically, the ST cell contains an intra-sequence temporal (IST) channel, which captures the changing patterns within each time series, and an inter-sequence spatial (ISS) channel, which captures the influence of other time series; the two channels are shown in Figure 4.6. The information from both channels is fused to generate the final prediction, aggregating both intra- and inter-sequence information, as shown in in Figure 4.7. The following introduction follows the notation convention in Section 4.5.1.

The ST cell

The IST channel. In Figure 4.6, the blue line illustrates the functioning of the IST channel for the t-th input–ouput pair, which takes $\ddot{\mathbf{x}}_t$, $\mathbf{H}_{t-1}^{\text{intra}}$, and $\mathbf{C}_{t-1}^{\text{intra}}$ ($t \in \{2, ..., T'\}$) as inputs. Here, $\mathbf{H}_{t-1}^{\text{intra}}$ is the hidden layer output matrix of the IST channel from the previous pair, and $\mathbf{C}_{t-1}^{\text{intra}}$ is the cell state of the IST channel from

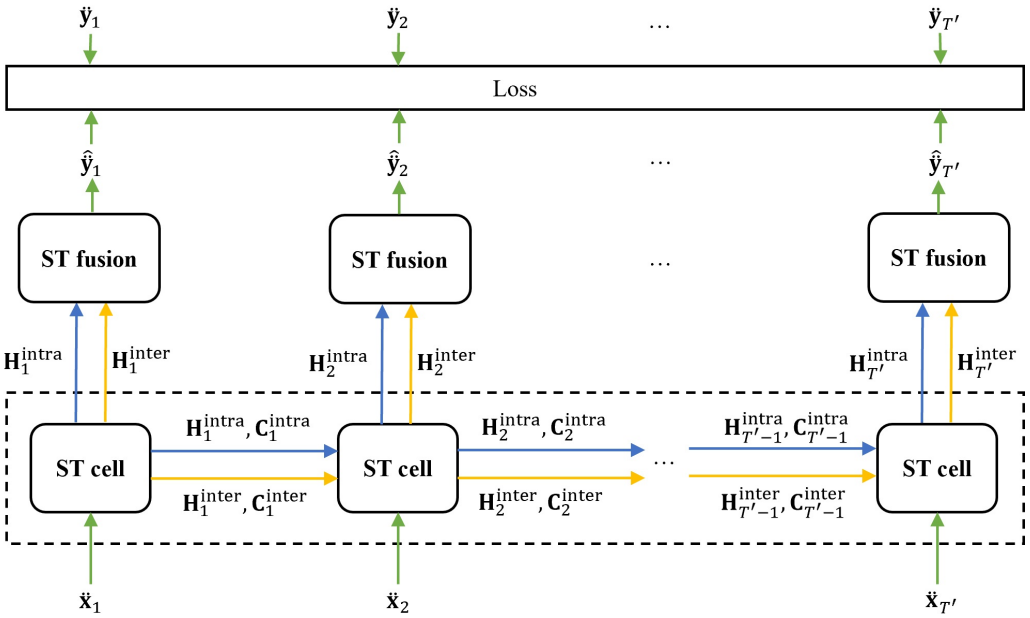

Figure 4.5: The structure and training process of the TC-LSTM

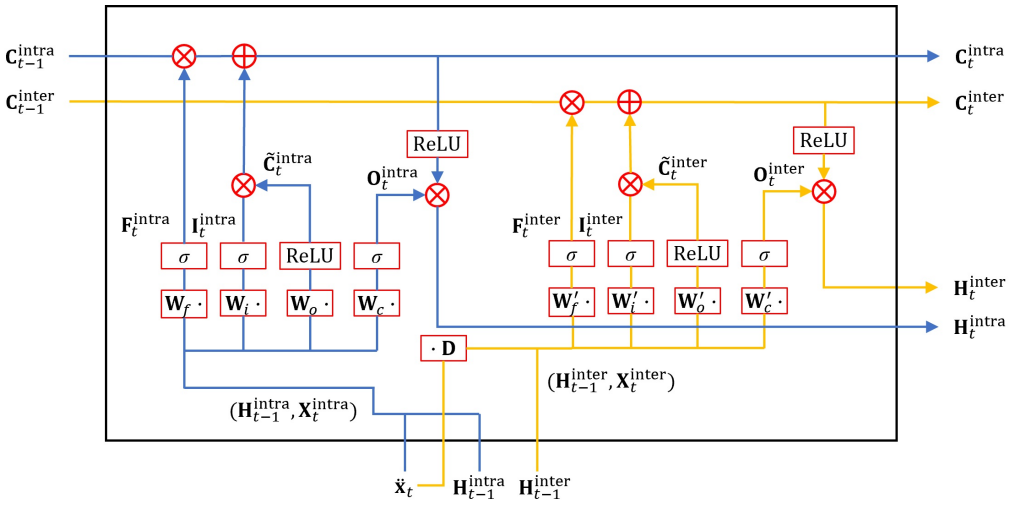

Figure 4.6: The structure of the ST cell

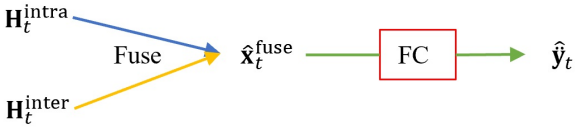

Figure 4.7: The structure of the ST fusion

the previous pair, all with dimension $\mathbb{R}^{\mu \times N'}$, where μ is the number of dimensions (neurons) in the embedding representations. The IST channel architecture comprises the cell state $\mathbf{C}_t^{\text{intra}}$, shown by the horizontal line running through the top of the chain, and three types of gates that control the information flow into and out of the cell state. These gates are the forget, input, and output gates, each of which is composed of a sigmoid neural net layer.

The forget gate determines the information to discard from the cell state of the last pair, controlled by a sigmoid layer concatenating the previous output $\mathbf{H}_{t-1}^{\text{intra}}$ and the current input feature $\mathbf{X}_t^{\text{intra}} = \ddot{\mathbf{x}}_t$, producing a value between 0 and 1 for each component in $\mathbf{C}_{t-1}^{\text{intra}}$. This value is stored in $\mathbf{F}_t^{\text{intra}} \in \mathbb{R}^{\mu \times N'}$ and calculated as follows:

$$\mathbf{F}_t^{\text{intra}} = \sigma\Big(\mathbf{W}_f\big(\mathbf{H}_{t-1}^{\text{intra}}, \mathbf{X}_t^{\text{intra}}\big) + \mathbf{b}_f\Big), \tag{4.23}$$

where σ denotes the sigmoid function, and $\mathbf{W}_f \in \mathbb{R}^{\mu \times (d+\mu)}$ and $\mathbf{b}_f \in \mathbb{R}^{\mu \times N'}$ denote the weight matrix and the bias matrix of the forget gate layer, respectively.

The input gate, controlling which cell state values to update, is managed by a sigmoid layer producing an output matrix $\mathbf{I}_t^{\text{intra}} \in \mathbb{R}^{\mu \times N'}$ through the following equation:

$$\mathbf{I}_t^{\text{intra}} = \sigma\Big(\mathbf{W}_i\big(\mathbf{H}_{t-1}^{\text{intra}}, \mathbf{X}_t^{\text{intra}}\big) + \mathbf{b}_i\Big), \tag{4.24}$$

where $\mathbf{W}_i \in \mathbb{R}^{\mu \times (d+\mu)}$ and $\mathbf{b}_i \in \mathbb{R}^{\mu \times N'}$ denote the weight matrix and the bias matrix of the input gate layer, respectively. Then, this layer's output is element-wise multiplied with a candidate matrix $\tilde{\mathbf{C}}_t^{\text{intra}} \in \mathbb{R}^{\mu \times N'}$, computed as follows:

$$\tilde{\mathbf{C}}_t^{\text{intra}} = \text{ReLU}\Big(\mathbf{W}_c\big(\mathbf{H}_{t-1}^{\text{intra}}, \mathbf{X}_t^{\text{intra}}\big) + \mathbf{b}_c\Big), \tag{4.25}$$

where $\mathbf{W}_c \in \mathbb{R}^{\mu\times(d+\mu)}$ and $\mathbf{b}_c \in \mathbb{R}^{\mu\times N'}$ denote the weight matrix and the bias matrix of the rectified linear unit (ReLU) layer, respectively.

In the IST channel, the forget gate $\mathbf{F}_t^{\text{intra}}$ determines which parts of the previous cell state $\mathbf{C}_{t-1}^{\text{intra}}$ to forget, whereas the input gate $\mathbf{I}_t^{\text{intra}} \otimes \tilde{\mathbf{C}}_t^{\text{intra}}$ decides which parts of the new input to keep, where $\otimes$ represents the element-wise product of the matrices. By combining these two, we can update the old cell state to the new cell state, as follows:

$$\mathbf{C}_t^{\text{intra}} = \mathbf{F}_t^{\text{intra}} \otimes \mathbf{C}_{t-1}^{\text{intra}} + \mathbf{I}_t^{\text{intra}} \otimes \tilde{\mathbf{C}}_t^{\text{intra}}. \tag{4.26}$$

Moving to the output of the IST channel, we use the updated cell state $\mathbf{C}_t^{\text{intra}}$ as the basis for the output. We first pass the concatenation of $\mathbf{H}_{t-1}^{\text{intra}}$ and $\mathbf{X}_t^{\text{intra}}$ through a sigmoid layer to obtain $\mathbf{O}_t^{\text{intra}}$, which determines which parts of the cell state to output. The equation is as follows:

$$\mathbf{O}_t^{\text{intra}} = \sigma\Big(\mathbf{W}_o\big(\mathbf{H}_{t-1}^{\text{intra}}, \mathbf{X}_t^{\text{intra}}\big) + \mathbf{b}_o\Big), \tag{4.27}$$

where $\mathbf{W}_o \in \mathbb{R}^{\mu\times(d+\mu)}$ and $\mathbf{b}_o \in \mathbb{R}^{\mu\times N'}$ denote the weight matrix and the bias matrix of the output gate layer, respectively.

Finally, the updated cell state $\mathbf{C}_t^{\text{intra}}$ passes through a ReLU layer and is multiplied with $\mathbf{O}_t^{\text{intra}}$ to obtain the output of the IST channel, denoted by $\mathbf{H}_t^{\text{intra}}$ as follows:

$$\mathbf{H}_t^{\text{intra}} = \mathbf{O}_t^{\text{intra}} \otimes \text{ReLU}\big(\mathbf{C}_t^{\text{intra}}\big). \tag{4.28}$$

The ISS channel. Similarly, the ISS channel comprises the cell state and three gates, depicted by the yellow line in Figure 4.6. The main difference between the IST channel and the ISS channel is that the input feature matrix $\ddot{\mathbf{x}}_t$ of the ISS

channel is first multiplied by a spatial relation matrix $\mathbf{D} \in \mathbb{R}^{N' \times N'}$, which embeds mutual influence among correlated time series (Wan et al., 2020). In this study, elements in the spatial relation matrix $\mathbf{D}$ are initialized (the initialization process is introduced in Section 4.6.1) and then fine-tuned during the training process. The overall functioning process of the ISS channel is shown mathematically as follows:

$$\mathbf{X}_t^{\text{inter}} = \ddot{\mathbf{x}}_t \mathbf{D}, \tag{4.29}$$

$$\mathbf{F}_t^{\text{inter}} = \sigma\Big(\mathbf{W}_f'\big(\mathbf{H}_{t-1}^{\text{inter}}, \mathbf{X}_t^{\text{inter}}\big) + \mathbf{b}_f'\Big), \tag{4.30}$$

$$\mathbf{I}_t^{\text{inter}} = \sigma\Big(\mathbf{W}_i'\big(\mathbf{H}_{t-1}^{\text{inter}}, \mathbf{X}_t^{\text{inter}}\big) + \mathbf{b}_i'\Big), \tag{4.31}$$

$$\tilde{\mathbf{C}}_t^{\text{inter}} = \text{ReLU}\Big(\mathbf{W}_c'\big(\mathbf{H}_{t-1}^{\text{inter}}, \mathbf{X}_t^{\text{inter}}\big) + \mathbf{b}_c'\Big), \tag{4.32}$$

$$\mathbf{C}_t^{\text{inter}} = \mathbf{F}_t^{\text{inter}} \otimes \mathbf{C}_{t-1}^{\text{inter}} + \mathbf{I}_t^{\text{inter}} \otimes \tilde{\mathbf{C}}_t^{\text{inter}}, \tag{4.33}$$

$$\mathbf{O}_t^{\text{inter}} = \sigma\Big(\mathbf{W}_o'\big(\mathbf{H}_{t-1}^{\text{inter}}, \mathbf{X}_t^{\text{inter}}\big) + \mathbf{b}_o'\Big), \tag{4.34}$$

$$\mathbf{H}_t^{\text{inter}} = \mathbf{O}_t^{\text{inter}} \otimes \text{ReLU}\big(\mathbf{C}_t^{\text{inter}}\big). \tag{4.35}$$

The ST fusion

As shown in Figure 4.7, the role of the fusion module is to aggregate the outputs of the two channels. We first sum the outputs of the two channels as follows:

$$\hat{\mathbf{x}}_t^{\text{fuse}} = \delta^{\text{intra}} \otimes \mathbf{H}_t^{\text{intra}} + \delta^{\text{inter}} \otimes \mathbf{H}_t^{\text{inter}}, \tag{4.36}$$

where $\delta^{\text{intra}} \in \mathbb{R}^{\rho \times N'}$ and $\delta^{\text{inter}} \in \mathbb{R}^{\rho \times N'}$ are the learnable matrices that balance the temporal and spatial dependencies, and ρ is the number of dimensions (neurons)

in the embedding representation of the fusion layer. Then, $\hat{\mathbf{x}}_t^{\text{fuse}} \in \mathbb{R}^{\rho \times N'}$ passes through a fully connected (FC) layer with a Tanh activation function to generate the final prediction as follows:

$$\hat{\ddot{\mathbf{y}}}_t = \text{Tanh}\big(\mathbf{W}'_{fc}\hat{\mathbf{x}}_t^{\text{fuse}} + \mathbf{b}'_{fc}\big), \tag{4.37}$$

where $\hat{\ddot{\mathbf{y}}}_t \in \mathbb{R}^{\tau \times N'}$, $\mathbf{W}'_{fc} \in \mathbb{R}^{\tau \times \rho}$, and $\mathbf{b}'_{fc} \in \mathbb{R}^{\tau \times N'}$.

Hyperparameters and prediction loss

The TC-LSTM is trained by back propagation, which is a method for computing loss gradients with respect to the weights by recursively applying the chain rule (Werbos, 1990). This allows the loss from the output layer to be propagated back to the preceding layers, enabling weight adjustments that minimize this loss. Backpropagation computes gradients efficiently, making it feasible to use gradient descent for training multilayer neural networks. The key hyperparameters in a neural network include the learning rate, epochs (number of iterations), and batch size, which collectively address the issues of underfitting and overfitting (Jain et al., 1996). The learning rate determines the step size at each iteration when updating the weights, controlling the speed at which weights are adjusted. Epochs refer to the number of times the entire training dataset is used for training. The batch size represents the number of examples in a mini-batch, which is a strict and nonempty subset of the entire training set. Examples in a mini-batch are processed by the network simultaneously to update the weights. Consequently, the total number of batches in an epoch is equal to the ratio of the training set size to the batch size.

The prediction loss of this task is the MSE between $\ddot{\mathbf{y}}_t$ and the predicted values $\hat{\ddot{\mathbf{y}}}_t$, shown as follows:

$$L_t(\omega) = \|\ddot{\mathbf{y}}_t - \hat{\ddot{\mathbf{y}}}_t\|_2^2, \tag{4.38}$$

where ω denotes all parameters in the TC-LSTM.

4.5.3 Distributional estimates in deep learning: MC dropout

In Model-MSD, we aim to predict the distribution of future fuel prices, considering the uncertainty of point predictions. In deep learning, Bayesian neural networks (BNNs) have attracted considerable attention as a principled framework for estimating uncertainty (Mackay, 1992; Neal, 2012). BNNs assign a prior to the network parameters and focus on determining the posterior distribution of the parameters instead of a point prediction (Zhu and Laptev, 2017). However, due to the complex nonlinearity and nonconjugacy of deep learning models, exact posterior inference is often infeasible, and traditional approximate Bayesian inference algorithms struggle to scale with large neural networks.

Recent advances in BNNs primarily employ variational inference, which optimizes the variational lower bound (Graves, 2011). Despite their potential, these algorithms require modifications to the neural network's loss function and training algorithm, which can be nontrivial. Practitioners often prefer solutions that leave the neural network architecture unchanged and can be applied to pre-trained models. Moreover, many inference algorithms introduce additional model parameters, sometimes doubling the quantity, which poses scaling challenges due to the large number of parameters used in practice (Zhu and Laptev, 2017).

Given these challenges, there is a growing preference for approaches that do

not require altering the neural network architecture and can be applied to pre-trained models. This consideration has led to the adoption of MC dropout (Gal and Ghahramani, 2016), a technique that maintains the existing network structure while providing a practical means for uncertainty estimation, irrespective of the underlying distributional assumptions.

MC dropout is grounded in the concept of drawing parallels between using dropout in deep learning models and an approximation to the probabilistic deep Gaussian process model (Damianou and Lawrence, 2013). However, it is important to emphasize that the primary purpose of this equivalence is to provide a Bayesian interpretation of dropout, rather than to restrict the application to Gaussian distributions alone. This Bayesian viewpoint enables a broader consideration of model uncertainty, encompassing various types of distributions, including those that are non-Gaussian.

At its core, by implementing stochastic dropouts after each hidden layer, the model outputs are effectively random samples from this distribution (Gal and Ghahramani, 2016). This approach allows for the generation of distributional estimates, which is particularly beneficial for predictions involving inherent uncertainties, such as in the case of fuel oil prices. The outputs, thus, offer a measure of uncertainty, which is invaluable for forecasting tasks characterized by unpredictability. For a comprehensive understanding of the theoretical foundations of the MC dropout framework, including its ties to dropout, Gaussian processes, and variational inference, Gal and Ghahramani (2016) provides an in-depth exploration.

In the context of predicting future values of N' time series over T periods using the MC dropout framework, let us consider the scenario where we have a

well-trained TC-LSTM model and historical input-output pairs $\{(\ddot{\mathbf{x}}_t, \ddot{\mathbf{y}}_t)\}_{t=1}^{T'}$. The input matrix for the next prediction is denoted as $\ddot{\mathbf{x}}_{T'+1} = [\dot{\mathbf{x}}'_{T-d+1}, ..., \dot{\mathbf{x}}'_T]^\top$. Let φ represent the dropout probability, and $|\mathcal{S}|$ be the number of scenarios for which we aim to predict the unknown future values $\tilde{\ddot{\mathbf{y}}}_{T'+1} = [\tilde{\dot{\mathbf{x}}}'_{T+1}, ..., \tilde{\dot{\mathbf{x}}}'_{T+\tau}]^\top$. The MC dropout algorithm, as outlined in Algorithm 4.1, is employed to estimate the distribution of these future values. The procedure is as follows:

Algorithm 4.1 MC Dropout

Input: Input observation $\ddot{\mathbf{x}}_{T'+1}$, trained prediction network, dropout probability φ, number of output scenarios $|\mathcal{S}|$.

Output: Predicted scenarios $\hat{\ddot{\mathbf{y}}}^s_{T'+1}$ $(s = 1, ..., |\mathcal{S}|)$.

1: **for** $s = 1, ..., |\mathcal{S}|$ **do**
2: Apply dropout to the TC-LSTM model with a rate of φ using the input observation $\ddot{\mathbf{x}}_{T'+1}$.
3: Generate the predicted output $\hat{\ddot{\mathbf{y}}}^s_{T'+1}$ for scenario s.
4: **end for**

This algorithm involves iteratively applying the dropout technique to the TC-LSTM model for each scenario, using a dropout rate of φ. By doing so, the model generates a set of scenario-specific predictions $\hat{\ddot{\mathbf{y}}}^s_{T'+1}$ for $s = 1, ..., |\mathcal{S}|$, which collectively represent the distributional estimate of the future time series values.

4.6 Computational Experiments

In this section, we carry out extensive computational experiments to validate the predictive superiority of the TC-LSTM model and compare the decision performance of the TDP and MSD frameworks. To this end, we employ two groups of data: one from the real world, as presented in Section 4.6.1, and the other synthetically generated, as detailed in Section 4.6.2. These experiments were conducted on a personal laptop equipped with a 12th Gen Intel Core i7-12700H processor,

operating at 2.30 GHz, and supported by 16.0 GB of RAM. The experimental codebase for this study was developed in Python, utilizing the Gurobi solver for solving optimization models.

4.6.1 Experiments using real-world data

Data description

The real-world dataset comprises a time series of fuel prices from 13 gloabl bunkering ports collected from Shipping Intelligence Network website.[5] The 13 ports are Rotterdam, Singapore, Fujairah, Houston, Gibraltar, Shanghai, Philadelphia, Hong Kong, Panama, Hamburg, Los Angeles, Santos, and Busan. Considering that since 2020, ships are required to utilize VLSFO globally, we mainly collect the weekly VLSFO prices of these 13 ports from November 22, 2019 to February 10, 2023. We plot the fuel price time series for the 13 ports over the period from November 22, 2019 to February 10, 2023 in Figure 4.8, and presents the descriptive statistics during the specified period in Table 4.1. It can be observed that fuel prices exhibit complex temporal patterns and fluctuations. They show differences across ports, which highlights the importance of considering the spatial aspect in modeling and predicting fuel prices. The mean fuel price ranges from $522.73/MT in Rotterdam to $613.40/MT in Los Angeles. The standard deviation of fuel prices also varies across ports, indicating the varying volatility of fuel prices. The minimum and maximum fuel prices range from $179.00/MT to $1,215.25/MT, reflecting the significant fluctuations in fuel prices over time.

[5] Available at `https://sin.clarksons.net/`, last accessed date: April 25, 2023.

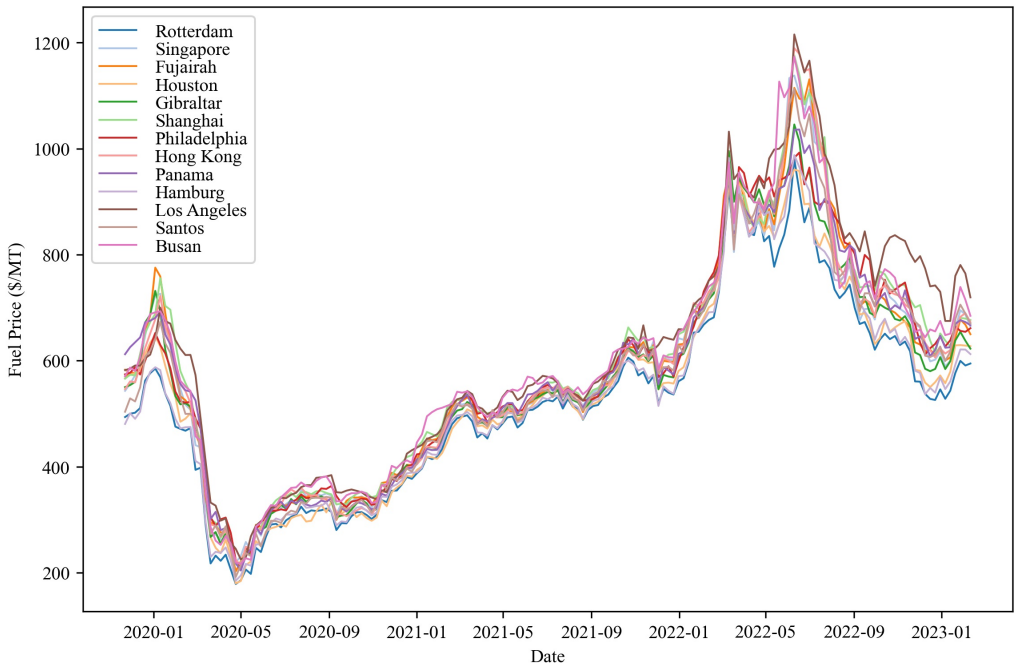

Figure 4.8: Fuel prices of the 13 global bunkering ports

Table 4.1: Descriptive information on fuel prices of the 13 global bunkering ports

Port	Mean Price ($/MT)	Standard Deviation	Minimum Price ($/MT)	Maximum Price ($/MT)
Rotterdam	522.73	181.49	179.00	980.00
Singapore	573.82	203.90	219.75	1,137.75
Fujairah	577.15	206.78	202.75	1,130.75
Houston	541.28	188.54	180.25	960.50
Gibraltar	562.17	195.05	204.75	1,045.50
Shanghai	589.65	208.38	216.25	1,175.75
Philadelphia	577.01	198.53	213.00	992.50
Hong Kong	576.83	210.73	214.75	1,189.25
Panama	574.23	197.72	204.50	1,036.00
Hamburg	533.78	187.31	185.25	987.75
Los Angeles	613.40	222.66	225.50	1,215.25
Santos	568.95	204.43	192.25	1,112.50
Busan	595.27	210.54	217.25	1,173.50

Prediction performance

In this study, we delve into the historical fuel price data of 13 global ports to assess and compare the predictive capabilities of different models. Compared with traditional statistical models, the ability of the LSTM to predict a single time series has been demonstrated in many studies (Wan et al., 2020; Siami-Namini et al., 2018). Therefore, we mainly compare the predictive capabilities of the LSTM and the TC-LSTM constructed in this study for multi-port fuel prices. This comparative analysis involves predicting the weekly fuel prices for these 13 ports for the upcoming τ weeks. For the purpose of this study, we consider various time horizons for τ, specifically $\tau \in \{4, 8, 12, 16, 20\}$. Consequently, the output from the predictive model for each horizon is a vector of dimensions $\tau \times 13$. Correspondingly, the input to the model is a matrix of $d \times 13$, where d, serving as a hyperparameter, is selected from $\{\tau, \tau+4, \tau+8, \tau+12\}$ for each τ value in the set $\{4, 8, 12, 16, 20\}$. To ensure a robust and comprehensive evaluation, the experimental setup includes a detailed split of training, validation, and test data, along with a thorough search and tuning of hyperparameters.

The specifics of this experimental framework, including the tuning results for all hyperparameters, are thoroughly detailed below. Considering that we have 169 weeks of data, we require the length of the sliding time window, e, to be 1. It is important to note that there is no contradiction between the setting here and the introduction of e in Section 4.5.1. Section 4.5.1 outlines the general guidelines for selecting the length of the sliding time window in practice, ensuring the optimal use of data and consecutive input–output pairs. For our simulation experiment, however, we purposefully choose $e = 1$. This decision is driven by the goal

to maximize data usage within the confines of our 169-week dataset. By setting $e = 1$, we are able to generate as many input-output pairs as possible, specifically $169 - \tau - d + 1$ pairs. This maximizes the utilization of available data, providing a rich dataset for training, validation, and testing. When splitting the dataset into training, validation, and test groups, the ratio is set as follows: 70% for training, 20% for validation, and 10% for testing. We apply max–min normalization to scale the data into the range of $[-1, 1]$. During the evaluation stage, we revert the predicted values back to their original scale. In addition to the hyperparameters mentioned earlier, we need to consider other hyperparameters, such as the number of neurons in the ST cell and the ST fusion layer, μ and ρ, respectively, the learning rate, and the batch size. Following the literature (Siami-Namini et al., 2018; Wan et al., 2020), the search range for these hyperparameters is as follows: for the number of neurons in the ST cell and the ST fusion layer, μ and ρ, respectively, the range is $\{32, 64, 128\}$; for the learning rate, the range is $\{0.1, 0.01, 0.001, 0.0001\}$; and for the batch size, the range is $\{32, 64, 128\}$. Furthermore, we restrict the maximum number of training epochs to 50. We utilize the validation set to identify the hyperparameter combination that minimizes the prediction error for the validation set. Subsequently, we merge the training and validation sets and employ the optimal hyperparameter configuration to predict data in the test set, yielding various error metrics. In addition, elements in the spatial relation matrix $\mathbf{D}$ are initialized using the reciprocal of the distance (the shortest sailing nautical miles) between two ports,[6] with diagonal elements set to 1. Subsequently, $\mathbf{D}$ is tuned during the training process. For each τ, the optimal parameter tuning results can be found in Table 4.2, and the prediction metrics under these optimal parameter tuning results

[6] Available at `http://ports.com/`.

for the test set are presented in Table 4.3.

Table 4.2: Tuning results of hyperparameters

τ	d	Learning rate		Batch size		μ		ρ	
		TC-LSTM	LSTM	TC-LSTM	LSTM	TC-LSTM	LSTM	TC-LSTM	LSTM
4	4	0.01	0.01	128	128	32	128	32	–
8	8	0.01	0.01	128	128	128	32	64	–
12	12	0.01	0.01	32	128	128	128	64	–
16	16	0.01	0.01	128	64	64	128	128	–
20	20	0.01	0.01	64	64	32	64	128	–

Table 4.3: Prediction performance of TC-LSTM and LSTM

τ	Test RMSE		Test MAE		Test MAPE	
	TC-LSTM	LSTM	TC-LSTM	LSTM	TC-LSTM	LSTM
4	16.172	16.662	12.649	12.940	1.874	1.914
8	30.125	31.096	23.317	24.162	3.463	3.621
12	32.431	37.488	22.931	30.236	3.349	4.577
16	29.203	42.039	20.154	34.649	2.978	5.332
20	65.648	70.074	49.188	58.237	7.356	9.089

[1] RMSE: root mean square error; MAE: mean absolute error; MAPE: mean absolute percentage error.

In terms of prediction accuracy, the TC-LSTM model consistently outperforms the LSTM model across all τ values. Specifically, the TC-LSTM achieves the best performance for $\tau = 4$, $\tau = 8$, $\tau = 12$, $\tau = 16$, and $\tau = 20$, with improvements of 2.94%, 3.12%, 13.49%, 30.53%, and 6.32% in RMSE, respectively. Moreover, the TC-LSTM model also surpasses the LSTM model in terms of MAE and MAPE for all τ values. These results indicate that the proposed TC-LSTM model is more effective than the LSTM model in capturing the temporal and spatial dependencies of multi-port fuel prices, thus leading to improved prediction accuracy.

The superior performance of the TC-LSTM model can be attributed to its ability to consider both temporal and spatial dependencies in the fuel price time series data. By incorporating this information, the model can better capture the complex dynamics of fuel prices across different ports and periods, resulting in more accurate predictions. This advantage is particularly pronounced for long prediction horizons, as evidenced by the significant improvements in RMSE for large values of τ. These improvements suggest that the TC-LSTM model is better than the LSTM model at handling the increased uncertainty and complexity associated with predicting fuel prices further into the future. By leveraging the accurate fuel price predictions provided by the TC-LSTM model, decision makers can make more informed and cost-effective decisions related to liner shipping services than with alternative models.

Decision performance

Here, we compare the decision performance of two prescriptive analytics frameworks for ship bunkering management: the TDP and MSD frameworks. We consider 13 real global ports and examine various instance scales by selecting four, six, eight, and ten ports randomly from the initial 13 to form shipping routes. In each instance scale, we assess ten routes, yielding ten cases for each number of ports. We simulate the ship bunkering decision-making process for a container ship navigating the entire shipping route in each case. To streamline our computational experiments and focus on the impact of fuel prices on ship bunkering decisions, we assume a one-week interval between two ports of call in all cases. Crucially, for decision-making purposes, the prediction target is modified to a vector representing the requisite future fuel prices at corresponding calling periods.

For instance, in a scenario with four ports of call, while the ship is at the first port, we predict fuel prices for the second, third, and fourth ports for the upcoming one, two, and three weeks, respectively. The bunkering decision at the first port is then made based on these predictions using the optimization model. This process is repeated at each subsequent port, with the predictive model updated and retrained using the latest weekly data.

For each case, the last set of input–output pairs serves as the test set, with a validation set size of 20 sets and the remainder as the training set. When generating the distributional estimates, we apply a dropout rate of 0.1, a scenario number of 200, and the hyperparameters shown in Table 4.2. The selection of the dropout rate and the number of scenarios is guided by decision error observations during validation, where the search ranges for these parameters are $\{0.05, 0.1, 0.2, 0.3\}$ for the dropout rate and $\{50, 100, 200, 300\}$ for the scenario number. Furthermore, regarding other parameters needed in the optimization models, we assume that the maximum tank capacity of the container ship is 1,500 MTs, the initial fuel inventory is a random value in $\mathbb{U}(200, 400)$, and the fuel consumption during each shipping leg is a random value in $\mathbb{U}(300, 500)$. The comparison benchmark (lower bound) is the perfect-foresight framework using real fuel prices from the test set. In addition, we utilize the mean values of historical fuel prices of each port to derive bunkering decisions, which is a common industry practice referred to as the mean-value framework. The overall decision performance, specifically the total bunkering cost, of all frameworks under consideration is displayed in Tables 4.4–4.7.

In the presented tables, we detail the decision performance across shipping routes with varying numbers of ports of call: four, six, eight, and ten. The per-

Table 4.4: Decision performance of shipping routes with four ports of call

Port number	Case	$Cost_{perf}$ ($\times 10^6$ \$)	$Cost_{mean}$ ($\times 10^6$ \$)	$Cost_{TDP}$ ($\times 10^6$ \$)	$Cost_{MSD}$ ($\times 10^6$ \$)	Gap1	Gap2	Gap3
4	1	1.197	1.215	1.215	**1.200**	1.457%	1.457%	**0.223%**
	2	1.199	1.221	1.216	**1.201**	1.797%	1.397%	**0.142%**
	3	1.194	1.212	**1.194**	1.200	1.533%	**0.000%**	0.534%
	4	1.186	1.210	1.210	**1.203**	1.968%	1.968%	**1.372%**
	5	1.194	1.216	**1.194**	1.201	1.817%	**0.000%**	0.618%
	6	1.186	1.207	**1.186**	1.203	1.751%	**0.000%**	1.381%
	7	1.177	1.224	**1.177**	1.188	3.997%	**0.000%**	0.956%
	8	1.180	1.230	**1.180**	1.189	4.247%	**0.000%**	0.784%
	9	1.177	1.216	**1.180**	1.181	3.378%	**0.272%**	0.393%
	10	1.172	1.213	**1.173**	1.182	3.484%	**0.077%**	0.780%
Average		1.186	1.216	**1.192**	1.195	2.543%	**0.517%**	0.718%
Variance		/	/	/	/	0.0001	0.0001	0.0000

[1] $Cost_{perf}$: the total bunkering cost of the perfect-foresight framework; $Cost_{mean}$: the total bunkering cost of the mean-value framework; $Cost_{TDP}$: the total bunkering cost of the TDP framework; $Cost_{MSD}$: the total bunkering cost of the MSD framework; Gap1=$\frac{Cost_{mean}-Cost_{perf}}{Cost_{perf}} \times 100\%$; Gap2=$\frac{Cost_{TDP}-Cost_{perf}}{Cost_{perf}} \times 100\%$; Gap3=$\frac{Cost_{MSD}-Cost_{perf}}{Cost_{perf}} \times 100\%$.

Table 4.5: Decision performance of shipping routes with six ports of call

Port number	Case	$Cost_{perf}$ ($\times 10^6$ \$)	$Cost_{mean}$ ($\times 10^6$ \$)	$Cost_{TDP}$ ($\times 10^6$ \$)	$Cost_{MSD}$ ($\times 10^6$ \$)	Gap1	Gap2	Gap3
6	1	1.608	1.682	**1.608**	1.610	4.610%	**0.000%**	0.160%
	2	1.607	1.686	**1.607**	1.610	4.920%	**0.000%**	0.200%
	3	1.608	1.687	1.619	**1.609**	4.950%	0.680%	**0.080%**
	4	1.608	1.689	1.619	**1.609**	5.020%	0.650%	**0.050%**
	5	1.607	1.691	**1.607**	1.615	5.230%	**0.000%**	0.480%
	6	1.608	1.689	**1.608**	1.615	4.980%	**0.000%**	0.410%
	7	1.601	1.678	**1.601**	1.604	4.780%	**0.000%**	0.130%
	8	1.601	1.682	**1.601**	1.604	5.090%	**0.000%**	0.170%
	9	1.601	1.680	**1.601**	1.603	4.890%	**0.000%**	0.080%
	10	1.603	1.688	**1.603**	**1.603**	5.340%	**0.000%**	**0.000%**
Average		1.605	1.685	**1.607**	1.608	4.980%	**0.130%**	0.180%
Variance		/	/	/	/	0.0000	0.0000	0.0000

formance evaluation is based on the total bunkering costs incurred by four distinct strategies: the perfect-foresight framework ($Cost_{perf}$), the mean-value framework ($Cost_{mean}$), the TDP framework ($Cost_{TDP}$), and the MSD framework ($Cost_{MSD}$).

Table 4.6: Decision performance of shipping routes with eight ports of call

Port number	Case	$Cost_{perf}$ ($\times 10^6$ \$)	$Cost_{mean}$ ($\times 10^6$ \$)	$Cost_{TDP}$ ($\times 10^6$ \$)	$Cost_{MSD}$ ($\times 10^6$ \$)	Gap1	Gap2	Gap3
8	1	2.208	2.342	**2.214**	2.234	6.03%	**0.24%**	1.15%
	2	2.202	2.337	**2.210**	2.238	6.14%	**0.36%**	1.64%
	3	2.204	2.334	**2.209**	2.241	5.91%	**0.24%**	1.71%
	4	2.200	2.346	**2.205**	2.250	6.66%	**0.24%**	2.26%
	5	2.202	2.337	**2.207**	2.255	6.13%	**0.24%**	2.40%
	6	2.200	2.344	**2.205**	2.258	6.56%	**0.24%**	2.63%
	7	2.213	2.341	**2.221**	2.238	5.76%	**0.36%**	1.15%
	8	2.205	2.346	**2.218**	2.244	6.39%	**0.59%**	1.75%
	9	2.214	2.350	**2.223**	2.248	6.16%	**0.40%**	1.54%
	10	2.207	2.357	**2.216**	2.253	6.81%	**0.42%**	2.10%
Average		2.205	2.343	**2.213**	2.246	6.25%	**0.33%**	1.83%
Variance		/	/	/	/	0.0000	0.0000	0.0000

Table 4.7: Decision performance of shipping routes with ten ports of call

Port number	Case	$Cost_{perf}$ ($\times 10^6$ \$)	$Cost_{mean}$ ($\times 10^6$ \$)	$Cost_{TDP}$ ($\times 10^6$ \$)	$Cost_{MSD}$ ($\times 10^6$ \$)	Gap1	Gap2	Gap3
10	1	2.489	2.627	**2.490**	2.540	5.55%	**0.07%**	2.07%
	2	2.487	2.627	**2.490**	2.540	5.65%	**0.15%**	2.15%
	3	2.488	2.634	**2.489**	2.549	5.88%	**0.07%**	2.47%
	4	2.486	2.637	**2.487**	2.549	6.10%	**0.07%**	2.54%
	5	2.487	2.640	**2.488**	2.559	6.15%	**0.07%**	2.90%
	6	2.486	2.643	**2.487**	2.559	6.32%	**0.07%**	2.94%
	7	2.497	2.639	**2.504**	2.548	5.69%	**0.28%**	2.06%
	8	2.493	2.639	**2.504**	2.548	5.87%	**0.44%**	2.22%
	9	2.497	2.648	**2.499**	2.555	6.04%	**0.07%**	2.31%
	10	2.494	2.650	**2.504**	2.555	6.26%	**0.39%**	2.46%
Average		2.490	2.638	**2.494**	2.550	5.95%	**0.17%**	2.41%
Variance		/	/	/	/	0.0000	0.0000	0.0000

Performance gaps, namely Gap1, Gap2, and Gap3, are quantified as the cost differences between the mean-value, TDP, and MSD frameworks in comparison to the perfect-foresight benchmark, respectively.

A discernible trend emerges from the data: as the number of ports increases, the performance gap between the perfect-foresight framework and the alternative strategies tends to widen. This trend reflects the increasing complexity and un-

certainties associated with more extensive routes. However, it is noteworthy that the TDP framework consistently outperforms the mean-value approach, often approaching the performance level of the perfect-foresight framework. The MSD framework, meanwhile, occupies a middle ground in terms of performance.

Interestingly, despite the theoretical advantages discussed in Section 4.4.2, the MSD framework does not consistently outperform the TDP framework. It surpasses the TDP in six specific instances: three in routes with four ports and another three in routes with six ports. This outcome, deviating from theoretical expectations, suggests that the MSD framework's effectiveness may be influenced by factors such as the variance in fuel prices between different ports and the total number of ports in a given route, analyzed below.

As demonstrated in the ship bunkering problem, decision making is more influenced by the relative fuel prices across ports (i.e., their rankings) rather than their absolute values. A larger variance means that even if the point predictions are not perfectly accurate in terms of absolute values, they can still effectively discern the relative rankings of prices across ports. This relative accuracy is crucial because, when these predictions align closely with the actual price rankings, the resulting decisions approximate those made with complete information. This phenomenon is particularly evident in scenarios with four and six ports, where the TC-LSTM model's accuracy in predicting relative rankings is substantiated. In 75% of cases with four and six ports, Gap 2 is 0%, indicating that decisions based on point predictions match those with perfect foresight.

However, when the variance in fuel prices across ports is smaller, predicting the exact ranking becomes more challenging, leading to increased decision-making uncertainty. In such cases, even minor perturbations might alter the rela-

tive rankings in point predictions. This is where distributional estimates become beneficial, as they provide a broader perspective on potential price variations.

Nonetheless, the empirical evaluation of the MSD framework reveals a bias toward specific point realizations. While theory suggests that distributional estimates should yield richer insights for multistage contextual stochastic problems, their practical application, especially when compared against empirical point values, does not always reflect this. These broader estimates might lead to skewed decisions if they inaccurately represent the actual fuel price rankings.

In terms of the impact of the number of ports of call, with an increasing number of ports, we expect more significant shifts in fuel price rankings and larger inter-port fuel price variances. This situation escalates the risk of distributional estimates obscuring decision making, thereby enhancing the appeal of the TDP framework, especially for routes with eight or ten ports.

In summary, while theoretical perspectives highlight the value of distributional estimates in providing a nuanced understanding of uncertainties, empirical evaluations can reveal a different reality. This discrepancy is pivotal in assessing the effectiveness of various prescriptive analytics frameworks. To further investigate this aspect, we conduct additional experiments using synthetically generated data. These experiments aim to explore how the variance in inter-port fuel prices and the number of ports affect decision outcomes, as detailed in Section 4.6.2.

4.6.2 Experiments using synthetic data

Data description

We create the synthetic dataset using historical fuel price data from the port of Shanghai (SH) as a basis. We select 12 additional domestic ports in China and generate their historical fuel price data by considering the distances between these ports and the port of Shanghai. Here, in addition to Shanghai, the 12 other domestic ports that we include are Dalian, Tianjin, Guangzhou, Jiangmen, Dandong, Foshan, Ningbo, Nanjing, Shenzhen, Xiamen, Qingdao, and Hangzhou. Specifically, the correlation of fuel prices between the port of Shanghai and port j is assumed to be $co_{\text{SH},j} = \exp\big(-\text{distance}(\text{SH}, j)/\eta\big)$, where η controls the value of the correlation. For each week t and port j, the weekly fuel price for port j is drawn from a normal distribution with a mean, denoted by X_t^{SH}, using the weekly fuel price of the port of Shanghai, and a standard deviation calculated by $\sqrt{X_t^{\text{SH}} \times (1 - co_{\text{SH},j})}$. We conduct three groups of experiments by setting $\eta = 100$, 1000, and 10000, respectively; a larger η represents a lower variance of inter-port fuel prices. The descriptive statistics of synthetic fuel prices for the 13 domestic ports in China when $\eta = 100$, 1,000, and 10,000 are shown in Tables 4.8– 4.10. As the value of η increases, the inter-port fuel price variance decreases, which is consistent with our expectation that a larger η represents a lower inter-port fuel price variance due to the reduced impact of distance. When $\eta = 100$, the inter-port fuel price variance is relatively high, indicating that the fuel prices of the 13 domestic ports in China are diverse. However, as η increases to 1,000 and 10,000, the inter-port fuel price variance decreases, leading to a reduced difference in fuel prices among ports. The remaining parameters and hyperparameters follow the settings in Section 4.6.1.

Table 4.8: Descriptive statistics of synthetic data when $\eta = 100$

Port	Mean Price ($/MT)	Standard Deviation	Minimum Price ($/MT)	Maximum Price ($/MT)
Shanghai	589.65	207.77	216.25	1,175.75
Dalian	590.33	208.78	210.05	1,184.18
Tianjin	589.08	208.10	214.68	1,176.64
Guangzhou	589.40	207.79	212.73	1,174.49
Jiangmen	589.92	209.60	209.96	1,164.61
Dandong	589.99	209.18	213.33	1,173.24
Foshan	589.52	208.08	212.72	1,165.22
Ningbo	589.85	208.16	212.32	1,176.61
Nanjing	589.52	208.19	216.66	1,172.63
Shenzhen	589.64	208.64	214.38	1,169.69
Xiamen	589.97	209.00	209.78	1,188.73
Qingdao	589.40	207.82	216.64	1,174.07
Hangzhou	589.29	208.78	215.72	1,185.04

Decision performance

Using synthetic data, we examine various instance scales, selecting four, six, eight, ten, and 12 ports randomly from the initial 13 domestic ports to form shipping routes. In each instance scale, we assess 10 cases with different routes for each number of ports. The overall decision performance, specifically the total bunkering cost, of the three frameworks (i.e., $\text{Cost}_{\text{mean}}$, Cost_{TDP}, and Cost_{MSD}) of interest is displayed in Table 4.11. Here, we only show the "votes" of each framework; that is, for each case, the framework that costs the least earns a point. We omit the results of the perfect-foresight framework, given that these results are always considered the ground-truth results, and calculate the average gaps of the mean-value, TDP, and MSD frameworks compared with the perfect-foresight framework for each number of port and η.

As shown in Table 4.11, we can draw some interesting observations regarding

Table 4.9: Descriptive statistics of synthetic data when $\eta = 1,000$

Port	Mean Price ($/MT)	Standard Deviation	Minimum Price ($/MT)	Maximum Price ($/MT)
Shanghai	589.65	207.77	216.25	1,175.75
Dalian	589.60	207.88	215.91	1,172.12
Tianjin	589.73	207.63	217.46	1,175.43
Guangzhou	589.73	207.71	217.98	1,177.02
Jiangmen	589.49	207.62	214.27	1,168.19
Dandong	589.95	207.96	215.55	1,175.85
Foshan	589.23	207.86	218.13	1,173.69
Ningbo	589.62	207.69	216.78	1,176.21
Nanjing	589.63	207.84	215.84	1,176.86
Shenzhen	589.72	207.85	216.14	1,176.51
Xiamen	589.60	207.79	216.00	1,179.42
Qingdao	589.51	207.68	215.89	1,178.15
Hangzhou	589.71	207.80	216.64	1,176.51

the impact of varying η values and port numbers on the performance of the three frameworks. When $\eta = 100$, the TDP framework dominates; it wins in all cases except when the number of ports is set to four, where the MSD framework slightly outperforms it. In this high-variance scenario, the TDP framework effectively captures the main trends in fuel prices, namely the true relative rankings among multi-port fuel prices, allowing it to make more accurate bunkering decisions than the other frameworks. Nevertheless, the MSD framework manages to achieve the lowest average gap in some cases when the number of ports is low, indicating that it can be useful in certain situations.

As the value of η increases to 1,000, the MSD framework becomes more competitive, consistently outperforming the point prediction framework in terms of votes when the number of ports is set to four and six. In this scenario, the low inter-port fuel price variance makes it more difficult for the TDP framework to accurately predict the true ranking among multi-port fuel prices, leading to less

Table 4.10: Descriptive statistics of synthetic data when $\eta = 10,000$

Port	Mean Price ($/MT)	Standard Deviation	Minimum Price ($/MT)	Maximum Price ($/MT)
Shanghai	589.65	207.77	216.25	1,175.75
Dalian	589.70	207.79	216.27	1,175.42
Tianjin	589.65	207.74	216.11	1,178.43
Guangzhou	589.65	207.75	216.77	1,174.27
Jiangmen	589.68	207.71	216.24	1,174.49
Dandong	589.65	207.83	215.75	1,175.49
Foshan	589.69	207.85	216.41	1,177.60
Ningbo	589.68	207.78	216.20	1,176.13
Nanjing	589.63	207.76	216.51	1,175.57
Shenzhen	589.71	207.80	217.06	1,176.09
Xiamen	589.60	207.69	216.30	1,175.32
Qingdao	589.66	207.78	216.39	1,176.36
Hangzhou	589.63	207.81	216.12	1,176.49

Table 4.11: Decision performance using synthetic data

η	Port number	Votes			Average Gaps		
		$\text{Cost}_{\text{mean}}$	Cost_{TDP}	Cost_{MSD}	Gap 1	Gap 2	Gap 3
100	4	0	4	**6**	1.893%	0.939%	**0.343**%
	6	0	**7**	3	3.990%	**0.945**%	0.987%
	8	0	**10**	0	4.126%	**0.616**%	0.723%
	10	0	**10**	0	3.199%	**1.465**%	2.933%
	12	0	**10**	0	2.825%	**1.578**%	*3.549*%
1,000	4	0	0	**10**	0.198%	0.197%	**0.021**%
	6	0	0	**10**	3.838%	1.080%	**0.832**%
	8	0	**10**	0	4.109%	**0.608**%	0.782%
	10	0	**10**	0	3.488%	**1.631**%	2.965%
	12	0	**10**	0	2.901%	**1.336**%	*3.715%*
10,000	4	0	0	**10**	1.789%	1.789%	**0.135**%
	6	0	0	**10**	3.565%	0.430%	**0.211**%
	8	0	**9**	1	4.048%	**0.520**%	0.623%
	10	0	**10**	0	3.177%	**1.345**%	2.848%
	12	0	**10**	0	2.920%	**1.467**%	*3.790%*

optimal decisions, whereas the MSD framework takes advantage of its ability to capture uncertainty.

Furthermore, we notice that as the number of ports increases to 10 and 12, the gap between the MSD framework and the perfect-foresight framework increases. This suggests that the MSD framework might face challenges in handling a large number of ports, even when the variance of inter-port fuel prices is low. This is because, as mentioned above, when there are more ports of call, distributional estimates introduce more noise to to the decision-making process than the TDP framework.

Finally, when $\eta = 10,000$, the MSD framework excels, winning in all cases where the number of ports is set to four or six. The low inter-port fuel price variance, combined with the low number of ports, highlights the strength of the MSD framework in capturing uncertainty and making better decisions as a result. In these cases, the average gap achieved by the MSD framework is the lowest among the three frameworks, further emphasizing its superior performance.

In summary, the MSD framework tends to excel when both the inter-port fuel price variance is low and the number of ports is small, effectively leveraging its ability to account for uncertainty. On the other hand, the TDP framework is more advantageous in high-variance scenarios, where the stability and predictability of relative fuel price rankings are higher. This analysis underscores the nuanced nature of decision making in ship bunkering management and the context-dependent suitability of different predictive analytics frameworks.

4.7 Conclusions

This study investigates the ship bunkering management problem in liner shipping in the presence of uncertain fuel prices. We construct a TC-LSTM model to predict multi-port time series fuel prices, considering both intra-dependencies within each time series and inter-dependencies among multi-port fuel prices. This leads to more accurate fuel price predictions for shipping companies than LSTM. In addition, we develop two prescriptive analytics optimization models for the ship bunkering management problem considering uncertain fuel prices: the TDP and MSD frameworks. The MC dropout technique is employed in the MSD framework to derive distributional estimates for the TC-LSTM model. Finally, we compare the performance of the proposed models using real-world and synthetic data. Our results show that the TC-LSTM model performs better than the traditional LSTM model in predicting multi-port fuel prices. Moreover, the decision performance of the two prescriptive analytics frameworks depends on the inter-port fuel price variance and the number of ports of call on the shipping route. The results obtained in our study provide valuable insights to liner shipping companies seeking to navigate uncertain fuel prices and improve their bunkering management decisions. By adopting suitable prescriptive analytics frameworks, companies can minimize bunkering costs and contribute to enhancing the sustainability of the shipping industry.

Chapter 5

Conclusions and Future Research Directions

5.1 Conclusions

This thesis has demonstrated the innovative application of prescriptive analytics across various challenges in maritime transportation. Through three focused studies, we have explored the data-driven optimization of ship maintenance, maritime routing, and bunkering decisions, each considering unique operational complexities and leveraging advanced machine learning (ML) techniques.

Chapter 2 illustrated a novel approach to ship maintenance optimization using a smart predict-then-optimize (SPO) framework. By integrating operational, repair, and risk costs into the training process of the ML model, the study achieved substantial cost reductions. Computational experiments confirmed that the proposed SPO-based scheme reduced the total operational costs by an average of 43.22% compared to traditional methods and by 10.44% compared to predict-then-

optimize-based schemes. This framework not only minimizes the operational costs but also enhances port efficiency by reducing the need for extensive port state control (PSC) inspections, thereby alleviating port congestion.

Chapter 3 addressed the optimization of PSC officer (PSCO) routing with a decision-focused learning approach. By embedding the optimization problem directly into the ML training process, this study surpassed the traditional two-stage framework in decision quality under certain situations. The use of undominated inspection templates and surrogate loss functions based on noise-contrastive estimation improved the scalability and computational feasibility of the approach, offering a more efficient and effective method for ship inspector routing that reduces unnecessary inspections and port congestion.

Chapter 4 tackled the challenge of optimizing bunkering decisions under uncertain fuel prices. The development of the two-channel long short-term memory (TC-LSTM) model provided superior multi-port fuel price predictions, which informed two advanced prescriptive analytics models: the two-stage contextual deterministic framework with point predictions (TDP) and the multistage contextual stochastic framework with distributional estimates (MSD). The study's findings highlighted the dependency of decision performance on fuel price variance and the number of port calls on each shipping service, offering critical insights for shipping companies aiming to minimize fuel costs and enhance operational sustainability.

Collectively, these studies underscore the transformative potential of integrating ML with optimization in maritime transportation. The methodologies developed not only foster economic efficiency but also contribute to sustainable industry practices by optimizing resource use and decision accuracy. As maritime opera-

tions continue to face dynamic challenges and evolving economic conditions, the insights from this thesis provide a valuable foundation for future innovations in the sector.

5.2 Future Research Directions

The first study proposes ship maintenance plans for ship operators, paving the way for future research. First, because of the varied expertise of PSCOs, PSC inspections at different ports have regional characteristics. For example, there may be subjective differences between the PSCOs regarding the criteria for recording a deficiency. In the future, PSC data at multiple ports can be used to generate a more robust ship maintenance plan. Second, future studies could design ship maintenance plans that consider more practical constraints, including but not limited to operational costs, ship maintenance time slot requirements, and the skills of maintenance crews for the deficiency conditions. On the one hand, when formulating the ship maintenance problem, we can further consider the repair decision of ship operators when they make trade-offs between the repair cost and risk cost for an identified deficiency. This consideration may transform the original one-stage optimization problem into a two-stage optimization problem where the solution complexity increases. Under this challenge, model transformation techniques would be needed when the optimization problem is plugged into the training process of ML models under the SPO framework. On the other hand, ship operators can obtain more accurate values of the three types of operational costs considering the requirements of different ship operators and ship types by consulting industrial experts. Third, instead of making inspection decisions separately for each deficiency

code, we can further plug the RF model used to predict ship detention probability into the training process of the SPO forest (SPOF). In doing so, we can model the risk cost as a comprehensive value influenced by the overall ship condition to consider the nonlinear relationship between having deficiencies under each code and the final detention outcome. And this comprehensive risk cost would influence the inspection decision backwardly in a coupled manner, and thus further influence the training process of the SPOF. Meanwhile, it is not hard to imagine that this methodology would no doubt need huge computational resources because the studied problem is transformed into a multi-output prescriptive problem with correlated optimization targets, and we need to call the detention probability predictor millions of times for all possible inspection schemes and all possible node splitting rules when training the SPOF. Therefore, we may further consider adopting an approximate splitting rule when constructing the prescriptive trees and reducing the size of inspection scheme pool by heuristics. Third, the SPO framework proposed in this study can be modified for the design of inspection plans for PSCOs in formal PSC inspections by considering the corresponding optimization objectives of PSC authorities. Finally, the SPO framework can be applied to other ML algorithms by taking advantage of their structural features. Their decision-making performance of these alternative ML algorithms for ship maintenance planning should be compared.

The second study proposes routing decisions for PSCOs under different prescriptive analytics frameworks, having the following limitations. Theoretically, regarding the adopted decision-focused learning framework, the proposed decision loss functions are all surrogate loss functions that approximate the ground-truth decision losses. To use the ground-truth decision losses, the proposed decision-

focused learning framework could be applied to other ML models by taking advantage of their structural features, such as random forest and k-nearest neighbor. The decision-making performance of these alternative ML algorithms for the PSCO routing problem should be analyzed and compared with the results obtained using artificial neural networks in our study. Practically, our formulation of the PSCO routing problem does not consider more complex real-world constraints, which may result in deviations from actual situations. For instance, foreign visiting ships might change their berthing locations while PSCOs conduct their inspections, making the PSCO problem into a dynamic routing problem. Moreover, the inspection time for each ship may vary depending on individual ship conditions. Future research could tackle these practical limitations and explore more advanced and realistic models to better represent the complexities of the PSCO routing problem.

The third study optimizes data-driven container ship bunkering decisions, having the following future research directions. First, building upon our foundational work with the TC-LSTM model, there exists an opportunity to incorporate other external factors that might influence fuel prices. Beyond our model, there is room to investigate other deep learning algorithms for potential complementary insights. Although the MC dropout technique is effective for our needs, exploring other methods for predicting distributional outcomes in deep learning can offer a broader toolkit for industry professionals. Second, while our current ship bunkering management model offers an operational-level approach, the dynamic nature of liner shipping allows for the continual integration of new constraints and objectives. Aspects like ship routing, emissions, and service quality are areas where future studies could expand, enhancing the decision-making landscape further. Third,

our exploration of the TDP and MSD frameworks opens the door for potential hybrid models that capitalize on the strengths of both. For instance, considering a mixed approach, where fuel price distributions are predicted for initial stages and point estimations for subsequent stages, might provide an enriched decision-making framework. Such endeavors would not replace our findings but would rather stand on the shoulders of our work, aiming to achieve even more nuanced results.

References

Adler, N., Brudner, A., Gallotti, R., Privitera, F., Ramasco, J.J., 2022. Does big data help answer big questions? The case of airport catchment areas & competition. Transportation Research Part B: Methodological 166, 444–467.

Adulyasak, Y., Cordeau, J.F., Jans, R., 2015. Benders decomposition for production routing under demand uncertainty. Operations Research 63(4), 851–867.

Bell, M.G.H., Liu, X., Rioult, J., Angeloudis, P., 2013. A cost-based maritime container assignment model. Transportation Research Part B: Methodological 58, 58–70.

Ben-Tal, A., EL Ghaoui, L., Nemirovski, A., 2009. Robust Programming. Princeton University Press, Princeton.

Berger, A., Pietra, S., Pietra, V., 1996. A maximum entropy approach to natural language processing. Computational Linguistics 22(1), 39–71.

Berk, L., Bertsimas, D., Weinstein, A. M., Yan, J., 2019. Prescriptive analytics for human resource planning in the professional services industry. European Journal of Operational Research 272(2), 636–641.

Bertsimas, D., Dunn, J., Mundru, N., 2019. Optimal prescriptive trees. INFORMS Journal on Optimization 1(2), 164–183.

Bertsimas, D., Brown, D.B., Caramanis, C., 2011. Theory and applications of robust optimization. SIAM Review 53(3), 464–501.

Bertsimas, D., Kallus, N., 2020. From predictive to prescriptive analytics. Management Science 66(3), 1025–1044.

Bertsimas, D., Koduri, N., 2021. Data-driven optimization: A reproducing kernel Hilbert space approach. Operations Research 70(1), 454–471.

Birge, J.R. and Louveaux, F., 2011. Introduction to Stochatic Programming. Springer, New York.

Brandt, T., Wagner, S., Neumann, D., 2021. Prescriptive analytics in public-sector decision-making: A framework and insights from charging infrastructure planning. European Journal of Operational Research 291(1), 379–393.

Breiman, L., 2001. Random forests. Machine Learning 45(1), 5–32.

Brier, G., 1950. Verification of forecasts expressed in terms of probability. Monthly Weather Review 78, 1–3.

Choi, J.S., 2017. Forecasting bunker price using system dynamics. Journal of Korea Port Economic Association 33(1), 75–87.

Chung, W., Kao, S., Chang, C., Yuan, C., 2020. Association rule learning to improve deficiency inspection in port state control. Maritime Policy & Management 47(3), 332–351.

Chiroma, H., Abdulkareem, S., Herawan, T., 2015. Evolutionary neural network model for west Texas intermediate crude oil price prediction. Applied Energy 142, 266–273.

Damianou, A., Lawrence, N.D., 2013. Deep Gaussian processes. In Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics. PMLR, Scottsdale, 207–215.

De, A., Choudhary, A., Turkay, M., Tiwari, M.K., 2021. Bunkering policies for a fuel bunker management problem for liner shipping networks. European Journal of Operational Research 289(3), 927–939.

De, A., Wang, J., Tiwari, M.K., 2020. Hybridizing basic variable neighborhood search With particle swarm optimization for solving sustainable ship routing and bunker management problem. IEEE Transactions on Intelligent Transportation Systems 21(3), 986–997.

De, A., Wang, J., Tiwari, M.K., 2021. Fuel bunker management strategies within sustainable container shipping operation considering disruption and recovery policies. IEEE Transactions on Engineering Management 68(4), 1089–1111.

Elmachtoub, A.N., Grigas, P., 2021. Smart "predict, then optimize". Management Science 68(1), 9–26.

Elmachtoub, A.N., Liang, J.C.N., McNellis, R., 2020. Decision trees for decision-making under the predict-then-optimize framework. In Proceedings of 2020 International Conference on Machine Learning. PMLR, Online, 2858–2867.

Eruguz, A., Tan, T., Houtum, G., 2017. A survey of maintenance and service logistics management: Classification and research agenda from a maritime sector perspective. Computers and Operations Research 85, 184–205.

Fagerholt, K., Heilig, L., Lalla-Ruiz, E., Meisel, F., Wang, S., 2023. Data-driven optimization and analytics for maritime logistics. Flexible Services and Manufacturing Journal 35(1), 1–4.

Ferber, A., Wilder, B., Dilkina, B., Tambe, M., 2020. MIPaaL: Mixed integer program as a layer. In Proceedings of 34th AAAI Conference on Artificial Intelligence. AAAI Press, New York, 1504–1511.

Gal, Y., Ghahramani, Z., 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the 33rd International Conference on Machine Learning. PMLR, New York, 1050–1059.

Gao, Z., Lu, G., Liu, M., Cui, M., 2008. A novel risk assessment system for port state control inspection. In Proceedings of 2008 IEEE International Conference on Intelligence and Security Informatics. IEEE, Taipei, 242–244.

Ghosh, S., Lee, L.H., Ng, S.H., 2015. Bunkering decisions for a shipping liner in an uncertain environment with service contract. European Journal of Operational Research 244(3), 792–802.

Giorgio, M., Guida, M., Pulcini, G., 2015. A condition-based maintenance policy for deteriorating units. An application to the cylinder liners of marine engine. Applied Stochastic Models in Business and Industry 31(3), 339–348.

Golden, B., Levy, L., Vohra, R., 1987. The orienteering problem. Naval Research Logistics 34, 307–318.

Goodfellow, I., 2015. On distinguishability criteria for estimating generative models. In Proceedings of 3rd International Conference on Learning Representations. ICLR, San Diego, 1–6.

Goossens, A., Basten, R., 2015. Exploring maintenance policy selection using the Analytic Hierarchy Process: An application for naval ships. Reliability Engineering and System Safety 142, 31–41.

Graves, A., 2011. Practical variational inference for neural networks. In Advances in Neural Information Processing Systems 24 (NIPS 2011). Curran Associates Inc., New York, 1–9.

Gu, Y., Wallace, S.W., Wang, X., 2019. Integrated maritime fuel management with stochastic fuel prices and new emission regulations. Journal of the Operational Research Society 70(5), 707–725.

Gupta, N., Nigam, S., 2020. Crude oil price prediction using artificial neural network. Procedia Computer Science 170, 642–647.

Gutmann, M., Hyvärinen, A., 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics. PMLR, Sardinia, 297–304.

He, L., Liu, S., Shen, Z., 2022. Smart urban transport and logistics: A business

analytics perspective. Production and Operations Management 31(10), 3771–3787.

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural Computation 9(8), 1735–1780.

Hu, Y., Kallus, N., Mao, X., 2022. Fast rates for contextual linear optimization. Management Science 68(6), 4236–4245.

Jain, A.K., Mao, J., Mohiuddin, K.M., 1996. Artificial neural networks: A tutorial. Computer 29(3), 31–44.

Kallus, N., 2017. Recursive partitioning for personalization using observation data. In Proceedings of 2017 International Conference on Machine Learning. PMLR, Sydney, 1789–1798.

Kallus, N., Mao, X., 2022. Stochastic optimization forests. Management Science 69(4), 1975–1994.

Kim, K., Lim, S., Lee, C.H., Lee, W.J., Jeon, H., Jung, J., Jung, D., 2022. Forecasting liquefied natural gas bunker prices using artificial neural network for procurement management. Journal of Marine Science and Engineering 10(12), 1814.

Kingma, D., Ba, J., 2015. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference for Learning Representations. ICLR, San Diego, 1–13.

Knapp, S., 2007. The Econometrics of Maritime Safety: Recommendations to Enhance Safety at Sea. Ph.D. dissertation, Erasmus University Rotterdam.

MacKay, D.J.C., 1992. A practical Bayesian framework for backpropagation networks. Neural Computation 4(3), 448–472.

Mandi, J., Demirovic, E., Stuckey, P, J., Guns, T., 2020. Smart predict-and-optimize for hard combinatorial optimization problems. In Proceedings of the AAAI Conference on Artificial Intelligence 34(2). AAAI Press, New York, 1603–1610.

Medsker, L.R., Jain, L.C., 2001. Recurrent neural networks. Design and Applications 5, 64–67.

Meng, Q., Du, Y., Wang, Y., 2016. Shipping log data based container ship fuel efficiency modeling. Transportation Research Part B: Methodological 83, 207–229.

Meng, Q., Wang, S., Andersson, H., Thun, K., 2014. Containership routing and scheduling in liner shipping: Overview and future research directions. Transportation Science 48(2), 265–280.

Meng, Q., Wang, Y., Du, Y., 2015. Bunker procurement planning for container liner shipping companies: Multistage stochastic programming approach. Transportation Research Record 2479(1), 60–68.

Meng, Q., Zhang, Y., Xu, M., 2017. Viability of transarctic shipping routes: A literature review from the navigational and commercial perspectives. Maritime Policy & Management 44(1), 16–41.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In Advances

in Neural Information Processing Systems 26 (NIPS 2013). Curran Associates Inc., New York, 3111–3119.

Moubray, J., 1997. Reliability-centered Maintenance. Industrial Press Inc. New York.

Mulamba, M., Mandi, J., Diligenti, M., Lombardi, M., Bucarey, V., Guns, T., 2021. Contrastive losses and solution caching for predict-then-optimize. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence. IJCAI, Montreal, 2833–2840.

Muñoz, M.A., Pineda, S., Morales, J.M., 2022. A bilevel framework for decision-making under uncertainty with contextual information. Omega 108, 102575.

Neal, R.M., 2012. Bayesian Learning for Neural Networks. Springer Science & Business Media, Berlin.

Ng, M.W., 2015. Container vessel fleet deployment for liner shipping with stochastic dependencies in shipping demand. Transportation Research Part B: Methodological 74, 79–87.

Notz, P. M., Wolf, P. K., Pibernik, R., 2023. Prescriptive analytics for a multi-shift staffing problem. European Journal of Operational Research 305(2), 887–901.

Notz, P.M., Pibernik, R., 2022. Prescriptive analytics for flexible capacity management. Management Science 68(3), 1756–1775.

Orojo, O., Tepper, J., McGinnity, T.M., Mahmud, M., 2019. A multi-recurrent network for crude oil price prediction. In Proceedings of 2019 IEEE Symposium Series on Computational Intelligence.IEEE, Xiamen, 2940–2945.

Qi, M., Shen, Z., 2022. Integrating prediction/estimation and optimization with applications in operations management. In Tutorials in Operations Research: Emerging and Impactful Topics in Operations, 36–58.

Sheng, X., Chew, E.P., Lee, L.H., 2015. (s,S) policy model for liner shipping refueling and sailing speed optimization problem. Transportation Research Part E: Logistics and Transportation Review 76, 76–92.

Sheng, X., Lee, L.H., Chew, E.K., 2014. Dynamic determination of vessel speed and selection of bunkering ports for liner shipping under stochastic environment. OR Spectrum 36(2), 455–480.

Siami-Namini, S., Tavakoli, N., Namin, A.S., 2018. A comparison of ARIMA and LSTM in forecasting time series. In Proceedings of 17th IEEE International Conference on Machine Learning and Applications. IEEE, Orlando, 1394–1401.

Soeffker, N., Ulmer, M. W., Mattfeld, D. C., 2022. Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. European Journal of Operational Research 298(3), 801–820.

Song, D., Dong, J., 2013. Long-haul liner service route design with ship deployment and empty container repositioning. Transportation Research Part B: Methodological 55, 188–211.

Stefanakos, C.N., Schinas, O., 2014. Forecasting bunker prices: A nonstationary, multivariate methodology. Transportation Research Part C: Emerging Technologies 38, 177–194.

Stefanakos, C.N, Schinas, O., 2015. Fuzzy time series forecasting of bunker prices: Nonstationary considerations. WMU Journal of Maritime Affairs 14, 177–199.

Sun, Z., Zheng, J., 2016. Finding potential hub locations for liner shipping. Transportation Research Part B: Methodological 93, 750–761.

Tian, X., Yan, R., Liu, Y., Wang, S., 2023. A smart predict-then-optimize method for targeted and cost-effective maritime transportation. Transportation Research Part B: Methodological 172, 32–52.

Tian, X., Yan, R., Wang, S., Liu, Y., Zhen, L., 2023. Tutorial on prescriptive analytics for logistics: What to predict and how to predict. Electronic Research Archive 31(4), 2265–2285.

Tjoa, E., Guan, C., 2021. A survey on explainable artificial intelligence (XAI) toward medical XAI. IEEE Transactions on Neural Networks and Learning Systems 32(11), 4793–4813.

Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D., 2009. Iterated local search for the team orienteering problem with time windows. Computer & Operations Research 36, 3281–3290.

Wan, H., Guo, S., Yin, K., Liang, X., Lin, Y., 2020. CTS-LSTM: LSTM-based neural networks for correlated time series prediction. Knowledge-Based Systems 191, 105239.

Wang, S., Gao, S., Tan, T., Yang, W., 2019. Bunker fuel cost and freight revenue optimization for a single liner shipping service. Computers & Operations Research 111, 67–83.

Wang, S., Meng, Q., 2015. Robust bunker management for liner shipping networks. European Journal of Operational Research 243(3), 789–797.

Wang, S., Tian, X., Yan, R., Liu, Y., 2022. A deficiency of prescriptive analytics—No perfect predicted value or predicted distribution exists. Electronic Research Archive 30(10), 3586–3594.

Wang, S., Yan, R., 2022. "Predict, then optimize" with quantile regression: A global method from predictive to prescriptive analytics and applications to multimodal transportation. Multimodal Transportation 1(4), 100035.

Wang, S., Yan, R., Qu, X., 2019. Development of a non-parametric classifier: Effective identification, algorithm, and applications in port state control for maritime transportation. Transportation Research Part B: Methodological 128, 129–157.

Wang, T., Wang, X., Meng, Q., 2018. Joint berth allocation and quay crane assignment under different carbon taxation policies. Transportation Research Part B: Methodological 117, 18–36.

Wang, X., Teo, C.C., 2013. Integrated hedging and network planning for container shipping's bunker fuel management. Maritime Economics & Logistics 15(2), 172–196.

Wang, Y., Meng, Q., 2021. Optimizing freight rate of spot market containers with uncertainties in shipping demand and available ship capacity. Transportation Research Part B: Methodological 146, 314–332.

Wang, Y., Meng, Q., Du, Y., 2015. Liner container seasonal shipping revenue management. Transportation Research Part B: Methodological 82, 141–161.

Wang, Y., Meng, Q., Kuang, H., 2018. Jointly optimizing ship sailing speed and bunker purchase in liner shipping with distribution-free stochastic bunker prices. Transportation Research Part C: Emerging Technologies 89, 35–52.

Werbos, P.J., 1990. Backpropagation through time: What it does and how to do it. Proceedings of the IEEE 78(10), 1550–1560.

Wilder, B., Dilkina, B., Tambe, M., 2019. Melding the data-decisions pipeline: decision-focused learning for combinatorial optimization. In Proceedings of the 33rd AAAI Conference on Artificial Intelligence. AAAI Press, New York, 1658–1666.

Wu, S., Chen, X., Shi, C., Fu, J., Yan, Y., Wang, S., 2022. Ship detention prediction via feature selection scheme and support vector machine (SVM). Maritime Policy & Management 49(1), 140–153.

Xu, R., Lu, Q., Li, W., Li, K., Zheng, H., 2007a. A risk assessment system for improving port state control inspection. In Proceedings of 2007 International Conference on Machine Learning and Cybernetics. IEEE, Hong Kong, 818–823.

Xu, R., Lu, Q., Li, K., Li, W., 2007b. Web mining for improving risk assessment in port state control inspection. In Proceedings of 2007 International Conference on Natural Language Processing and Knowledge Engineering. IEEE, Beijing, 427–434.

Xu, Z., Lee, C.-Y., 2016. New lower bound and exact method for the continuous berth allocation problem. Operations Research 66(3), 778–798.

Yan, R., Wang, S., 2019. Ship inspection in port state control—Review of current research. Smart Transportation Systems 2019, 233–241.

Yan, R., Wang, S., Cao, J., Sun, D., 2021a. Shipping domain knowledge informed prediction and optimization in port state control. Transportation Research Part B: Methodological 149, 52–78.

Yan, R., Wang, S., Falgerholt, K., 2020. A semi-"smart predict then optimize" (semi-SPO) method for efficient ship inspection. Transportation Research Part B: Methodological 142, 100–125.

Yan, R., Wang, S., Peng, C., 2021b. An artificial intelligence model considering data imbalance for ship selection in port state control based on detention probabilities. Journal of Computational Science 48, 101257.

Yan, R., Wang, S., Zhen, L., 2023. An extended smart "predict, and optimize" (SPO) framework based on similar sets for ship inspection planning. Transportation Research Part E: Logistics and Transportation Review 173, 103109.

Yan, R., Zhuge, D., Wang, S., 2021c. Development of two high-efficient and innovative inspection schemes for PSC inspection. Asia Pacific Journal of Operational Research 38(3), 2040013.

Yang, L., Kwan, M.P., Pan, X., Wan, B., Zhou, S., 2017. Scalable space-time trajectory cube for path-finding: A study using big taxi trajectory data. Transportation Research Part B: Methodological 101, 1–27.

Yang, Z., Qu, Z., 2016. Quantitative maritime security assessment: A 2020 vision. IMA Journal of Management Mathematics 27(4), 453–470.

Yang, Z., Yang, Z., Yin, J., 2018a. Realizing advanced risk-based port state control inspection using data-driven Bayesian networks. Transportation Research Part A: Policy and Practice 110, 38–56.

Yang, Z., Yang, Z., Yin, J., Qu, Z., 2018b. A risk-based game model for rational inspections on port state control. Transportation Research Part E: Logistics and Transportation Review 118, 477–495.

Yao, Z., Ng, S.H., Lee, L.H., 2012. A study on bunker fuel management for the shipping liner services. Computers & Operations Research 39(5), 1160–1172.

Yegnanarayana, B., 2009. Artificial Neural Networks. PHI Learning Pvt. Ltd., Delhi.

Zhang, A., Zheng, Z., Teo, C.P., 2022. Schedule reliability in liner shipping timetable design: A convex programming approach. Transportation Research Part B: Methodological 155, 499–525.

Zhen, L., Wang, S., Zhuge, D., 2017. Dynamic programming for optimal ship refueling decision. Transportation Research Part E: Logistics and Transportation Review 100, 63–74.

Zhu, L., Laptev, N., 2017. Deep and confident prediction for time series at uber. In Proceedings of 2017 IEEE International Conference on Data Mining Workshops. IEEE, New Orleans, 103–110.