Citation for published version:
Tian, X, Yan, R, Wang, S & Laporte, G 2023, 'Prescriptive analytics for a maritime routing problem', *Ocean and Coastal Management*, vol. 242, 106695. https://doi.org/10.1016/j.ocecoaman.2023.106695

DOI:
10.1016/j.ocecoaman.2023.106695

Publication date:
2023

Document Version
Peer reviewed version

Link to publication

Publisher Rights
CC BY-NC-ND

University of Bath

Alternative formats
If you require this document in an alternative format, please contact:
openaccess@bath.ac.uk

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. Jul. 2024

Prescriptive Analytics for a Maritime Routing Problem

Xuecheng Tian[a], *Ran Yan*[b,1], *Shuaian Wang*[a], *Gilbert Laporte*[c,d]
[a]*Department of Logistics and Maritime Studies, Faculty of Business, The Hong Kong Polytechnic University, Hung Hom, Hong Kong*
[b]*School of Civil and Environmental Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore*
[c]*Department of Decision Sciences, HEC Montréal, Montréal, Québec, Canada*
[d]*School of Management, University of Bath, Bath, United Kingdom*

Abstract: Port state control (PSC) serves as the final defense against substandard ships in maritime transportation. The port state control officer (PSCO) routing problem involves selecting ships for inspection and determining the inspection sequence for available PSCOs, aiming to identify the highest number of deficiencies. Port authorities face this problem daily, making decisions without prior knowledge of ship conditions. Traditionally, a predict-then-optimize framework is employed, but its machine learning (ML) models' loss function fails to account for the impact of predictions on the downstream optimization problem, potentially resulting in suboptimal decisions. We adopt a decision-focused learning framework, integrating the PSCO routing problem into the ML models' training process. However, as the PSCO routing problem is NP-hard and plugging it into the training process of ML models requires that it be solved numerous times, computational complexity and scalability present significant challenges. To address these issues, we first convert the PSCO routing problem into a compact model using undominated inspection templates, enhancing the model's solution efficiency. Next, we employ a family of surrogate loss functions based on noise-contrastive estimation (NCE) for the ML model, requiring a solution pool treating suboptimal solutions as noise samples. This pool represents a convex hull of feasible solutions, avoiding frequent reoptimizations during the ML model's training process. Through computational experiments, we compare the predictive and prescriptive qualities of both the two-stage framework and the decision-focused learning framework under varying instance sizes. Our findings suggest that accurate predictions do not guarantee good decisions; the decision-focused learning framework's performance may depend on the optimization problem size and the training dataset size; and using a solution pool containing noise samples strikes a balance between training efficiency and decision performance.

Keywords: prescriptive analytics; predict-then-optimize; decision-focused learning; port state control (PSC) inspection; maritime routing

1. Introduction

Maritime safety is crucial, as maritime transportation transports over 80% of global goods. 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., 2021), effectively selecting substandard ships for inspection is essential. Existing research, such as Yan et al.

[1] Corresponding author. Email addresses: xuecheng-simon.tian@connect.polyu.hk (X Tian), ran-angela.yan@connect.polyu.hk (R Yan), wangshuaian@gmail.com (S Wang), gilbert.laporte@cirrelt.net (G Laporte)

(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 A.1 in Appendix A). Therefore, this paper innovatively uses PSC inspection data for the PSCO routing problem to maximize the number of deficiencies identified while considering various practical constraints.

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. Prescriptive 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, 2018b; Yan et al., 2020, 2021a, 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 A.2 in Appendix A), 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).

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 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 paper is organized as follows. Section 2 reviews the related literature. Section 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 4 describes the traditional two-stage framework. Based on this framework, Section 5 describes a decision-focused learning framework that uses a new family of noise-contrastive loss functions. Section 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 7 concludes the paper and outlines future research directions.

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 naïve (TAN) 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.

Xu et al. (2007a) developed a support vector machine (SVM) model to predict ship detention using both generic and historical factors. Xu et al. (2007b) subsequently enhanced their prediction performance by considering new features extracted by web mining technology. Gao et al. (2007) further predicted ship detention by integrating the SVM model, the *k*-nearest neighbor model, and the bag-of-words model.

Yang et al. (2018a) predicted the ship detention probability of bulk carriers within the Paris Memorandum of Understanding (MoU) using a data-driven Bayesian network model based on TAN learning. To determine a port's optimal inspection scheme, Yang et al. (2018b) combined the results of the ship detention prediction model with a game model that considers the benefits associated with both port authorities and ship operators. Recently, Wu et al. (2022) predicted ship detention with an SVM model using an analytic hierarchy process and gray relational analysis to select input features. To address the issues caused by the low-probability detention outcome, Yan et al. (2021b) adopted a balanced random forest model to predict ship detention. Tian and Wang (2023) proposed a cost-sensitive Laplacian logistic regression model for ship detention prediction, addressing the two challenges arising from the imbalanced and unlabeled data.

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. (2023) 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., 2021), 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. (2023).

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 directly 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. PSCO Routing Problem

Section 3.1 mathematically presents the PSCO routing problem. Section 3.2 transforms the original PSCO routing problem into a more compact formulation by generating undominated inspection templates. Finally, Section 3.3 conducts several groups of computational experiments to compare the solution efficiency of the two proposed models.

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., 2021). 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. (2021), 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 in the morning and from 14:00 to 17:00 in the afternoon. 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 $N+3$ $(i = 0,...,N+2)$ location indices to be considered in this problem, including N $(i = 1,...,N)$ 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 $(i = 0,...,N+2)$ 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}.

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 actual numbers of ship deficiencies by $\mathbf{d} := \{d_i \mid i = 1,...,N\}$. The sets of decision variables are denoted by $\mathbf{x} := \{x_{ijm} \mid i, j = 0,...,N+2; m = 1,...,M\}$, where $x_{ijm} = 1$ if a visit to location i is followed by a visit to location j by PSCO m and 0 otherwise; $\mathbf{y} := \{y_{im} \mid i = 1,...,N; m = 1,...,M\}$, where $y_{im} = 1$ if ship i is assigned to be inspected by PSCO m and 0 otherwise; and $\mathbf{s} := \{s_{im} \mid i = 0,...,N+2; m = 1,...,M\}$, 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:

[M1]

$$\max_{\mathbf{y}} z(\mathbf{d}, \mathbf{y}) = \max_{\mathbf{y}} \sum_{m=1}^{M} \sum_{i=1}^{N} d_i y_{im} \tag{1}$$

subject to

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

$$\sum_{i=0}^{N+1} x_{ikm} = \sum_{j=1}^{N+2} x_{kjm} = y_{km} \quad k = 1,...,N+1; m = 1,...,M \tag{3}$$

$$s_{im} + t_{ij} + t'_i - s_{jm} \le P(1 - x_{ijm}) \quad i, j = 0,...,N+2; m = 1,...,M \tag{4}$$

$$\sum_{m=1}^{M} y_{im} \le 1 \quad i = 1,...,N \tag{5}$$

$$y_{im} = 1 \quad i = 0, N+1, N+2; m = 1, \ldots, M \tag{6}$$

$$O_i - P(1 - y_{im}) \leq s_{im} \quad i = 0, \ldots, N+2; m = 1, \ldots, M \tag{7}$$

$$s_{im} + t'_i \leq C_i + P(1 - y_{im}) \quad i = 0, \ldots, N+2; m = 1, \ldots, M \tag{8}$$

$$x_{ijm}, y_{im} \in \{0,1\} \quad i, j = 0, \ldots, N+2; m = 1, \ldots, M \tag{9}$$

$$s_{im} \geq 0 \quad i, j = 0, \ldots, N+2; m = 1, \ldots, M. \tag{10}$$

Objective function (1) maximizes the number of deficiencies identified among all foreign visiting ships to be inspected. Constraint (2) guarantees that all PSCOs start work at location 0 and end work at location $N+2$. Constraints (3) and (4) guarantee each PSCO's connectivity and timeline. Constraints (5) mean that each ship is inspected no more than once. Constraints (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 (7) and (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.

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. Through the route generation method, 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 Appendix B, which follows the method shown in Yan et al. (2021a), and the method of routing the available PSCOs is illustrated in **Procedure 1**.

Through **Procedure B.1** shown in Appendix B, we obtain the set of all undominated inspection templates, denoted by $\mathbf{H}$. After obtaining the set of undominated inspection templates $\mathbf{H}$ and parameters η_i^h (binary parameter indicating whether ship i is contained in inspection template 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 $u_i \in \mathbf{u}$, $i = 1, \ldots, N$, which equals 1 if and only if ship i is inspected by one PSCO. Then, model M2 is as follows:

[M2]

$$\max_{\mathbf{u}} z(\mathbf{d}, \mathbf{u}) = \max_{\mathbf{u}} \sum_{i=1}^{N} d_i u_i \tag{11}$$

subject to

$$\sum_{h \in \mathrm{H}} y'_h \leq M \tag{12}$$

$$u_i \leq \sum_{h \in \mathrm{H}} \eta_i^h y'_h \quad i = 1, \ldots, N \tag{13}$$

$$y'_h \in \{0,1\} \quad h \in \mathbf{H} \tag{14}$$

$$u_i \in \{0,1\} \quad i = 1, \ldots, N. \tag{15}$$

Objective function (14) maximizes the total number of deficiencies identified. Constraint (15) provides that the maximum number of adopted inspection templates cannot exceed the number of available PSCOs. Constraints (16) indicate the relationship between u_i and y'_h.

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 S_o containing $|S_o|$ ships, adopting this inspection template requires visiting $|S_o|+3$ locations and generating $(|S_o|+1)!$ candidate routes for the PSCOs to complete their tasks. The above-defined earliest return time to the office is denoted by $\zeta_{|S_o|+3}$. The **Subprocedure** to find the optimal route for a selected inspection template is introduced as follows.

Subprocedure. Find the optimal route for an inspection template.

```
Input: a selected inspection template $S_o$; set of locations $i \in \{0, \ldots, N+2\}$ (including berthing
locations of ships in $S_o$, lunch break location, and starting and ending location); duration spent at
each location $t_i$ $(i = 0, \ldots, N+2)$; time window of each location $[O_i, C_i]$ $(i = 0, \ldots, N+2)$.
Output: Optimal_route.
--------------------------------------------------------------------------------
Define set $\mathbf{V}_c$ that contains all of the routes of starting work, inspecting the ships in $S_o$, having
lunch break, and ending work. Denote a candidate route as $p_j$, $p_j \in V_c$, $j = 1, \ldots, (|S_o|+1)!$ and the
earliest return time to the office for $p_j$ as $\zeta_{p_j}$.
Initialize Optimal_route $= \varnothing$, Earliest_finish_time $= \infty$.
For $j = 1, \ldots, (|S_o|+1)!$ do:
    Test the feasibility of a candidate route $p_j$ using Proposition B.1.
    If $p_j$ is feasible and $\zeta_{p_j} \leq$ Earliest_finish_time:
        Earliest_finish_time $\leftarrow \zeta_{p_j}$.
        Optimal_route $\leftarrow p_j$.
    End if
End for
```

After obtaining the optimal routes for all of the selected inspection templates, 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 $\mathbf{S}_d$ containing the ships that have already been inspected. The overall procedure of PSCO routing after solving M2 is shown in **Procedure 1**.

Procedure 1. PSCO routing after solving M2.

```
Input: set of undominated inspection templates $\boldsymbol{H}$, optimal solution $y'_h$.
Output: Scheduling results.
Define $\mathbf{H}' = \{h \mid y'_h = 1\}$ as the set of selected inspection templates, and the size of this set is defined by $|\mathbf{H}'|$.
Denote $\mathbf{S}_d = \varnothing$ as the set of inspected ships.
For $j = 1,...,|\mathbf{H}'|$ do:
    If $j = 1$:
        Determine the optimal route $p_1$ for inspection template $h_1$ by applying Subprocedure.
        Assign $p_1$ to PSCO 1.
    Else:
        Update $\mathbf{S}_d$ by merging the last inspection template $h_{j-1}$: $\mathbf{S}_d \leftarrow \mathbf{S}_d \cup h_{j-1}$.
        Delete $\mathbf{S}_d$ from the current inspection template $h_j$: $h_j \leftarrow h_j \setminus \mathbf{S}_d$.
        Determine the optimal route $p_j$ for inspection template $h_j$ by applying Subprocedure.
        Assign $p_j$ to PSCO $j$.
    End if
End for
For $m = |\mathbf{H}'|+1,...,M$ do:
    Not assign any route to PSCO $m$.
End for
```

Finally, by observing Constraint (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.

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 $t'_{e_ie_j}$ 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 $t'_{e_ie_j}$. 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 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 2.

Table 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 2. The ranges of input integer parameters for models M1 and M2

Parameter	O_i	C_i	e_i	d_i
Range	8~11	13~17	2~5	0~10

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.

As we can see from Table 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. Compared 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 paper will use M2 as the target optimization problem, which is plugged into the ML algorithm in the following sections.

Table 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 Procedure B.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 Procedure 1 (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[2]			0.00%	0.00%	0.00%	0.09%

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 $D=\{(\mathbf{a}_i,d_i)\}_{i=1}^{R}$ 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 $\hat{d}=f(\boldsymbol{\omega},\mathbf{a})$. This ML model is trained to minimize a specified loss function using dataset D, such as the mean squared error (MSE) loss

[2] $\text{Gap of M2 to M1} = \dfrac{\text{objective function value of M2} - \text{objective function value of M1}}{\text{objective function value of M1}} \times 100\%.$.

function L_{MSE} for a regression task defined as follows:

$$L_{MSE} = \frac{1}{R}\sum_{i=1}^{R}(\hat{d}_i - d_i)^2. \tag{16}$$

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

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} L_{MSE} = \arg\min_{\boldsymbol{\omega}} \frac{1}{R}\sum_{i=1}^{R}(\hat{d}_i - d_i)^2 . \tag{17}$$

Then, when presented with a new example with feature vector $\mathbf{a}$, model f with the optimal $\boldsymbol{\omega}^*$ can be applied to predict the number of deficiencies of the new example $\hat{d} = f(\boldsymbol{\omega}^*, \mathbf{a})$. Finally, the predicted values $\hat{d}$ 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 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., $\boldsymbol{\omega}$ 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 D.1** in Appendix D depicts the two-stage framework including a standard gradient-descent learning procedure.

5. The Decision-Focused Learning Framework

This section introduces the decision-focused learning framework. Section 5.1 defines the regret loss. Section 5.2 introduces a new family of noise-contrastive loss functions. Section 5.3 describes the gradient-descent decision-focused learning framework using the noise-contrastive losses proposed in Section 5.2. We mainly follow the decision-focused learning framework proposed by Mulamba et al. (2021).

5.1 The regret 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 suboptimal 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 $f'(\boldsymbol{\omega}',\mathbf{a}_i)$, is trained to generate predictions $\hat{d}_i'$ for ship i that can provide optimal decisions with respect to the real values of d_i. To measure the accuracy of the prescribed decisions, instead of using the loss function (19), we adopt the 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

$$\boldsymbol{\omega}'^* = \arg\min_{\boldsymbol{\omega}'} L_{regret} = \arg\min_{\boldsymbol{\omega}'} \frac{1}{T}\sum_{j=1}^{T}\left[z(\mathbf{d}_j,\mathbf{u}^*(\mathbf{d}_j)) - z(\mathbf{d}_j,\mathbf{u}^*(f'(\boldsymbol{\omega}',\mathbf{A}_j)))\right], \tag{18}$$

where $\mathbf{A}_j=(\mathbf{a}_1,\mathbf{a}_2,...,\mathbf{a}_N)$ is an $N\times q$ feature matrix for instance j (recall that a ship has q features), and $f'(\boldsymbol{\omega}',\mathbf{A}_j)$ is the vector of the predicted number of deficiencies $\hat{\mathbf{d}}_j'$. By observing Eq. (21), we find that the regret loss is the sum of the difference between the objective function values derived from 1) the perfect solution $\mathbf{u}^*(\mathbf{d})$ under the real deficiency number vector $\mathbf{d}$; and 2) the optimal solution $\mathbf{u}^*(\hat{\mathbf{d}}')$ under the predicted deficiency number vector $\hat{\mathbf{d}}'$.

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 $\hat{\mathbf{d}}'$. 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.

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. Using the contrastive losses proposed by Mulamba et al. (2021), we adopt 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 (Mulamba et al., 2021). 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 paper thus uses an exponential distribution to fit model M2 (Mulamba et al., 2021).

Let U denote the state space of the feasible solutions of an optimization problem $\max_{\mathbf{u}\in\mathsf{U}} z(\mathbf{d},\mathbf{u})$ with input data $\mathbf{A}$, where $\mathbf{u}$ is a feasible solution that satisfies $\mathbf{u}\in\mathsf{U}$. Then, we define the following exponential distribution over U, which represents the probability of deriving solution $\mathbf{u}$ under prediction $f'(\boldsymbol{\omega}',\mathbf{A})$:

$$\mathrm{P}(\mathbf{u}\,|\,f'(\boldsymbol{\omega}',\mathbf{A})) = \frac{1}{\mathsf{Z}}\exp\left(z(f'(\boldsymbol{\omega}',\mathbf{A}),\mathbf{u})\right), \tag{19}$$

where Z normalizes the distribution over the state space U and is expressed as

$$\mathsf{Z} = \sum_{\mathbf{u}'\in\mathsf{U}} \exp\left(z(f'(\boldsymbol{\omega}',\mathbf{A}),\mathbf{u}')\right). \tag{20}$$

If $\mathbf{u}^*(f'(\boldsymbol{\omega}',\mathbf{A}))$ is the maximizer of $\max_{\mathbf{u}\in\mathsf{U}} z(\mathbf{d},\mathbf{u})$ for an instance with input data $\mathbf{A}$, it can maximize Eq. (22) among all $\mathbf{u}\in\mathsf{U}$. This implies that if we can learn the parameter $\boldsymbol{\omega}'$, which can maximize the likelihood of $\mathrm{P}(\mathbf{u}^*(\mathbf{d})\,|\,f'(\boldsymbol{\omega}',\mathbf{A}))$, we can obtain the true perfect solution $\mathbf{u}^*(\mathbf{d})$ with the highest probability under the prediction $f'(\boldsymbol{\omega}',\mathbf{A})$. Consequently, for all training instances, our goal is to learn the parameters in $\boldsymbol{\omega}'$ that can maximize the likelihood 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 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, but not a perfect solution; we denote the solution pool of noise samples by U'. In Section 5.3, we describe the method of establishing the solution pool. In Appendix C, we introduce the detailed four forms of decision loss functions based on NCE that are proposed by Mulamba et al. (2021).

5.3 Gradient-descent decision-focused learning with noise samples

Based on the four types of contrastive losses shown in Appendix C, the main problem to be solved now is how to formulate the solution pool of noise samples U'. We note that any feasible solution in U is a noise sample in U'. However, finding all of the feasible solutions in U is time-consuming and nearly impossible, especially for large-scale combinatorial problems. Therefore, following Mulamba et al. (2021), we first initialize U' by solving models using the real values of $\mathbf{d}$ before the training process, and then expand while obtaining a new solution by solving model M2 using the predicted $\hat{\mathbf{d}}$ during the training process. **Algorithm D.2** in Appendix D shows the procedure of the gradient-descent decision-focused learning framework with noise samples (Mulamba et al., 2021).

Notably, there are three main differences between **Algorithm D.1** and **Algorithm D.2**. First, the data records used to train the ML model in **Algorithm D.2** 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 $\boldsymbol{\omega}'$, **Algorithm D.2** initializes the solution pool 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 U, because the noise samples in U' can represent the convex hull of 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 U (Mulamba et al., 2021). Third, there is a parameter p_{solve} representing the probability of calling a solver to obtain the current prescribed solution in **Algorithm D.2**. 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; otherwise, an $\arg\max$ operation on the solution pool is performed to find an existing solution that maximizes the estimated objective function value. 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.

6. Computational Experiments

This section presents the results of our computational experiments. In Section 6.1, we describe our dataset and the settings for each ML model. In Section 6.2, we compare the performance of the two-stage framework and the decision-focused learning framework and present several interesting findings. In Section 6.3, we conduct a sensitivity analysis on p_{solve}.

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 MoU, which governs the Asia-Pacific region. The PSC inspection records are retrieved from the Asia Pacific Computerized Information System[3] provided by the Tokyo MoU, and the ship-related factors are obtained from the World Shipping Register database[4]. 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 paper considers 14 auxiliary features that are closely related to ship condition, according to the literature (Yan et al., 2020, 2021b),

[3] https://apcis.tmou.org/public/.
[4] https://world-ships.com/.

namely, ship age, gross tonnage (GT), 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, 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 fivefold 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.

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, 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}. The best hyperparameter tuples for each model are shown in Table E.1 in Appendix E.

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. 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 4.

Table 4. The computational results of 30 experiments using models with different loss functions and 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 Eq. (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 5 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 $C_{600}^{8}=3.96\times10^{17}$ 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 1.

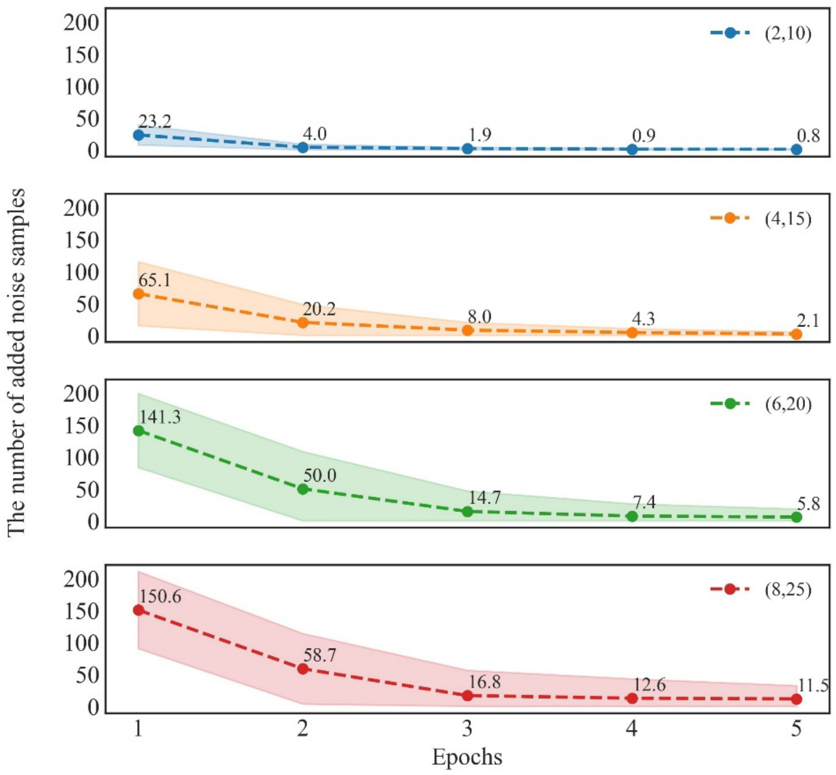

Figure 1. The number of added noise samples in each epoch for experiments using the MAP-variant loss under different instance sizes

As shown in Figure 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 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.

6.3 The influence of parameter p_{solve}

In **Algorithm D.2**, 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 D.2**. 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 2 and 3.

Figure 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 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 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.

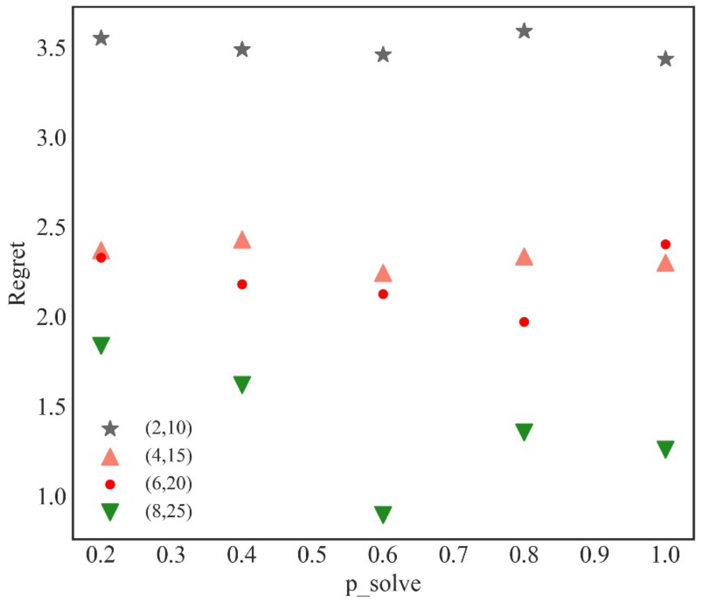

Figure 2. The average regret of the 30 experiments using models adopting the MAP-variant loss with different p_{solve} and different instance sizes

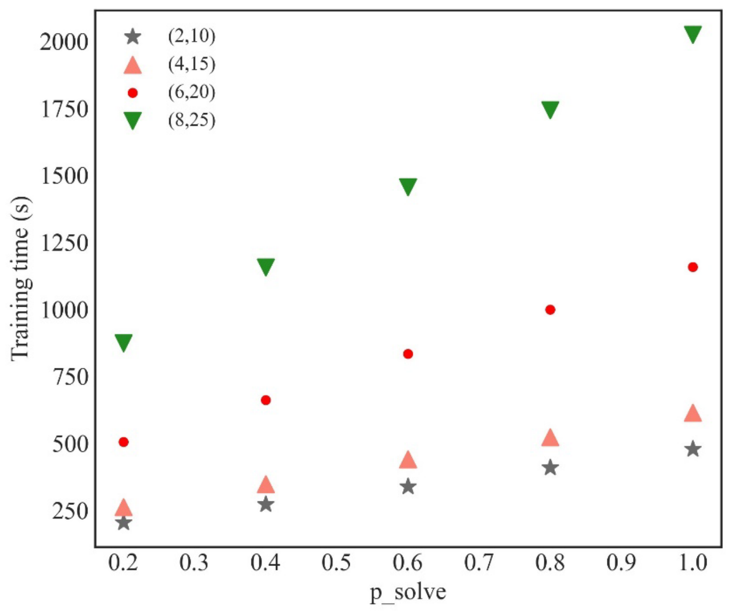

Figure 3. The average training time of the 30 experiments using models adopting the MAP-variant loss with different p_{solve} and different instance sizes

Figure 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 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 p_{solve} 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 between efficiency and accuracy.

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 depend 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.

As the first study to propose routing decisions for PSCOs under different prescriptive analytics frameworks, our research has 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 ANNs 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.

Acknowledgment

This work was supported by the National Natural Science Foundation of China [Grant Nos. 71831008, 72071173] and the Research Grants Council of the Hong Kong Special Administrative Region, China [Project numbers 15201121, HKSAR RGC TRS T32-707-22-N]. This work is supported by the Start-Up Grant from Nanyang Technological University, Singapore.

References

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

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.

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.

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

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, 1504–1511.

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, 242–244.

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 ICLR 2015, 1–6.

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, 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.

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

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

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, 1603–1610.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems 26, 3111–3119.

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 2021 International Joint Conference on Artificial Intelligence, 2833–2840.

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.

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.

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.

Tian, X., Wang, S., 2023. Cost-sensitive Laplacian Logistic regression for ship detention prediction. Mathematics 11(1), 119.

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.

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.

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.

Wilder, B., Dilkina, B., Tambe, M., 2019. Melding the data-decisions pipeline: decision-focused learning for combinatorial optimization. In Proceedings of the 33 AAAI Conference on Artificial Intelligence, 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, 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, 427–434.

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, 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.

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

Appendix A. Illustrative examples

Example A.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. □

Example A.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. □

Appendix B. Procedures of generating undominated inspection templates (Yan et al, 2021a)

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 $C_N^L = N!/\left(L!(N-L)!\right)$. Denote a combination of L ships by the set S, where $|S| = L$. We then define set 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.

Procedure B.1 illustrates the basic idea of selecting the inspection templates 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 set 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. 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 is the l^{th} 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:

$$\zeta_l \geq \lambda_{\alpha_l} \quad l = 1,...,N+3 \tag{21}$$

$$\zeta_l + t_{\alpha_l} \leq \bar{\lambda}_{\alpha_l} \quad l = 1,...,N+3 \tag{22}$$

$$\zeta_{l+1} \geq \zeta_l + t_{\alpha_l} + t'_{l,l+1} \quad l = 1,...,N+2, \tag{23}$$

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

Proposition B.1: For a candidate route, whether constraints (11)–(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, $l = 2,\dots,L+3$, let its start time $\zeta_l^* = \max\{\zeta_{l-1}^* + t_{\alpha_{l-1}} + t'_{l-1,l}, \lambda_{\alpha_l}\}$, $l = 2,\dots,L+3$; if $\zeta_l^* \le \overline{\lambda}_{\alpha_l} - t_{\alpha_l}$, $l = 1,\dots,L+3$, then the candidate route is feasible, otherwise it is infeasible.

Property B.1: For two inspection templates S and S', if $S \ne S'$ and $S \subseteq S'$, then inspection template S is dominated by inspection template S'. If inspection template 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.

Procedure B.1. Generate undominated inspection templates.

```
Input: set of locations $i \in \{0,\dots,N+2\}$; duration spent at each location $t_i$ $(i = 0,\dots,N+2)$; time window of each location $[O_i, C_i]$ $(i = 0,\dots,N+2)$.
Output: the set of undominated inspection templates $\mathbf{H}$, binary variable parameter $\eta_i^h$ indicating whether ship $i$ is contained in inspection template $h$.
Initialize $\mathbf{H} = \varnothing$, $\eta_i^h = 0$, $i = 1,\dots,N$, $h = 0$.
For $L = 0,1,2,3$ do:
    Formulate all combinations containing $L$ ships among all visiting ships denoted by $\mathbf{Q}$.
    For each combination $q \in \mathbf{Q}$ do:
        Initialize feasibility = False.
        Define set $\mathbf{V}$ that contains all 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 B.1.
            If $v$ is feasible:
                $\mathbf{H} \leftarrow \mathbf{H} \cup \{q\}$.
                Set $\eta_i^h = 1$ for those locations included in $q$.
                Set $h \leftarrow h+1$.
                Update feasibility = True.
                Break.
            End if
        End for
        If feasibility = True:
            Continue.
        End if
    End for
End for
Delete dominated inspection templates in $\mathbf{H}$ using Property B.1.
```

Appendix C. Four forms of contrastive loss functions (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 U', where $\mathsf{U}' \subset \mathsf{U} \setminus \{\mathbf{u}^*(\mathbf{d})\}$. These noise samples constitute the solution pool that can be regarded as an approximation of Z. Therefore, U' is the solution pool that we need. Next, our goal is to learn the parameter $\boldsymbol{\omega}'$ by maximizing the product of the ratios between the probability of the perfect solution $\mathbf{u}^*$ under the prediction $f'(\boldsymbol{\omega}', \mathbf{A})$ and the probability of any noise sample $\mathbf{u}''$ in U' under the prediction $f'(\boldsymbol{\omega}', \mathbf{A})$ for any instance with input data $\mathbf{A}$, which is expressed as

$$\begin{aligned}
\boldsymbol{\omega}'^* &= \arg\max_{\boldsymbol{\omega}'} \log \prod_{j=1}^{T} \prod_{\mathbf{u}'' \in \mathsf{U}'} \frac{\mathrm{P}(\mathbf{u}_j^* \mid f'(\boldsymbol{\omega}', \mathbf{A}_j))}{\mathrm{P}(\mathbf{u}'' \mid f'(\boldsymbol{\omega}', \mathbf{A}_j))} \\
&= \arg\max_{\boldsymbol{\omega}'} \log \prod_{j=1}^{T} \prod_{\mathbf{u}'' \in \mathsf{U}'} \frac{\exp\left(z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \mathbf{u}_j^*)\right)}{\exp\left(z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \mathbf{u}'')\right)} \\
&= \arg\max_{\boldsymbol{\omega}'} \sum_{j=1}^{T} \sum_{\mathbf{u}'' \in \mathsf{U}'} \left(z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \mathbf{u}_j^*) - z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \mathbf{u}'')\right).
\end{aligned} \tag{24}$$

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

$$L_{NCE} = \sum_{j=1}^{T} \sum_{\mathbf{u}'' \in \mathsf{U}'} \left(z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \mathbf{u}'') - z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \mathbf{u}_j^*)\right). \tag{25}$$

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

MAP-basic loss. Furthermore, instead of considering all of the noise samples in 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 with the highest probability of achieving the optimal objective function value for each instance under the current prediction. Similarly, learning the parameter $\boldsymbol{\omega}'$ by maximizing the product of the ratios between the probabilities of the perfect solution $\mathbf{u}^*$ and the noise sample in U' with the highest likelihood of generating the optimal objective function value based on the current prediction can be expressed as

$$\begin{aligned}
\boldsymbol{\omega}'^* &= \arg\max_{\boldsymbol{\omega}'} \log \prod_{j=1}^{T} \frac{\mathrm{P}(\mathbf{u}_j^* \mid f'(\boldsymbol{\omega}', \mathbf{A}_j))}{\mathrm{P}(\hat{\mathbf{u}}_j^* \mid f'(\boldsymbol{\omega}', \mathbf{A}_j))} \\
&= \arg\max_{\boldsymbol{\omega}'} \log \prod_{j=1}^{T} \frac{\exp\left(z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \mathbf{u}_j^*)\right)}{\exp\left(z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \hat{\mathbf{u}}_j^*)\right)} \\
&= \arg\max_{\boldsymbol{\omega}'} \sum_{j=1}^{T} \left(z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \mathbf{u}_j^*) - z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \hat{\mathbf{u}}_j^*)\right).
\end{aligned} \tag{26}$$

where $\hat{\mathbf{u}}_j^* = \arg\max_{\mathbf{u}'' \in \mathsf{U}'} [z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \mathbf{u}'')]$. Accordingly, the above equation can be transformed into the following MAP-basic loss function:

$$L_{MAP} = \sum_{j=1}^{T} \left(z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \hat{\mathbf{u}}_j^*) - z(f'(\boldsymbol{\omega}', \mathbf{A}_j), \mathbf{u}_j^*) \right). \quad (27)$$

NCE-variant loss and MAP-variant loss. We note that the objective function of model M2 is a linear function. Therefore, the original L_{NCE} and L_{MAP} can be rewritten in the following linear form:

$$L_{NCE} = \sum_{j=1}^{T} \sum_{\mathbf{u}^n \in \mathrm{U}'} \left(f'(\boldsymbol{\omega}', \mathbf{A}_j)(\mathbf{u}^n - \mathbf{u}_j^*) \right), \quad (28)$$

$$L_{MAP} = \sum_{j=1}^{T} \left(f'(\boldsymbol{\omega}', \mathbf{A}_j)(\hat{\mathbf{u}}_j^* - \mathbf{u}_j^*) \right). \quad (29)$$

By observing Eq. (28) and Eq. (29), we find that if $\mathbf{d}_j$ for instance j is predicted to be $\mathbf{0}$, namely, $f'(\boldsymbol{\omega}', \mathbf{A}_j) = 0$, L_{NCE} and L_{MAP} are 0, which is the minimum loss. To avoid this case, we introduce variants of Eq. (25) and Eq. (27) by replacing $f'(\boldsymbol{\omega}', \mathbf{A}_j)$ with $f'(\boldsymbol{\omega}', \mathbf{A}_j) - \mathbf{d}_j$. This modification can be regarded as adding a regularization term to keep $f'(\boldsymbol{\omega}, \mathbf{A}_j)$ close to $\mathbf{d}_j$. Therefore, we can obtain the NCE-variant loss and MAP-variant loss as follows:

$$L_{NCE-v} = \sum_{j=1}^{T} \sum_{\mathbf{u}^n \in \mathrm{U}'} \left((f'(\boldsymbol{\omega}', \mathbf{A}_j) - \mathbf{d}_j)(\mathbf{u}^n - \mathbf{u}_j^*) \right), \quad (30)$$

$$L_{MAP-v} = \sum_{j=1}^{T} \left((f'(\boldsymbol{\omega}', \mathbf{A}_j) - \mathbf{d}_j)(\hat{\mathbf{u}}_j^* - \mathbf{u}_j^*) \right). \quad (31)$$

By observing Eq. (30) and Eq. (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 $\mathbf{u}_j^*$, which is the ultimate goal of decision-focused learning.

Appendix D. Algorithms for two-stage and decision-focused learning frameworks

Algorithm D.1. Two-stage framework.

```
Input: Training data $D = \{(\mathbf{a}_i, d_i)\}_{i=1}^{R}$.
Hyperparameters: learning rate $\alpha$, epochs, batch size.
Initialize $\boldsymbol{\omega}$.
For each epoch do:
    For each batch do:
        For each example $\mathbf{a}$ do:
            Predict the number of deficiencies of $\mathbf{a}$, denoted by $\hat{d} = f(\boldsymbol{\omega}, \mathbf{a})$.
        End for
        Calculate the accumulated MSE loss $L_{MSE}^{b}$ for the set of examples in a batch.
        Update $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \alpha \frac{\partial L_{MSE}^{b}}{\partial \hat{\mathbf{d}}^{b}} \frac{\partial \hat{\mathbf{d}}^{b}}{\partial \boldsymbol{\omega}}$, where $\hat{\mathbf{d}}^{b}$ denotes the vector of predicted values of
        examples in a batch.
    End for
End for
Predict the numbers of deficiencies of a set of new examples $\mathbf{A}'$, denoted by a vector
$\hat{\mathbf{d}}' = f(\boldsymbol{\omega}^{*}, \mathbf{A}')$, where $\boldsymbol{\omega}^{*}$ is the final optimal weight of the trained ANN.
Input $\hat{\mathbf{d}}'$ into the downstream optimization problem M2 to generate PSCO routing decisions.
```

Algorithm D.2. Decision-focused learning framework (Mulamba et al., 2021).

```
Input: Auxiliary parameters in model M2; training data $D' = \{(\mathbf{A}_j, \mathbf{d}_j)\}_{j=1}^{T}$, a fixed parameter $p_{solve}$.
Hyperparameters: learning rate $\alpha'$, epochs, batch size.
Initialize $\boldsymbol{\omega}'$, $\mathsf{U}' = \{\mathbf{u}^{*}(\mathbf{d}_j) \mid (\mathbf{A}_j, \mathbf{d}_j) \in D'\}$.
For each epoch do:
    For each batch do:
        For each instance $\mathbf{A}$ do:
            Predict the number of deficiencies of the examples in instance $\mathbf{A}$, denoted by
            $\hat{\mathbf{d}} = f'(\boldsymbol{\omega}', \mathbf{A})$.
            If a random number between 0 and 1 is smaller than $p_{solve}$:
                Obtain $\mathbf{u}(\hat{\mathbf{d}})$ by solving model M2 with $\hat{\mathbf{d}}$.
                $\mathsf{U}' \leftarrow \mathsf{U}' \cup \{\mathbf{u}(\hat{\mathbf{d}})\}$.
            Else:
                $\mathbf{u}' = \arg\max_{\mathbf{u} \in \mathsf{U}'} (z(\mathbf{u}, \hat{\mathbf{d}}))$.
                $\mathsf{U}' \leftarrow \mathsf{U}' \cup \{\mathbf{u}'\}$.
            End if
        End for
        Calculate the accumulated regret loss $L_{b}^{u^{*}}$ for the set of instances in a batch.
        $\boldsymbol{\omega}' \leftarrow \boldsymbol{\omega}' - \alpha' \frac{\partial L_{b}^{u^{*}}}{\partial \hat{\mathbf{D}}^{b}} \frac{\partial \hat{\mathbf{D}}^{b}}{\partial \boldsymbol{\omega}'}$ where $\hat{\mathbf{D}}^{b}$ dnotes the matrix of the predicted number of
        deficiencies of the examples in a batch.
    End for
End for
```

Appendix E. Hyperparameter tuning results

Table E.1. Hyperparameter tuning for each model

Instance	Method	Loss function	Hyperparameter		
			Learning rate	Epochs	Batch size
(2,10)	Decision-focused learning	NCE_basic	0.0005	25	32
		NCE_variant	0.0001	25	16
		MAP_basic	0.01	25	16
		MAP_variant	0.05	5	16
	Two-stage	MSE	0.05	5	16
(4,15)	Decision-focused learning	NCE_basic	0.0005	10	8
		NCE_variant	0.0005	5	8
		MAP_basic	0.05	25	16
		MAP_variant	0.01	5	16
	Two-stage	MSE	0.005	25	32
(6,20)	Decision-focused learning	NCE_basic	0.001	5	16
		NCE_variant	0.0001	25	16
		MAP_basic	0.05	15	16
		MAP_variant	0.0005	5	16
	Two-stage	MSE	0.01	25	32
(8,25)	Decision-focused learning	NCE_basic	0.0005	5	16
		NCE_variant	0.0001	15	16
		MAP_basic	0.05	20	16
		MAP_variant	0.0005	5	16
	Two-stage	MSE	0.005	15	32